Parallel Multi-objective Memetic RBFNNs Design and Feature Selection for Function Approximation Problems

Alberto Guillén[1], Héctor Pomares[2], Jesús González[2], Ignacio Rojas[2], L.J. Herrera[2], and A. Prieto[2]

1. Department of Informatics, University of Jaen, Spain.
2. Department of Computer Technology and Architecture, University of Granada, Spain.

Abstract. The design of Radial Basis Function Neural Networks (RBFNNs) still remains as a difficult task when they are applied to classification or to regression problems. The difficulty arises when the parameters that define an RBFNN have to be set, these are: the number of RBFs, the position of their centers and the length of their radii. Another issue that has to be faced when applying these models to real world applications is to select the variables that the RBFNN will use as inputs. The literature presents several methodologies to perform these two tasks separately, however, due to the intrinsic parallelism of the genetic algorithms, a parallel implementation will allow the algorithm proposed in this paper to evolve solutions for both problems at the same time. The parallelization of the algorithm not only consists in the evolution of the two problems but in the specialization of the crossover and mutation operators in order to evolve the different elements to be optimized when designing RBFNNs. The subjacent Genetic Algorithm is the Non-Sorting Dominated Genetic Algorithm II (NSGA-II) that helps to keep a balance between the size of the network and its approximation accuracy in order to avoid overtraining networks. Another of the novelties of the proposed algorithm is the incorporation of local search algorithms in three stages of the algorithm: initialization of the population, evolution of the individuals, and final optimization of the Pareto front. The initialization of the individuals is performed hybridizing clustering techniques with the Mutual Information theory (MI) to select the input variables. As the experiment will show, the synergy of the different paradigms and techniques combined by the presented algorithm allow to obtain very accurate models using the most significant input variables.

1 Introduction

The problem of function approximation, also know as non-linear regression, has been successfully tackled using Radial Basis Function Neural Networks (RBFNNs) [14]. Formally, the functional approximation problem can be formulated as, given a set of observations $\{(\boldsymbol{x}_k; y_k); k = 1, ..., n\}$ with $y_k = F(\boldsymbol{x}_k) \in \mathbb{R}$ and $\boldsymbol{x}_k \in \mathbb{R}^d$, it is desired to obtain a function $\mathcal{F}$ so $y_k \simeq \mathcal{F}(\boldsymbol{x}_k)$. Once this function is learned, it will be possible to generate new outputs from input data that were not specified in the original data set.

The reason to use RBFNNs [2], is because they have the capability of approximating any continuous function defined on a compact set. An RBFNN is a two-layer, fully connected network in which each neuron implements a gaussian function. These kind of functions are very appropriate for function approximation because they are continuous, differentiable, provide a softer output, and improve the interpolation capabilities.

The real problem that arises when it is desired to approximate a function using an RBFNN is how to design the RBFNN. The parameters to be set to create a RBFNN are: the number and the position of the centers of the RBFs and their radii. The weights of the output layer can be calculated

optimally solving a linear equation system. The solution space for the problem of initializing these variables is infinite since they are real values, on top of this, the risk of stalling in local minima is quite high.

The literature presents a wide variety of algorithms which are based on Genetic Algorithms (GAs) [6] and local search or descent gradient methods [12]. These techniques have shown a good performance, however in [15], Memetic Algorithms (MAs) were presented as Evolutionary algorithms that hybridize the global optimization characteristics of GAs with local search techniques that allowed the GAs to perform a more deep exploitation of the solutions.

The two objectives of designing an RBFNN with the maximum generalization capabilities and the minimum approximation error with the training data is translated into defining the topology (number of RBFs) of the network. The more neurons are in the network, the smaller the approximation error will be although the more the generalization capabilities will be decreased. This fact defines our task as a multiobjective problem which is can be solved applying Multi-Objective GAs (MOGAs).

Another related issue is which variables the RBFNNs should consider. In real world applications there are many variables that can be useless. Having a number of irrelevant or redundant input variables can lead to overfitting, higher computational cost and to a poor generalization of the model. The algorithm presented in this paper evolves the input variables for the RBFNNs after a initialization based in the Mutual Information theory.

The algorithm presented on this paper will combine the techniques of local search, multiobjective optimization and mutual information system combined with pure variable selection through genetic algorithms to design a RBFNNs that approximates a function with accuracy and generalization capabilities.

