RBF Neural Networks, Multiobjective Optimization and Time Series Forecasting

J. González, I. Rojas, H. Pomares and J. Ortega

Department of Computer Architecture and Computer Technology
University of Granada
Campus de Fuentenueva
E. 18071 Granada (Spain)

Abstract. This paper presents the problem of optimizing a radial basis function neural network from training examples as a multiobjective problem and proposes an evolutionary algorithm to solve it properly. This algorithm incorporates some heuristics in the mutation operators to better guide the search towards good solutions. An application to the Mackey-Glass chaotic time series is presented. The prediction accuracy of the proposed method is compared with that of other approaches in terms of the root mean squared error.

1 Introduction

The automatic optimization of a Radial Basis Function Neural Network (RBFNN) from training data [10, 11] is a problem in which two clearly competing objectives must be satisfied. The model's prediction error must be minimized in order to achieve a well fitted model, while the number of Radial Basis Functions (RBFs) should be as low as possible to obtain a reliable interpolator. The problem here is how to minimize both objectives simultaneously. Improving one of them will probably cause a worse behavior in the remaining one. This kind of problems is known as Multi-Objective Problems (MOPs), and their solutions are usually sub-optimal for each objective in particular, but "acceptable" taking all the objectives into account, where "acceptable" is totally subjective and problem dependent.

The algorithms proposed in the literature to construct RBFNNs from examples try to find a unique model with a compromise between its complexity and its prediction error. This is not an adequate approach. In MOPs there are usually more than one alternative optimal solutions (having different compromises between their multiple objectives) that should be considered equivalent. Thus, conventional optimization techniques have great difficulty to be adapted to solve MOPs because they weren't designed to deal with more than one solution simultaneously. Nevertheless, Evolutionary Algorithms (EAs) maintain a population of potential solutions for the problem, thus making easier their adaption to solve MOPs [2]. In particular, the fitness of the individuals must be adapted to comprise all the objectives to be satisfied and some new mutation operators must be designed to alter the structure of RBFNNs.

2 Multiobjective evolution

The great difference between single objective and multiple objective problems is that the set of solutions is not completely ordered for MOPs. For two objective vectors, the following relations can be defined:

$$\begin{array}{ll} f(\iota_1) = f(\iota_2) \Longleftrightarrow f_i(\iota_1) = f_i(\iota_2) & \forall i \in 1,2,...,n_{obj} \\ f(\iota_1) \le f(\iota_2) \Longleftrightarrow f_i(\iota_1) \le f_i(\iota_2) & \forall i \in 1,2,...,n_{obj} \\ f(\iota_1) < f(\iota_2) \Longleftrightarrow (f(\iota_1) \le f(\iota_2)) \wedge (f(\iota_1) \ne f(\iota_2)) & \end{array} \quad (1)$$

where n_{obj} is the number of competing objectives. Taking into account the above relations the Pareto-dominance criterion can be defined as:

$$\begin{array}{lcl} \iota_1 \prec \iota_2 \quad (\iota_1 \text{ dominates } \iota_2) & \Longleftrightarrow & f(\iota_1) < f(\iota_2) \\ \iota_1 \preceq \iota_2 \; (\iota_1 \text{ weakly dominates } \iota_2) & \Longleftrightarrow & f(\iota_1) \le f(\iota_2) \\ \iota_1 \sim \iota_2 \quad (\iota_1 \text{ is indifferent to } \iota_2) & \Longleftrightarrow & f(\iota_1) \not\le f(\iota_2) \wedge f(\iota_2) \not\le f(\iota_1) \end{array} \quad (2)$$

A Pareto-optimum solution is defined as an individual that cannot be dominated by any one in the solution set:

$$\nexists \iota_i \in D : \iota_i \prec \iota_j \quad (3)$$

A good multiobjective algorithm should find as many Pareto-optimum solutions as possible, to provide the final user the possibility of choosing the right solution following his own criteria.

There are several ways of adapting EAs to solve MOPs. In this paper is used the approach proposed in [2], which estimates an scalar dummy fitness for each individual in the current population based in its rank. The rank of each individual is defined as:

$$\text{rank}(\iota_j^t) = 1 + dom_j^t \quad (4)$$

where dom_j^t represents the number of individuals dominating ι_j^t in the current population. Once that each individual is assigned a rank, a dummy fitness is obtained for all the individuals by interpolating between the maximum and minimum rank in the population. This simple modification allows a generic EA [8] to solve a MOP transparently, that is, without changing any other of its components.

