Using Fuzzy Logic to Improve a Clustering Technique for Function Approximation

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

Department of Computer Architecture and Computer Technology, University of Granada, 18017 Granada, Spain

Abstract

Clustering algorithms have been applied in several disciplines successfully. One of those applications is the initialization of Radial Basis Function (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.

Key words: Clustering, RBF, Neural Networks, Function approximation, fuzzy.

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 [5,12]. Formally, a function approximation problem can be formulated as, given a set of observations $\{(\vec{x}_k; y_k); k = 1, ..., n\}$ with $y_k = F(\vec{x}_k) \in \mathbb{R}$ and $\vec{x}_k \in \mathbb{R}^d$, it is desired to obtain a function $\mathcal{G}$ so $y_k = \mathcal{G}\ (\vec{x}_k) \in \mathbb{R}$ with $\vec{x}_k \in \mathbb{R}^d$. 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 [10,16]. Clustering techniques have been applied to classification problems [8], 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)[6].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 d entries and one output starting from a set of values $\{(\vec{x}_k; y_k); k = 1, ..., n\}$ with $y_k = F(\vec{x}_k) \in \mathbb{R}$ and $\vec{x}_k \in \mathbb{R}^d$, has a set of parameters that have to be optimized:

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

where $C = \{\vec{c}_1, ..., \vec{c}_m\}$ 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 $\phi(\vec{x}_k; \vec{c}_j, r_j)$ 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 [3,14]. The procedure to design an RBFNN for functional approximation problem is shown below:

(1) Initialize RBF centers $\vec{c}_j$
(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 distortion function that has to be minimized in order to converge to a solution:

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

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:

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

The first addend in this expression calculates a normalized distance (in the interval [0,1]) between $F(\vec{x}_k)$ 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 [4], 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:

$$\vec{c}_j = \frac{\sum_{\vec{x}_k \in C_j} \vec{x}_k \omega_{kj}}{\sum_{\vec{x}_k \in C_j} \omega_{kj}} \tag{4}$$

$$o_j = \frac{\sum_{\vec{x}_k \in C_j} F(\vec{x}_k) \omega_{kj}}{\sum_{\vec{x}_k \in C_j} \omega_{kj}} \tag{5}$$

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 [13] 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 [5]. This means that the migration step will iterate, moving centers that make a small contribution to the total distortion 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 [2] 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 slowly. 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, it is more adequate to use a fuzzy partition since an input vector can activates several neurons at the same time. The CFA algorithm 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 ω (3). The calculation of ω implies the election of a minimum contribution value ($\vartheta_{\min}$) that will affect seriously the performance and the computing time of the algorithm. In order to avoid the establishment of a parameter, ICFA removes this threshold defining a new weighing parameter, w, which is calculated by:

$$w_{kj} = |F(\vec{x}_k) - o_j| \tag{6}$$

where $F(\vec{x}_k)$ is the function output for the input $\vec{x}_k$ and o_j is the estimated output of $\vec{c}_j$. There is no need to normalize the weighing parameter because we will consider normalized functions and the parameter ϑ_{min} is not necessary since the way in which w is used ensures the convergence of the algorithm independently of this parameter.

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_{jk} for the matrix $U = [u_{jk}]$ 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 (6) 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:

$$D_{kjW} = \|\vec{x}_k - \vec{c}_j\|^2 \cdot w_{kj}^2 \tag{7}$$

where $w_{kj} = |Y_k - o_j|$. The distortion function to be minimized is redefined as:

$$J_h(U, C, W) = \sum_{k=1}^{n} \sum_{j=1}^{m} u_{jk}^h D_{kjW} \tag{8}$$

constrained by:

- $\sum_{j=1}^{m} u_{jk} = 1 \ \forall k = 1...n$
- $0 < \sum_{k=1}^{n} u_{jk} < n \ \forall j = 1...m.$

