Multiobjective RBFNNs Designer for Function Approximation: An Application for Mineral Reduction

Alberto Guillén, Ignacio Rojas, Jesús González, Héctor Pomares, L.J. Herrera and Francisco Fernández

University of Granada

Abstract. Radial Basis Function Neural Networks (RBFNNs) are well known because, among other applications, they present a good performance when approximating functions. The function approximation problem arises in the construction of a control system to optimize the process of the mineral reduction. In order to regulate the temperature of the ovens and other parameters, it is necessary a module to predict the final concentration of mineral that will be obtained from the source materials. This module can be formed by an RBFNN that predicts the output and by the algorithm that designs the RBFNN dynamically as more data is obtained. The design of RBFNNs is a very complex task where many parameters have to be determined, therefore, a genetic algorithm that determines all of them has been developed. This algorithm provides satisfactory results since the networks it generates are able to predict quite precisely the final concentration of mineral.

1 Introduction

Many dynamic optimization problems can be found during the process of the nickel production with the CARON technology. These problems require reaching a balance between the immediate gaining and the optimum behavior of the systems through the time. As an example, it can be considered the process of the mineral reduction. In this process, an expenditure of technologic petroleum is used to establish the thermic profile for the ovens that determine the different chemist reactions for the correct process. This is a complex task that nowadays requires a human operator to take decisions based on his experience and intuition. Therefore, it would be very helpful if a support decision system can be designed and implemented. Figure 1 shows the input, output and control variables that will be used to characterize the model. All these variables are registered by a SCADA system that generates the data used for the experiments.

The problem consists in the optimization of the necessities of the technological petroleum through the analysis of the data dynamically obtained. Since there are several necessities, we are tackling a multiobjective optimization problem. These kind of problems do not have an unique solution since the set of solutions cannot be completed sorted because, for some cases, it is impossible to decide

which solution is better. The set of solutions that cannot be improved is know as the optimal Pareto.

The mineral reduction process can be characterized by the following functions:

- Extractions = f_1 (Input mineral, Oven temperature, Reducing agents)
- Oven temperatures = f_2 (Input mineral, Chamber temperatures)
- Reducing agents = f_3 (Input mineral, Additive petroleum, Petroleum in chambers)

The problem then consists in the learning of the three different functions that relate the input vectors with the corresponding output. This is possible since the data will be measured directly from the source, and once these functions are learned, it will be possible to generate new values not defined in the training sets that will help to take decisions in order to optimize the process.

Figure 2 shows an hybrid process for the dynamic optimization. In the process, there is a module that has to approximate the behavior of the system in order to predict it and to give this information to the Neuro-programming optimization module that will provide the information to take decisions.

Most of the dynamic neuro-programming methods start with an initial sequence of control actions that are used to compute an initial value of the fitness function that will be used. This initial situation can be improved by modifying the the initial control actions. The algorithm proposed in this paper will be used in the function approximation module shown in Figure 2. This module has to include a predictor element, the RBFNN, and an algorithm that designs the network since the number of inputs vectors grows dynamically each 8 hours. This time frame is big enough to be able to use a genetic algorithm that will design a RBFNN that will approximate the input data.

The algorithm presented in this paper is able to design this kind of networks providing excellent results as it will be shown in the experiment section.

2 RBFNN description

The problem to be tackled consists in designing an RBFNN that approximates a set of given values. The use of this kind of neural networks is a common solution since they are able to approximate any function [10, 11]. Formally, a function 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 $\sum_{k=1}^{n} ||y_k - \mathcal{F}(\boldsymbol{x}_k)||^2$ is minimum. The purpose of the design is to be able to obtain outputs from input vectors that were not specified in the original training data set.

