Output Value-Based Initialization for Radial Basis Function Neural Networks

Alberto Guillén, Ignacio Rojas, Jesús González, Héctor Pomares and L.J. Herrera
University of Granada, Department of Computer Technology and Architecture

November 11, 2005

Abstract.

The use of Radial Basis Function Neural Networks (RBFNNs) to solve functional approximation problems has been addressed many times in the literature. When designing an RBFNN to approximate a function, the first step consists of the initialization of the centers of the RBFs. This initialization task is very important because the rest of the steps are based on the positions of the centers. Many clustering techniques have been applied for this purpose achieving good results although they were constrained to the clustering problem. The next step of the design of an RBFNN, which is also very important, is the initialization of the radii for each RBF. There are few heuristics that are used for this problem and none of them use the information provided by the output of the function, but only the centers or the input vectors positions are considered. In this paper, a new algorithm to initialize the centers and the radii of an RBFNN is proposed. This algorithm uses the perspective of activation grades for each neuron, placing the centers according to the output of the target function. The radii are initialized using the center's positions and their activation grades so the calculation of the radii also uses the information provided by the output of the target function. As the experiments show, the performance of the new algorithm outperforms other algorithms previously used for this problem.

Keywords: RBF, neural networks, RBFNN, initialization, function approximation, radii, centers, LaTeX

1. Introduction

Formally, the functional 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 $\mathcal{F}$ so $y_k \simeq \mathcal{F}(\vec{x}_k)$. 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.

For the functional approximation problem, Artificial Neural Networks (ANNs) have the capacity of estimating a model from empirical data. Therefore, they can act as interpolators so they return an output for any input, even if this input was not in the training data.

ANNs are computational systems inspired by the structure, processing method, and learning ability of a biological brain. In the artificial system, a neuron corresponds to a processing unit which, given an input, it uses a function to compute an output. These systems are largely used in many areas of science and engineering applications involving classification, optimization and function approximation (Elanayar and Shin, 1994).

Within the ANNs, it exists a variety of types depending on the activation function or the number of layers. One of them is the RBFNN that was introduced by Broomhead and Lowe in (Broomhead and Lowe, 1988). These RBFNNs have the capability of approximating any given function (Chen and Chen, 1995; Park and Sandberg, 1991; Poggio and Girosi, 1990), therefore they have been successfully applied to many problems related with nonlinear regression and function approximation (Karayiannis et al., 2003; Schilling

© 2007 *Kluwer Academic Publishers. Printed in the Netherlands.*

et al., 2001; Sigitani et al., 1999). The main characteristic of this kind of ANN is that it has a two-layer, fully connected network in which each neuron implements a gaussian function as follows:

$$\phi(\vec{x};\vec{c}_i,r_i) = \exp\left(\frac{-||\vec{c}_i-\vec{x}||^2}{r_i^2}\right) \ , i=1,...,m \tag{1}$$

where $\vec{x}$ is an input vector, $\vec{c}_i$ is the center and r_i is the radius of the i-th RBF. Gaussian functions are very appropriate for function approximation because they are continuous, differentiable, provide a softer output, and improve the interpolation capabilities (Bors, 2001; Rojas et al., 1998).

The output layer implements a weighted sum of all the outputs from the hidden layer:

$$\mathcal{F}(\vec{x};C,R,\Omega) = \sum_{i=1}^{m} \phi(\vec{x};\vec{c}_i,r_i)\cdot\Omega_i \tag{2}$$

where Ω_i are the output weights, which show the contribution of a hidden layer to the corresponding output unit and can be obtained optimally by solving a linear equation system. After the initialization of the centers and the radii, once all the previous elements have been calculated, a local search algorithm can be applied to make a fine tuning of the model. Other methodologies to train RBFNNs have been presented in (Karayiannis, 1999; Randolph-Gips and Karayiannis, 2003), where reformulated and cosine RBFs are presented. The methodology is based on a gradient descent that does not use any initialization technique. Genetic Algorithms have been also applied to design RBFNNs to approximate functions (González et al., 2003).

This paper presents the Output Value-Based Initializer (OVI) algorithm, which calculates the initial values for the centers and radii of an RBFNN according to the output values of the target function. To accomplish this task, the Output Value-Based Initializer (OVI) algorithm calculates a hypothetic activation value for each center in order to place it and, once the centers are placed, the radii are calculated using this activation value.

The remaining of this paper, in order to make it self-contained, is organized as follows: Section 2 describes the details of the centers and radii initialization problem, Section 3 describes the OVI algorithm, Section 4 shows the experimental results and in Section 5 conclusions are drawn.

2. Statement of the problem

The design of an RBFNN to approximate a given function requires a sequence of steps. The first one is the initialization of the RBF centers. This initialization task has been traditionally solved using clustering algorithms (Baraldi and Blonda, 1999a; Baraldi and Blonda, 1999b; Uykan et al., 2000). For example in (Karayiannis and Mi, 1997) and in (Zhu et al., 1999) the initial model to design the RBFNN is created using clustering. Traditional clustering algorithms have been used to initialize centers (Bezdek, 1981; Duda and Hart, 1973; Patanè and Russo, 2001), but they were designed to solve classification problems. In these kind of problems, the input data have to be assigned to a pre-defined

set of labels. This does not fit well enough when working with function approximation since a continuous interval of real numbers is defined to be the output of the input data.

Related to the previous point, when classifying, if a label is not assigned correctly, the error could be greatly increased. In the functional approximation problem is not like that, if the generated output value is near the real output, the error may not increase too much.

The initialization of the centers of an RBFNN, designed to approximate a given function, is not a trivial task and it should be done using the information provided by the output of the target function, otherwise, the centers could be placed in areas where the target function is easy to model, making them useless.

The target function can have an output with a high variability in some areas and with other areas where the variability is not that high. Because of this, if many centers are placed in the areas where the function is not too variable, the ANN will not have enough neurons to approximate the variable zones. Classical clustering techniques ignore the output of the function, just considering the input data vectors, but there are some other clustering techniques that use the information provided by the target function output. These algorithms are known as $input - output$ clustering algorithms and some of them have been applied to the initialization of the RBF centers for function approximation problem (González et al., 2002; Pedrycz, 1998; Runkler and Bezdek, 1999).

In this framework, the need of using specific algorithms designed to initialize the centers instead of using traditional clustering algorithms is obvious. Several algorithms have been designed specifically for this task, but they all were inspired in previous clustering algorithms (Guillén et al., 2005b; Guillén et al., 2005a; Leski, 2003; Pedrycz, 1998). This motivated the design of the algorithm proposed in this paper. This new algorithm uses the information provided by the output of the function to be approximated so it concentrates the centers through the areas with a high variability, placing less centers where the function is not too variable.