This function is minimized using an alternating optimization procedure in the same way as in [2], which consists in the Picard iteration algorithm that on each iteration calculates the membership function, then the positions of the centers and finally their expected outputs using the following equations:

$$u_{jk} = \left(\sum_{i=1}^{m} \left(\frac{D_{jkW}}{D_{ikW}} \right)^{\frac{1}{h-1}} \right)^{-1} \qquad \vec{c}_j = \frac{\sum\limits_{k=1}^{n} u_{jk}^h \vec{x}_k w_{kj}^2}{\sum\limits_{k=1}^{n} u_{jk}^h w_{kj}^2}$$

$$o_j = \frac{\sum\limits_{k=1}^{n} u_{jk}^h Y_k d_{jk}^2}{\sum\limits_{k=1}^{n} u_{jk}^h d_{jk}^2} \tag{9}$$

where d_{jk} is the euclidean distance between $\vec{c}_j$ and $\vec{x}_k$, 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 derived by simply setting the derivative of the distortion function (extended by Lagrange multipliers to incorporate the constraints) with respect to the parameter to be optimized (U,C and O) equal to zero, 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 distortion 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 distortion function to be minimized. To decide what centers will be migrated, a fuzzy rule that selects centers that have a distortion value above the average is used. By doing this, centers that do not add a significant distortion value to the distortion function are excluded because their placement is correct and they do not need help from another 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

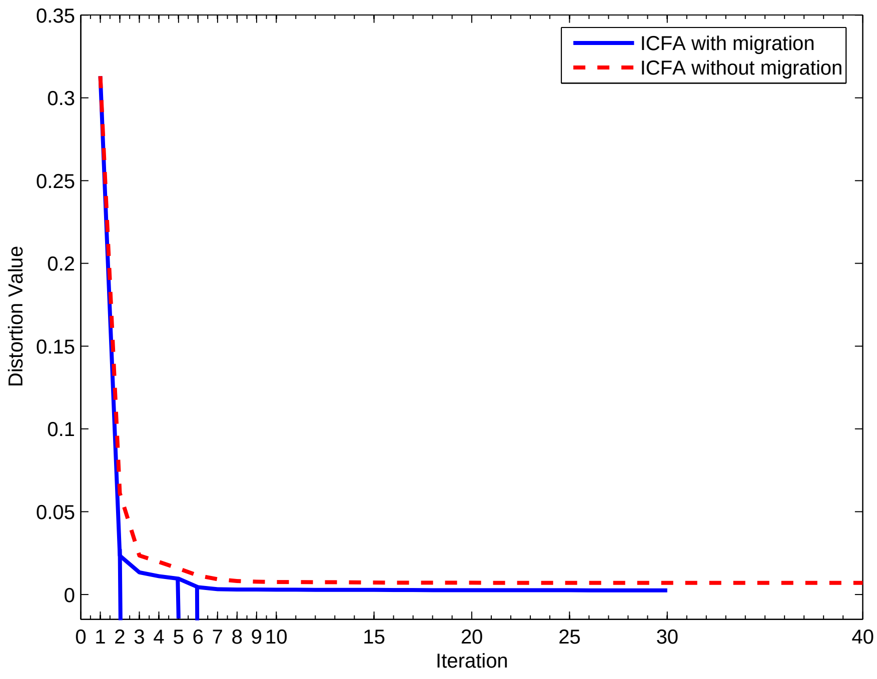

Fig. 1. Values for the distortion function with and without migration when approximating f_1 with 5 centers. The migration was performed in steps 2,5 and 6.

in CFA. If the error is smaller than the one before the migration step, the migration is accepted, otherwise is rejected.

Figure 1 shows the values of the distortion function on each iteration of the algorithm for two executions, one with migration and other without migration. The target function is f_1 (Fig. 3) that will be introduced in the experiments section. In the figure, it is easy to appreciate how adding the migration process, it is possible to escape from local minima and reach smaller distortion values.

4.5 ICFA General Scheme

Once all the elements that compose the algorithm have been described, the general scheme that ICFA follows is shown in Figure 2.