An RBFNN $\mathcal{F}$ with fixed structure to approximate an unknown function F with n entries and one output starting from a set of values $\{(\boldsymbol{x}_k; y_k); k = 1, ..., n\}$ with $y_k = F(\boldsymbol{x}_k) \in \mathbb{R}$ and $\boldsymbol{x}_k \in \mathbb{R}^d$, has a set of parameters that have to be optimized:

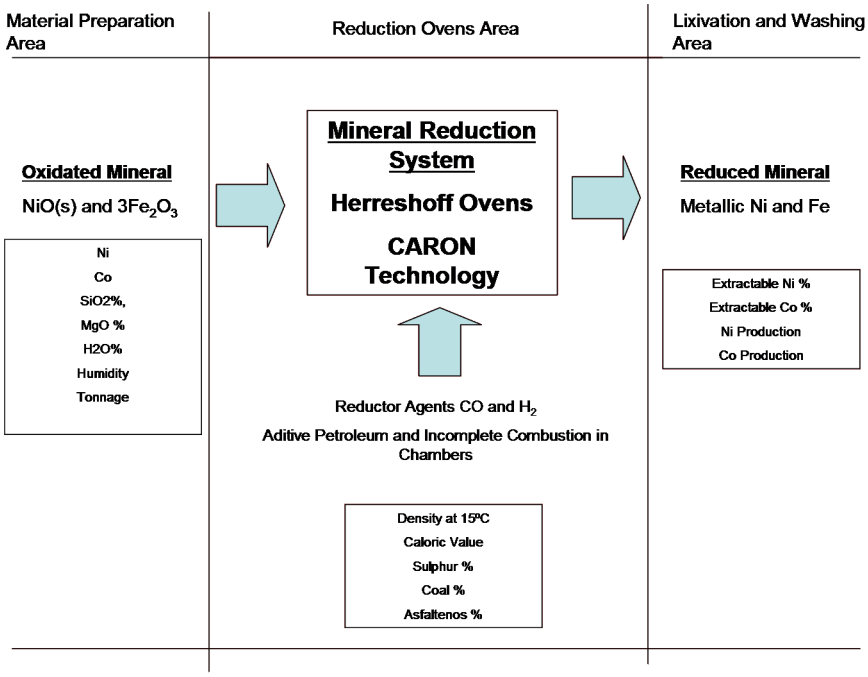

Fig. 1. Mineral reduction process

$$\mathcal{F}(\boldsymbol{x}_k; C, R, \Omega) = \sum_{j=1}^{m} \phi(\boldsymbol{x}_k; \boldsymbol{c}_j, r_j) \cdot \Omega_j \tag{1}$$

where $C = \{\boldsymbol{c}_1, ..., \boldsymbol{c}_m\}$ is the set of RBF centers, $R = \{r_1, ..., r_m\}$ is the set of values for each RBF radius, $\Omega = \{\Omega_1, ..., \Omega_m\}$ is the set of weights and $\phi(\boldsymbol{x}_k; \boldsymbol{c}_j, r_j)$ represents an RBF. The activation function most commonly used for classification and regression problems is the Gaussian function because it is continuous, differentiable, it provides a softer output and improves the interpolation capabilities [2, 13].

The procedure to design an RBFNN starts by setting the number of RBFs in the hidden layer, then the RBF centers $\boldsymbol{c}_j$ must be placed and a radius r_j has to be set for each of them. Finally, the weights Ω_j can be calculated optimally by solving a linear equation system [4].

3 Multiobjective Algorithm for Function Approximation: MOFA

This section describes the algorithm that could be used in the prediction and function approximation module within the system described in the previous

