Using a Mahalanobis-Like Distance to Train Radial Basis Neural Networks

J.M. Valls, R. Aler, and O. Fernández

Carlos III University - Computer Science Department,
Avenida de la Universidad, 30 - 28911 Leganés (Madrid), Spain
jvalls@inf.uc3m.es

Abstract. Radial Basis Neural Networks (RBNN) can approximate any regular function and have a faster training phase than other similar neural networks. However, the activation of each neuron depends on the euclidean distance between a pattern and the neuron center. Therefore, the activation function is symmetrical and all attributes are considered equally relevant. This could be solved by altering the metric used in the activation function (i.e. using non-symmetrical metrics). The Mahalanobis distance is such a metric, that takes into account the variability of the attributes and their correlations. However, this distance is computed directly from the variance-covariance matrix and does not consider the accuracy of the learning algorithm. In this paper, we propose to use a generalized euclidean metric, following the Mahalanobis structure, but evolved by a Genetic Algorithm (GA). This GA searches for the distance matrix that minimizes the error produced by a fixed RBNN. Our approach has been tested on two domains and positive results have been observed in both cases.

1 Introduction

Radial Basis Neural Networks (RBNN) [1, 2] are originated from the use of radial basis functions, in the solution of the real multivariate interpolation problem [3, 4]. As the Multilayer perceptron (MLP) they can approximate any regular function [5]. Due to its local behavior and to the linear nature of its output layer, their training is faster than MLP training [5] and this fact makes them useful for a wide variety of applications. The most used radial basis functions are Gaussian functions, defined by equation 1.

$$\phi_m(x_k) = e^{-\frac{\|\mathbf{c_m}-\mathbf{x_k}\|^2}{2\sigma_m^2}} \tag{1}$$

Where $\phi_m(x_k)$ represents the activation function for neuron m when an input pattern x_k is presented. The vector c_m is the center of the neuron m, and σ_m is its deviation or width.

One of the problems of RBNN is the symmetrical nature of their activation function, making that the activation of a neuron when a pattern is presented, only depends on the euclidean distance from this pattern to the neuron center regardless of the importance of each attribute. This could be solved by altering the metric used in the activation function.

The Mahalanobis distance is a metric used in Statistics in order to normalize different attributes and take into account the correlation among them. This distance is computed according to expression 2.

$$d_{ij} = [(\mathbf{x_i} - \mathbf{x_j})^T \mathbf{S}^{-1} (\mathbf{x_i} - \mathbf{x_j})]^{1/2} = [(\mathbf{x_i} - \mathbf{x_j})^T \mathbf{M}^T \mathbf{M} (\mathbf{x_i} - \mathbf{x_j})]^{1/2} \tag{2}$$

Where d_{ij} is the Mahalanobis distance between vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, S is the variance-covariance matrix of all vectors in the data set and M is the so-called Mahalanobis matrix[6, 7, 8]. This distance can be used to improve prediction accuracy in those learning systems that use distances [9]. However, the Mahalanobis distance is independent of the learning system used and of the error produced on the training data, because it is computed from the points in the data set only (more specifically, it is computed from the variance-covariance matrix).

In this paper, we propose to use a Mahalanobis-like distance in the activation function of the RBNN so that different attributes are treated differently according to their relevance. But instead of computing the distance using the variance-covariance matrix S, a matrix will be built in order to minimize the error of the network. This will be achieved by a genetic algorithm [10] whose individuals are generalized euclidean distance matrices and whose fitness function depends on the prediction accuracy attained by the network using the matrix.

2 Description of the Method

In this paper we use a standard Genetic Algorithm (GA) [10] to evolve distance matrices. A genetic algorithm is a kind of heuristic search. The algorithm maintains a set of candidate solutions (or population of individuals) and applies the search operators on them (also called genetic operators: mutation and crossover). The search is guided by a heuristic (or fitness) function. We have used a generational Genetic Algorithm with elitism and tournament selection. Matrices in the individuals are coded by representing each of their components in binary format. The fitness function is computed by training a RBNN on a set of training data and determining the training error. Thus, the GA tries to find the distance matrix that minimizes the RBNN training error. The number of hidden neurons is fixed from the start.