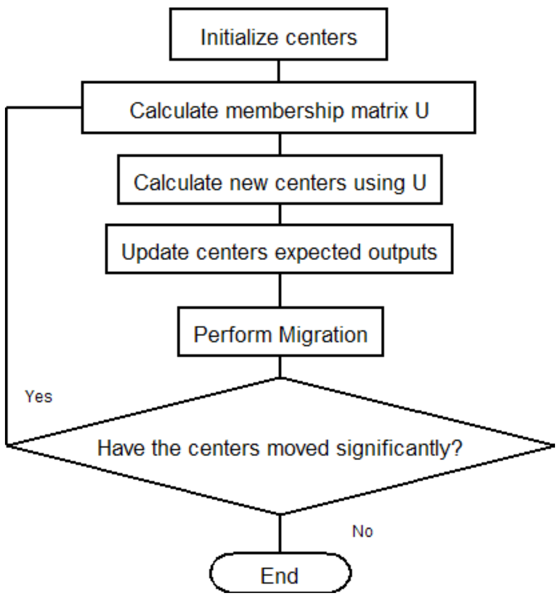
Initialize centers
Calculate membership matrix U
Calculate new centers using U
Update centers expected outputs
Perform Migration
Yes
Have the centers moved significantly?
No
End

Fig. 2. General scheme for the ICFA algorithm

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. Proceeding like this, all random elements of the previous algorithm are excluded, obtaining the maximum robustness. The centers expected outputs must be initialized using the same value, thanks to this, all the centers will be in the same condition respect to the target function output so the weighting parameter will influence the centers in the same way in the first iteration of the algorithm. The initialization value is not too important and does not influence in a significant way the final configuration of the centers although a fixed value of 1 is assigned in order to avoid any random element in the algorithm. This leads to obtain always the same output for a fixed input, providing a standard deviation of zero when multiple executions are run with the same input.

5 Experimental Results

Several experiments were done to show how the proposed algorithm improves the results of its predecessor, the CFA algorithm.

To compare the results provided by the different algorithms it will be used the normalized mean squared error (NRMSE), which is defined as:

$$\mathbf{NRMSE} = \sqrt{\frac{\sum_{k=1}^{n} \left(F(\vec{x}_k) - \mathcal{F}(\vec{x}_k; C, R, \Omega)\right)^2}{\sum_{k=1}^{n} \left(F(\vec{x}_k) - \bar{F}\right)^2}} \tag{10}$$

Where $\bar{F}$ is the average of the output of the input vectors.

The steps followed to generate the RBFNN, once the algorithms placed the centers, are:

- Get the radius of the RBFs using the k-neighbors algorithm with k=1
- Calculate weights optimally by solving a linear equation system.

Once the RBFNNs were obtained, it was applied a local search algorithm (Levenberg-Marquardt [11]) to improve results by adjusting RBF centers and radii. The RBF function used to compute the output was: $e^{-\frac{||\vec{x}_k - \vec{c}_j||^2}{r_j^2}}$

5.1 Experiment 1

The data used for this experiment consists in the real values of reflectance for different wavelengths of emitted by a piece of iberian pig. Samples were measured by using a portable Vis/NIR Spectrometer FieldSpec Pro JR (Analytical Spectral Devices, Inc; Boulder, Colorado, USA) and were scanned applying directly the sound on the meet and recording the reflectance in the 350-2500 nm spectral range. The spectrum of each muscle was the average of 10 successive scans. Previous to the measure, the spectrometer was calibrated using a white spectralon 3.62. Figure 3 represents the data once it was normalized, the training set is represented by crosses and the test set is represented by points. It could be useful to have a model that interpolates the reflectance of the meat since this is strongly related with its tenderness. These models could be analyzed to see if the differences between them correspond to variations on the tenderness of the meat and from that, the tenderness could be predicted. Table 1 shows the approximation errors for the training set before and after applying the local search algorithm and Table 2 shows the approximation errors for the test set. The ICFA algorithm is compared with the CFA algorithm, the Gustafson-Kessel (GK) [7] and the FCM algorithm.

The results show how much the performance of the CFA algorithm has been improved. The new algorithm places the centers in better positions obtaining an small approximation error right after the initialization procedure. After applying the local search algorithm, it is shown how the initialization provided by the ICFA algorithm is better since it allows to find better local minima.