Multi-Objective Evolutionary Algorithms (MOEAs) are a robust optimization technique that have been successfully applied to several optimization problems. Its strength is based on their simplicity and easy implementation. However, MOEAs are a generic optimization technique whose results can be improved if some expert knowledge about the problem to be solved is incorporated in them. These changes produce algorithms that hybridizes the robustness and strength of EAs and the expertness of some heuristics for the problem. This adapted

MOEAs obtain better results than the generic ones because they can guide the search towards solutions that the expert knowledge expects to be superior. The easiest way of incorporating this expert knowledge of the problem in an EA is to construct some mutation operators specific for the problem to be solved. These operators differ from the original ones in the sense that they do not apply blind changes to the individuals they affect. They try to improve the mutated individual by analyzing and altering it using some heuristics. A complete description of the MOEA used in this paper can be found in [3], here we will focus only en the expert mutation operators designed for this problem and on how they can guide the search towards better solutions.

3 Proposed mutation operators

The purpose of the MOEA described in this paper is to search for RBFNNs with different compromises of complexity and prediction error, so that the final user can choose among diverse possibilities the model that better solve his requirements. Thus, some mutation operators have to be designed to alter the structure of the net, adding or deleting RBFs until good nets are obtained. These mutation operators incorporate some heuristics to alter the RBFNNs "cleverly", detecting the RBFs that less contribute to the net output and eliminating them, or, on the other hand, estimating the input regions that increase the prediction error and adding there more RBFs to obtain a better model.

3.1 RBF Splitting

This mutation operator is based in a splitting mechanism for RBFs proposed in [6], but with some modifications because the original mechanism was oriented to classification problems and the proposed algorithm will be used to forecasting time series. The basic functioning of this operator is as follows:

- detect that RBF producing the highest error increment in the prediction error, and
- split it into two RBFs to obtain a net that covers better the input space.

The prediction error is caused by some RBFs that become highly activated by some input vectors and produce an output value different of the expected one. The RBFs that are not activated by an input vector don't contribute to the prediction error for that input vector. Thus, the prediction error can be proportionally divided between all the RBFs that are activated by an input example as:

$$e^j = \sum_{k=1}^{n} \frac{\phi_j(\boldsymbol{x}^k)}{\sum\limits_{i=1}^{m} \phi_i(\boldsymbol{x}^k)} \left|\mathcal{F}(\boldsymbol{x}^k) - y^k\right|, \quad j = 1, ..., m \tag{5}$$

where n is the number of input examples, m is the number of RBFs in the net, ϕ_j is the j-th RBF, $\mathcal{F}(\boldsymbol{x}^k)$ is the prediction of the net for the input vector $\boldsymbol{x}^k$, and y^k is its expected prediction. A high value of e^j shows that the j-th RBF is not sufficient to learn the input examples that activate it, thus the bigger e^j, the higher probability of division for ϕ_j. So, each RBF ϕ_h is assigned a division probability inversely proportional to e^j and a basis function is randomly selected according to these probabilities.

Once one RBF ϕ_j has been selected, the 2-means algorithm is run with the input examples that are closer to the center of ϕ_j than to any other RBF center, obtaining two new positions for two new RBFs, ϕ_{j_1} and ϕ_{j_2}, that will substitute ϕ_j in the affected net. The radii for ϕ_{j_1} and ϕ_{j_2} are calculated using the CIV heuristic [6] and the optimum weights for all the weights of the new net are obtained using the Cholesky method [9].

3.2 OLS based pruning

Orthogonal Least Squares (OLS) [1] is one of the most widely used method to prune the less relevant RBFs of a net. Basically, this method calculates a vector of error reduction ratios $\boldsymbol{err}$ where each one of its components $[err]_j$ gives an idea of the output variance explained by its associated RBF ϕ_j. The lower $[err]_j$, the less relevant is ϕ_j in the RBFNN output. OLS allows this mutation operator to assign a prune probability to each RBF based in its associated error reduction ratio. Less important RBFs will have more likelihood to be deleted, while more relevant RBFs will be more deletion-protected.

Once that an RBF is selected and deleted, the weights of the remaining basis functions are optimally recalculated using the Cholesky method.

3.3 SVD based pruning

Another good heuristic to prune RBFs is the Singular Value Decomposition (SVD) of the activation matrix P of the RBFNN [5]. This orthogonal transformation provides a vector $\boldsymbol{\sigma}$ of Singular Values (SVs), each one of them concerning one of the RBFs in the net. These SVs reveal the degree of linear independence of the columns of P. Columns with a low (nearly null) singular value are almost linearly dependent and may cause a singular activation matrix. Thus, if these columns are identified and deleted, the system will become simpler and more robust.