The design of an RBFNN also requires the initialization of the radii for each RBF. Usually this step has been performed using heuristics as the K-Nearest Neighbor (KNN) algorithm (Moody and Darken, 1989) or the Closest Input Vector (CIV) algorithm (Karayiannis and Mi, 1997). These heuristics ignore the output of the function when initializing the radii, they only use the center positions or the input vector positions. The new proposed algorithm makes an initialization using the center positions and the activation grade of each input vector depending on its output, this allows the algorithm to improve the results in comparison with the classical heuristic employed for this task.

3. OVI: An Output Value-Based Initializer

This section contains the description of the new algorithm that initializes both the centers and the radii of an RBFNN to approximate a function. First, the general scheme and the several components of the algorithm are briefly presented, then, in the following subsections, these elements are described in detail.

3.1. OVI general scheme

The proposed algorithm minimizes a distortion function using an alternating optimization procedure where the activation grades for each neuron and the neuron positions are updated on each iteration. A migration step has been added to the algorithm to ensure that each neuron reduces the distortion in an equitable way by being activated by several input vectors. Thanks to this migration step it is possible to avoid local minima that could be found in the iteration process. At each iteration, an updating step and a migration are performed. This is repeated until the positions of the centers do not change significantly. Once the positions of the centers are fixed, matrix A contains the final activation grades that are used in the calculation of the radii for each RBF. The general scheme that the proposed algorithm follows is shown in Figure 1.

The algorithm starts with a pre-initialization of the centers using the Fuzzy C-means algorithm as it has been done in the literature (Zhang and Leung, 2004) to obtain the maximum robustness. If the algorithm starts from a random initialization of the centers, it achieves similar configurations with a very small variation but, after applying a local search algorithm to adjust the centers and the radii, this small variation in the positions of the centers can affect the final result in a serious way.

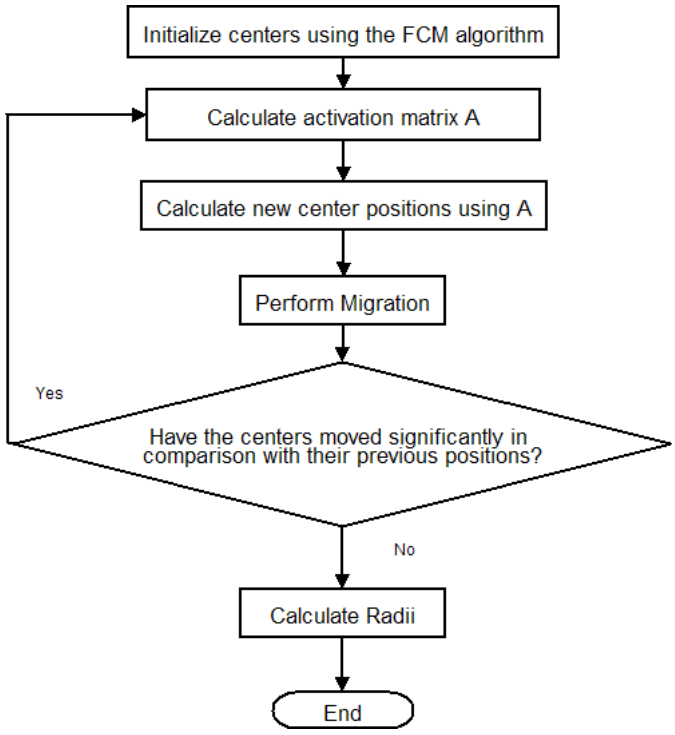

Figure 1. OVI General Scheme

3.2. **Objective Distortion Function**

As it was mentioned in the statement of the problem, the initialization of the centers of an RBFNN that will approximate a function must be done carefully. The centers should be more concentrated in the areas where the function is more variable. Thus, the output of the function to be approximated must be considered as important information when initializing the centers.

Since the centers have to be placed in the input vectors space, it is necessary to include in the algorithm an element that allows it to keep a reference between the output and the input space. The objective is to keep a balance between the coordinates in the input space and the output it generates. If the two elements described, the output of the function and its coordinates in the input space, are considered, it is possible to define a distortion function:

$$\delta = \sum_{k=1}^{n} \sum_{i=1}^{m} D_{ik}^2 a_{ik}^l |Y_k^p| \tag{3}$$

where D_{ik} represents the Euclidean distance from a center c_i to an input vector $\vec{x}_k$, a_{ik} is the activation value that determines how important the input vector $\vec{x}_k$ is for the center $\vec{c}_i$, l is a parameter to control the degree of overlapping between the neurons, Y_k is the preprocessed output of the input vector $\vec{x}_k$, and p allows the influence of the output when initializing the centers to increase or decrease.

This distortion function combines the information provided by a coordinate in the input vector space and its corresponding output in such a way that, if a neuron is near an input vector and the output of the input vector is high, the activation value of that neuron respect that input vector will be high.

The original output of the function is preprocessed before starting the algorithm. The idea is to think of the output space as a flat surface ($Y(\vec{x}_k) = 0$), where some of the values of this surface have been modified by an n-dimensional element, obtaining y_k. From this, we can assume that the most common value of the target function is constant and equal 0. The preprocessing of the output is performed by making the most frequent output value equal 0. This can be easily performed by subtracting the fuzzy mode to each one of the output values y_k. Once this situation is achieved, all the most common values that have an output around 0 will not affect the distortion significantly.

Let $A = [a_{ik}]$, be the activation matrix, this matrix is restricted by the following conditions:

- $\sum_{i=1}^{m} a_{ik} = 1 \ , \forall k = 1...n$
- $0 < \sum_{k=1}^{n} a_{ik} < n \ , \forall i = 1...m.$

If a_{ik} is high, it means that the neuron (center i) is significantly activated by the input vector $\vec{x}_k$. The first restriction forces every input vector to activate at least one neuron, 1 being the maximum grade of activation. The second restriction forces each neuron to share every input vector with the others, so all neurons will have an activation value

even though this can be very small. Thanks to this restriction, it is easier to maintain an overlap between the RBFs allowing the ANN to make a softer interpolation of the target function.

