RBF Neural Network, Basis Functions and Genetic Algorithm

Eric P. Maillard
GESMA
BP 42
29240 Brest-Naval, France
E-mail: maillard@gesma.fr

Didier Gueriot
UHA, TROP Laboratory
4, rue des frères Lumière
68093 Mulhouse, France
E-mail:gueriot@gesma.fr

Abstract

Radial Basis Function (RBF) network is an efficient function approximator. Theoretical researches focus on the capabilities of the network to reach an optimal solution. Unfortunately, few results concerning the design and training of the network are available. When dealing with a specific application, the performances of the network dramatically depend on the number of neurons and on the distribution of the hidden neurons in the input space. Generally, the network resulting from learning applied to a predetermined architecture, is either insufficient or over-complicated. In this study, we focus on genetic learning for the RBF network applied to prediction of chaotic time series. The centers and widths of the hidden layer neurons basis function – defined as the barycenter and distance between two input patterns – are coded into a chromosome. It is shown that the basis functions which are also coded as a paramater of the neurons provide an additional degree of freedom resulting in a smaller optimal network. A direct inversion of matrix provides the weights between the hidden layer and the output layer and avoids the risk of getting stuck into a local minimum. The performances of a network with Gaussian basis functions is compared with those of a network with genetic determination of the basis functions on the Mackey-Glass delay differential equation.

1. Introduction

When dealing with approximation and prediction by neural networks, one of the most common model choice is the multilayer perceptron. It has been shown that this network is a special case of the generalized radial basis function [4]. This class of networks has been proven to be capable of universal approximation [4]. Futhermore, a solid background of mathematical studies is available on the problem of denseness, uniqueness of interpolation and convergence rate. Since prediction is a special case of approximation, i.e. a mapping from a continuous input space onto a continuous output space applied outside the learning regions, RBF networks can be trained on those classes of applications and reach optimum performances provided that the network are carefully designed. Overfitting of the data leads to poor generalization and long running time while a network with an underestimated number of neurons will not reach convergence. Most RBF learning paradigms train sequentially the hidden layer then the output layer. While learning of the output layer is rather straightforward, the hidden layer must be carefully designed to achieve good performances. We propose a global genetic designing and learning of the hidden layer. The neurons are coded as a weighted combination of two training patterns associated to an individual activation function whose width is also determined genetically. Individual activation function coded for each hidden neuron is compared to a common Gaussian activation of those neurons on a benchmark problem.

2. Radial Basis Function

The RBF network is a two-stage neural network implementing a mapping $\xi : R^m \longrightarrow R^p$. Its architecture is presented in figure 1. Each neuron i of the first layer consists of a center $\boldsymbol{\mu_i}$ and width $\boldsymbol{\sigma_i}$. The euclidean distance between the input signal $\mathbf{S}$ and the center of the neuron is first evaluated, then an activation function – also known as a basis function (BF) – ϕ such as the Gaussian function is applied. Several authors have proposed other functions among which six functions are used in our genetic algorithm:

0-7803-4122-8/97 $10.00©1997 IEEE

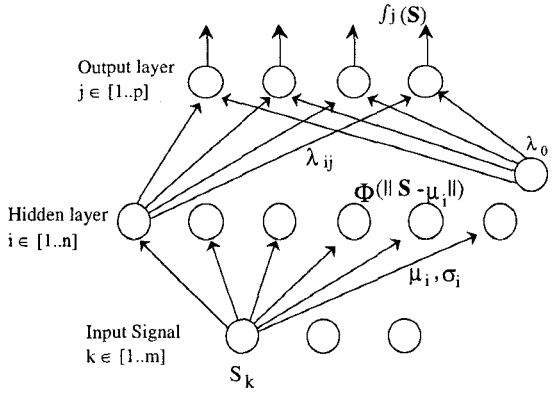


Figure 1: Architecture of an RBF network

- the Gaussian function:

$$\phi(\kappa) = e^{(\kappa^2/\beta^2)}$$

- the piecewise linear function:

$$\phi(\kappa) = \kappa$$

- the cubic approximation:

$$\phi(\kappa) = \kappa^3$$

- the thin plate spline:

$$\phi(\kappa) = \kappa^2 \log(\kappa)$$

- the multiquadratic function:

$$\phi(\kappa) = (\kappa^2 + \beta^2)^{1/2}$$

- the inverse multiquadratic function:

$$\phi(\kappa) = (\kappa^2 + \beta^2)^{-1/2}$$

The outputs of the hidden layer are combined linearly by the neurons of the second layer according to:

$$f_j(\mathbf{S}) = \lambda_0 + \sum_{i=1}^{n} \lambda_{ij} \Phi\left(||\mathbf{S} - \mu_i||\right)$$

