Soft Comput (2010) 14:953–971
DOI 10.1007/s00500-009-0488-z

ORIGINAL PAPER

CO^{2}RBFN: an evolutionary cooperative–competitive RBFN design algorithm for classification problems

María D. Perez-Godoy · Antonio J. Rivera · Francisco J. Berlanga · María José Del Jesus

Published online: 16 September 2009

Abstract This paper presents a new evolutionary cooperative–competitive algorithm for the design of Radial Basis Function Networks (RBFNs) for classification problems. The algorithm, CO^{2}RBFN, promotes a cooperative–competitive environment where each individual represents a radial basis function (RBF) and the entire population is responsible for the final solution. The proposal considers, in order to measure the credit assignment of an individual, three factors: contribution to the output of the complete RBFN, local error and overlapping. In addition, to decide the operators' application probability over an RBF, the algorithm uses a Fuzzy Rule Based System. It must be highlighted that the evolutionary algorithm considers a distance measure which deals, without loss of information, with differences between nominal features which are very usual in classification problems. The precision and complexity of the network obtained by the algorithm are compared with those obtained by different soft computing methods through statistical tests. This study shows that CO^{2}RBFN obtains RBFNs with an appropriate balance between accuracy and simplicity, outperforming the other methods considered.

M. D. Perez-Godoy · A. J. Rivera · F. J. Berlanga · M. J. Del Jesus (✉)
Departamento de Informática, Universidad de Jaén, Jaén, Spain
e-mail: mjjesus@ujaen.es

M. D. Perez-Godoy
e-mail: lperez@ujaen.es

A. J. Rivera
e-mail: arivera@ujaen.es

F. J. Berlanga
e-mail: berlanga@ujaen.es

Keywords Radial Basis Function Networks · Evolutionary Algorithms · Cooperative–Competitive Evolutionary Design · Classification · Fuzzy Rule Base Systems

1 Introduction

Radial Basis Function Networks (RBFNs) are one of the most important Artificial Neural Network (ANN) paradigms in the machine design field. An RBFN is a feedforward ANN with a single layer of hidden units, called radial basis functions (RBFs). The first research on neural networks based on RBFs (Broomhead and Lowe 1988; Powell 1985; Moody and Darken 1989) was carried out at the end of the 1980s. From then until now the overall efficiency of RBFNs has been proved in many areas like pattern classification (Buchtala et al. 2005), function approximation (Park and Sandberg 1991), time series prediction (Whitehead and Choate 1996) and multiple specific applications such as credit assessment (Lacerda et al. 2005), face recognition (Er et al. 2005), process control (Huang et al. 2008), medical diagnosis (Maglogiannis et al. 2008; Marcos et al. 2008), and time series forecasting (Sun et al. 2005), among others.

The main features of an RBFN are:

- It is composed of a set of RBFs which have a characteristic locally tuned response that depends on the centre and the width (radius) of each RBF.
- It has a simple topological structure, with only one hidden layer.
- It is one of the few existing interpretable ANN models and gives us the possibility of extracting rules (Jin and Sendhoff 2003; Jang and Sun 1993).

- It has universal approximation capability (Park and Sandberg 1991, 1993).

The objective of any RBFN design process is to determine centres, widths and the linear output weights connecting the RBFs to the output neuron layer. The most traditional learning procedure has two stages: first, unsupervised learning of centres and widths is used, and finally output weights are established by means of supervised learning. Clustering techniques (Pedrycz 1998) are normally used to adjust the centres. Regarding the widths, they may all be given the same value, may reflect the width of the previously calculated clusters (i.e. RBFs), or may be established as the average distance between RBFs, among other possibilities. In order to obtain the weights in the second stage, algorithms such as Least Mean Square (LMS) (Widrow and Lehr 1990) or Singular Value Decomposition (SVD) (Golub and Van Loan 1996) can be used.

As well as this typical methodology, different design strategies for RBFN design can be found in the literature. Due to the fact that RBFNs were initially used for function approximation, most of the methods are based on traditional optimization techniques such as regularization (Orr 1995), orthogonalization of regressors (Chen et al. 1991), gradient-based (Neruda and Kudová 2005), or Levenberg–Marquardt (Ampazis and Perantonis 2002). These techniques can be used to decide the RBFs to aggregate or eliminate and may be considered as forward or backward selection methods (Peng et al. 2006).