The objective is to minimize the distortion function so the neurons are only activated when it is necessary. The minimum of the distortion function will be reached when the distance from a center to an input vector with a high output value is the smallest. If this occurs, the value for a_{ik} will be the highest for this center, but very small for the other centers. An alternating optimization procedure is used to minimize the distortion function. Let $C = [\vec{c}_i]$ be the matrix containing the coordinates of all the centers. On each iteration, the algorithm calculates the matrix A from the matrix C supposing that C is the matrix where a minimum for the distortion function is obtained. Once a new A is calculated it is possible to recalculate the matrix C using the previous A. This iterative process ends when any of the two matrices do not change significantly.

The equations to calculate these matrices at each iteration are:

$$a_{ik} = \left(\sum_{j=1}^{m} \left(\frac{D_{ik}}{D_{jk}} \right)^{\frac{2}{l-1}} \right)^{-1} \tag{4}$$

$$\vec{c}_i = \frac{\sum_{k=1}^{n} a_{ik}^l \vec{x}_k |Y_k^p|}{\sum_{k=1}^{n} a_{ik}^l |Y_k^p|} \tag{5}$$

where D_{ik} is the Euclidean distance between $\vec{x}_k$ and $\vec{c}_i$, Y_k is the preprocessed output of the target function, and l is the parameter to control the degree of overlapping between the neurons.

These equations are obtained by applying Lagrange multipliers and calculating the derivative of the objective function, therefore, convergence is guaranteed (see Appendix 1).

3.3. **Migration step**

The iteration of the steps presented above makes a gradient descent to reach a minimum of the distortion function defined in (3). It uses an iterative process that can get stuck in a local minima, specially when working with non continuous data sets. This problem can be easily solved introducing a migration step as it was done in (Patanè and Russo, 2001).

The migration step aims to find the optimal vector quantization where each center makes equal contribution to the total distortion (Gersho, 1979). The migration algorithm iterates many times, moving centers on each iteration, until each center contributes equally to the distortion function defined to be minimized. The distortion never increases so the convergence is not altered by the migration step.

The migration step provides more robustness to the algorithm since it helps to find the same configuration independently of the starting point of the centers.

Each center is assigned an individual distortion value that is defined as:

$$\text{dist}_i = \sum_{k=1}^{n} D_{ik}^2 a_{ik}^l |Y_k^p| \tag{6}$$

so the sum of all the distortions for each center is equal to the global distortion function that it is being minimized. Using this distortion measure, it is possible to assign an utility for each center. A center is useful if it increases the global distortion in comparison with the others. The utility of a center is defined as:

$$\text{ut}_i = \frac{\text{dist}_i}{\overline{\text{dist}}} \tag{7}$$

where $\overline{\text{dist}}$ is the mean distortion of the centers.

After calculating the utility, there will be useful centers, with utility > 1, and centers with utility ≤ 1 that are not useful. In order to make sure that each center contributes equally to the total distortion, a center with a low utility will be moved near a center with utility greater than 1. After the movement of the center, this modification will be accepted if the total distortion has been decreased, otherwise, the moved center will be marked as processed and another one will be considered. This will be repeated until all the centers with utility greater than 1 have been processed. The migration algorithm is represented in Figure 2.

There is a fixed criteria to select the centers to be migrated: the destination of the migration will be the center with maximum utility and the center that will be moved will be the one with the smallest utility.

When a center is migrated, several adjustments must be made. The standard deviation ($\vec{\sigma}$) of the input vectors belonging to the destination center is calculated. Then, if $\vec{c}_i$ is the original position of the destination center, the new position will be $\vec{c}_i + \vec{\sigma}$ and the position of the migrated center will be $\vec{c}_i - \vec{\sigma}$. Once the new positions are calculated, the whole matrix A is recalculated using the new center positions, from this, the new distortion value can be found.

3.4. Calculating the Radii of the RBFs

As it was commented in Section 2, after the initialization of the centers, the values of the radii of the gaussian RBFs has to be chosen. In (Park and Sandberg, 1993; Park and Sandberg, 1991) it was demonstrated that if all the radii have the same value, the network can still be an universal approximator, but, in (Benoudjit and Verleysen, 2003; Musavi et al., 1992) it is shown that if each center can define its own value for the radius, the performance of the network can be increased. Therefore, it is important to assign a correct value for the radii.

The OVI algorithm calculates how much an input vector will activate a neuron in function of its output value. From this, the value of the radius can be set as the distance to the farthest input vector that activates a center. In order to do that, a threshold has to be established to decide when an input vector activates or does not activate a center. To illustrate this, the function f_1 (Figure 3) defined as:

$$f_1(x) = \frac{\sin(2\pi x)}{e^x}, \; x \in [0, 10] \tag{8}$$

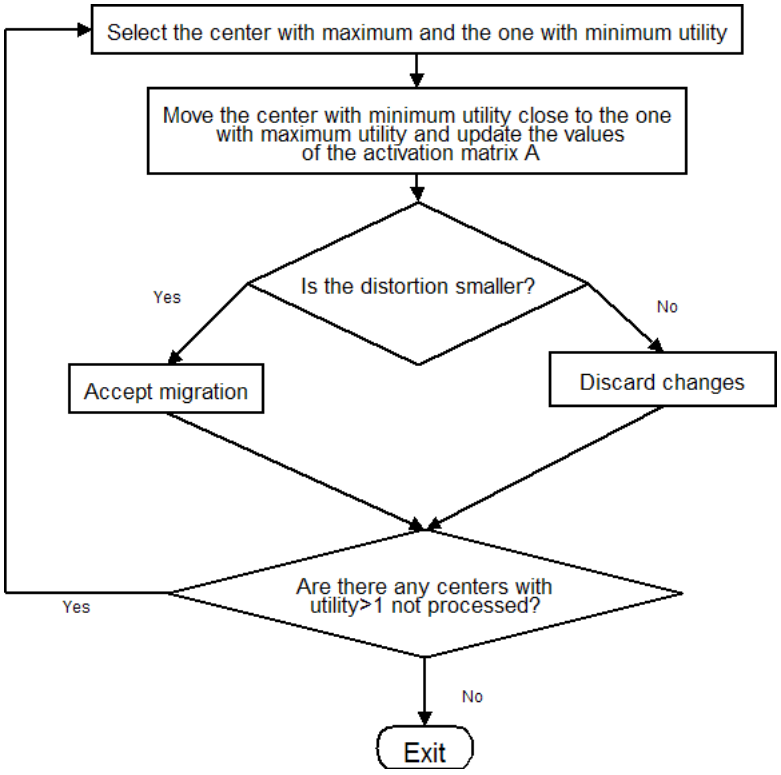

Figure 2. Migration algorithm

has been used. This function has been designed in such a way that shows the importance of selecting a proper value for the centers and the radii. In some areas, the RBFs need large values for the radii and in other areas, the radii have to be small. This fact makes the election of a proper value for the radii very difficult. After the execution of the algorithm, the activation grades are stored in the A matrix. These values are depicted graphically in Figure 4. In this figure, it is easy to appreciate which input vectors activate each neuron. Using the values of the A matrix, an activation threshold ($\vartheta_{overlap}$) that allows us to calculate the distance to the farthest input vector that activates a neuron can be established. The proposed algorithm selects the radius for each center independently of the positions of the other centers, unlike in the KNN heuristic (Moody and Darken, 1989), and it allows to maintain an overlap between the RBFs, unlike in the CIV heuristic (Karayiannis and Mi, 1997).

