A Possibilitic Approach to RBFN Centers Initialization

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 techniques have always been oriented to solve classification and pattern recognition problems. This clustering techniques have been used also to initialize the centers of the Radial Basis Function (RBF) when designing an RBF Neural Network (RBFNN) that approximates a function. Since classification and function approximation problems are quite different, it is necessary to design a new clustering technique specialized in the problem of function approximation. In this paper, a new clustering technique it is proposed to make the right initialization of the centers of the RBFs. The novelty of the algorithm is the employment of a possibilistic partition of the data, rather than a hard or fuzzy partition as it is commonly used in clustering algorithms. The use of this kind of partition with the addition of several components to use the information provided by the output, allow the new algorithm to provide better results and be more robust than the other clustering algorithms even if noise exits in the input data.

1 Introduction

The 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 $\boldsymbol{x}_k \in \mathbb{R}^d$. To solve this problem, Radial Basis Function Neural Networks (RBFNN) are used because of their capability as universal approximators [5, 11].

The design of a RBFNN is performed by following several steps, in which the initialization of the centers of the RBFs is the first one. The use of a clustering algorithm is a common solution for a first initialization of the centers [9, 15]. These clustering algorithms were designed for classification problems [8] instead of for the function approximation problem so the results they provide can be improved significantly. Clustering algorithms try to classify the set of input data assigning a set of predefined labels, however, in the function approximation problem, the output of the function belongs to a continuous interval. Clustering algorithms do not use the information provided by the function output ignoring the variability of the function. In the function approximation problem, the information provided by the output of the function to be approximated is needed to obtain a correct placement of the centers. Centers must be placed in the areas where the function is more variable and therefore, it will be needed more RBFs to be able to model the variations of the function, meanwhile, in the areas where the function is not that variable, less centers will be needed to approximate the function.

It is necessary to design a clustering algorithm oriented to the function approximation problem in order to make a right initialization of the RBFs centers. In this paper, a new algorithm to solve this task is proposed. It is based on a mixed fuzzy-possibilistic approach improving results, as it will be shown in the experiments section, in comparison with traditional clustering algorithms and clustering algorithms designed specifically for the function approximation problem.

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:

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

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 [3, 13]. 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 Previous Clustering Algorithms

This section will describe several clustering algorithms that have been used to determine the centers when designing RBFNN for functional approximation problems.

3.1 Fuzzy C-means (FCM)

This algorithm uses a fuzzy partition of the data where an input vector belongs to several clusters with a membership value. The objective 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 minimum square method to minimize the function in Equation 2 [2], we get the equations to reach the solution trough and iterative process:

$$u_{ik} = \left(\sum_{j=1}^{m} \left(\frac{D_{ikA}}{D_{jkA}} \right)^{\frac{2}{h-1}} \right)^{-1} \tag{3}$$

$$\boldsymbol{c}_i = \frac{\sum\limits_{k=1}^{n} u_{ik}^{h} \boldsymbol{x}_k}{\sum\limits_{k=1}^{n} u_{ik}^{h}} \tag{4}$$

where $D_{jkA} = ||\boldsymbol{x_k} - \boldsymbol{c_j}||^2$, and $|| \cdot ||$ is the inner product norm in $\mathbb{R}^d$. Using this iterative process we calculate matrix U from matrix C or vice versa starting from a random initialization of any of the matrices: ($C_{t-1} \rightarrow U_t \rightarrow C_t$ or $U_{t-1} \rightarrow C_t \rightarrow U_t$). The stop criteria is usually considered as $\|C_{t-1} - C_t\| < \mathit{threshold}$, this is, when the centers do not move significantly.

3.2 Improved Possibilistic C-means (IPCM)

The Possibilistic C-means [10] determines a possibilistic partition of the data, in which a possibilistic membership measures the absolute degree of typicality of a point to a cluster. This approach is robust because noisy points will not affect significantly the possibilistic partition as they would in a fuzzy partition. This algorithm tends to find identical clusters [1] so an improved version of this algorithm is proposed in [14]. This new approach combines a fuzzy partition with a possibilistic partition determining the following function to be minimized:

$$J_h(U^{(p)}, U^{(f)}, C; X) = \sum_{k=1}^{n} \sum_{i=1}^{m} (u_{ik}^{(f)})^{h_f} (u_{ik}^{(p)})^{h_p} d_{ik}^2 + \sum_{i=1}^{m} \eta_i \sum_{k=1}^{n} (u_{ik}^{(f)})^{h_f} (1 - u_{ik}^{(p)})^{h_p} \tag{5}$$

where:

- $u_{ik}^{(p)}$ is the possibilistic membership of x_k in the cluster i.
- $u_{ik}^{(f)}$ is the fuzzy membership of x_k in the cluster i.
- h_p and h_f are the weighting exponents for the possibilistic and the fuzzy membership functions.
- η_i is a scale parameter that is calculated using:

$$\eta_i = \frac{\sum\limits_{k=1}^{n} (u_{ik}^{(f)})^{h_f} (u_{ik}^{(p)})^{h_p} d_{ik}^2}{(u_{ik}^{(f)})^{h_f} (u_{ik}^{(p)})^{h_p}} \tag{6}$$

As in the previous algorithms, an iterative process drives to the solution.