2 A Parallel Evolutionary Feature Selector and RBFNN Designer for Function Approximation: pEFSFA

This section briefly, due to space limitations, describes the proposed algorithm that optimizes the inputs, the structure and the parameters of RBFNNs for function approximation problems. This algorithm implements and combines several paradigms such us MAs, MOGAs and PGAs. The synergy of these techniques results in a robust algorithm which is able to design proper RBFNNs.

2.1 Representing RBFNNs in the Individuals

As it was shown in the Introduction, to design an RBFNN it is needed to specify: 1) the variables that the RBFNN will receive as inputs 2) the number of RBFs 3) the position of the centers of the RBFs 4) the length of the radii and 5) the weights for the output layer.

An individual will encode, as a binary vector, the input variables that will take of each input vector, then, the position of the centers in that input variable space and the radii are stored as real numbers. The binary encoding was chosen because of its simplicity and its discrete solution space.

2.2 Initial Population

The infinite solution space for the problem of setting the centers and the radii and the large (although it depends on the problem) solution space for the problem of selecting the input variables,

makes very important the initialization of the population in areas of the solution space where the solutions can be considered adequate. Therefore, this subsection deals with the method used to initialize the individuals.

Mutual Information Systems Since the individuals represent the set of input variables using a binary vector, the number of posible solutions is 2^d. At first, this might not seem to high but, to evaluate the goodness of each one of those possible solutions, infinite RBFNNs could be designed. Therefore, a preprocessing of these input variables that indicates which variables are the most significant for the output becomes necessary. For this purpose, the Mutual Information (MI) (also called cross-entropy) concept will be employed to obtain a ranking of the significance of each variable [1].

Once the MI $MI_i, i = 1...d$ has been estimated for each variable, the MI is normalized in [0,1] and the variables of an individual are calculated as follows: $variables_{ind} = round(\boldsymbol{MI}_{norm}.*\boldsymbol{rand})$ where *round* is a function that rounds a real value to the closest integer and $\boldsymbol{rand}$ is a vector with random values in [0,1]. This initialization method allows the values with higher ranking to have more chances to be selected as inputs for the RBFNNs.

Clustering The initial population is generated using the clustering algorithms presented in [10–12] in order to supply good individuals that will make easier and faster to find good solutions. It also includes individuals generated randomly in order to keep diversity in the population.

All the clustering algorithms used base their operation mode in minimizing a distortion function with a local search by an alternating optimization procedure. After the initialization of half of the individuals of the population with the clustering techniques, a few iterations of a local search algorithm are applied to each one and the results are appended to the population. This procedure increments the diversity and, as experimentally has been proven, improves the quality of the results.

The size of the RBFNNs coded by each individual should be small in the initialization step for two reasons: 1) to make the initialization as fast as possible and 2) to allow the genetic algorithm to determine the sizes of the RBFNNs from an incremental point of view, saving the computational effort that would be required to deal with big networks from the first generations.

2.3 Crossover Operators

Four crossover operators have been designed specifically to make the offspring similar to their parents but modifying at least one of the four elements that the algorithm is evolving, these are: the topology of the network, the position of the centers, the length of the radii, and the input variables.

Crossover operator 1: Neurons exchange. The operator will exchange only one neuron, selected randomly, between the two individuals. This crossover operator exploits the genetic material of each individual in a simple and efficient way without modifying the structure of the network. Before adding the exchanged neuron, the operator makes sure that the neuron that will be introduced is not already in the network in order to avoid the competing convention problem [9].

Crossover operator 2: Addition of the neuron with the smallest local error. This operator consists in the addition into one parent of the neuron with the smallest local error belonging to the other parent. As before, if the neuron is already in the network, the next with the smallest local error is chosen and son on. The local error is defined as the sum of the errors between the real

output and the output generated by the RBFNN considering the input vectors that activate that neuron. Since the algorithm follows an incremental approach, there might be the possibility of one parent owning all the RBFs of the other parent. This means that the exploration of the topologies is converging, then, the BLX-α crossover, described below, is applied in order to generate two different offsprings.