Another important paradigm for RBFN design is Evolutionary Computation (Bäck et al. 1997; Holland 1975; Goldberg 1989), a general stochastic optimization framework inspired by natural evolution. In any evolutionary process, the codification scheme is one of the most important characteristics. There are two main ways of representing a solution in the RBFN design:

- The Pittsburgh approach, which codifies the whole RBFN in a chromosome (Harpham et al. 2004).
- The cooperative–competitive approach, which represents an RBF in each individual. In this way, each RBF competes for survival and all the RBFs in the population cooperate towards a definite solution (Whitehead and Choate 1996). The key matter in these models is the computation of the role of each individual (credit assignment), considering both the accuracy and complexity of the final RBFN.

The Pittsburgh approach has to deal with two problems: to face up to with the high dimensionality of the search space and to define an adequate competition and cooperation among different components of the RBFN. The cooperative–competitive approach reduces the search space for the GA, and the representation of the solution with an adequate credit assignment function makes easier the obtaining of an RBFN composed by a few number of accurate RBFs, which are not overlapping and represents the information of the data examples.

The objective of our algorithm, CO^{2}RBFN, is to obtain simple and accurate RBFNs. A simple RBFN is composed of a low number of RBFs, which precisely represent the knowledge about the patterns of their environment and which are correctly located in the space of patterns (with minimum overlapping). In addition, the RBFs must work well together in order to obtain an RBFN with an adequate generalization.

In this way, our proposal, CO^{2}RBFN, follows the cooperative–competitive evolutionary approach for the design of RBFNs applied to classification problems. In order to obtain simple and accurate networks, this evolutionary paradigm is reinforced with the remaining design components:

- a fitness function (which considers three factors for each RBF; contribution to the output of the complete RBFN, local error and overlapping),
- some specific evolutionary operators based on analysing the environment of the RBFs,
- a distance measure which deals, without loss of information, with differences between nominal features which are very usual in classification problems, and
- a Fuzzy Rule Based System (FRBS) which decides the operators' application probability over a certain RBF.

The paper is organized as follows: in Sect. 2 RBFNs and the classification problem framework are described. In Sect. 3, a revision of the evolutionary proposals for RBFN design is shown. The CO^{2}RBFN algorithm is explained in Sect. 4. To prove the efficiency of the proposal, a deep comparative statistical study has been carried out with different evolutionary RBFN design paradigms, neural network models and rule induction methods. The experimental environment is described in Sect. 5. The analysis of the results is presented in Sect. 6. Finally, in Sect. 7, the main conclusions are shown.

2 RBFNs and the classification problem framework

An RBFN is a feed-forward neural network with three layers: an input layer with n nodes, a hidden layer with m neurons or RBFs, and an output layer with one or several nodes (Fig. 1). Each input node corresponds to a feature of the input pattern. The m neurons of the hidden layer are activated by a radially symmetric basis function, $\phi_i\colon R^n \to R$, which can be defined in several ways (Rojas et al. 1997).

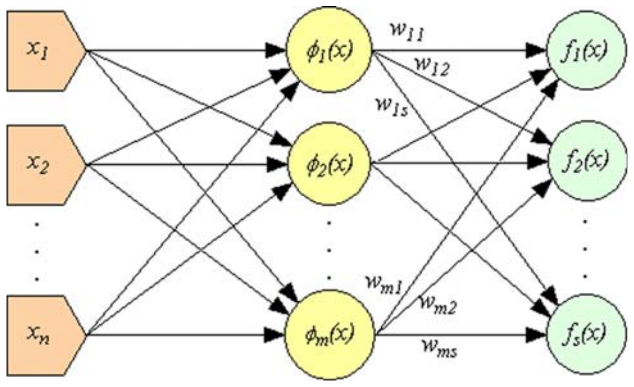

Fig. 1 RBFN topology

From all the possible choices for ϕ_i, the Gaussian function is the most widely used: $\phi_i(\vec{x}) = \phi_i(e^{-(\|\vec{x}-c_i\|/d_i)^2})$, where $\vec{c}_i \in R^n$ is the centre of basis function ϕ_i, $d_i \in R$ is the width (radius), and $\|\|$ is typically the Euclidean norm on R^n. The output of one basis function will be high when the input vector and the centre of this basis function are closer, always taking into account the value of the radius. The weights w_{ij} show the contribution of an RBF to the respective output node, and therefore output nodes implement the weighted sum of RBF outputs (Eq. 1).