The learning algorithm for the centers of the hidden layer neurons is usually an unsupervised clustering algorithm such as K-means [3] or Kohonen learning [5]. The simpliest approach is to set the center of the hidden neurons onto randomly chosen patterns of the training set. This coding scheme will be applied after improvment in the genetic version of the RBF. The width of these neurons may be set to a fixed value, it could be evaluated by simple methods such as setting it to the mean distance to the k-nearest neighbors of the neuron or it can be computed using a statistical approach through the estimation of local covariance matrices.

Learning of the weights from the hidden layer to the output layer can be achieved using a gradient descent method, however this approach inherits the risk of getting stuck into a local minimum thus failing to provide an optimum solution. In this paper, a direct method based on pseudoinverse matrix computation is used. The RBF weights are chosen to minimize a square error function between the actual output of the network and the desired output d over the X patterns $\mathbf{S}$. It has been proven that the weights λ_{ij} can be expressed as :

$$\lambda_{ij} = \sum_{l=1}^{n} (R_{il})^{-1} B_{lj}$$

where:

- $R_{il} = \sum_{k=1}^{X} \Phi_i(S_k).\Phi_l(S_k)$ (correlation matrix)
- $B_{lj} = \sum_{k=1}^{X} \Phi_l(S_k).d_j(S_k)$ (output matrix)

The main difficulty in operating this network concerns the optimization of the hidden layer. When a dynamic architecture is preferred to a predefined one, the optimization method often consists in a gradient descent algorithm [2] which can get trapped in local minima. Furthermore, the activation function has an influence on the final state of the optimization process.

In the next section, a genetic algorithm capable of determining the most appropriate choice for each neuron will be presented.

3. Genetic training of the Basis Function neurons

In response to the drawbacks described in the previous section, we propose a new approach combining network tailoring and training. In the range of evolutionnary programming, genetic algorithms provide a powerful engine to find solutions to non-linear problems. In our application, it means to succeed in computing the parameters of a self-organizing RBF in order to achieve correct classification of the training set pattern while minimizing the number of BF neurons. Using genetic terminology [7], in our data representation, each allele codes a parameter of a neuron, so a five gene sequence codes the characteristics of a BF neuron, and a whole chromosome, the set of parameters of the hidden layer (fig. 2). All the chromosomes are brought

together to form a population which represents the current potential solutions for our classification problem. The classical coding approach consists in defining each

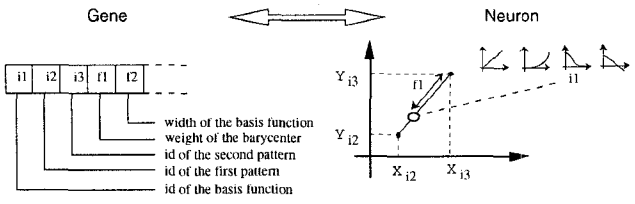


Figure 2: Decoding the chromosome constituted by a list of 5-gene sequences representing neurons

weight of the hidden neurons by an allele. A major drawback of this method is the increase of the chromosome length proportionnally to the dimension number of the input space. A major improvment was proposed in [1], the neurons are centered on patterns of the training set. The search space of the genetic algorithm is dramatically reduced but so are the freedom degrees of the possible solutions. We propose a modification of this coding scheme to span thoroughly the input space. A neuron is now defined as the weighted barycenter of two input patterns. Furthermore to improve the adaptation capacity of the network, the activation function of the hidden neurons (i.e. the BFs) is chosen dynamically among those presented in the previous section. Since some of them contain an additional vector parameter (the so-called width), this vector is also added to the gene translation of the neuron.

