# A Fuzzy-Possibilistic Fuzzy Ruled Clustering Algorithm for RBFNNs Design

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

Department of Computer Architecture and Technology
Universidad de Granada. Spain

**Abstract.** This paper presents a new approach to the problem of designing Radial Basis Function Neural Networks (RBFNNs) to approximate a given function. The presented algorithm focuses in the first stage of the design where the centers of the RBFs have to be placed. This task has been commonly solved by applying generic clustering algorithms although in other cases, some specific clustering algorithms were considered. These specific algorithms improved the performance by adding some elements that allow them to use the information provided by the output of the function to be approximated but they did not add problem specific knowledge. The novelty of the new developed algorithm is the combination of a fuzzy-possibilistic approach with a supervising parameter and the addition of a new migration step that, through the generation of RBFNNs, is able to take proper decisions on where to move the centers. The algorithm also introduces a fuzzy logic element by setting a fuzzy rule that determines the input vectors that influence each center position, this fuzzy rule considers the output of the function to be approximated and the fuzzy-possibilistic partition of the data.

## 1 Introduction

The problem of approximating a given function using an RBFNN $\mathcal{F}$ 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. RBFNNs have been widely used to solve this problem because of their capability to approximate any function [6, 13].

An RBFNN $\mathcal{F}$ with fixed structure to approximate an unknown function $F$ with $d$ variables and one output is defined as:

$$\mathcal{F}(\boldsymbol{x}_k; C, R, \Omega) = \sum_{i=1}^{m} \phi(\boldsymbol{x}_k; \boldsymbol{c}_i, r_i) \cdot \Omega_i \tag{1}$$

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 $\phi(\boldsymbol{x}_k; \boldsymbol{c}_i, r_i)$ 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, 15].

The first step of initialization of the position of the RBF centers has been commonly carried out using clustering algorithms [11, 17]. Classical clustering algorithms have been used to solve classification task where the objective is to identify to classify the input data assigning a discrete set of predefined labels, however, in the function approximation problem, the output of the function belongs to a continuous interval. The output of the function is an important element that must be considered when initializing the centers, they should be concentred where the output is more variable since these areas require more RBFs to be modelled.

This paper proposes a new algorithm to solve the task of the initialization of the centers. It is based on a mixed fuzzy-possibilistic supervised approach that, with the use of fuzzy logic and problem specific knowledge, will provide an adequate placement of the centers. The consequence of this is that the final RBFNN obtained after following the rest of the steps dependent from the center initialization, approximates the given function with a smaller error than if other algorithms were applied.

## 2 Previous Clustering Algorithms

This section describes several clustering algorithms that have been used to determine the centers when designing RBFNNs for function approximation problems.

### 2.1 Fuzzy C-means (FCM)

This algorithm presented in [1] uses a fuzzy partition of the data where an input vector belongs to several clusters with a membership value. It defines an objective distortion function to be minimized is:

$$J_h(U, C; X) = \sum_{k=1}^{n} \sum_{i=1}^{m} u_{ik}^h \|\boldsymbol{x}_k - \boldsymbol{c}_i\|^2 \tag{2}$$

where $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ are the input vectors, $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_m\}$ are the centers of the clusters, $U = [u_{ik}]$ is the matrix where the degree of membership is established by the input vector to the cluster, and $h$ is a parameter to control the degree of the partition fuzziness. After applying the least square method to minimize the function in Equation 2, we get the equations to reach the solution trough an iterative process.

### 2.2 Fuzzy Possibilistic C-means (FPCM)

In this approach developed in [12], a combination of a fuzzy partition and a possibilistic partition is presented. The point of view of the authors is that the membership value of the fuzzy partition is important to be able to assign a hard

label to classify an input vector, but at the same time, it is very useful to use the typicality (possibility) value to move the centers around the input vectors space, avoiding undesirable effects due to the presence of outliers. The distortion function to be minimized is:

$$J_h(U, C, T; X) = \sum_{k=1}^{n} \sum_{i=1}^{m} (u_{ik}^h + t_{ik}^{h_2}) \|\boldsymbol{x}_k - \boldsymbol{c}_i\|^2 \tag{3}$$

with the following constraints: $\sum_{i=1}^{m} u_{ik} = 1 \ \forall k = 1...n$ and $\sum_{k=1}^{n} t_{ik} = 1 \ \forall i = 1...m$.