$$f_j(\vec{x}) = \sum_{i=1}^{m} w_{ij}\phi_i(\vec{x}) \qquad (1)$$

In a classification environment, the RBFN has to perform mapping from an input space X^n to a finite set of classes C with k classes, in the following way: considering a training set S with p patterns:

$$S = \{(\vec{x}_u, c_u) | \vec{x}_u \in X^n, c_u \in C, \quad u = 1, \ldots, p\} \qquad (2)$$

where $\vec{x}_u$ is the feature vector and c_u is the class it belongs to.

- Usually in the classification scenario, the number of outputs of the RBFN corresponds to the number of classes (k), and each class c_u is specifically assigned to an output node.
- In order to train the network, the membership to class c_u is encoded into a binary vector $\vec{z}_u \in \{0,1\}^k$ through the relation $\vec{z}^{\,i}_u = 1$ iff $c_u = i$, and $\vec{z}^{\,i}_u = 0$ otherwise.
- After training, the output of the output node j, $f_j(\vec{x}_u)$ for a given input vector $\vec{x}_u$ can be interpreted as the class membership probability. Usually, the output node with maximum activation will be taken as the output class for the network.
- In a simple classification environment, each basis function ϕ_i will cover a set of patterns of a given class c_u. If the network weights have been correctly trained, the RBF ϕ_i will obtain a high activation value for these patterns belonging to class c_u, and therefore they will be properly classified by the RBFN.

This behaviour, together with the above-mentioned characteristics such as simple topology or local response, confers to RBFNs a degree of interpretability higher than most ANNs. In fact, the functional equivalence between RBFNs and interpretable systems like fuzzy inference systems has been demonstrated by Jang and Sun (1993), and different methods for the extraction of interpretable fuzzy rules from a trained RBFN have been proposed (Jin and Sendhoff 2003). To extract interpretable fuzzy rules from an RBFN, the number of RBFs should be kept small and there should be no very similar basis function. These two characteristics are considered in our proposal.

A typical problem of several classification models, RBFNs included, consists of working with datasets that cannot only have numerical attributes (i.e. real or integer) but also nominal ones. One of the key characteristics of any RBFN model is pattern evaluation, which implies the calculation of the distance between the pattern and the RBF centres of the network. Euclidean distance is typically used when the given attributes are numerical, obtaining satisfactory results. This metric is only defined for numerical attributes. With nominal attributes, a definition of the similarity (dissimilarity) measure becomes less trivial (Esposito et al. 2000a). A simple but commonly used measure is the Overlap Metric (OM) (Stanfill and Waltz 1986), also called Hamming distance for binary attributes. Under this metric, for two possible values, the distance is defined as zero when these values are identical and as one otherwise. However, this metric implies loss of information because it considers that all attributes values are of equal distance from each other, not taking into account different degrees of similarity. There exist alternative dissimilarity measures such as the Value Difference Metric (VDM) (Wilson and Martinez 1997), and Adaptive Dissimilarity Matrix (ADM) (Cheng et al. 2004), among others. For the VDM two feature values are considered to be closer if they have similar classifications (i.e. more similar correlations with the output classes). On the other hand, ADM takes into account the possible correlation between the attributes. In Cheng et al. (2004), the RBFN model is used to test the efficiency of the OM, VDM and ADM dissimilarity measures and it is demonstrated that VDM and ADM outperform OM. It is more difficult to conclude which dissimilarity measure is the most efficient when comparing VDM and ADM results. Finally, we have decided to use the VDM measure keeping in mind that it is, from a computational point of view, simpler and more efficient than ADM. Nevertheless, the definition of a suitable distance for both numerical and nominal attributes is necessary. The distance we have chosen is HVDM (Wilson and