It is important to notice how robust is the new algorithm in comparison with the other algorithms. Another remarkable fact is that the RBFNNs generated using the ICFA algorithm do not suffer from overfitting, obtaining small test errors.

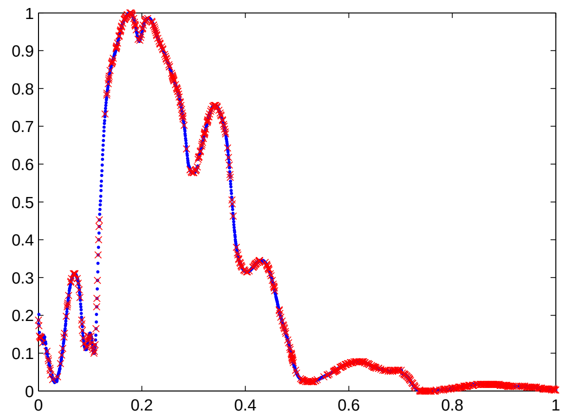

Fig. 3. Target Function f_1

5.2 Experiment 2

The second experiment is taken from the UCI Machine Learning Repository [15]. The goal is to predict the fuel consumption of several kinds of vehicles considering the following features: displacement, horsepower, weight, acceleration and model year. The proposed algorithm is compared with the Modified Gath-Geva fuzzy clustering algorithm (MGGC) [1], the ANFIS algorithm [9] included in the MATLAB fuzzy toolbox and with the Fuzzy Model Identification toolbox (FMID) based on the GK algorithm. From the 392 entries, 196 samples were used for training and the rest for test. In [1], the algorithms were used to design TS fuzzy models with four rules, therefore the RBFNNs generated using the CFA and ICFA algorithms had 4 neurons.

Table 3 shows the approximation errors for the training and test sets. The smallest approximation error is achieved by the ANFIS algorithm however, due to overfitting, the test error is the highest. The ICFA provides the second smallest approximation error for the training set and the smallest approximation error for the test set. This demonstrates that the RBFNNs designed using the ICFA algorithm keep their generalization capability.

Before LS				
Clusters	**FCM**	**GK**	**CFA**	**ICFA**
5	0.270(0.015)	0.310(0.000)	0.316(0.029)	0.219(0)
6	0.255(0.013)	0.316(0.092)	0.316(0.004)	0.207(0)
7	0.276(0.040)	0.265(0.044)	0.403(0.139)	0.237(0)
8	0.250(0.034)	0.273(0.049)	0.258(0.007)	0.480(0)
9	0.259(0.062)	0.256(0.029)	0.231(0.043)	0.206(0)
10	0.206(0.051)	0.232(0.022)	0.235(0.011)	0.168(0)
11	0.254(0.059)	0.192(0.041)	0.216(0.033)	0.159(0)

After LS				
Clusters	**FCM**	**GK**	**CFA**	**ICFA**
5	0.145(0.023)	0.182(0.001)	0.173(0.003)	0.167(0)
6	0.160(0.018)	0.141(0.022)	0.172(0.005)	0.085(0)
7	0.133(0.035)	0.096(0.045)	0.181(0.012)	0.090(0)
8	0.108(0.053)	0.070(0.017)	0.163(0.010)	0.046(0)
9	0.072(0.050)	0.065(0.022)	0.102(0.046)	0.045(0)
10	0.044(0.004)	0.088(0.057)	0.110(0.046)	0.041(0)
11	0.059(0.020)	0.061(0.025)	0.063(0.032)	0.030(0)

Table 1
Mean and Standard Deviation (in brackets) of the approximation error (NRMSE) for function f_1 before and after local search algorithm (LS)

5.3 Experiment 3

The third experiment is also taken from the UCI Machine Learning Repository [15] and consists, as it is described in the UCI database, in:

"a system involving a servo amplifier, a motor, a lead screw/nut, and a sliding carriage of some sort. It may have been on of the translational axes of a robot on the 9th floor of the AI lab. In any case, the output value is almost certainly a rise time, or the time required for the system to respond to a step change in a position set point."