Let $T = [t_{ik}]$, then, the constraint shown above requires each row of $T$ to sum up to 1 but its columns are free up to the requirement that each column contains at least one non-zero entry, thus, there is a possibility of input vectors not belonging to any cluster. The main improvement in comparison with the FCM algorithm is that, since there are some input vectors that can be outliers, the membership function for them will be small not forcing all clusters to share them.

### 2.3 Possibilistic Centers Initializer (PCI)

This algorithm [10] adapts the algorithm proposed in [9] using a mixed approach between a possibilistic and a fuzzy partition, combining both approach as it was done in [16]. The objective function to be minimized is defined as:

$$J_h(U^{(p)}, U^{(f)}, C, W; X) = \sum_{k=1}^{n} \sum_{i=1}^{m} (u_{ik}^{(f)})^{h_f} (u_{ik}^{(p)})^{h_p} D_{ikW}^2 + \sum_{i=1}^{m} \eta_i \sum_{k=1}^{n} (u_{ik}^{(f)})^{h_f} (1 - u_{ik}^{(p)})^{h_p} \tag{4}$$

where $u_{ik}^{(p)}$ is the possibilistic membership of $x_k$ in the cluster $i$, $u_{ik}^{(f)}$ is the fuzzy membership of $\boldsymbol{x}_k$ in the cluster $i$, $D_{ikW}$ is the weighted euclidean distance, $\eta_i$ is a scale parameter that is calculated by: $\eta_i = \frac{\sum_{k=1}^{n} (u_{ik}^{(f)})^{h_f} \|\boldsymbol{x}_k - \boldsymbol{c}_i\|^2}{(u_{ik}^{(f)})^{h_f}}$

This function is obtained by replacing de distance measure in the FCM algorithm by the objective function of the PCM algorithm, obtaining a mixed approach. The scale parameter determines the relative degree to which the second term in the objective function is compared with the first. This second term forces to make the possibilistic membership degree as big as possible, thus, choosing this value for $\eta_i$ will keep a balance between the fuzzy and the possibilistic memberships.

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 [14] as an extension of Hard C-means.

## 3 Fuzzy-Possibilistic Centers Initializer (FPCI)

This section introduces the proposed algorithm, describing first the elements that characterize the algorithm: the supervising parameter, the migration process and the distortion function. Then the main body of the algorithm, which integrates the previous components, is introduced.

### 3.1 Supervising parameter

Classical clustering techniques do not consider the output of the function that will be approximated by the RBFNN, so there is a need of introducing an element that influences the way in which the centers are placed. This effect can be achieved by changing the similarity criteria that defines the clusters.

In [8], it was presented the concept of expected output of a center which is an hypothetic value for each center that gives a position on the output axis. This element makes possible to compare the output of the centers and the real output of the input vectors. The calculation of the supervising parameter $w$ is performed by:

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

where $o_i$ represents the expected output of a center. The parameter $w$ is used in the algorithm in order to modify the similarity criteria during the clustering process, reducing the distance between a center and an input vector if they have similar outputs. Therefore, the distance calculation is performed by:

$$D_{ikW} = d_{ik} \cdot w_{ik} \tag{6}$$

where $d_{ik}$ is the euclidean distance between the center $\boldsymbol{c}_i$ and the input vector $\boldsymbol{x}_k$.

### 3.2 Distortion function

As classical clustering algorithms, the proposed algorithm defines a distortion function that has to be minimized. The distortion function is based in a fuzzy-possibilistic approach as it was presented in [12] although it contains the elements required for a supervised learning. The function is:

$$J_h(U, C, T, W; X) = \sum_{k=1}^{n} \sum_{i=1}^{m} (u_{ik}^{h_f} + t_{ik}^{h_p}) D_{ikW}^2 \tag{7}$$

restricted to the constraints: $\sum_{i=1}^{m} u_{ik} = 1 \ \forall k = 1...n$ and $\sum_{k=1}^{n} t_{ik} = 1 \ \forall i = 1...m$.