Martinez 1997), which uses the Euclidean distance for numerical attributes:

$$\mathrm{HVDM}(x, y) = \sqrt{\sum_{a=1}^{n} d_a^2(x, y)} \tag{3}$$

where n is the number of attributes, and the function $d_a(x,y)$ returns a distance between the two values x and y for attribute a in the following way:

$$d_a(x, y) = \begin{cases} 1 & \text{if } x \text{ or } y \text{ is unknown} \\ VDM_a(x, y) & \text{if } a \text{ is nominal} \\ E_a(x, y) & \text{if } a \text{ is numerical} \end{cases} \tag{4}$$

The Euclidean distance is a well-known distance for numerical attributes, defined as:

$$E(x, y) = \sqrt{\sum_{a=1}^{n} (x_a - y_a)^2} \tag{5}$$

VDM provides an appropriate distance function for nominal attributes:

$$\mathrm{VDM}_a(x, y) = \sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|^q = \sum_{c=1}^{C} \left| P_{a,x,c} - P_{a,y,c} \right|^q \tag{6}$$

where

- $N_{a,x}$ is the number of instances in the training set having x as the value for attribute a;
- $N_{a,x,c}$ is the number of instances in the training set having x as the value for attribute a and c as the output class;
- C is the number of output classes in the problem domain;
- q is a constant (we have used $q = 1$).
- $P_{a,x,c}$ is the conditional probability that the output class is c, given that attribute a has the value x, i. e., $P(c|a)$. $P_{a,x,c}$ is defined as:

$$P_{a,x,c} = \frac{N_{a,x,c}}{N_{a,x}} \tag{7}$$

3 The evolutionary design of RBFNs

As previously mentioned, there are two problem areas associated with the design of ANNs in general and so with the design of RBFNs (Sanchez 2002): determining the optimal architecture and the optimal parameters.

The first problem is simplified in the case of RBFNs since there is one hidden layer and the number of data vector in the training set defines an upper bound for the number of hidden nodes. There are different proposals for the determination of hidden layer nodes (RBFs) based on Akaike's information criterion (Chen et al. 1990) that provides a compromise between network complexity and performance or on sequential learning (increasing or decreasing the number of RBFs) as (Peng et al. 2006; Holcomb and Morari 1991; Lee and Kil 1991; Musavi et al. 1992). In Sundararajan et al. (1999), a review of this area can be found.

The second problem in the RBFN design is focused on the basis centres, widths and weights, which must be representative of the whole data set. The most usual approach for this problem, the clustering algorithms (Pedrycz 1998), usually does have the problem of determining the number of hidden nodes a priori and so it can only achieve a local optimal solution (Harpham et al. 2004).

These problems have been addressed by evolutionary algorithms (EAs) (Schaffer et al. 1992; Yao 1993, 1999; Balakrishnan and Honavar 1995; Harpham et al. 2004). The following taxonomy presents the main areas where the EAs have been applied to RBFN design (Yao 1999; Harpham et al. 2004):

1. Evolving network architecture. In the RBFN design, the determination of the network architecture implies the obtaining of the number of hidden nodes. This problem is usually addressed with evolutionary proposals together with the RBFN parameters in Pittsburgh approaches (Burdsall and Giraud-Carrier 1997; Sergeev et al. 1998; Xue and Watton 1998).
2. Evolving RBFN parameters (centres, widths and weights of the RBFs). The use of EAs to optimize the connection weights could eliminates the possibility of converging to a local minimum but usually this problem is not address with EAs in an independent way but with other parameters as basis width (Sumathi et al. 2001; Sheta and De Jong 2001; Jiang et al. 2003). In Vesin and Gruter (1999), Dawson et al. (2000) proposals which evolve only the basis centre and width can be found.
3. Optimizing the data set. The dimensionality of the learning problem can be drastically reduced selecting an optimal subset of training data, which is used for training the RBFN (Billings and Zheng 1995; Sergeev et al. 1998). On the other hand, the selection of the most relevant attributes for the RBFN design is not deeply studied in the specialized bibliography (Fu and Wang 2003) and it can be addressed by means EAs (Fu and Wang 2002; Ferreira et al. 2003; Pérez-Godoy et al. 2008).