3.3 Clustering for Function Approximation (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:

$$\frac{\sum\limits_{j=1}^{m} \sum\limits_{\boldsymbol{x}_k \in C_j} \|\boldsymbol{x}_k - \boldsymbol{c}_j\|^2 \omega_{kj}}{\sum\limits_{j=1}^{m} \sum\limits_{\boldsymbol{x}_k \in C_j} \omega_{kj}} \tag{7}$$

where ω_{kj} weights the influence of each input vector in the final position a center.

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 partitioned, the centers and their estimated outputs have to be updated, this process is done iteratively using the equations shown below:

$$\boldsymbol{c}_j = \frac{\sum\limits_{\boldsymbol{x}_k \in C_j} \boldsymbol{x}_k \omega_{kj}}{\sum\limits_{\boldsymbol{x}_k \in C_j} \omega_{kj}} \quad o_j = \frac{\sum\limits_{\boldsymbol{x}_k \in C_j} F(\boldsymbol{x}_k) \omega_{kj}}{\sum\limits_{\boldsymbol{x}_k \in C_j} \omega_{kj}} \quad . \tag{8}$$

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 [12] as an extension of Hard C-means and the objective is to find an optimal vector quantization where each center makes an equal contribution to the total distortion [5].

3.4 Fuzzy Clustering for Function Approximation (FCFA)

The FCFA algorithm [6, 7] is based in the CFA and FCM. The main difference between CFA and FCFA is the application of fuzzy logic to the algorithm. FCFA performs a fuzzy partition of the data and iterates in the same way as FCM does, thus, it improves the speed of the algorithm in comparison with CFA because it only needs one step of actualization instead of an internal loop. The algorithm considers the input data as the input data vectors concatenated with their outputs. Proceeding like this, the expected output of a center correspond with its last coodinate. FCFA also makes a modification in the migration process performing a pre-selection of the centers to be migrated modifying the criteria used to decide if a center should be migrated or not. This pre-selection is based on a fuzzy ruled system.

4 Possibilistic Centers Initializer (PCI)

The new algorithm proposed uses a possibilistic partition and a fuzzy partition, combining both approach as it was done in [14]. The objective function $J_h(U^{(p)}, U^{(f)}, C, W; X)$ to be minimized is defined as:

$$\sum_{k=1}^{n}\sum_{i=1}^{m}(u_{ik}^{(f)})^{h_f}(u_{ik}^{(p)})^{h_p}D_{ikW}^2 + \sum_{i=1}^{m}\eta_i\sum_{k=1}^{n}(u_{ik}^{(f)})^{h_f}(1-u_{ik}^{(p)})^{h_p} \tag{9}$$

where:

- $u_{ik}^{(p)}$ is the possibilistic membership of x_k in the cluster i.
- $u_{ik}^{(f)}$ is the fuzzy membership of x_k in the cluster i.
- D_{ikW} is the weighted euclidean distance.
- η_i is a scale parameter that is calculated using:

$$\eta_i = \frac{\sum_{k=1}^{n}(u_{ik}^{(f)})^{h_f}d_{ik}^2}{(u_{ik}^{(f)})^{h_f}} \tag{10}$$

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 η_i will keep a balance between the fuzzy and the possibilistic memberships. When calculating η_i, the distance is not weighted because the estimated outputs, in the initialization of the algorithm, are not appropriate to calculate w.

4.1 Weighting Parameter

To make the output of the target function to be approximated influence the placement of the centers, it is necessary to change the similarity criteria in the clustering process. The combination of the possibilistic and the fuzzy approach has to be influenced by the output of the function to be minimized. In order to do this, the euclidean distance used as the similarity criteria will be weighted using the parameter w. The calculation of w is obtained by:

$$w_{kj} = |F(\boldsymbol{x}_k) - o_j| \tag{11}$$

where o_j represents the expected output of a center, this is, the hypothetic position of the center c_j in the output axis. The euclidean distance d_{ij} between a center i and an input vector will be weighted using the following equation:

$$D_{ijW} = d_{ij} \cdot w_{ij}. \tag{12}$$

Proceeding this way, D_{ijW} will be small if the center is near the input vector and they have similar output values. Thus a center can own input vectors that are far from him if they have similar output values, and will not own input vectors that, even though are near the center, have a big difference in the output values. This will allow the algorithm to place more centers where the output of the target function to be approximated is more variable.