Each radius is defined as:

$$r_i = \max\{\ D_{ik}\ /\ a_{ik} > \vartheta_{overlap}\ ,\ 1 \leq i \leq m\ ,\ 1 \leq k \leq n\ \}\ , \tag{9}$$

this is, r_i is equal to the Euclidean distance between $\vec{c}_i$ and the farthest input vector $\vec{x}_k$ that activates $\vec{c}_i$ with a value over the threshold $\vartheta_{overlap}$. The selection of a threshold makes the algorithm more flexible, because it can increase or decrease the degree of overlap between the RBFs. This is illustrated in Figure 5 which represents the centers

and their radii using several values for $\vartheta_{overlap}$ after the execution of the algorithm using the function f_1. For small values of $\vartheta_{overlap}$ there is a big overlap between the RBFs but this overlap decreases when the threshold becomes higher.

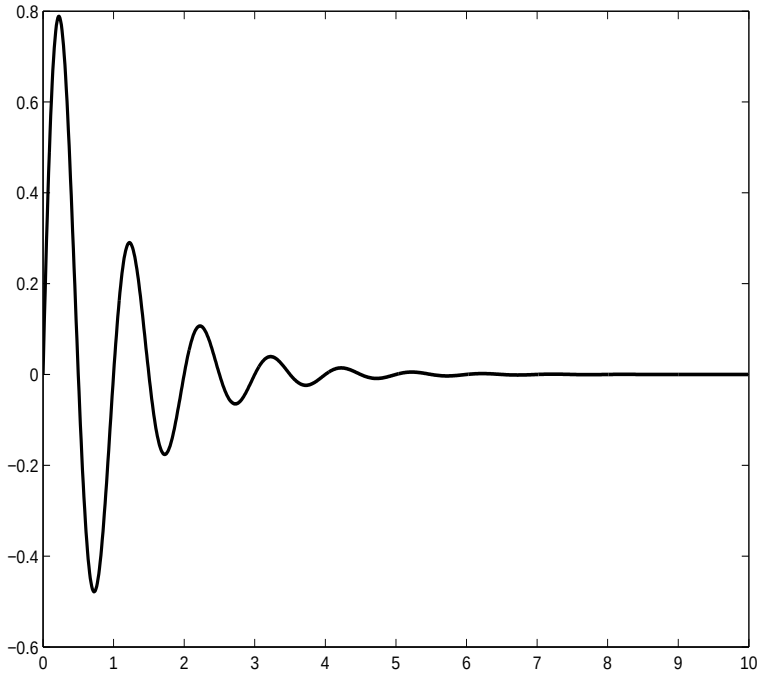

Figure 3. Target function f_1

4. Experiments

In this section the behavior of the algorithm in function of several parameters is analyzed and it is compared with other algorithms that have been used to initialize the centers of RBFNNs or the complete RBFNN.

The first experiment shows how much the parameter p controls the influence of the output of the target function. In this experiment, the OVI algorithm is compared with traditional heuristics used to calculate the values of the radii.

In the second experiment, the proposed algorithm is compared with several classical clustering techniques and with some specific clustering algorithms designed for the center initialization task.

A two dimensional function is approximated in the third experiment. The algorithm proposed is compared with several paradigms that used this benchmark function to show the accuracy of their approximations.

Finally, in the last experiment, a more complex example of function approximation is used. The input data consists in the the benchmark data originated from Box and Jenkins concerning an identification of a gas oven. The performance of the proposed

algorithm is compared against another clustering algorithm, the Generalized Weighted Conditional Fuzzy Clustering (GWCFC).

4.1. Experiment 1: Influence of p

The proposed algorithm has one parameter, p, that allows it to regulate how much the output will influence the initialization of the centers. This experiment shows several executions using different values for this parameter. The target is the unidimensional function f_1, which was presented before. The parameter p allows us to concentrate more centers in the variable interval as it is shown in Figure 6. In this figure the centers positions after several executions of the algorithm varying the value of p are represented. The approximation errors resulting from these executions after generating the RBFNN are shown in Table I. In the first column, both radii and centers are obtained using the OVI algorithm. In the second and third columns, the centers were taken from the output of the OVI algorithm and the radii were calculated using other common heuristics, such as KNN and CIV. This shows the radii initialization that the OVI algorithm performs in comparison with the classical heuristic, showing that the proposed algorithm significantly outperforms the other two heuristics independently of the value of the parameter p.

Another remarkable characteristic is the robustness of the results obtained using the proposed algorithm, that allows to obtain the same configuration from different initializations, providing very similar approximation errors on several executions for the same parameters.

Table I. Average and standard deviation of the approximation error (NRMSE) for function f_1 using several values of p and several heuristics for the radii initialization.

p	**OVI**	**OVI-KNN**	**OVI-CIV**
1	0.044 (0.000)	0.167 (0.021)	0.107 (0.017)
1.5	0.041 (0.000)	0.131 (0.001)	0.153 (0.000)
2.5	0.105 (0.000)	0.151 (0.000)	0.151 (0.000)
4.5	0.102 (0.000)	0.155 (0.000)	0.231 (0.000)

As it is shown in Figure 6, the bigger the value of p is, the closer the centers are placed in the interval where the variability of the function is higher. The results show that, for small values of p, the approximations are good but when p is highly augmented, the error increases. This is because all the centers are concentrated in the interval where the function is more variable and no centers are placed in the other areas therefore, these areas can not be modelled. An advisable value of p could be 1.5, this value has been used for all the following experiments allowing the algorithm to obtain appropriate results. The election of the value of p makes the algorithm more flexible if there is a chance of a human expert determining its value.

