Improving Clustering Technique for Functional Approximation Problem using Fuzzy Logic: ICFA algorithm

A. Guillén, I. Rojas, J. González, H. Pomares, L.J. Herrera, O. Valenzuela, and A. Prieto

Department of Computer Architecture and Computer Technology
Universidad de Granada. Spain

Abstract. Clustering algorithms have been applied in several disciplines successfully. One of those applications is the initialization of Radial Basis Functions (RBF) centers composing a Neural Network, designed to solve functional approximation problems. The Clustering for Function Approximation (CFA) algorithm was presented as a new clustering technique that provides better results than other clustering algorithms that were traditionally used to initialize RBF centers. Even though CFA improves performance against other clustering algorithms, it has some flaws that can be improved. Within those flaws, it can be mentioned the way the partition of the input data is done, the complex migration process, the algorithm's speed, the existence of some parameters that have to be set in order to obtain good solutions, and the convergence is not guaranteed. In this paper, it is proposed an improved version of this algorithm that solves the problems that its predecessor has using fuzzy logic successfully. In the experiments section, it will be shown how the new algorithm performs better than its predecessor and how important is to make a correct initialization of the RBF centers to obtain small approximation errors.

1 Introduction

Designing an RBF Neural Network (RBFNN) to approximate a function from a set of input-output data pairs, is a common solution since this kind of networks are able to approximate any function [4, 9]. 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 $\mathcal{G}$ so $y_k = \mathcal{G}(\boldsymbol{x}_k) \in \mathbb{R}$ with $\boldsymbol{x}_k \in \mathbb{R}^d$. 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 initialization of the centers of RBFs is the first step to design an RBFNN. This task has been solved traditionally using clustering algorithms [8] [10]. Clustering techniques have been applied to classification problems [6], where the task to solve is how to organize observed data into meaningful structures. In classification problems, the input data has to be assigned to a pre-defined set of labels,

thus, if a label is not assigned correctly, the error will be greatly increased. In the functional approximation problem, a continuous interval of real numbers is defined to be the output of the input data. Thus, if the generated output value is near the real output, the error does not increase too much.

In this context, a new clustering algorithm for functional approximation problems was designed in our research group: Clustering for Functional Approximation (CFA)[5].The CFA algorithm uses the information provided by the function output in order to make a better placement of the centers of the RBFs. This algorithm provides better results in comparison with traditional clustering algorithms but it has several elements that can be improved.

In this paper, a new algorithm is proposed, solving all the problems presented in the CFA algorithm using fuzzy logic techniques, and improving results, as it will be shown in the experiments section.

2 RBFNN Description

A 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:

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

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, 7]. The procedure to design an RBFNN for functional approximation problem is shown below:

1. Initialize RBF centers $\boldsymbol{c}_j$
2. Initialize the radius r_j for each RBF
3. Calculate the optimum value for the weights Ω_j

The first step is accomplished by applying clustering algorithms, the new algorithm proposed in this paper will initialize the centers, providing better results than other clustering algorithms used for this task.

3 Clustering For Function Approximation Algorithm: CFA

This algorithm uses the information provided by the objective function output in such a way that the algorithm will place more centers where the variability of the output is higher instead of where there are more input vectors.

To fulfill this task, the CFA algorithm defines a set $O = \{o_1, ..., o_m\}$ that represents a hypothetic output for each center. This value will be obtained as a weighted mean of the output of the input vectors belonging to a center.

CFA defines an objective function that has to be minimized in order to converge to a solution:

$$\frac{\sum_{j=1}^{m} \sum_{\boldsymbol{x}_k \in C_j} \|\boldsymbol{x}_k - \boldsymbol{c}_j\|^2 \omega_{kj}}{\sum_{j=1}^{m} \sum_{\boldsymbol{x}_k \in C_j} \omega_{kj}} \tag{2}$$

where ω_{kj} weights the influence of each input vector in the final position a center. The bigger the distance between the expected output of a center and the real output of an input vector is, the bigger the influence in the final result will be. The calculation of w is obtained by:

$$\omega_{kj} = \frac{|F(\boldsymbol{x}_k) - o_j|}{\max_{i=1}^{n} \{F(\boldsymbol{x}_i)\} - \min_{i=1}^{n} \{F(\boldsymbol{x}_i)\}} + \vartheta_{\min}, \quad \vartheta_{\min} > 0. \tag{3}$$

The first addend in this expression calculates a normalized distance (in the interval [0,1]) between $F(\boldsymbol{x}_k)$ and o_j, the second addend is a minimum contribution threshold. The smaller $\vartheta_{\min}$ becomes, the more the centers are forced to be in areas where the output is more variable.