In order to determine the appropriate M matrix by using GA, individuals must be properly encoded. We have chosen matrix M to be symmetrical, to ensure that $M^T M$ is invertible, although in the future this restriction could be removed in favor of less restrictive conditions. In that case, only the diagonal and the upper half of the matrix coefficients must be encoded to a binary representation in order to build the chromosome of the individual. Each matrix element is a real number that must be encoded to a binary representation with a fixed number of bits, following a fixed-point representation with a single bit for the sign. Hence, the chromosome is a string of bits formed by the binary representation of each matrix element belonging to the diagonal or the upper half of the matrix. I.e. if the matrix is

$$\mathbf{M} = \begin{pmatrix} \mathbf{m}_{11} & \mathbf{m}_{12} & \ldots & \mathbf{m}_{1d} \\ m_{21} & \mathbf{m}_{22} & \ldots & \mathbf{m}_{2d} \\ \vdots & \vdots & \vdots & \vdots \\ m_{d1} & m_{d2} & \ldots & \mathbf{m}_{dd} \end{pmatrix}$$

The corresponding string chromosome will be:

$$\{B(m_{11}), B(m_{12}), \ldots, B(m_{1d}), B(m_{22}), B(m_{23}), \ldots, B(m_{dd})\}$$

where $B(m_{ij})$ is the binary representation of m_{ij}.

Each individual represents a matrix M that will determine the distance function to be used for the neurons activation (see Eq. 2). The goal of this work consists of improving the accuracy or the RBNN; hence, if the network error is small it means that the corresponding distance function is good; thus, the individual representing the M matrix must have a big fitness value. The fitness function chosen in this work is given in equation 3.

$$fitness = -log_2 E \tag{3}$$

where E is the mean squared error committed by the network on the training data. It has been chosen so that fitness increases when error decreases. This function also manages to amplify differences between individuals whose error is close to zero. This is important to increase evolutionary pressure in the latest stages of GA-evolution, when all individuals are very good.

In the next the sequential structure of the proposed method is summarized.

---

1. Create the initial population. A set of random chromosomes is generated. These chromosomes represent different distance functions to be used in the Radial Basis Functions of the networks.
2. Evaluate the fitness of each element of the current population. In order to perform this point, RBNN with a fixed number of hidden neurons are trained using the distance function determined by each individual of the population. Training errors of these networks are used to calculate the fitness of each individual.
3. Apply genetic operators to the population in order to create the next generation.
4. If the number of generations is lower than the maximum, go to step 2
5. Return the highest fitness matrix

---

3 Empirical Evaluation

The purpose of this section is to validate empirically our approach. Two sets of experiments will be carried out. First, a synthetic domain, where the solution is known, will be posed to the system. Next, the well-known Mackey-Glass regression problem will be tested.

Table 1. Parameters of the Genetic Algorithm for the gaussian domain

Generations	30
Tournament size	2
Population size	20
Elitism	1
Crossover probability	0.6
Mutation probability	0.03

3.1 Synthetic Domain

This domain follows a bi-variate gaussian shape ($\mu = (0.5, 0.5)$, $\sigma^2 = 0.002$). However, instead of the euclidean distance, a generalized euclidean distance with matrix M will be used instead (see Eq. 2 and matrix 4). In euclidean space, the result is a rotated and stretched gaussian (i.e. non-symmetrical). In short, the goal is to approximate the function given by Eq. 5, where M is given by Eq. 4.