The solution is reached by an alternating optimization approach where all the elements defined in the function to be minimized (Equation 7) are actualized iteratively. For the new algorithm proposed in this paper, the equations are:

$$u_{ik} = \frac{1}{\sum_{j=1}^{m} \left( \frac{t_{ik}^{(h_p-1)/2} D_{ikW}}{t_{jk}^{(h_p-1)/2} D_{jkW}} \right)^{\frac{2}{h_f-1}}} \qquad t_{ik} = \frac{1}{1 + \left( \frac{D_{ikW}}{\eta_i} \right)^{\frac{1}{h_p-1}}} \tag{8}$$

$$c_i = \frac{\sum_{k=1}^{n} t_{ik}^{h_p} u_{ik}^{h_f} x_k w_{ik}^2}{\sum_{k=1}^{n} t_{ik}^{h_p} u_{ik}^{h_f} w_{ik}^2} \qquad o_i = \frac{\sum_{k=1}^{n} t_{ik}^{h_p} u_{ik}^{h_f} y_k d_{ik}^2}{\sum_{k=1}^{n} t_{ik}^{h_p} u_{ik}^{h_f} d_{ik}^2} \tag{9}$$

These equations are obtained by differentiating $J_h(U, T, C, W; X)$ (Equation 7) with $u_{ik}$, $t_{ik}$, $\boldsymbol{c}_i$ and $o_i$, therefore the convergence is guaranteed.

### 3.3 Migration step

The migration step will allow the algorithm to escape from local minima found during the iterative process, that minimizes the distortion function, and to work with non continuous data sets.

The idea of a migration step was developed in [14], and it was also used in [10], but the migration the new algorithm uses is totally different to the previous ones because it adds problem specific knowledge.

Since the problem is to design an RBFNN to approximate a given function, an RBFNN is generated to decide where to move the centers and if the migration step was correct. The migration is accepted if the distortion function has been decreased or if the error from the output of the RBFNN generated after the migration is smaller than the error from the RBFNN generated before.

To design the RBFNN on each migration step, since the centers are already placed, it is only needed to get the values for each radius, which is calculated using the KNN algorithm with K=1. Once we have these two parameters, the weights for each RBF are computed optimally by solving a linear equation system [7].

Each center is assigned an utility value that represents how important is that center in the current partition. The utility of a center is calculated with the following algorithm:

$error_i$ = Accumulated error between the hypothetic output of the center $\boldsymbol{c}_i$ and the output of the vectors that belong to that center
$numvec_i$ = number of vectors that belong to $\boldsymbol{c}_i$

**for each center** $\boldsymbol{c}_i$ **with** $i = 1...m$
  **if** $error_i < mean(error)$
    select $\boldsymbol{c}_i$
  **else**
    **if** $numvec_i < mean(numvec)$
      select $\boldsymbol{c}_i$

```
        end
    end
    distortion_i= error_i · numvec_i
end
utility_i=distortion_i/mean(distortion)
```

This process excludes the centers that own a big number of input vectors and have a small error from those vectors from the set of centers to be source or destination of the migration. Once the pre-selection of the centers has been done, the center that will be migrated is the one with minimum utility and the center receiving the other center is the one with maximum utility.

Due to the addition of the parameter $w$ both the fuzzy and the possibilistic membership functions loose their interpretability. The combination of them using a fuzzy rule will allow us to decide which input vectors are owned by each cluster. This process is necessary in order to choose which centers will be migrated.

```
for each center c_i
    for each input vector x_k
        distance=||x_k − c_i||
        if (distance< near)
            B_ik = (u_ik · t_ik)/(distance+1 + w_ik)
        else
            B_ik = (u_ik · t_ik)/(distance+1)
        end
    end
end
```

The aim of the rule is that if an input vector is *near* a center, the value of the output of that input vector must influence the position of the center, otherwise, the difference between the expected and the real output is not as important since the center is far. This method alleviates the problem of an input vector being owned by a center just because they have the same output values. In order to set a value for the threshold *near*, the columns of the matrix $B$ must be normalized. An empirical value of 0.05 has been shown to provide a good performance as it will be shown in the experiments.

The complete algorithm for the migration is shown in Figure 1.