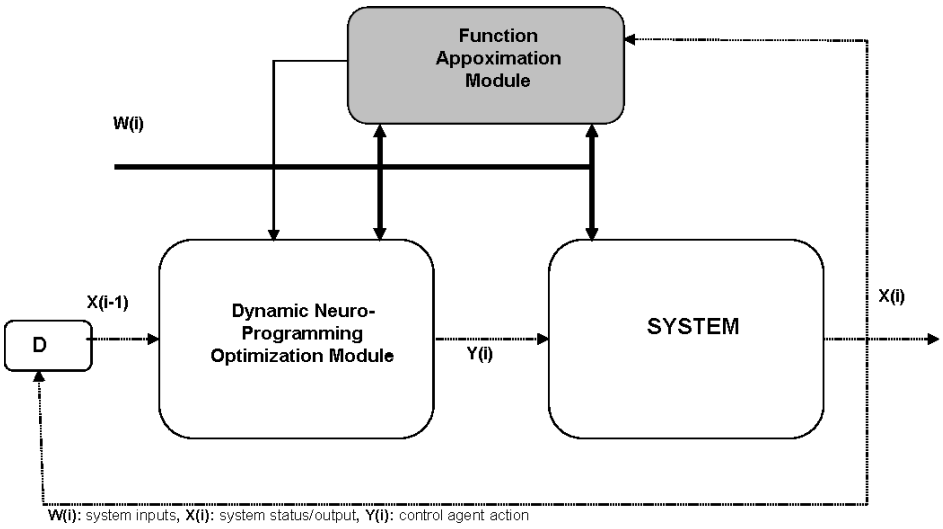

Fig. 2. Hybrid process for the dynamic optimization

section. This algorithm is based in the popular multiobjective non-dominated sorting genetic algorithm in its second version (NSGA-II) [3]. The two objectives in the algorithm are to obtain the network with the smallest error and with the smallest number of RBFs. The bigger the networks become, the more expensive is its manipulation within the genetic algorithm, making it run very slowly and the network must be retrained each 8 hours. This section will introduce the new elements that have been incorporated to fit the original algorithm to the design of RBFNNs.

3.1 Representing RBFNN in the Individuals

As it was shown in the Introduction, to design an RBFNN it is needed to specify:

1. the number of RBFs
2. the position of the centers of the RBFs
3. the length of the radii
4. the weights for the output layer

The individuals in the population of the algorithm will contain the first three elements in a vector of real numbers. Instead of including the weights, the approximation error is stored in order to save computational effort by the time the individuals will be compared.

In the following subsections the concept of local error of an RBF will be referred. The local error is defined as the sum of the errors between the real

output and the output generated by the RBFNN but, instead of considering all the input vectors, only the ones that activate each RBF will be selected. To know if an input vector activates a neuron, its activation function is calculated for each input vector and if it is higher than a determined threshold, the input vector activates the neuron.

3.2 Initial Population

The initial population is generated using clustering algorithms in order to supply good individuals that will make easier and faster to find good solutions. These clustering algorithms are:

- Fuzzy C-means (FCM): This clustering algorithm [1] performs a fuzzy partition of the input data where the same input vector can belong to several clusters at the same time with a membership degree.
- Improved Clustering for Function Approximation (ICFA): this algorithm [5] uses supervised clustering in order to identify the areas where the function is more variable. To do this, it defines the concept of estimated output of a center to assign a value for the center in the output axis.
- Possibilistic Centers Initializer (PCI) and Fuzzy-Possibilistic Clustering for Function approximation (FPCFA): these algorithms [6] modify the way the input vectors are shared between the centers of the clusters. In the ICFA algorithm, a fuzzy partition was defined. In these two algorithms the fuzzy partition is replaced by the ones used in [14] and in [9] respectively.

There are also included individuals generated randomly in order to not to loose diversity in the population.

After this initialization of the population, very few iterations of a local search algorithm ($Levenberg-Marquardt$ [8]) are run and the results are concatenated to the population. This have been proved to improve the quality of the results because the population becomes more diverse since the clustering algorithms are quite robust and could generate individuals that are too similar.

The size of the RBFNNs belonging to the first generation should be small for two reasons:

1. make the initialization as fast as possible
2. allow the genetic algorithm to determine the sizes of the RBFNNs from an incremental point of view, saving the computational effort that would suppose to deal with big networks from the first generations.

The cross operators will have the chance to increment the number of RBFs and with the mutation operators there will be the possibility of removing useless RBFs.

3.3 Crossover operators

The original crossover operator over a binary or real coded chromosome cannot be performed with the individuals of this algorithm because each groups of genes have different meanings. Two crossover operators were designed for these individuals, and experimentally it was concluded that the application of both operators with the same probability provided better results than applying only one of them.

Crossover operator 1: Neurons exchange This crossover operator, conceptually, would be the most similar one to the original crossover. Since the individuals represent an RBFNN with several neurons, the cross of two individuals will be the result of exchanging one neuron. This is exchange is represented in Figure 3. The advantages of this crossover operator is that it exploits the genetic material of each individual without modifying the structure of the network, the other advantage is its simplicity and efficiency.