The CFA algorithm is structured in three basic steps: Partition of the data, centers and estimated output updating and a migration step.

The partition is performed as it is done in Hard C-means [3], thus, a Voronoi partition of the data is obtained. Once the input vectors are partitionated, the centers and their estimated outputs have to be updated, this process is done iteratively using the equations shown below:

$$\boldsymbol{c}_j = \frac{\sum_{\boldsymbol{x}_k \in C_j} \boldsymbol{x}_k \omega_{kj}}{\sum_{\boldsymbol{x}_k \in C_j} \omega_{kj}} \quad o_j = \frac{\sum_{\boldsymbol{x}_k \in C_j} F(\boldsymbol{x}_k) \omega_{kj}}{\sum_{\boldsymbol{x}_k \in C_j} \omega_{kj}} \ . \tag{4}$$

The algorithm, to update centers and estimated outputs, has an internal loop that iterates until the total distortion of the partition is not decreased significantly.

The algorithm has a migration step that moves centers allocated in input zones where the target function is stable, to zones where the output variability is higher. The idea of a migration step was introduced in [11] as an extension of Hard C-means.

CFA tries to find an optimal vector quantization where each center makes an equal contribution to the total distortion [4]. This means that the migration step will iterate, moving centers that make a small contribution to the error to the areas where centers make a bigger contribution.

3.1 Flaws in CFA

CFA has some flaws that can be improved, making the algorithm more robust and efficient and providing better results.

The first disadvantage of CFA is the way the partition of the data is made. CFA makes a hard partition of the data where an input vector can belong uniquely to a center, this is because it is based on the Hard C-means algorithm. When Fuzzy C-means [1] was developed, it demonstrated how a fuzzy partition of the data could perform better than a hard partition. For the functional approximation problem, it is more logical to apply a fuzzy partition of the data because an input vector can activate several neurons with a certain degree of activation, in the same way an input vector can belong to several centers in a fuzzy partition.

The second problem is the setting of a parameter which influences critically the results that can be obtained. The parameter is $\vartheta_{\min}$, the minimum contribution threshold. The smaller this parameter becomes, the slower the algorithm becomes and the convergence becomes less warranted. The need of a human expert to set this parameter with a right value is crucial when it is desired to apply the algorithm to different functions, because a wrong value, will make the algorithm provide bad results.

The third problem of CFA is the iterative process to converge to the solution. The convergence is not demonstrated because it is presented as a weighted version of Hard C-means, but the equations proposed do not warrant the convergence of the algorithm. The iterative method is quite inefficient because it has to iterate many times on each iteration of the main body of the algorithm.

The last problem CFA presents is the migration process. This migration step is quite complex and makes the algorithm run very slow. It is based on a distortion function that require as many iterations as centers, and adds randomness to the algorithm making it not too robust.

4 Improved CFA Algorithm: ICFA

Let's introduce the new elements in comparison with CFA, and let's see the reasons why this new elements are introduced.

4.1 Input Data Partition

As it was commented before, for the functional approximation problem, is better to use a fuzzy partition, but CFA uses a hard partition of the data. In ICFA, in the same way as it is done in Fuzzy C-means, a fuzzy partition of the data is used, thus, an input vector belongs to several centers at a time with a certain membership degree.

4.2 Parameter w

In CFA, the estimated output of a center is calculated using a parameter w (3). The calculation of w implies the election of a minimum contribution value ($\vartheta_{\min}$) that will affect in a serious way the performance and the computing time of the algorithm.

In order to avoid the establishment of a parameter, ICFA removes this threshold, and the difference between the expected output of a center and the real output of the input data is not normalized. Thus, the calculation of w is done by:

$$w_{kj} = |F(\boldsymbol{x}_k) - o_j| \tag{5}$$

where $F(\boldsymbol{x})$ is the function output and o_j is the estimated output of $\boldsymbol{c}_j$.