4.2. Experiment 2: 1D Function

This experiment compares the proposed algorithm with the traditional clustering algorithms that have been used to initialize the centers and with the two *input − output* algorithms CFA and FCFA. The target function f_1, previously introduced, is the one that was used to compare CFA and FCFA with other clustering techniques.

Table II shows the approximation errors after generating the RBFNN using the KNN algorithm with K=1 for all the algorithms except for the OVI algorithm. In the results, the approximation errors of the OVI algorithm are smaller than the error obtained using the other algorithms. It is very important to remark on the robustness of the algorithm, that starting from a random point, it is able to reach the same configuration, as it is the most robust of all the other compared algorithms.

Table II. Average and standard deviation of the approximation error (NRMSE) for function f_1 using several numbers of neurons (m).

m	**HCM**	**FCM**	**ELBG**	**CFA**	**FCFA**	**OVI**
6	0.759 (0.263)	0.783 (0.317)	0.152 (0.122)	0.090 (0.008)	0.103 (0.060)	0.041 (0.031)
7	0.344 (0.281)	0.248 (0.115)	0.111 (0.072)	0.081 (0.016)	0.069 (0.017)	0.029 (0.012)
8	0.227 (0.368)	0.150 (0.162)	0.093 (0.064)	0.053 (0.028)	0.048 (0.031)	0.014 (0.001)
9	0.252 (0.386)	0.300 (0.160)	0.073 (0.057)	0.056 (0.027)	0.027 (0.024)	0.010 (0.001)
10	0.087 (0.100)	0.285 (0.334)	0.064 (0.039)	0.047 (0.015)	0.011 (0.015)	0.007 (0.003)

4.3. Experiment 3: 2D Function

This experiment consists in the approximation of the two dimensional function f_5 (Figure 7) that was originally proposed in (Cherkassky and H.Lari-Najafi, 1991) and is defined as:

$$f_5(x_1, x_2) = 42.659(0.1 + x_1(0.05 + x_1^4 - 10x_1^2x_2^2 + 5x_2^4)) \quad x_1, x_2 \in [-0.5, 0.5] \qquad (10)$$

This function has been used as a benchmark in several papers. In (Cherkassky et al., 1996) the approximation was done using multilayer perceptrons (MLP) and in (Pomares, 2000) it was used an algorithm for identification of rule-based fuzzy systems. The target function was also approximated in (Friedman, 1981) using the Projection Pursuit (PP), and a Multivariate Adaptive Regression Splines (MARS) algorithm was used in (Friedman, 1991). A multi-objective GA was used in (González et al., 2003) to design RBFNNs to approximate this function.

All the algorithms have been compared used a training set of 400 inputs generated randomly from a grid of 20×20 (Figure 8). The approximations have been tested with a data set of 961 input vectors generated randomly over a grid of 31×31.

The results are shown in Table III, where all the previous algorithms described in the paragraphs above are compared with the proposed algorithm. The OVI algorithm improves the results significantly providing smaller approximation errors. Another aspect

of note is the robustness of the proposed algorithm, that finds the same configuration independently of the starting positions of the centers as it occurred in the previous experiments.

4.4. Experiment 4: Time series prediction

In this experiment, the benchmark data originated from Box and Jenkins (Box and Jenkins, 1976), concerning an identification of a gas oven is used. To obtain the data, air and methane were delivered into a gas oven to obtain a mixture of gases containing CO_2. There were originally 296 data points $y(t), u(t)$, from $t = 1$ to $t = 296$. The value $y(t)$ is the output CO_2 concentration and $u(t)$ is the input gas flow rate. The objective is to predict $y(t)$ based on $\{y(t-1), y(t-2), y(t-3), y(t-4), u(t-1), u(t-2), u(t-3), u(t-4), u(t-5),\ u(t-6)\}$.
This reduces the number of effective data points to 290 and set the dimension of the problem to 10.

The OVI algorithm is compared with the GWCFC algorithm (Leski, 2003). The RBFNN proposed in (Leski, 2003) is a gaussian with six neurons. In (Leski, 2003), the algorithm is compared also with FCM. The GWCFC algorithm defines three contexts with two centers on each one, the linguistic labels for these contexts are *big*, *medium* and *small*.

The author makes several executions modifying several parameters from the algorithm in order to obtain better results. The measure of the error is done by the RMSE, which is defined as:

$$RMSE = \sqrt{\frac{\sum_{k=1}^{n} (y_k - \mathcal{F}(\vec{x}_k))^2}{n}} \tag{11}$$

Table IV shows the approximation errors obtained by the GWCFC, FCM and OVI. The new algorithm significantly improves the results of the other two clustering algorithms, demonstrating again how it can initialize the centers and the radii of the RBFs in a better way than other algorithms proposed for this purpose. One of the main advantages of the proposed algorithm is that it does not need a human expert to tune the parameters or set membership functions as it is required in the conditional clustering.

Figure 9 shows the output value of the real function and the approximation obtained using the new algorithm.

5. Conclusions

Radial Basis Function Neural Networks have been successfully applied to the problem of function approximation. Nevertheless, The design of an RBFNN still remains as a difficult task since it requires the determination of several parameters that are crucial to obtain a good approximation.

In this paper, a new algorithm that provides an initialization of the centers and radii has been proposed. The results provided by the new algorithm outperformed the

results provided by other algorithms in the literature as it has been demonstrated in the experimental results. The success of the new algorithm resides in the idea of a generic activation function that keeps a balance between the output of the target function and the coordinates of the input vectors. The output of the algorithm is used to calculate the values of the radii for each RBF performing much better than other heuristic used for this task.

A very interesting feature of the OVI algorithm is its robustness. The standard deviations obtained are very small and, if they are compared with the other algorithms, it can be concluded that the new algorithm is much more robust than the other algorithms used in the experiments.

Appendix

The distortion function is defined as:

$$\delta = \sum_{k=1}^{n} \sum_{i=1}^{m} D_{ik}^2 a_{ik}^l |Y_k^p| \tag{12}$$

If the values of the matrix C are fixed, the values for matrix A can be calculated by Equation 4.

To obtain the Equation to update centers, fixed a matrix A and making the derivative equal zero we have that:

$$\frac{\partial \delta}{\partial c_i} = \sum_{k=1}^{n} \sum_{i=1}^{m} a_{ik}^l (2\vec{x}_k - 2\vec{c}_i)|Y_k^p| = 0 \ , \tag{13}$$

from where it is easy to obtain Equation 5.

Acknowledgements

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

References

Baraldi, A. and Blonda, P. (1999a). A Survey of Fuzzy Clustering Algorithms for Pattern Recognition – Part I. *IEEE Transactions on Systems, Man and Cybernetics*, 29(6):786–801.