"This is an interesting collection of data provided by Karl Ulrich. It covers an extremely non-linear phenomenon - predicting the rise time of a servomech-

Clusters	**FCM**	**GK**	**CFA**	**ICFA**
5	0.154(0.020)	0.190(0.001)	0.184(0.007)	0.174(0)
6	0.163(0.013)	0.151(0.024)	0.180(0.006)	0.092(0)
7	0.139(0.034)	0.102(0.045)	0.187(0.011)	0.094(0)
8	0.115(0.055)	0.078(0.016)	0.167(0.010)	0.050(0)
9	0.078(0.049)	0.072(0.024)	0.111(0.048)	0.050(0)
10	0.050(0.007)	0.097(0.058)	0.120(0.045)	0.046(0)
11	0.075(0.020)	0.071(0.022)	0.075(0.036)	0.034(0)

Table 2
Mean and Standard Deviation (in brackets) of the approximation error (NRMSE) for the test set for function f_1

Table 3
Mean and Standard Deviation (in brackets) of the approximation error (NRMSE) for the MPG data set

	Training	**Test**
FMID	0.3396	0.3818
ANFIS	0.2506	11.8222
MGGC	0.3459	0.3688
CFA	0.3459	0.3637
ICFA	0.3306	0.3494

anism in terms of two (continuous) gain settings and two (discrete) choices of mechanical linkages."

The data has 167 instances with 4 attributes and one output:

1. motor: A,B,C,D,E 2. screw: A,B,C,D,E 3. pgain: 3,4,5,6 4. vgain: 1,2,3,4,5 5. class: 0.13 to 7.10

The motor and screw attributes were mapped to numbers from 1 to 5.

From the original 167 instances, 165 were taken and the K-fold cross validation was performed with K=11, this provides 11 training sets of 150 instances and 11 test sets of 15 instances. Results are shown in Table 4. The RBFNNs generated using the ICFA algorithm approximate better the training sets and are able to interpolate the output of the function in a better way than the ones generated using the other algorithms, providing smaller errors with the test data sets.

Table 4
Mean and Standard Deviation (in brackets) of the approximation error (NRMSE) for function the servo-robot problem with the training and test data using cross validation

Training

Clusters	**FCM**	**CFA**	**ICFA**
5	0.300(0.090)	0.302(0.076)	0.257(0.072)
6	0.312(0.091)	0.278(0.053)	0.260(0.066)
7	0.249(0.070)	0.271(0.069)	0.207(0.045)
8	0.211(0.046)	0.239(0.043)	0.187(0.050)
9	0.226(0.064)	0.210(0.041)	0.204(0.067)
10	0.202(0.082)	0.230(0.054)	0.192(0.057)
11	0.177(0.042)	0.207(0.028)	0.193(0.062)
12	0.197(0.060)	0.191(0.055)	0.175(0.049)
13	0.204(0.060)	0.189(0.055)	0.168(0.044)
14	0.171(0.034)	0.175(0.050)	0.168(0.031)

Test

Clusters	**FCM**	**CFA**	**ICFA**
5	0.281(0.112)	0.257(0.089)	0.235(0.090)
6	0.238(0.094)	0.267(0.103)	0.228(0.127)
7	0.182(0.095)	0.242(0.100)	0.185(0.067)
8	0.168(0.087)	0.220(0.107)	0.134(0.046)
9	0.174(0.080)	0.151(0.078)	0.174(0.100)
10	0.126(0.069)	0.168(0.112)	0.127(0.076)
11	0.136(0.063)	0.164(0.079)	0.132(0.065)
12	0.109(0.038)	0.124(0.074)	0.112(0.063)
13	0.111(0.050)	0.124(0.071)	0.082(0.072)
14	0.101(0.105)	0.085(0.041)	0.079(0.061)

5.4 Execution times

Many aspects from the CFA algorithm have been changed in the new proposed algorithm, the result of these changes is not only that the approximation errors