Crossover operator 2: Addition of the neuron with the smallest error This operator consists in the addition of the neuron with the smallest local error belonging to the other individual and it is represented in Figure 3. If the neuron with the smallest local error is very similar to another in the other network, the neuron with the second smallest error is chosen and so on. This operator will give the opportunity to increase the number of RBFs in one individual, allowing the algorithm to explore more topologies. A refinement step is performed right after the crossing, this refinement consists in the prune of the RBFs which does not influence the output of the RBFNN, to do this, all the weights that connect the processing units to the output layer are calculated and the neurons that do not have a significant weight will be removed.

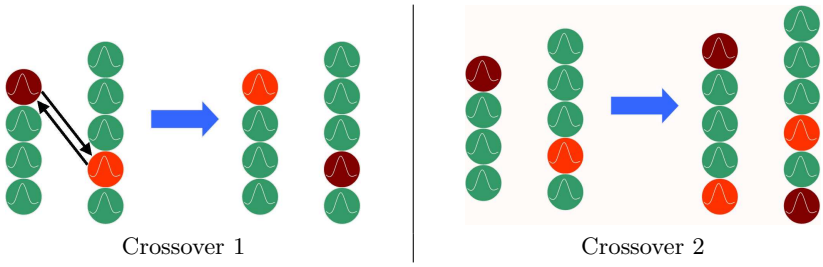

Fig. 3. Crossover operators 1 and 2

3.4 Mutation Operators

The mutation operators proposed for this algorithm can be separated in two categories:

– mutations without any knowledge
– 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 won't be completely random. As it occurs with the crossover operators, if we divide these subset of mutation operators and perform different runs, the results obtained are worse than if we run the algorithm using both kind of mutation operators.

Mutations without any knowledge. There are four operators that are completely random:

– The first one is the deletion 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 two fist operators modify the structure of the network meanwhile the third and the forth modify the parameters of the network. The third and the fourth operators refer to the real coded genetic algorithms as presented in [7].

Mutations with expert knowledge. These mutation operators use the information provided by the output of the function to be approximated. As the previous operators, these will take care of the structure of the network, adding and removing RBFs in a RBFNN, and will also modify the value of the parameters of the RBFs. 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 removes one RBF from the RBFNN, the RBF to be removed is the one with less local error. This could seem not too logical at first, but it allows to keep more diversity in the population since one of the cross operators adds the neuron of the individual with less local error, so combining this to elements, the genes will remain in the population but avoiding redundant elements and allowing to search for new areas in the input vector space.
– The third operator consists in the application of a local search algorithm (Levenberg-Mardquardt) to tune the positions of the centers and their radii but being aware that with these movements the error will be decreased for sure. This operator must be used carefully and only few iterations should be done, otherwise the population will converge too fast to a local minima.

4 Experiments

The data used in the experiments were obtained by measuring the following parameters from the real system:

– *Inputs*: Ni, Fe, Co, Si, Mg, Ton, Temp Ovens (9), *Output*: Additive petroleum index

The data were obtained measuring each 8 hours the different elements, so the prediction of the next value must be done in 8 hours time, the proposed genetic algorithm is able to provide appropriate results in that time frame. For the experiments, it was used the data obtained in 70 days so the size of the data set has 210 instances of 26 variables each. From this 210 instances, 180 were used for training and the rest for test.

The algorithm proposed in this paper will be compared with other evolutionary strategy presented in [12] where the authors propose a new evolutionary procedure to design optimal RBFNNs. It defines a self-organizing process into a population of RBFs based on the estimation of the fitness for each neuron in the population, which depends on three factors: the weight and width of each RBF, the closeness to other RBFs, and the error inside each RBF width. The algorithm also uses operators that, according to a set of fuzzy rules, transform the RBFs. All these elements allowed the authors to define cooperation, speciation, and niching features in the evolution of the population.