To induce this population (100 individuals) to evolve towards a set of suitable solutions, basic genetic operators such as mutation (rate: 1%) are used, but to improve computational performance, optimized mechanisms are tailored. The principle of n-dynamic crossovers (rate: 80%) to speed up the combination of high performance schemata is applied. The phenotypic crossover is an inter-sequence mechanism which results in preserving the neuron identity thus allowing the network size to evolve. The genotypic crossover is a classical one adapted to variable length chromosomes. Furthermore, fresh blood strategies to possibly prevent the population from wasting time in gathering near local optima as well as elitist selection are used (convergence towards the best solution garanteed as indicated in [8]).

The fitness function is defined as the classification performances of the network coded by a chromosome. The fitness evaluation process consists in computing the weights between the hidden layer and the output layer to test this network on a training set. Those weights are estimated through the matrix inversion mechanism described in the previous section. Matrix inversion prevents the risk of local minimum inherent to gradient descent methods. Such a fitness function allows the evaluation of a network defined by a subset of sequences found in a chromosome. It is thus similar to partial function estimation used in messy genetic algorithms.

4. Experimental results

We had presented in a previous paper the effectivness of the genetic approach applied to RBF learning [6]. In particular, we emphasized the genetic determination of the BFs. On classification problems the variety of BFs provides a additionnal degree of freedom for the network to achieve efficient classification while remaining small. To succeed, the learning algorithm must comply with opposite constraints. On one hand, a better classification rate on the training set can almost always be obtain using more neurons. On the other hand, the generalization capacity tends to decline when too many neurons are involved. On classification problems, a compromise was reached by discretizing the energy function : $E = \sum_n \delta(P_n)$ with $\delta(P_n) = 1$ if the winning output neuron (i.e. neuron with maximum activation) represents the class of the pattern $P(n)$ and $\delta(P_n) = 0$ otherwise. This energy function is not tailored for approximation or prediction since the notion of "correct classification" bears no meaning here. An energy function which translates the two constraints was chosen on the model of the Akaike Information Criterion [1]:

$$E(n_n, \phi, n) = N \times \log(err) + 4 \times n_n$$

where:

$$err = \frac{1}{X} \sum_{k=1}^{X} \left(d_k - \sum_{i=1}^{n_n} \lambda_i \phi_i(S_k, \mu_i, \sigma_i) \right)^T \times \left(d_k - \sum_{i=1}^{n_n} \lambda_i \phi_i(S_k, \mu_i, \sigma_i) \right)$$

This energy function balances the performances of the network with the number of neurons.

To test the ability of the proposed algorithm to determine a near-optimum architecture for the prediction of function, our genetic RBF was tested on a benchmark known as the Mackey-Glass delay differential equation. The purpose is to predict futur values of a time series given the nearest past values. This time

	$\tau = 17$		$\tau = 30$	
	GRBF	MRBF	GRBF	MRBF
Best	200	116	200	123.5
Avg.	199.5	87.5	199	90.2
Worst	200	78	187	78
Smallest	187.5	72	183.5	68
Largest	200	116	200	126

Table 1: Number of BF neurons for the Gaussian RBF (GRFB) and the multi-function RBF (MRBF) on the Mackey-Glass problems

series is described by:

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t)$$

In this study two values of τ were used: $\tau = 17$ and $\tau = 30$. The resulting time series are known to be chaotic and thus difficult to approximate. The results of the Gaussian RBF are compared to those of the multi-function RBF.