Crossover operator 3: BLX-α crossover over the centers and radii. The previous operators modify the topology of the network, however, if only these are applied, the centers and the radii would not be able to evolve to proper values. To be able to modify the centers and the radii, the BLX-α [4] crossover is employed.

Crossover operator 4: Binary crossover over the input variables: 2 point crossover. This crossover is a generalization of the simple crossover, 2 crossover points a and b with $0 < a < b < h + 1$ are randomly selected and the segments of the parents, the genes in the positions $i_j^1, j = a...b$ and $i_j^2, j = a...b$, are exchanged to generate the offspring.

2.4 Mutation Operators

The mutation operators proposed for this algorithm can be separated in two categories: 1) mutations without any knowledge and 2) mutations using expert knowledge.

The mutation without any knowledge refers to those changes that are performed in a random way, those changes can affect both the structure and the parameters of the RBFNNs. The objective of these operators is to add randomness in the search process to avoid the convergence to local minima. The mutation operators with expert knowledge are mutations that affect also the structure and the parameters of the RBFNNs but using some information in such a way that the changes are not completely random.

Mutations without any knowledge. There are five operators that are completely random, the two fist operators modify the structure of the network meanwhile the third and the forth modify the parameters of the network:

- The first one is the addition of an RBF in one random position over the input vectors space setting his radio also with a random value. All the random values are in the interval [0,1] since the input vectors and their output are normalized.
- The second operator is the opposite to the previous one, deleting an existing RBF. This mutation must be constrained and not be applied when the individual has less than two neurons.
- The third one adds to all the coordinates of a center a random distance which value is chosen in the interval [-0.5,0.5].
- The forth one has exactly the same behavior than the third one but changing the value of the radius of the selected RFB.
- The fifth operator sets/unsets one gene that codifies a variable in the chromosome, selecting/deselecting a variable for the input of the network.

Mutations with expert knowledge. These mutation operators use the information provided by the output of the function to be approximated. The operators are:

- The first operator inserts one RBF in the position of the input vector with the highest error. To select this position the output of the RBFNN is calculated and then it is compared with the

output of the target function, the center will be placed in the position of the point where the difference between the real output and the generated output is greater.
- The second operator introduces RBF close to the neuron which has the smallest local error.
- The third operator consists in the application of a local search algorithm (Levenberg-Mardquardt) to tune the positions of the centers and their radii. Only a few iterations should be done, otherwise the population will converge too fast to a local minima.

2.5 Fitness Function

The fitness function for an individual is a vector with two components: the number of RBFs and the approximation error (Normalized Root Mean Squared Error) of the RBFNNs it encodes. In order to avoid overtraining, a k-fold crossvalidation method is used, the training set is divided in three folds, generating three training sets of data and three test sets. For each training set, the optimal weights are computed and then the test error is obtained. The fitness is the average of the three test errors. The reason to choose K equal to three is because if K is too large, the evaluation of the individuals could become too expensive, making the algorithm run too slow.

2.6 Parallelization Paradigm

The proposed algorithm has been parallelized dividing the population in sub populations that are evolved in different processors, this method has received several names, such as coarse-grained parallel GAs, distributed GAs, or island-models GAs.

There are several reasons to parallelize a multiobjective algorithm such as the desire to reduce the execution time and to explore the large solution space. However these elements could come straight forward from an adequate parallel implementation of any kind of GA. Therefore, the purpose of the parallel implementation performed with the algorithm is not only the improvement in the execution time with the consequent major exploration of the solution space but with the purpose of use the ideas introduced in [13]. The authors propose that having several islands using different crossover operators (heterogenous distributed GAs) the quality of the results could be improved. Recently, this was shown to be quite effective when designing RBFNNs with MOGAs using specialised crossover and mutation operators [8].

The proposed algorithm performs a specialisation of the crossover and mutation operators, dividing them in four islands, each island will evolve a determined aspect of the RBFNN:

- Island 1: This island is specialised in the evolution of the input variables for each RBFNN. Therefore, the crossover it uses is crossover 1 and the mutation it performs is the binary mutation over the binary part of the chromosome.
- Island 2: The task of evolving the structure of the networks will be done in this island. It will use the crossover 2, so it can increase and decrease the number of neurons. The mutation operators that will be applied are the ones that modify the number of neurons.
- Island 3: The third island has the target of evolving the parameters of the centers and the radii of the RBFs, thus, crossover 3 (BLX-α) is applied to modify the values of these two parameters. The mutations only modify the real values encoded in the individuals.
- Island 4: The last island exploits as well the genetic material already contained in the individuals exchanging the neurons through the crossover 4 and applying the same mutation operators than the island 3.