With the aforementioned idea in mind, this mutation operator assigns a pruning probability to each RBF inversely proportional to its SV and deletes a basis function randomly selected, having less relevant RBFs more likelihood to be pruned than those more important ones. After the deletion, the weights of the remaining RBFs are optimally obtained using the Cholesky method.

4 Results

The algorithm presented above have been tested with the time series generated by the Mackey-Glass time-delay differential equation [7]:

$$\frac{ds(t)}{dt} = \alpha \cdot \frac{s(t-\tau)}{1+s^{10}(t-\tau)} - \beta s(t) \tag{6}$$

Following previous studies [12], the parameters have been fixed to $\alpha = 0.2$, $\beta = 0.1$, obtaining a chaotic time series without a clearly defined period; it will not converge or diverge, and it is very sensitive to initial conditions.

As in [4], to obtain the time series value at integer points, we applied the fourth-order Runge-Kutta method to find the numerical solution for the above equation. The values $s(0) = 1.2$, $\tau = 17$, and $s(t) = 0$ for $t < 0$ are assumed. This data set can be found in the file `mgdata.dat` belonging to the Fuzzy Logic Toolbox of Matlab 5.

Following the conventional settings for predicting these time series, we will predict the value $s(t+6)$ from the current value $s(t)$ and the past values $s(t-6)$, $s(t-12)$, and $s(t-18)$, thus, the training vectors for the model will have the following format:

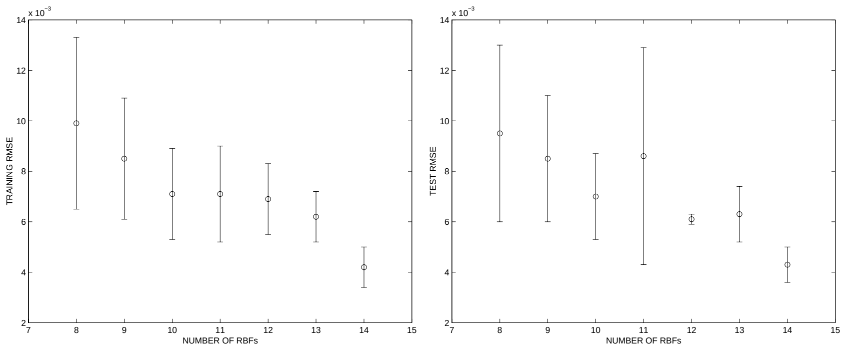
$$[s(t-18), s(t-12), s(t-6), s(t), s(t+6)] \tag{7}$$

The first 500 input vectors have been used to train the model and the next 500 vectors have been used to test the RBFNNs obtained. The proposed algorithm has been run 5 times with a population of 25 individual during 1000 generations, and the best solutions found have been applied the Levenberg-Marquard minimization algorithm to fine-tune their parameters. Table 1 shows minimum, mean and standard deviation of the best RMSEs obtained in five different runs of the algorithm using a population of 25 individuals.

Table 2 shows the best solutions found by other approaches. The comparison of these results with those obtained by the proposed algorithm (see table 1)

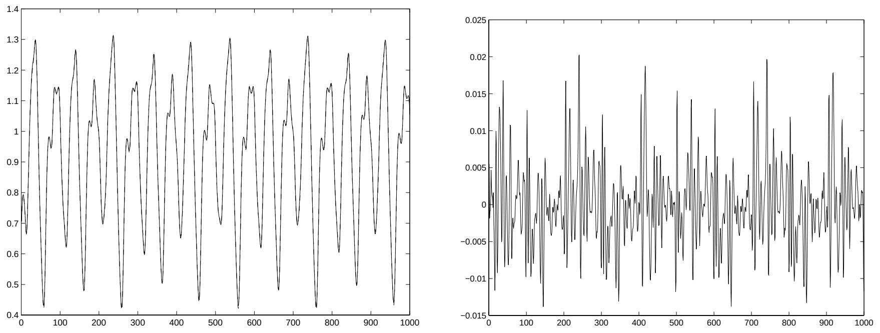
Num.	Training RMSE			Test RMSE		
RBFs	Min.	Mean	St. Dev.	Min.	Mean	St. Dev.
8	0.0065	0.0099	0.0034	0.0057	0.0095	0.0035
9	0.0061	0.0085	0.0024	0.0061	0.0085	0.0025
10	0.0055	0.0071	0.0018	0.0054	0.0070	0.0017
11	0.0055	0.0070	0.0019	0.0056	0.0086	0.0043
12	0.0061	0.0069	0.0014	0.0060	0.0061	0.0002
13	0.0057	0.0062	0.0010	0.0056	0.0063	0.0011
14	0.0037	0.0042	0.0008	0.0039	0.0043	0.0007

Table 1. Minimum, mean and standard deviation of the training and test errors for all the different RBFNN structures.