The first criterion is the final size of the networks, i.e. the number of BF neurons. Table 1 summarizes the results. As expected, the Gaussian network needs more neurons than the multi-function network does. The use of several BF provides an additional degree of freedom to the training algorithm. The smaller size of the final networks makes up for this increase of the solution space dimension. It should be pointed out that during learning the upper weights are determined through matrix inversion. Dealing with larger network is more penalizing than dealing with an increased solution space. The multi-function RBF population reached the hundredth generation ten times faster than the Gaussian RBF networks. This results mainly from the rapidly growing size of the "mean" network among the Gaussian RBFs. Thus, the genetic learning algorithm takes full advantage of the multiple BFs. Table 2 summarizes the CPU time for various tests. The second criterion

	$\tau = 17$		$\tau = 30$	
CPU Time	GRBF	MRBF	GRBF	MRBF
Seconds	58866	5682	56974	4874
Norm.	12.07	1.16	11.68	1.00

Table 2: CPU time for the Gaussian RBF (GRFB) and the multi-function RBF (MRBF) on the Mackey-Glass problems

is the prediction accuracy. The results on the Mackey-Glass problems are almost similar for both $\tau = 17$ and $\tau = 30$. The results presented here concern $\tau = 17$ due to the available space. The prediction results and errors are presented in fig. 3 and fig. 4. It is clear from these results that both networks achieved an effective approximation of the underlying dynamic of the chaotic time series since the short term predictions on a test set are correct. The only visible difference comes from the points of the time series with high gradient. If not important for short time prediction, this kind of error might prove rather penalizing on long term predictions. A finer analysis of the respective errors proves that the multi-function RBF outperforms the Gaussian one in terms of mean value of the error (-3.79×10^{-4} for the Gaussian RBF, -6.45×10^{-5} for the multi-function RBF) and of standard deviation of the error (0.0051 for the Gaussian RBF, 0.0019 for the multi-function RBF). The evolution of the network population is presented in fig. 5 and fig. 6. The best individual of the multi-function RBF population achieves better performances on the training set with fewer neurons than the best individual of the Gaussian RBF population. On the presented run, the final sizes of the best individual from the two populations are dramatically different. This smaller size results in better generalization capabilities proven by the performances on a test set. This result emphasizes the importance of achieving learning with smaller networks which are not subject to overfitting. The performances on the training sets are improved while preserving the generalization capacity of the network.

References

[1] S. A. Billings and G. L. Zheng. Radial basis function network configuration using genetic algorithms. *Neural Networks*, 8(6):877–890, 1995.

[2] A. Cichocki and R. Unbehauen. *Neural networks for optimization and signal processing.* John Wiley & Sons, 1993.

[3] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* John Wiley & Sons, 1973.

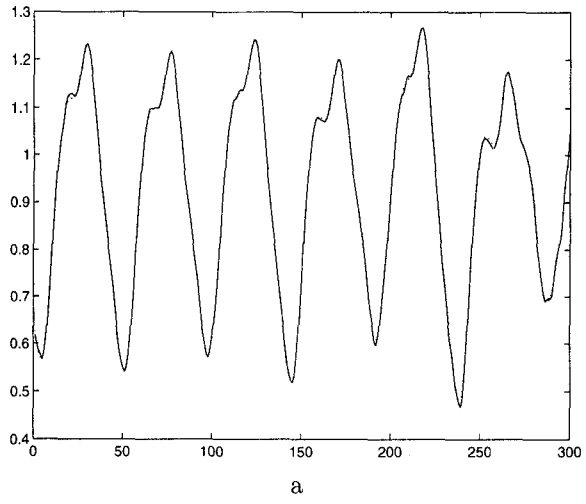
[4] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269, 1995.

[5] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78:1464–1480, 1990.

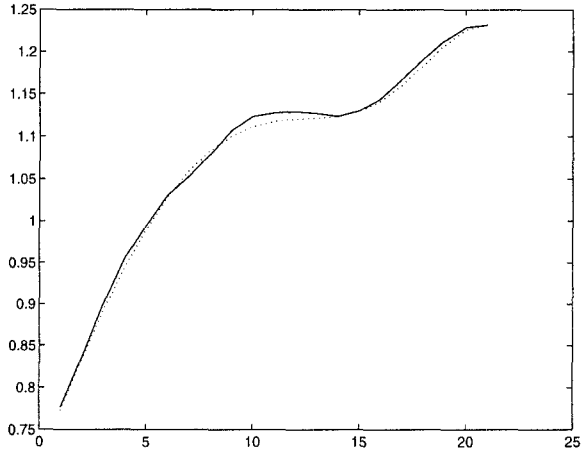
[6] E. P. Maillard and D. Guériot. Designing an optimal RBF classifier using genetic algorithms. In *World Congress on Neural Networks*, volume x, pages xxx–xxx, San Diego, September 1996.