The islands communicate with each other thanks to a migration mechanism where each island sends its Pareto front to all the other islands, this topology is know as fully connected and its represented in Figure 1. Once each island receives all the Pareto fronts, it removes the repeated individuals and append the immigrants to the population of parents and children, performing then the Non-dominated sorting procedure. Proceeding like this, we remain the diversity of each island but introducing the best individuals of the other populations, maintaining a global Pareto front in all the islands.

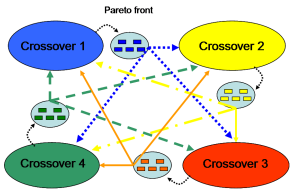

Fig. 1. Fully connected topology

2.7 Algorithm Scheme

The proposed algorithm combines the two approaches presented in [3,5]: it performs a local search at the beginning of the algorithm, then, during the execution a local search can be applied as a mutation and finally a local search is applied to the individuals in the Pareto front. The proposed algorithm combines one of the best MOGAs (NSGA-II) with all the elements that make the MAs outperform classical GAs and, thanks to the parallelization and to the use of specific crossover and mutation operators, the diversity is kept. All these elements allow the algorithm to obtain high quality RBFNNs which are able to approximate with a high precision but remaining the generalization capabilities, as it will be shown in the following section.

3 Experiments

In this experiment, a two dimensional function is generated using a gaussian RBFNN ($e^{-\frac{||\boldsymbol{x}_k - \boldsymbol{c}_i||^2}{r_i^2}}$) over a grid of 20x20 points using the following parameters that where randomly extracted using a random number generator.

The proposed algorithm was feeded with modified input data: it was added 8 more dimensions so the total dimension of the input function was 10. The input data was obtained as follows: $XD = [rand, X_1 + (rand * 0.5), rand, rand * 0.5, X_1, rand, X_2, X_2 + (rand * 0.5), X_1 + (rand * 0.15), X_2 + (rand * 0.15)]$ where $rand$ represents a column vector of random numbers in [0,1], X_1 is all the values for the first dimension of the original data and X_2 is all the values for the second dimension of the original data. Thus, the input data that the algorithm received consisted in pure random values which have no relation with the output, distorted real input values, and the real values. The algorithm should be able to identify the right dimensions and design an RBFNNs that approximates the output using only those dimensions, discarding the other ones. However, there

are two dimensions that are slightly distorted (X_9 and X_{10}), if these are combined together they could provide nearly the same information than the real dimension, this is known as *redundancy*. Nevertheless, since the experiment uses a synthetic example where the real dimensions are know, satisfactory results will select the real values.

The algorithm was executed 20 times going through 300 generations using the following parameters:

- population size = 50. The population size was chosen after performing several experiments giving several values to this parameter within a range of 30 and 70. The reason of these bounds is because we want the islands to have diversity but also to exploit their populations since each one of them uses specialized operators, therefore the size of the population must be not too small neither too big.
- migration rate = 20. The migration rate, this is, the number of generations between migration and migration, is 20 because of the same reasons above. Each island needs a minimum of generations to evolve and evaluate the information from the other islands before exchanging information (individuals) again.

Further studies will incorpore autoregulation mechanism to set the value of the migration rate and stop criteria based on the diversity of the population and the evolution of the Pareto front. Other parameters such as the crossover and mutation probabilities took their values according to what have been established as common in the literature (0.8 and 0.1 respectively). The parallel code was implemented using MATLAB with the interface to use MPI functions available in [7].

The results obtained by the algorithm were satisfactory, being able to identify the original variables and providing very accurate models: mean training error = 0.0684 (0.0921); mean test error = 0.1100 (0.0703), where the error measure is the Normalized Root Mean Square error and the number in brackets is the standard deviation. Clustering algorithms combined with local search were run using 500 individuals obtaining, as best result: training error = 0.1216; test error = 0.1799. The genetic algorithm was executed as well without any local search method obtaining the following results: mean training error = 0.1184 (0.1021); mean test error = 0.1579 (0.0917). Hence, it has been shown how, once again, the memetic approach can improve the results in comparison with the other techniques and how the algorithm is capable to solve successfully the problem of the design RBFNNs to approximate a given function selecting the variables that will take as inputs.