$$M = \begin{pmatrix} 0.2 & 0.75 \\ 0.75 & 1.0 \end{pmatrix} \tag{4}$$

$$e^{-\frac{(\mathbf{x}-0.5)^T \mathbf{M}^T \mathbf{M}(\mathbf{x}-0.5)}{2*0.002}} \tag{5}$$

Obviously, a RBNN with a single neuron centered on (0.5, 0.5) will not be able to correctly learn this function, because the activation function uses a euclidean distance which is symmetric. But our GA should be able to learn the matrix M used to generate the domain. In order to get a proof-of-concept using this simple problem, we trained our system using a single neuron centered on (0.5, 0.5) with a $\sigma^2 = 0.008$ (four times the σ^2 used to generate the domain). The GA was run using the parameters shown in Table 1.

In addition, 3 bits were used for the integer part, and 5 bits for the fractionary part. Only symmetric matrices were allowed. After 30 generations, the following matrix was obtained (see Eq. 6), which approached very well the function (it achieved a $5.867x10^{-5}$ error).

$$M = \begin{pmatrix} 0.46875 & 1.50000 \\ 1.50000 & 2.00000 \end{pmatrix} \tag{6}$$

Matrix 6 does not match matrix 4 (the one used to generate the domain), although it can be seen that their components approximately double the ones in the domain matrix. In any case, it is the activation functions that must be the same, in order for the 1 neuron RBNN to approximate perfectly the function. That is, the following equality has to be satisfied (see Eq. 7):

$$(1/\sigma_1^2)[M_1^T M_1] = (1/\sigma_2^2)[M_2^T M_2] \tag{7}$$

where σ_1^2 and M_1 refer to the parameters used to generate the domain, σ_2^2 is the variance of the neuron, and M_2 is the matrix obtained by the genetic algorithm. This equality is almost satisfied, as Eq. 8 shows.

$$\begin{pmatrix} 301.25 & 450 \\ 450 & 781.25 \end{pmatrix} \sim \begin{pmatrix} 283.2 & 421.8 \\ 421.8 & 757.8 \end{pmatrix} \tag{8}$$

It is interesting to remark that even though the σ^2 of the neuron (0.008) was not the same than the one used to generate the domain ($\sigma^2 = 0.002$), the GA managed to fit the domain function by appropriately scalating the components of the evolved matrix.

3.2 The Mackey-Glass Domain

The Mackey-Glass time series is widely regarded as a benchmark for comparing the generalization ability of RBNN[11],[12],[13]. The task for the RBNN is to predict the value of the time series at point $x[t+50]$ from the earlier points $(x[t], x[t-6], x[t-12], x[t-18])$. It is a chaotic time series created by Eq. 9:

$$\frac{dx(t)}{dt} = -bx(t) + a\frac{x(t-\tau)}{1+x(t-\tau)^{10}} \tag{9}$$

1474 patterns were generated for the Mackey-glass series, and values were normalized in (0,1). First, we ran some preliminary experiments in order to determine the number of neurons required. The minimum error was obtained with about 25 neurons. Also, these preliminary tests showed that 400 learning cycles and a 0.002 learning rate were reasonable values in this domain.

We tested two configurations of the system: allowing only diagonal matrices, and allowing general symmetrical matrices. The first case is equivalent to have a generalized euclidean distance, where every attribute is weighted by a factor. The most relevant attributes will be weighted by a larger number (see Eq. 10). The second case (the symmetrical matrix) can also consider correlations between attributes. Table 2 summarizes the parameters used. Two bits were used for the integer part, and three for the fractionary part.

$$d(A,B) = \sqrt{\left(\sum_{i=0}^{i=N} C_i * (A_i - B_i)^2\right)} \tag{10}$$

Table 3 displays the results comparing performance of a RBNN using an euclidean distance and evolved distances. 5-fold crossvalidation results are shown for both a diagonal matrix and a general symmetric matrix. Improvements of

Table 2. Parameters of the Genetic Algorithm for the Mackey-Glass problem

Generations	50
Tournament size	2
Population size	15
Elitism	1
Crossover probability	0.7
Mutation probability	0.01

Table 3. Comparison of results between euclidean and evolved distances (5-fold crossvalidation)

Distance used	Error	Improvement (%)
RBNN euclidean	0.013981	
RBNN GA diagonal	0.010337	26%
RBNN euclidean	0.015789	
RBNN GA symmetrical	0.014835	6%

26% and 6% (respectively) can be observed. In this domain, using a diagonal matrix is better than using a symmetrical matrix.