Table 1 shows respectively the approximation errors using the training set and the test set. The two evolutionary algorithms are compared also with the CFA algorithm that was designed specifically to perform the initialization step in the design of RBFNN for function approximation. Once the algorithms were executed a local search algorithm was applied to their results. These tables show how the proposed algorithm overcomes the other approaches when approximating the training data and the test data set. The other two algorithms are able to approximate the training set with a very small error when many RBFs are used, however, when they try to approximate the test data, the error increases significantly. The networks generated by the proposed algorithm have the ability of

approximating the training error quite precisely but without loosing generality so the test error is still small, not like when the other algorithms are used.

Table 1. Mean of the approximation error (NRMSE) for the training and test data

Training				Test			
RBFs	**CFA**	**Rivera**	**MOFA**	RBFs	**CFA**	**Rivera**	**MOFA**
3	0.623	0.428	0.282	3	4.963	4.963	1.995
4	0.566	0.327	0.140	4	1.204	2.235	0.954
5	0.590	0.313	0.170	5	1.644	1.382	1.405
7	0.182	0.169	0.0140	7	1.309	1.410	0.870
8	0.061	0.054	3.333e-5	8	1.495	1.403	0.047
9	0.006	0.007	6.264e-6	9	1.479	1.339	0.047
10	0.003	0.002	4.513e-6	10	1.125	1.129	0.047
11	0.026	0.006	4.385e-6	11	1.128	1.105	0.043
12	0.017	0.002	2.946e-6	12	1.024	0.995	0.045
13	0.004	0.001	2.939e-6	13	0.730	0.755	0.047
14	8.264e-4	4.854e-5	2.896e-6	14	0.830	0.520	0.030
16	6.264e-4	3.644e-5	2.527e-6	16	0.519	0.312	0.029

5 Conclusions

This paper has presented a system that can be used to control and optimize the mineral extraction from source materials. One of the modules that builds the system is in charge of predicting the final amount of extracted mineral from empirical data obtained previously. The module consists in an RBFNN, that is able to predict quite precisely the real output of material, and in a genetic algorithm that trains the network within the time frame required by the system. A multiobjective genetic algorithm that designs the RBFNNs for the prediction module was presented, obtaining a very good performance when it was compared against other techniques for the design of RBFNNs.

Acknowledgements This work has been partially supported by the Spanish CICYT Project TIN2004-01419 and the European Commission's Research Infrastructures activity of the Structuring European Research Area programme, contract number RII3-CT-2003-506079 (HPC-Europa)

References

1. J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum, New York, 1981.

2. A. G. Bors. Introduction of the Radial Basis Function (RBF) networks. *OnLine Symposium for Electronics Engineers*, 1:1–7, February 2001.
3. Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Computation*, 6(2):182–197, 2002.
4. 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.
5. 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.
6. 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.
7. F. Herrera, M. Lozano, and J. L. Verdegay. Tackling real-coded genetic algorithms: operators and tools for the behavioural analysis . *Artificial Intelligence Reviews*, 12(4):265–319, 1998.
8. D. W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Inequalities. *SIAM J. Appl. Math.*, 11:431–441, 1963.
9. N. R. Pal, K. Pal, and J. C. Bezdek. A Mixed C–Means Clustering Model. In *Proceedings of the 6th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'97)*, volume 1, pages 11–21, Barcelona, July 1997.
10. J. Park and J. W. Sandberg. Universal approximation using radial basis functions network. *Neural Computation*, 3:246–257, 1991.
11. T. Poggio and F. Girosi. Networks for approximation and learning. In *Proceedings of the IEEE*, volume 78, pages 1481–1497, 1990.
12. A. J. Rivera Rivas, J. Ortega Lopera, I. Rojas Ruiz, and M. J. del Jesus Daz. Co-evolutionary Algorithm for RBF by Self-Organizing Population of Neurons. *Lecture Notes in Computer Science*, (2686):470–477, June 2003.
13. I. Rojas, M. Anguita, A. Prieto, and O. Valenzuela. Analysis of the operators involved in the definition of the implication functions and in the fuzzy inference proccess. *International Journal of Approximate Reasoning*, 19:367–389, 1998.
14. J. Zhang and Y. Leung. Improved possibilistic C–means clustering algorithms. *IEEE Transactions on Fuzzy Systems*, 12:209–217, 2004.