4.2 Iterative process

As in all the previous algorithms based on a fuzzy or a possibilistic partition, the solution is reached by an alternating optimization approach where all the elements defined in the function to be minimized (Equation 9) are actualized iteratively. For the new algorithm proposed in this paper, the equations are:

$$u_{ik}^{(p)} = \frac{1}{1 + \left(\frac{D_{ikW}}{\eta_i}\right)^{\frac{1}{h_p-1}}} \tag{13}$$

$$u_{ik}^{(f)} = \frac{1}{\sum_{j=1}^{m} \left(\frac{(u_{ik}^{(p)})^{(h_p-1)/2} D_{ikW}}{(u_{jk}^{(p)})^{(h_p-1)/2} D_{jkW}}\right)^{\frac{2}{h_f-1}}} \tag{14}$$

$$c_i = \frac{\sum_{k=1}^{n} (u_{ik}^{(p)})^{(h_p)} (u_{ik}^{(f)})^{(h_f)} x_k w_{ik}^2}{\sum_{k=1}^{n} (u_{ik}^{(p)})^{(h_p)} (u_{ik}^{(f)})^{(h_f)} w_{ik}^2} \tag{15}$$

$$o_i = \frac{\sum_{k=1}^{n} (u_{ik}^{(p)})^{(h_p)} (u_{ik}^{(f)})^{(h_f)} Y_k d_{ik}^2}{\sum_{k=1}^{n} (u_{ik}^{(p)})^{(h_p)} (u_{ik}^{(f)})^{(h_f)} d_{ik}^2} \tag{16}$$

These equations are obtained by differentiating $J_h(U^{(p)}, U^{(f)}, C, W; X)$ (Equation 9) with respect $u_{ik}^{(p)}$, $u_{ik}^{(f)}$, c_i and o_i. This approach is the same followed in FCM, IPCM and the convergence is guaranteed.

4.3 General Scheme

The PCI algorithm follows the scheme shown below to place the centers:

```
Initialize $C_1$ using Fuzzy C-means
Initialize $O_1$ using fuzzy membership function
**Do**
    Calculate $w$
    Calculate the distance between $C_i$ and $X$
    Calculate the new $U_i$
    Calculate the new $C_i$ from $U_i$
    Calculate the new $O_i$ from $C_i$
    i=i+1
**While**(abs($C_{i-1}$-$C_i$<$threshold$)
```

As in [14], the FCM algorithm is used to find a proper start point, making the algorithm much more robust. In the first step, the expected output o_i of each center c_i will correspond with the output value of the input vector that belongs to c_i with the highest membership value.

5 Experimental Results

The experiment will consists in the approximation of the function f_1 represented in Fig. 1, that has been generated using an RBFNN with the following parameters:

centers	radii	weights
119.029	15.595	-3.748
175.106	4.279	-1.016
39.796	16.350	3.855
71.087	9.164	1.857
93.030	37.518	4.095

200 points uniformly distributed were generated between the interval [1,399] using the RBFNN described above with a gaussian activation function . The function has been designed to show the importance of the output variability on the target function. In this function there is an interval where the variability of the output is high and another interval where the function is almost constant. This fact will make the initialization of the centers very important because the centers will have to be concentrated in the areas where the function is more variable.

Once the clustering algorithm were executed and the corresponding RBFNNs were generated, the normalized root mean squared error (NRMSE) has been used in order to determine the quality of the approximation.

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.

FCM, IPCM, CFA, FCFA and PCI were executed several times providing the results shown in Table 1. In this table it is shown the approximation error right after the initialization procedure. In Table 2 are shown the results after applying a local search algorithm (Levenberg-Marquardt) to make a fine tune of the RBF centers and radii. The results are depicted in Fig. 2.

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

Clusters	**FCM**	**IPCM**	**CFA**	**FCFA**	**PCI**
4	0.633(0.002)	0.592(2E-4)	0.595(0.022)	0.361(0.035)	0.444(1E-4)
5	0.619(0.001)	0.584(0.001)	0.515(0.035)	0.345(0.034)	0.343(0.001)
6	0.544(0.003)	0.537(0.001)	0.412(0.041)	0.312(0.046)	0.287(3E-4)

Table 2. Mean and Standard Deviation of the approximation error (NRMSE) for function f_1 after local search algorithm.