[7] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springler-Verlag, second extended edition, 1994.

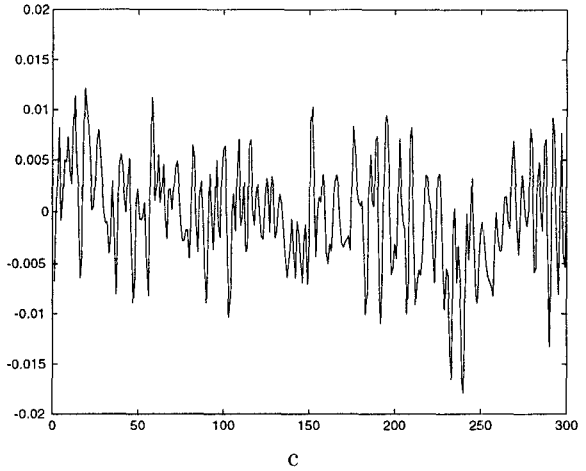
[8] G. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101, January 1994.



a

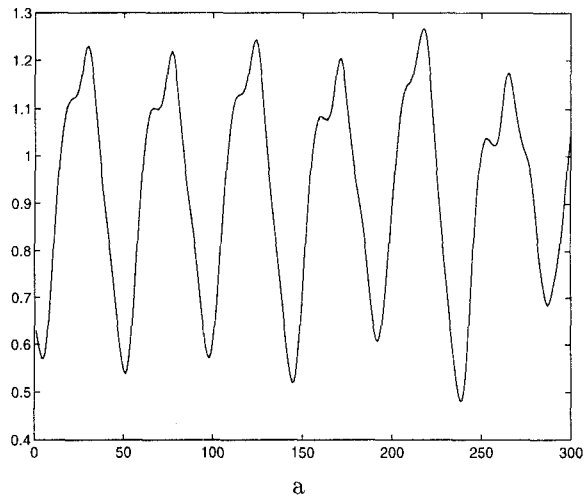


b

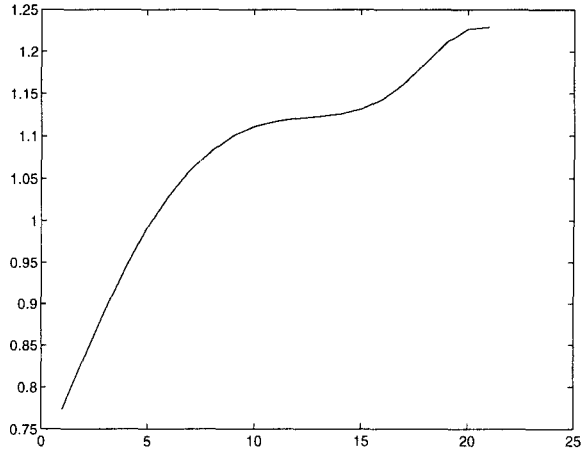


c

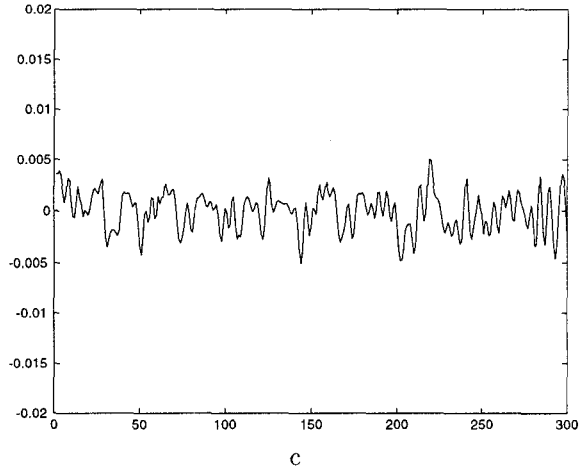
Figure 3: Prediction of the Mackey-Glass function by genetic gaussian RBFs (a: original and approximated Mackey-Glass test series, b: zoom on the previous figures, c: error of the short term prediction)