### 3.4 General Scheme

The FPCI algorithm follows the scheme shown in Figure 2. 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 conditions so

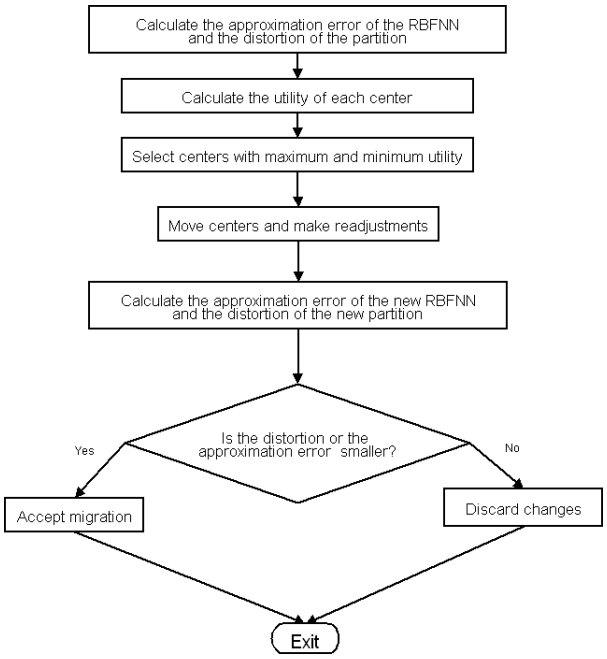


**Fig. 1.** FPCI migration algorithm.

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.

## 4 Experimental Results

The target function that will be used in this experiment was presented in [3] and it has been used as a benchmark in [4, 5]. The function is defined as:

$$f_1(x_1, x_2) = 1.9\left(1.35 + e^{x_1}\sin\left(13\left(x_1 - 0.6\right)^2\right)e^{-x_2}\sin\left(7x_2\right)\right), \quad x_1, x_2 \in [0, 1] \tag{10}$$

Figure 3 shows the original function and the training data set that was obtained by selecting randomly 400 points from the original function. The number of test input vectors was 1900, such a big difference between the size of the data sets is to show the generalization abilities of the RBFNN that, learning from a reduced number of examples, are able to generalize and obtain a good approximation of the complete function.