Clusters	**FCM**	**IPCM**	**CFA**	**FCFA**	**PCI**
4	0.182(0.011)	0.187(0.015)	0.186(0.012)	0.149(0.030)	0.100(1E-4)
5	0.308(0.001)	0.256(0.026)	0.178(0.045)	0.104(0.043)	0.048(0.022)
6	0.187(0.069)	0.125(0.045)	0.098(0.071)	0.082(0.019)	0.0002(3E-4)

The results show how there is a need of using specific clustering algorithms to initialize the centers when designing an RBFNN for the functional approximation problem. Classical clustering do not use the information provided by the output of the target function to be approximated. Thus they are not able to detect the areas where the output is more variable, providing poor results. The approximation error decreases significantly when specific clustering algorithms designed for this task are employed.

The new algorithm proposed in this paper has shown that can performs better than all the previous clustering. PCI not only performs better, but is more robust because the standard deviations of the solutions are very small, indicating that it finds the same configuration on each execution of the algorithm. The previous clustering algorithms for functional approximation were not too robust, so it can be appreciated how a mixed fuzzy-possibilistic approach can solve that problem.

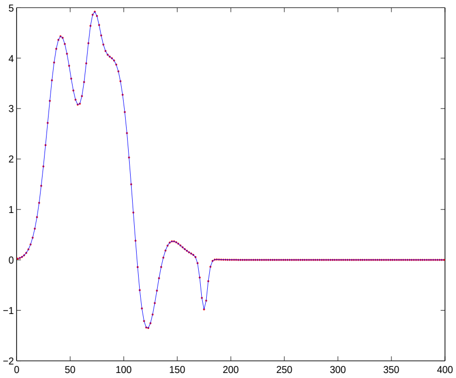
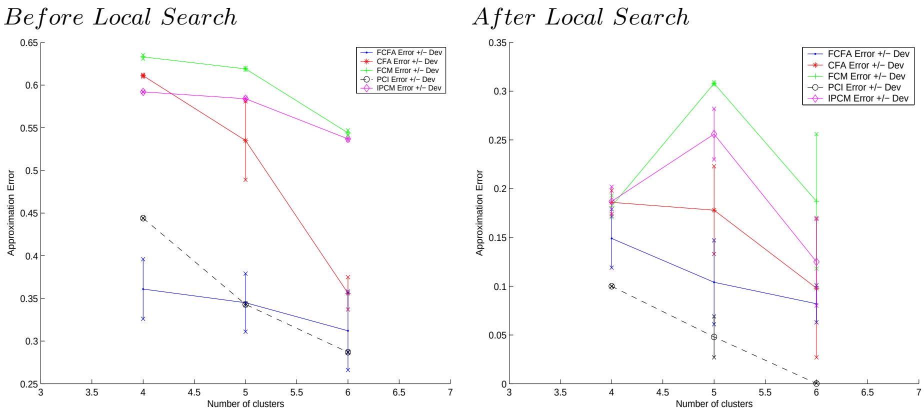


Fig. 1. Target Function (blue line) and training set (red dots).


Before Local Search

After Local Search

Fig. 2. Mean and Standard Deviation of the approximation error (NRMSE) before and after local search algorithm.

6 Conclusions

RBFNNs provides good results when they are used for functional approximation problems. The first step in the design of those RBFNNs was performed by clustering algorithms. In this paper, a new clustering algorithm designed specifically for the center initialization task has been presented. The novelty of this algorithm in comparison with other clustering algorithms designed for this task, is the use of a mixed possibilistic and fuzzy approach when making the partition of the data. The results shown how this approach allow to make a better initialization of the centers, providing better results when approximating functions.

References

1. M. Barni, V. Capellini, and A. Mecocci. Comments on 'A posibilistic approach to clustering'. *IEEE Transactions on Fuzzy Systems*, 4:393–396, June.
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. A. Guillén, I. Rojas, J. Gonzalez, H. Pomares, and L. J. Herrera. Clustering based algorithm for rfbs centers initialization. In *XI SIGEF Congress, Techniques and Methodologies for the Information and Knowledge Economy*, pages 44–63, Reggio Calabria-Messina, November 2004.
7. A. Guillén, I. Rojas, J. Gonzalez, H. Pomares, L.J. Herrera, and A. Prieto. Using a new clustering algorithm to design rbf networks for functional approximation problem. In *Proceedings of the Learning'04 International Conference*, pages 19–24, Elche, October 2004.
8. J. A. Hartigan. *Clustering Algorithms.* New York: Wiley, 1975.
9. N. B. Karayannis 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.
10. R. Krishnapuram and J. M. Keller. A Possibilistic Approach to Clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, May 1993.
11. J. Park and J. W. Sandberg. Universal approximation using radial basis functions network. *Neural Computation*, 3:246–257, 1991.
12. G. Patanè and M. Russo. The Enhanced-LBG algorithm. *Neural Networks*, 14(9):1219–1237, 2001.
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. *Int. J. 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.
15. Q. Zhu, Y. Cai, and L. Liu. A global learning algorithm for a RBF network. *Neural Networks*, 12:527–540, 1999.