a

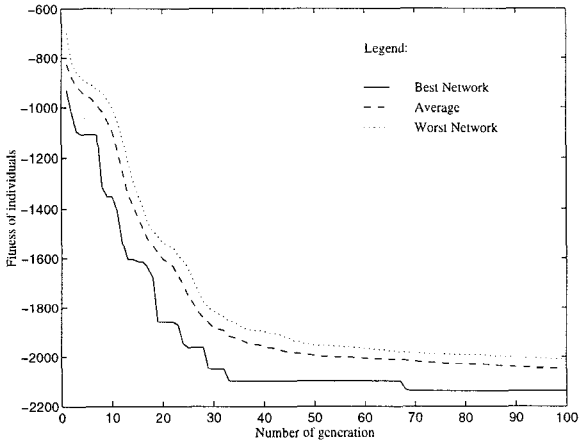


b



c

Figure 4: Prediction of the Mackey-Glass function by genetic multi-function RBFs (a: original and approximated Mackey-Glass test series, b: zoom on the previous figures, c: error of the short term prediction)



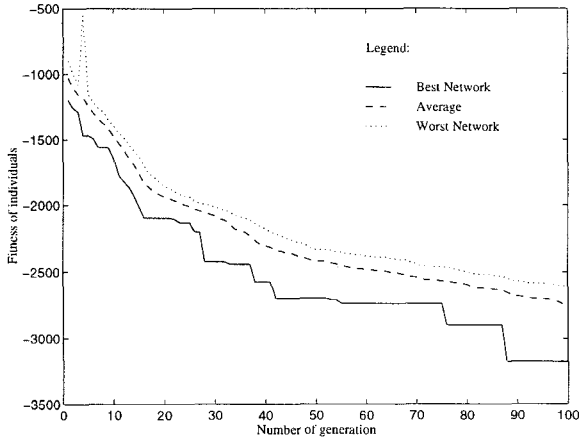
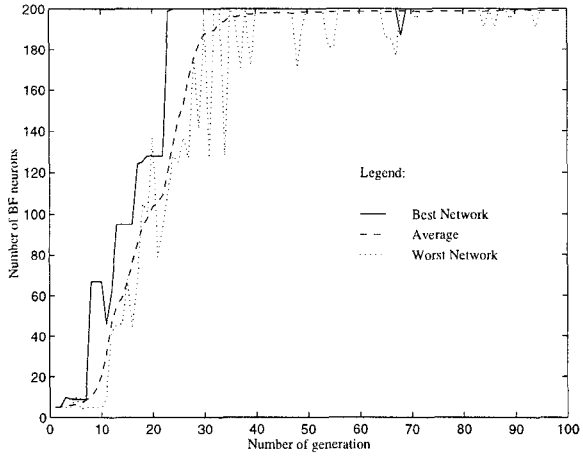


Figure 5: Performances of genetic RBFs on the Mackey-Glass problem (upper part: fitness of the Gaussian RBF, lower part: fitness of the multi-function RBF)



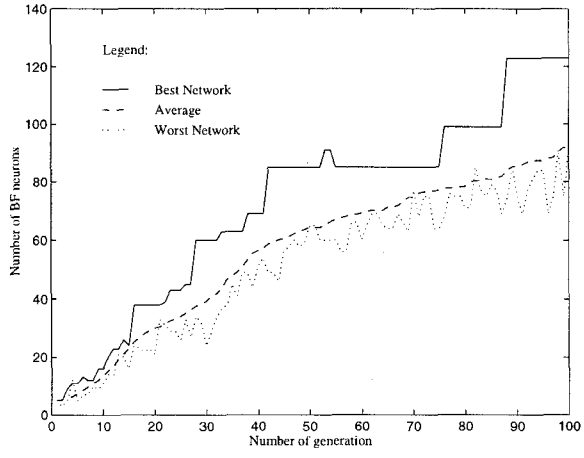


Figure 6: Sizes of genetic RBFs on the Mackey-Glass problem (upper part: number of neurons for the Gaussian RBF, lower part: number of neurons for the multi-function RBF)