Baraldi, A. and Blonda, P. (1999b). A Survey of Fuzzy Clustering Algorithms for Pattern Recognition – Part II. *IEEE Transactions on Systems, Man and Cybernetics*, 29(6):786–801.

Benoudjit, N. and Verleysen, M. (2003). On the kernel widths in radial basis function networks. *Neural Processing Letters*, 18(2):139–154.

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

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

Box, G. E. P. and Jenkins, G. M. (1976). *Time series analysis, forecasting and control*. Holden Day: San Francisco, CA.

Broomhead, D. S. and Lowe, D. (1988). Multivariate functional interpolation and adaptive networks. *Complex Systems*, 2:321–355.

Chen, T. and Chen, H. (1995). Approximation capability to functions of several variables, nonlinear functionals, and operators by radial basis function networks. *IEEE Transactions on Neural Networks*, 6(4):904–910.

Cherkassky, V., Gehring, D., and Mulier, F. (1996). Comparison of adaptative methods for function estimation from samples. *IEEE Transactions on Neural Networks*, 7:969–984.

Cherkassky, V. and H.Lari-Najafi (1991). Constrained topological mapping for nonparametric regression analysis. *Neural Networks*, 4(1):27–40.

Duda, R. O. and Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.

Elanayar, S. and Shin, Y. C. (1994). Radial Basis Function Neural Networks for approximation and estimation of nonlinear stochastic dynamic systems. *IEEE Transactions on Neural Networks*, 5:594–603.

Friedman, J. (1981). Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823.

Friedman, J. H. (1991). Multivariate adaptive regression splines (with discussion). *Annals of Statistics*, 19:1–141.

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

González, J., Rojas, I., Ortega, J., Pomares, H., Fernández, F., and Díaz, A. (2003). 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.

González, J., Rojas, I., Pomares, H., Ortega, J., and Prieto, A. (2002). A new Clustering Technique for Function Aproximation. *IEEE Transactions on Neural Networks*, 13(1):132–142.

Guillén, A., Rojas, I., González, J., Pomares, H., Herrera, L., Valenzuela, O., and Prieto, A. (2005a). Improving Clustering Technique for Functional Approximation Problem Using Fuzzy Logic: ICFA algorithm. *Lecture Notes in Computer Science*, 3512:272–280.

Guillén, A., Rojas, I., González, J., Pomares, H., and Herrera, L. J. (2005b). RBF centers initialization using fuzzy clustering technique for function approximation problems. *Fuzzy Economic Review*, 10(2).

Karayiannis, N. B. (1999). Reformulated Radial Basis Neural Networks Trained by Gradient Descent. *IEEE Transactions on Neural Networks*, 10(3):657–671.

Karayiannis, N. B., Balasubramanian, M., and Malki, H. A. (2003). Evaluation of cosine radial basis function neural networks on electric power load forecasting. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 2100–2105.

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

Leski, J. M. (2003). Generalized Weighted Conditional Fuzzy Clustering. *IEEE Transactions on Fuzzy Systems*, 11(6):709–715.

Moody, J. and Darken, C. (1989). Fast learning in networks of locally-tunned processing units. *Neural Computation*, 1(2):281–294.

Musavi, M. T., Ahmed, W., Chan, K., Faris, K., and Hummels, D. (1992). On the training of radial basis functions classifiers. *Neural Networks*, 5(4):595–603.

Park, J. and Sandberg, I. (1993). Approximation and Radial Basis Function Networks. *Neural Computation*, 5:305–316.

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

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

Pedrycz, W. (1998). Conditional Fuzzy Clustering in the Design of Radial Basis Function Neural Networks. *IEEE Transactions on Neural Networks*, 9(4):601–612.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. In *Proceedings of the IEEE*, volume 78, pages 1481–1497.

Pomares, H. (2000). Nueva metodología para el diseño automático de sistemas difusos. *PhD. dissertation, University of Granada, Spain.*

Randolph-Gips, M. M. and Karayiannis, N. B. (2003). Cosine Radial Basis Function Neural Networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 96–101.

Rojas, I., Anguita, M., Prieto, A., and Valenzuela, O. (1998). 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.

Runkler, T. A. and Bezdek, J. C. (1999). Alternating Cluster Estimation: A New Tool for Clustering and Function Approximation. *IEEE Transactions on Fuzzy Systems*, 7(4):377–393.

Schilling, R. J., Carroll, J. J., and Al-Ajlouni, A. F. (2001). Approximation of Nonlinear Systems with Radial Basis Function Neural Networks. *IEEE Transactions on Neural Networks*, 12(1):1–15.

Sigitani, T., Iiguni, Y., and Maeda, H. (1999). Image interpolation for progressive transmission by using radial basis function networks. *IEEE Transactions on Neural Networks*, 10(2):381–389.

Uykan, Z., Gzelis, C., Celebei, M. E., and Koivo, H. N. (2000). Analysis of Input–Output Clustering for Determining Centers of RBFN. *IEEE Transactions on Neural Networks*, 11(4):851–858.

Zhang, J. and Leung, Y. (2004). Improved possibilistic C–means clustering algorithms. *IEEE Transactions on Fuzzy Systems*, 12:209–217.