4 Conclusions

These paper introduced a new algorithm to design RBFNNs which must approximate a given function. The novelties of the proposed algorithm respect the other approaches found in the literature are:

- In the field of feature extraction, the algorithm has presented the hybridation of two techniques such us MI and the classical genetic approach.
- Another innovation in the feature extraction field is the parallel evolution of the models (RBFNNs) that evaluate the quality of the individuals encoding the set of variables to be selected.
- In the field of MAs, the algorithm presents a new three steps structure where the local search is applied at the three stages of the algorithm: initialization of the individuals, evolution of the individuals and final optimization of the Pareto front.

– In in the field of parallel MOGAs, the algorithm proposes a parallel implementation of the NSGA-II algorithm which is able to specialize in the evolution of the different objectives to be satisfied and perform other task at the same time (feature selection) without influencing the evolution of the objectives considered in the multiobjective optimization.

This work has been partially supported by the Spanish CICYT Project TIN2004-01419

References

1. http://www.klab.caltech.edu/∼kraskov/MILCA/.
2. D. S. Broomhead and D. Lowe. Multivariate functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.
3. K. Deb and T. Goel. Controlled elitist non-dominated sorting genetic algorithms for better convergence. In *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 67–81. Springer-Verlag, 2001.
4. L. J. Eshelman, A. Caruana, and J. D. Schaffer. Realcoded genetic algorithms and interval schemata. In J. David Schaffer, editor, *Foundation of Genetic Algorithms 2*, pages 187–202. Morgan Kaufmann, 1993.
5. X. Gandibleux, H. Morita, and N. Katoh. The supported solutions used as a genetic information in a population heuristic. In *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 429–442. Springer-Verlag, 2001.
6. J. González, I. Rojas, J. Ortega, H. Pomares, F.J. Fernández, and A. Díaz. Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation. *IEEE Transactions on Neural Networks*, 14(6):1478–1495, November 2003.
7. A. Guillén. *Writing programs in MATLAB using any implementation of MPI.* http://atc.ugr.es/ aguillen, 2005.
8. A. Guillén, I. Rojas, J. Gonz ález, H. Pomares, L.J. Herrera, and B. Paechter. Improving the Performance of Multi-objective Genetic Algorithm for Function Approximation Through Parallel Islands Specialisation. *Lecture Notes in Artificial Intelligence*, 4304:1127–1132, 2006.
9. A. Guillén, I. Rojas, J. González, H. Pomares, L. J. Herrera, and F. Fernández. Multiobjective RBFNNs Designer for Function Approximation: An Application for Mineral Reduction. *Lecture Notes in Computer Science*, 4221:511–520, 2006.
10. A. Guillén, I. Rojas, J. González, H. Pomares, L. J. Herrera, and A. Prieto. A Fuzzy-Possibilistic Fuzzy Ruled Clustering Algorithm for RBFNNs Design. *Lecture Notes in Computer Science*, 4259:647–656, 2006.
11. A. Guillén, I. Rojas, J. González, H. Pomares, L.J. Herrera, O. Valenzuela, and A. Prieto. A Possibilistic Approach to RBFN Centers Initialization. *Lecture Notes in Computer Science*, 3642:174–183, 2005.
12. A. Guillén, I. Rojas, J. González, H. Pomares, L.J. Herrera, O. Valenzuela, and A. Prieto. Improving Clustering Technique for Functional Approximation Problem Using Fuzzy Logic: ICFA algorithm. *Lecture Notes in Computer Science*, 3512:272–280, June 2005.
13. F. Herrera and M. Lozano. Gradual distributed real-coded genetic algorithms. *IEEE-EC*, 4(1):43, April 2000.
14. N. B. Karayiannis, M. Balasubramanian, and H. A. Malki. Evaluation of cosine radial basis function neural networks on electric power load forecasting. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 2100–2105, July 2003.
15. P. Moscato. A memetic approach for the travelling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. *Parallel Computing and Transputer Applications*, pages 177–176, 1992.