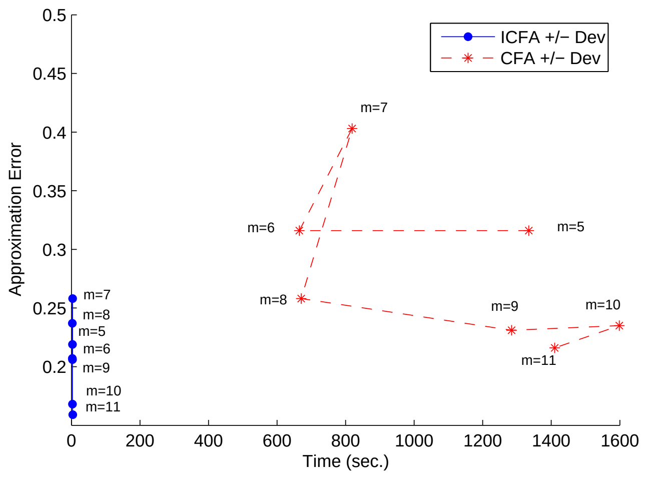

Fig. 4. Execution time and approximation errors for the different executions using several number of centers and f_1 as input.

are smaller but the execution time has been significantly reduced. As it was commented before, the updating of the centers positions and their expected outputs have been simplified to just one calculation instead of an internal loop within the algorithm. The migration process was also accelerated by removing the loop that processed all the centers on each iteration, just doing one iteration considering only the centers with the maximum and the minimum utility. To illustrate this, Figure 4 depicts the execution time and the error obtained for function f_1. This figure shows how with the ICFA algorithm can obtain better results in less time.

6 Conclusions

RBFNNs provide good results when they are used for functional approximation problems. The CFA algorithm was designed in order to make an adequate initialization of the centers of the RBFs overcoming the results provided by the clustering algorithms that were used traditionally for this task. CFA had some problems 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 providing smaller approximation errors in less time.

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

- It has been demonstrated how important an initialization step is when designing RBFNN for functional approximation problems. The design of specific clustering algorithms shows how the results can be improved significantly.

- The new algorithm proposed in this paper outperforms the CFA algorithm which was shown to be very effective for the centers initialization task. The improvements obtained are: smaller execution times, maximum robustness and smaller approximation errors.

Acknowledgments

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

References

[1] János Abonyi, Robert Babuska, and Ferenc Szeifert. Modified gath-geva fuzzy clustering for identification of takagi-sugeno fuzzy models. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 32(5):612–621, 2002.

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

[3] A. G. Bors. Introduction of the Radial Basis Function (RBF) networks. *OnLine Symposium for Electronics Engineers*, 1:1–7, February 2001.

[4] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* New York: Wiley, 1973.

[5] A. Gersho. Asymptotically Optimal Block Quantization. *IEEE Transanctions on Information Theory*, 25(4):373–380, July 1979.

[6] 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.

[7] E.E. Gustafson and W.C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. *IEEE CDC*, pages 761–766, 1979.

[8] J. A. Hartigan. *Clustering Algorithms.* New York: Wiley, 1975.

[9] Jyh-Shing Roger Jang and Chuen-Tsai Sun. Neuro-fuzzy modeling and control. *Proceedings of the IEEE*, 1995.

[10] N. B. Karayiannis and G. W. Mi. Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques. *IEEE Transactions on Neural Networks*, 8:1492–1506, November 1997.

[11] D. W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Inequalities. *SIAM J. Appl. Math.*, 11:431–441, 1963.

[12] J. Park and J. W. Sandberg. Universal approximation using radial basis functions network. *Neural Computation*, 3:246–257, 1991.

[13] G. Patanè and M. Russo. The Enhanced-LBG algorithm. *Neural Networks*, 14(9):1219–1237, 2001.

[14] 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.

[15] C.L. Blake S. Hettich and C.J. Merz. UCI repository of machine learning databases. 1998.

[16] Q. Zhu, Y. Cai, and L. Liu. A global learning algorithm for a RBF network. *Neural Networks*, 12:527–540, 1999.