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

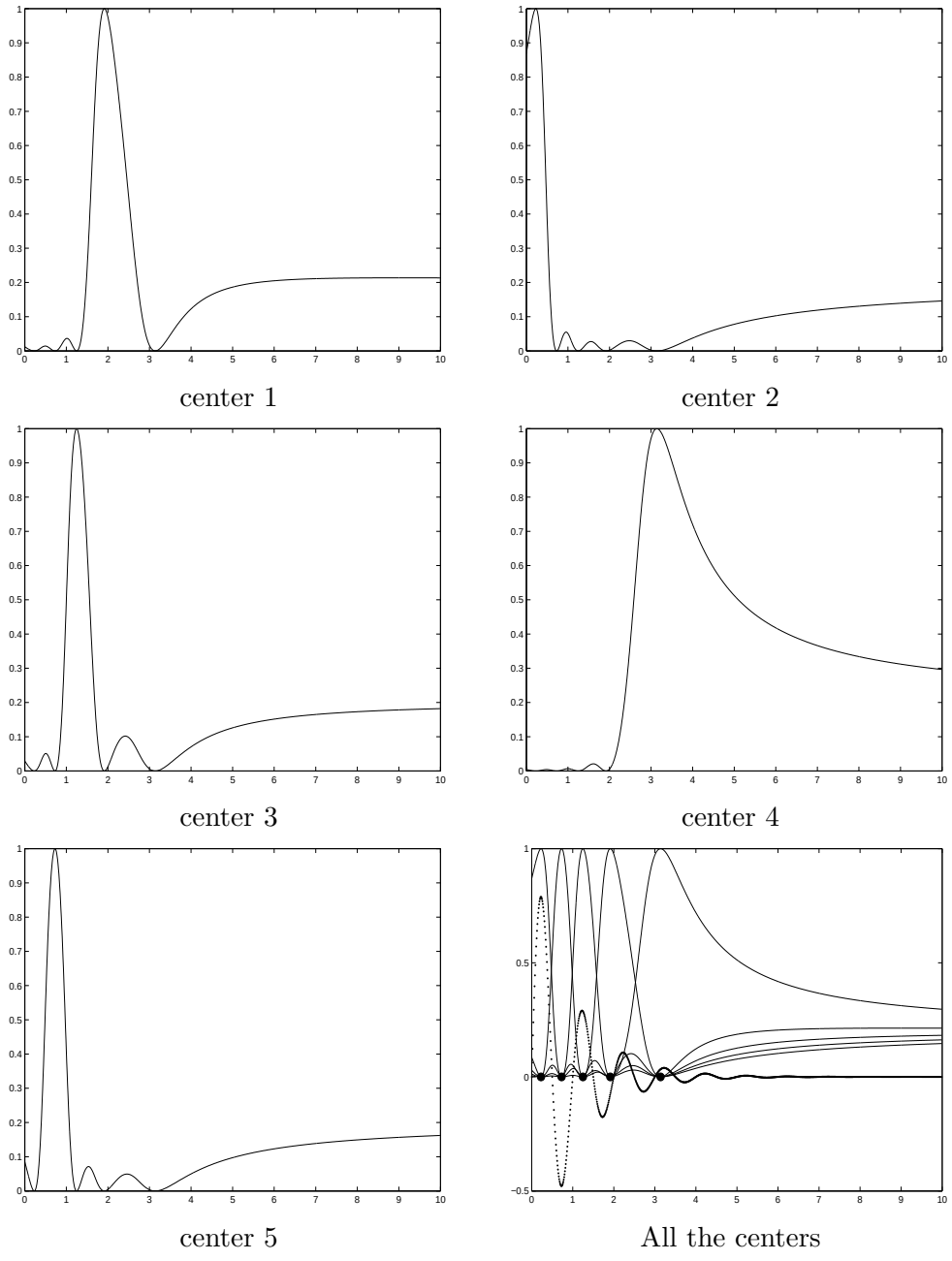

Figure 4. Activation grades for 5 centers after the execution of the algorithm using f_1

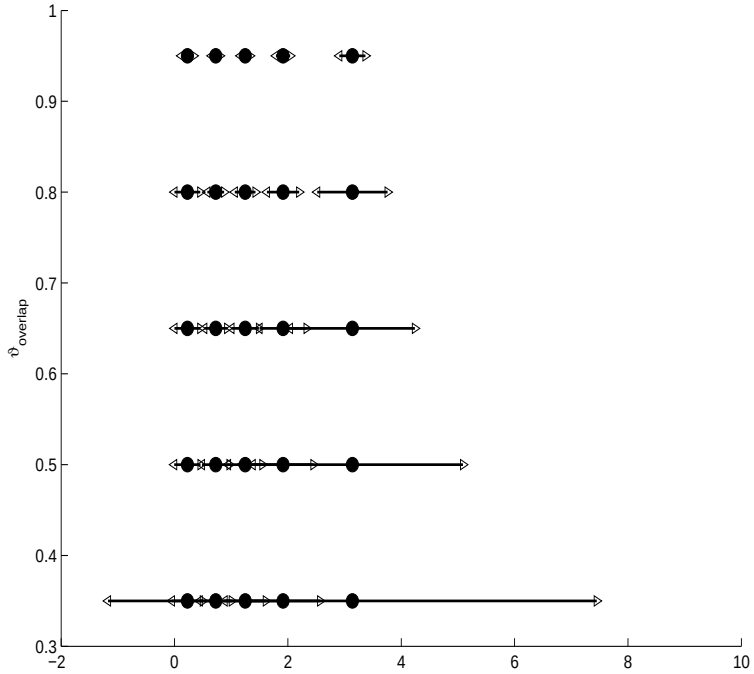

Figure 5. Centers positions and radii values using several values for $\vartheta_{overlap}$