In order to get a better understanding of results in this domain, we observed the values of the components of the matrices evolved by the GA. As we used a 5-fold crossvalidation procedure, 5 matrices were evolved. It can be observed that none of the components outside the diagonal are significantly different than 0 (taking into account the 5 folds, the median for these components is very close to 0). This means that for this domain, a symmetrical matrix does not give any advantage and that only a diagonal matrix is required. It can also be observed in both the diagonal and symmetrical matrices, that attributes 1 and 4 get larger components in the matrix than attributes 2 and 4. This means that in this domain, variables 1 and 4 are more relevant for the regression problem.

4 Conclusions

One of the problems of RBNN is the symmetrical nature of their activation function: the activation of a neuron only depends on the euclidean distance from the input pattern to the neuron center, without taking into account the importance of different attributes. This problem can be solved by altering the metric used in the activation function. The learning method presented in this work uses a Mahalanobis-like distance function: instead of computing the Mahalanobis matrix from the variance-covariance matrix of all vectors in the data set, it is determined in such a way that minimizes the error of the network. This is achieved by a genetic algorithm whose individuals are generalized distance matrices and whose fitness function depends on the prediction accuracy attained by the network.

Our GA approach has been tested on two domains: a simple synthetic one and the Mackey-Glass time series. It has been shown that using both diagonal and symmetrical evolved matrices improves prediction accuracy over a purely euclidean distance.

In the future, we would like to test our approach using problems with a larger dimensionality, to study the effect of evolving large matrices. We would also like to explore other evolutionary approaches, like evolution strategies, which are perhaps more suited to evolve structures with real numbers. Finally, it would be interesting to understand the characteristics of domains where a symmetrical matrix is better than a purely diagonal one.

References

1. J.E. Moody and C. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, 1:281–294, 1989.
2. Joydeep Ghosh and Arindam Nag. *An Overview of Radial Basis Function Networks.* R.J. Howlett and L.C. Jain (Eds). Physica Verlag, 2000.
3. D.S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptative networks. *Complex Systems*, 2:321–355, 1988.
4. M. Powell. The theory of radial basis function approximation in 1990. *Advances in Numerical Analysis*, 3:105–210, 1992.
5. J. Park and I. W. Sandberg. Universal approximation and radial-basis-function networks. *Neural Computation*, 5:305–316, 1993.
6. C.G.Atkenson, A.W.Moore, and S.Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.
7. J. T. Tou and R. C. Gonzalez. *Pattern Recognition Principles.* Addison-Wesley, 1974.
8. S. Weisberg. *Aplied Linear Regression.* New York: John Wiley and Sons, 1985.
9. F. Babiloni, L. Bianchi, F. Semeraro, J. del R-Millan, J. Mourino, A. Cattini, S. Salinari, M.G. Marciani, and F. Cincotti. Mahalanobis distance-based classifiers are able to recognize eeg patterns by using few eeg electrodes. In *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, volume 1, pages 651–654, 2001.
10. John H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1975.
11. A. Leonardis and H. Bischof. An efficient mdl-based construction of rbf networks. *Neural Networks*, 11:963–973, 1998.
12. M. J. L. Orr. Introduction to radial basis neural networks. *Technical Report. Centre for Cognitive Science, University of Edinburgh*, 1996.
13. L. Yingwei, N. Sundararajan, and P. Saratchandran. A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation*, 9:461–478, 1997.