(a) (b)

Fig. 1. Mean and standard deviation of the RMSE in five runs of the algorithm.



(a) (b)

Fig. 2. (a) Original time series (solid line) and predicted values (dashed line) for training and test data. (b) Training and test residuals.

reveals that the use of expert mutation operators enhances significantly the search results. All minimum performance indices are superior to those obtained by other approaches, while the standard deviation from the mean RMSE are small enough to conclude the robustness of our approach. As an example, figure 2 shows the original series (solid line), and the predicted values (dashed line) for the training and test data (a), and the training and error residuals (b) for the best RBFNN of 10 basis functions found.

5 Conclusions

Identifying the adequate structure for a RBFNN created from training data is not an easy problem. There exist several optimal solutions depending on the chosen compromise between the prediction error and the complexity of the model.

Approach		Test RMSE
Linear Predictive Method		0.55
Auto Regressive Model		0.19
L-X. Wang	T-Norm: Prod.	0.0907
	T-Norm: Min.	0.0904
Cascade Correlation ANN		0.06
6^{th}-order Polynomial		0.04
D. Kim & C. Kim	5 MFs/var.	0.0492
(Genetic Algorithm	7 MFs/var.	0.0423
+ Fuzzy System)	9 MFs/var.	0.0379
Retropropagation ANN		0.02
ANFIS (ANN + Fuzzy Logic)		0.007

Table 2. RMSE of other approaches

MOEAs are able to find a sufficient number of Pareto-optimal solutions, providing the final user a wide variety of possibilities, in order he can choose the solution that better satisfies his requirements.

MOEAs can also be specialized by means of expert mutation operators that improve the search results. These expert mutation operators can incorporate some heuristics for the problem, such as OLS and SVD for the pruning mechanism, or an error sharing scheme to identify RBFs producing bigger errors, to perform "clever changes" when altering an individual. Particularly, the proposed algorithm has found very good RBFNNs for the prediction of the Mackey-Glass time series.

Acknowledgement

This work has been partially supported by the CICYT Spanish projects TAP97-1166 and TIC2000-1348.

References

1. S. Chen, C. F. N. Cowan, and P. M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans. Neural Networks*, 2:302–309, 1991.
2. C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423. Morgan Kaufmann, 1993.
3. J. González, I. Rojas, H. Pomares, M. Salmerón, and A. Prieto. Evolutive identification of fuzzy systems for time series prediction. In A. Ollero, S. Sánchez, B. Arrue, and I. Baturone, editors, *Actas del X Congreso Español sobre Tecnologías y Lógica Difusa, ESTYLF 2000*, pages 403–409, Sevilla, Spain, Sept. 2000.

4. J. S. R. Jang. Anfis: Adaptive network-based fuzzy inference system. *IEEE Trans. Syst., Man, Cybern.*, 23:665–685, May 1993.
5. P. P. Kanjilal and D. N. Banerjee. On the application of orthogonal transformation for the design and analysis of feed-forward networks. *IEEE Trans. Neural Networks*, 6(5):1061–1070, 1995.
6. N. B. Karayiannis and G. W. Mi. Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques. *IEEE Trans. Neural Networks*, 8(6):1492–1506, Nov. 1997.
7. M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289, 1977.
8. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 3rd edition, 1996.
9. H. Pomares, I. Rojas, J. Ortega, J. González, and A. Prieto. A systematic approach to a self-generating fuzzy rule-table for function approximation. *IEEE Trans. Syst., Man, Cyber. Part B*, 30(3):431–447, June 2000.
10. I. Rojas, J. González, A. Cañas, A. F. Díaz, F. J. Rojas, and M. Rodriguez. Short-term prediction of chaotic time series by using rbf network with regression weights. *Int. Journal of Neural Systems*, 10(5):353–364, 2000.
11. I. Rojas, H. Pomares, J. González, J. L. Bernier, E. Ros, F. J. Pelayo, and A. Prieto. Analysis of the functional block involved in the design of radial basis function networks. *Neural Processing Letters*, 12(1):1–17, Aug. 2000.
12. B. A. Whitehead and T. D. Choate. Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction. *IEEE Trans. Neural Networks*, 7(4):869–880, July 1996.