4.3 Objective Function and Iterative Process

In order to make the centers closer to the areas where the target function is more variable, a change in the similarity criteria used in the clustering process it is needed. In Fuzzy C-means, the similarity criteria is the euclidean distance. Proceeding this way, only the coordinates of the input vectors are used, thus, the membership values u_{ik} for the matrix $U = [u_{ik}]$ for a given center will be small for the input vectors that are far from that center, and the values will be big if the input vector is close to that center. For the functional approximation problem, this is not always true because, given a center, its associated cluster can own many input vectors even if they are far from this center but they have the same output values.

To consider these situations, the parameter w is introduced (5) to modify the values of the distance between a center and an input vector. w will measure the difference between the estimated output of a center and the output value of an input vector. The smaller w is, the more the distance between the center and the vector will be reduced. This distance is calculated now by modifying the norm in the euclidean distance:

$$D_{kjW} = \|\boldsymbol{x}_k - \boldsymbol{c}_j\|^2 \cdot w_{kj}^2. \tag{6}$$

where $w_{kj} = |Y_k - o_i|$. The objective function to be minimize is redefined as:

$$J_h(U, C, W) = \sum_{k=1}^{n} \sum_{i=1}^{m} u_{ik}^h D_{kjW}. \tag{7}$$

This function is minimized applying the LS method, obtaining the following equations that will converge to the solution:

$$u_{ik} = \left(\sum_{j=1}^{m} \left(\frac{D_{ikW}}{D_{jkW}} \right)^{\frac{2}{h-1}} \right)^{-1} \qquad \boldsymbol{c}_i = \frac{\sum_{k=1}^{n} u_{ik}^h \boldsymbol{x}_k w_{ik}^2}{\sum_{k=1}^{n} u_{ik}^h w_{ik}^2}$$

$$\boldsymbol{o}_i = \frac{\sum_{k=1}^{n} u_{ik}^h Y_k d_{ik}^2}{\sum_{k=1}^{n} u_{ik}^h d_{ik}^2} \tag{8}$$

where d_{ij} is the euclidean distance between c_i and x_j, and $h > 1$ is a parameter that allow us to control how fuzzy will be the partition and usually is equal to 2.

These equations are the equivalence of the ones defined for CFA (4) where the centers and their expected outputs are updated. These equations are obtained applying Lagrange multipliers and calculating the respect derivatives of the function, so convergence is warranted, unlike in CFA. ICFA, requires only one step of updating, being much more efficient than CFA where an internal loop is required on each iteration of the algorithm to update the centers and the outputs.

4.4 Migration Step

As in CFA, a migration step is incorporated to the algorithm. CFA's migration iterates many times until each center contributes equally to the error of the function defined to be minimized. On each iteration, all centers are considered to be migrated, making the algorithm inefficient and, since it adds random decisions, the migration will affect directly to the robustness of the final results.

ICFA only makes one iteration and instead of considering all centers to be migrated, it performs a pre-selection of the centers to be migrated. The distortion of a center is the contribution to the error of the function to be minimized. To decide what centers will be migrated, it is used a fuzzy rule that selects centers that have a distortion value above the average.By doing this, centers that do not add a significant error to the objective function are excluded because their placement is correct and they do not need help from other center.

There is a fixed criteria to choose the centers to be migrated, in opposite to CFA where a random component was introduced at this point. The center to be migrated will be the one that has assigned the smallest value of distortion. The destination of the migration will be the center that has the biggest value of distortion. The repartition of the input vectors between those two it is like in CFA. If the error is smaller than the one before the migration step, the migration is accepted, otherwise is rejected.

4.5 ICFA General Scheme

Once all the elements that compose the algorithm have been described, the general scheme that ICFA follows is:

Do

Calculate the weighted distance between C_i and X using w
Calculate the new U_i, C_i using U_i and O_i using C_i
Migrate

While(abs(C_{i-1}-C_i<$threshold$)

In ICFA, the start point is not a random initialization of matrix U as in Fuzzy C-means. In the new algorithm, centers will be distributed uniformly through the input data space and their estimated outputs will be equal to the difference between the maximum and the minimum value of the output function. Proceeding like this, all random elements of the previous algorithm are excluded, obtaining the maximum robustness.