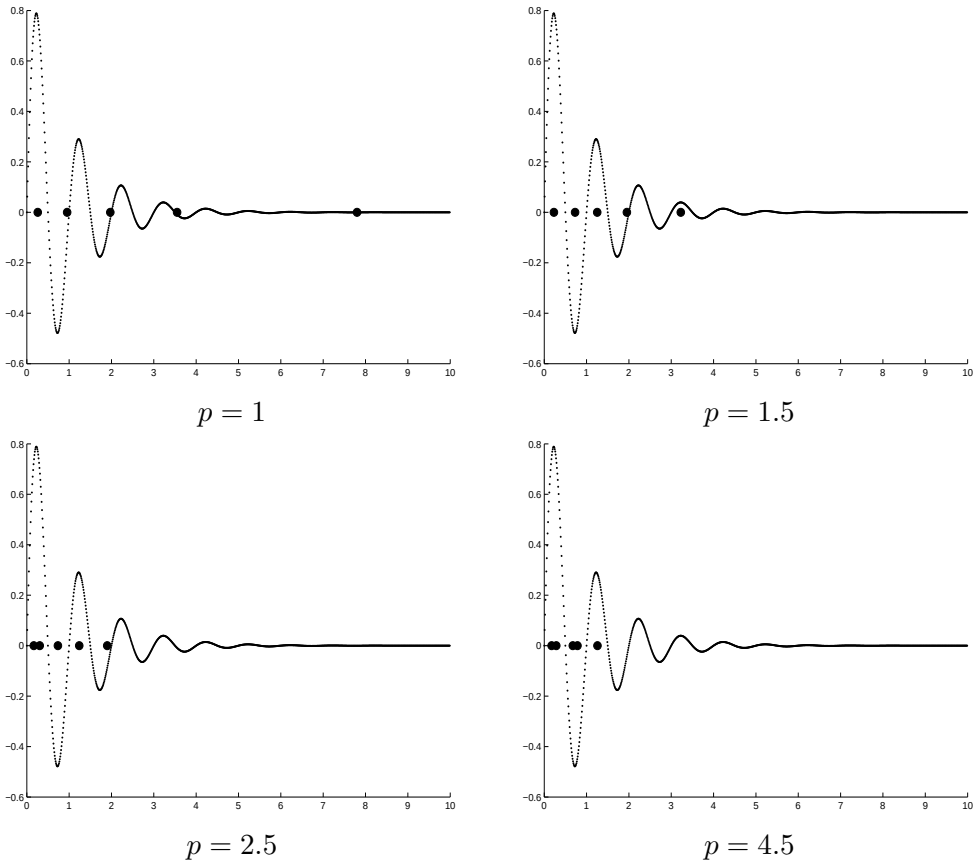

Figure 6. Centers placement using several values of p

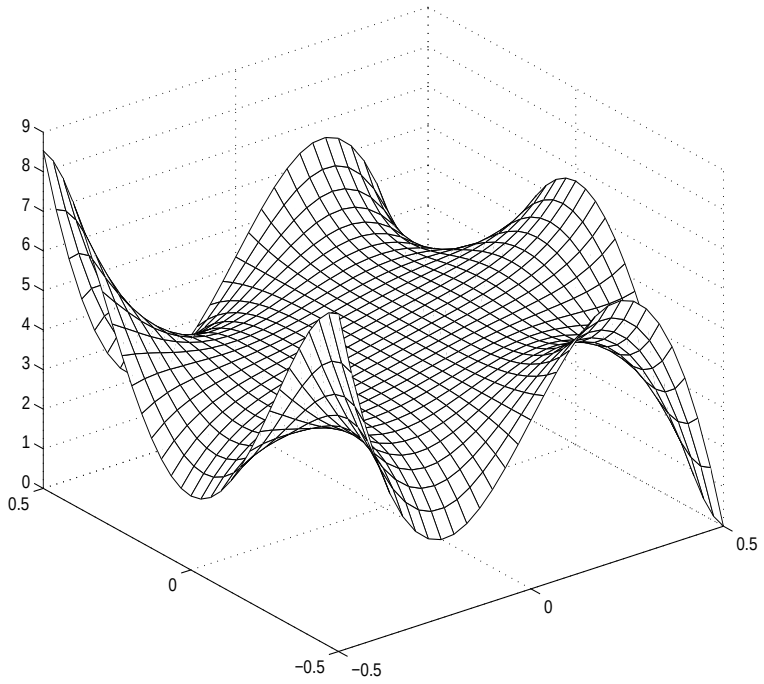

Figure 7. Target function f_5

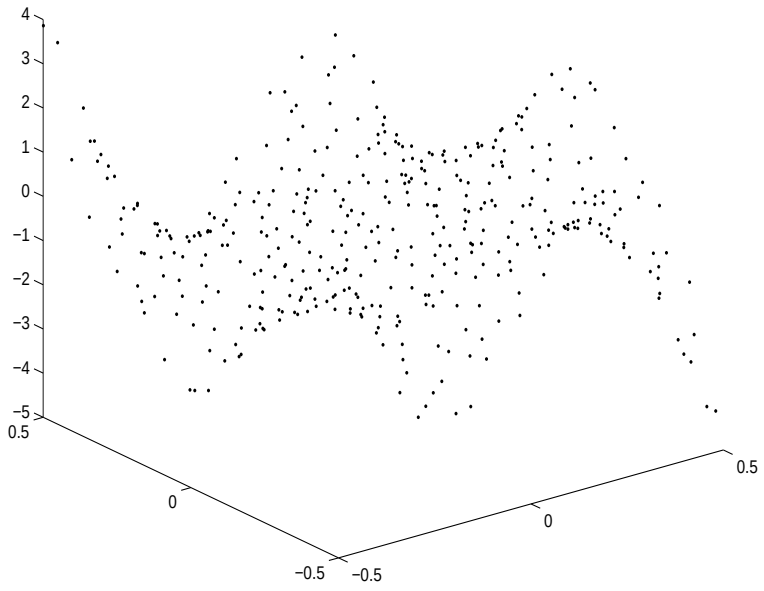

Figure 8. Training set for function f_5

Table III. Mean approximation error and standard deviation when approximating f_5 using several numbers of neurons or fuzzy rules (m).

Algorithm	m	Test NRMSE
MLP	15	0.308
PP	–	0.504
CTM	–	0.131
MARS	–	0.190
Cherkassky *et al*, 1996	40	0.038
Gonzalez,2003	13	0.023 (0.001)
	14	0.021 (0.005)
	15	0.017 (0.005)
	16	0.016 (0.009)
	17	0.015 (8.5E-5)
Proposed Algorithm	13	0.004 (0.002)
	14	0.002 (0.001)
	15	0.002 (0.001)
	16	0.002 (0.001)
	17	0.001 (0.000)

Table IV. Approximation errors (RMSE) for the Box-Jenkins data set using the FCM, GWCM and the OVI algorithms.

Algorithm	RMSE
GWFCM	0.187
FCM	0.204
OVI	0.1749

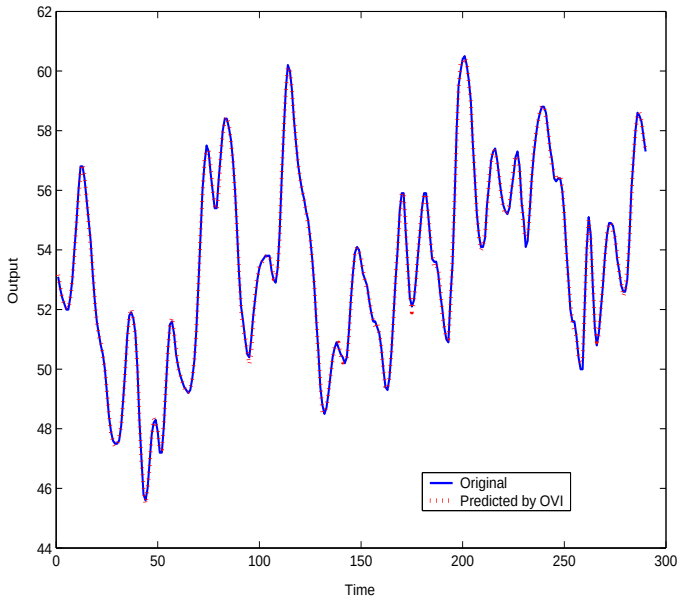

Figure 9. Output values for the Box-Jenkins data set using the OVI algorithm.