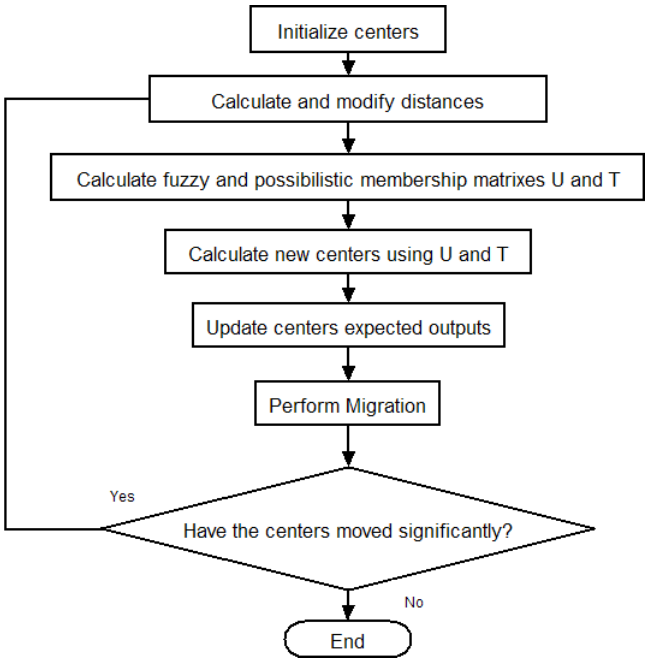


**Fig. 2.** General scheme of the FPCI algorithm.

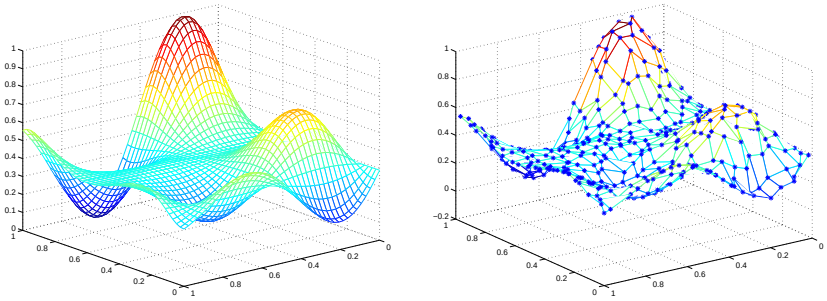

**Fig. 3.** Target function $f_1$ and the training set.

Once the centers were placed by all the algorithms, the radii of the RBFs were calculated using the KNN heuristic with k=1, and then, a local search algorithm (Levenberg-Marquardt) was applied to obtain a fine tuning of these parameters. Table 1 shows the approximation errors obtained after designing the RBFNNs using the algorithms described above.

**Table 1.** Mean and Standard Deviation of the approximation error (NRMSE) for function $f_1$ for the training and test set

| Clusters | **FCM** | **FPCM** | **PCI** | **FPCI** |
|---|---|---|---|---|
| 8 | 0.155(0.039) | 0.163(0.037) | 0.204(0.042) | 0.145(0) |
| 9 | 0.160(0.019) | 0.136(0.035) | 0.112(0.023) | 0.130(0) |
| 10 | 0.108(0.030) | 0.118(0.035) | 0.102(0.011) | 0.085(0) |
| 11 | 0.128(0.073) | 0.077(0.012) | 0.071(0.015) | 0.062(0) |
| 8 | 0.153(0.040) | 0.160(0.036) | 0.245(0.119) | 0.143(0) |
| 9 | 0.157(0.017) | 0.135(0.034) | 0.129(0.061) | 0.131(0) |
| 10 | 0.107(0.027) | 0.116(0.032) | 0.100(0.011) | 0.087(0) |
| 11 | 0.126(0.050) | 0.078(0.012) | 0.071(0.015) | 0.062(0) |

These results show how the best performance is obtained by the proposed algorithm. In general, the other three algorithms have the drawback of lack of robustness. The results obtained using the FPCM show how the combination of a fuzzy partition with a possibilistic one leads to a better performance. The PCI algorithm uses both types of partitions with the addition of an element to consider the output of the function to be approximated, this element makes possible to obtain better results than the other non-supervised algorithms. The new algorithm combines the advantages of using the same type of fuzzy possibilistic partition of the FPCM algorithm with the addition of the supervising parameter and the migration step. These three elements together lead to obtain the smallest approximation errors when is compared with the other algorithms.

Regarding the use of RBFNN to approximate a function, the results the algorithms present show how well they perform when interpolating the modelled function because for all the algorithms, the training and test errors do not differ significantly.

## 5 Conclusions

Within the different methods to solve the function approximation problem, RBFNNs have shown their good performance although their design still represents a difficult problem. The RBF center initialization is the first stage in the design, and the success of the rest of the stages depends critically in the way the centers are placed. This paper presents a new algorithm that combines several elements of previous algorithms used to place the centers, adding a new element that uses problem specific knowledge. This new element consists in the design

of an RBFNN that helps to decide where to move the centers in the migration stage. The migration incorporates also a fuzzy rule that alleviates the problem of loosing interpretability in the supervised method, making more adequate the election of the centers that should be migrated.

## 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. V. Cherkassky and H.Lari-Najafi. Constrained topological mapping for nonparametric regression analysis. *Neural Networks*, 4(1):27–40, 1991.
4. J. H. Friedman. Multivariate adaptive regression splines (with discussion). *Annals of Statistics*, 19:1–141, 1991.
5. J.H. Friedman. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981.
6. A. Gersho. Asymptotically Optimal Block Quantization. *IEEE Transanctions on Information Theory*, 25(4):373–380, July 1979.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. J. Park and J. W. Sandberg. Universal approximation using radial basis functions network. *Neural Computation*, 3:246–257, 1991.
14. G. Patanè and M. Russo. The Enhanced-LBG algorithm. *Neural Networks*, 14(9):1219–1237, 2001.
15. 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.
16. J. Zhang and Y. Leung. Improved possibilistic C–means clustering algorithms. *IEEE Transactions on Fuzzy Systems*, 12:209–217, 2004.
17. Q. Zhu, Y. Cai, and L. Liu. A global learning algorithm for a RBF network. *Neural Networks*, 12:527–540, 1999.