5 Experimental Results

To compare the results provided by the different algorithms, it will be used the normalized root mean squared error (NRMSE)

The radii of the RBFs were calculated using the k-neighbors algorithm with k=1. The weights were calculated optimally by solving a linear equation system.

Table 1 shows the errors when approximating the function f_1 (Fig. 1) using the ICFA, Fuzzy C-means and CFA algorithms. In Fig. 1 are represented graphically the results shown in Table 1. Function f_1 is defined as:

$$Y = sin(2piX)e^{-X} - 34sin(piX)e^{-75/X}. \tag{9}$$

Table 1. Mean and Standard Deviation of the approximation error (NRMSE) for function f_1.

Clusters	**FCM**	**CFA**	**FCFA**
6	0.961(0.3E-4)	0.965(0.010)	0.734(0)
7	0.957(0.1E-4)	0.907(0.071)	0.676(0)
8	0.945(0.4E-4)	0.867(0.060)	0.568(0)
9	0.978(0.001)	0.819(0.049)	0.543(0)
10	0.935(0.001)	0.792(0.055)	0.509(0)

The results clearly show the improvement in performance of ICFA in comparison with CFA and its predecessors, not improving only the results, but the robustness.

6 Conclusions

RBFNNs provides good results when they are used for functional approximation problems. The CFA algorithm was designed in order to make the right initialization of the centers for the RBFs improving the results provided by the clustering algorithms that were used traditionally for this task. CFA had some mistakes and disadvantages that could be improved. In this paper, a new algorithm which fix all the problems in CFA is proposed. This new algorithm performs much better than its predecessor.

From the analysis of the results, the following conclusions are obtained:

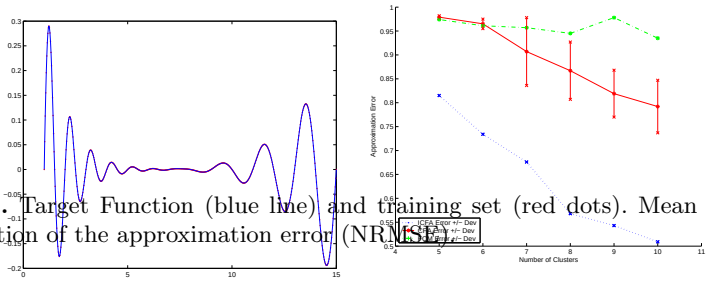

Fig. 1. Target Function (blue line) and training set (red dots). Mean and Standard Deviation of the approximation error (NRMSE).

– It has been demonstrated how important an initialization step is when designing RBFNN for functional approximation problems.
– All problems in CFA were solved so the new algorithm obtains better results.

Acknowledgments

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

References

1. J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, Nueva York, 1981.
2. Adrian G. Bors. Introduction of the Radial Basis Function (RBF) networks. *On-Line Symposium for Electronics Engineers*.
3. R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. New York:wiley, 1973.
4. A. Gersho. Asymptotically Optimal Block Quantization. *IEEE Transanctions on Information Theory*, 25(4):373–380, July 1979.
5. J. González, I. Rojas, H. Pomares, J. Ortega, and A. Prieto. A new Clustering Technique for Function Aproximation. *IEEE Transactions on Neural Networks*, 13(1):132–142, January 2002.
6. J. A. Hartigan. *Clustering Algorithms*. New York:wiley, 1975.
7. A. Prieto I. Rojas, M. Anguita and O. Valenzuela. Analysis of the operators involved in the definition of the implication functions and in the fuzzy inference proccess. *Int. J. Approximate Reasoning*, 19:367–389, 1998.
8. N. B. Karayannis and G.W. Mi. Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques.
9. J. Park and J. W. Sandberg. Universal approximation using radial basis functions network. *Neural Computation*, 3:246–257, 1991.
10. Y. Cai Q. Zhu and L. Liu. A global learning algorithm for a RBF network.
11. M. Russo and G. Patan. Improving the LBG Algorithm. *Lecture Notes in Computer Science*, 1606:621–630, 1999.