As mentioned earlier, in the evolutionary design of RBFNs most of the proposals address the different problems by means hybrid algorithms where the EA optimizes the basis centres and widths and the architecture and the second stage uses a supervised learning method in order to obtain the weights (Chaiyaratana and Zalzala 1998; Sergeev et al. 1998; Xue and Watton 1998; Vesin and Gruter 1999; Moechtar et al. 1999; Dawson et al. 2000; Chen et al. 1999).

In any evolutionary algorithm, and therefore in those for the evolutionary design of RBFNs, two main aspects must be considered:

1. The way to represent and manage the solutions.
2. The way to compute the goodness of any candidate solution.

Regarding the first aspect, in the specialized bibliography most of the evolutionary proposals for the design of RBFNs (Lacerda et al. 2005; Harpham et al. 2004; Rivas et al. 2004) codifies a complete RBFN by means an individual and the population of RBFNs evolves through different operators. It is called Pittsburgh representation scheme. In the specialized bibliography, there are some proposals which represents a solution by means this Pittsburgh scheme: with binary encoded (Sergeev et al. 1998; Vesin and Gruter 1999; Moechtar et al. 1999; Dawson et al. 2000; Sumathi et al. 2001; Du and Zhang 2008), integer (Billings and Zheng 1995) or real encoded (Leung et al. 2002; Esposito et al. 2000b) chromosomes.

Nevertheless, according to Potter and De Jong (2000) evolutionary computation has some difficulties in solving certain types of problems, especially when an individual represents a complete solution composed of independent subcomponents. An alternative to the classical (Pittsburgh) approach is the cooperative–competitive evolutionary strategy (Whitehead and Choate 1996; Potter and De Jong 2000), which provides a framework where an individual of the population represents only a part of the solution, evolving in parallel, competing to survive but at the same time cooperating in order to find a common solution (the complete RBFN). This approach has the advantage of being computationally less complex, since an individual does not represent the whole solution but only a part of it. With this approach, two main problems must be addressed: the credit assignment, or the fitness allocated to each individual according to its contribution to the final solution, and the mechanism used in order to maintain diversity among individuals of the population.

In the bibliography, there are some proposals concerning the design of RBFNs based on cooperative–competitive evolutionary strategies (Whitehead and Choate 1996; Topchy et al. 1997; Li et al. 2008). The most traditional is (Whitehead and Choate 1996) where an individual represents an RBF and the population the whole network. Individual credit assignment is defined depending on the weight of the RBF. In the same way, in the algorithm described in (Topchy et al. 1997) an individual is an RBF and credit assignment is calculated according to the efficiency of the RBF or its contribution to the correct output of the network. In Li et al. (2008), a co-evolutionary RBFN design method, interacting co-adapted subpopulations evolve independently. Each individual in a population represents a particular component (group of RBFs) of the RBFN. The fitness of an individual from a particular subpopulation is assessed by associating it with representatives from other subpopulations.

The authors of this paper have proposed evolutive methods for function approximation and time series forecast (Rivera et al. 2007) using the cooperative–competitive paradigm. In addition, a first approach to solve classification problems has been described in Pérez-Godoy et al. (2007).

Regarding the computation of the quality of the candidate solutions, managing only one objective in order to optimize RBFNs (the error, for instance) may lead us to obtain RBFNs with high complexity, i.e. a high number of RBFs. This is because it is easier to reduce error with those RBFNs having many RBFs than with those having few neurons. Evolutionary multiobjective optimization (Deb 2001) can be used in order to optimize several objectives such as error and complexity (the number of RBFs) for example. In these evolutionary algorithms, all the objectives must be taken into account in order to define both the fitness and the relation order among individuals. The goal is to obtain a set of optimized solutions (non-dominated solutions) with similar quality rather than a single optimal solution. Different multiobjective evolutionary algorithms applied to RBFN design have been proposed in (González et al. 2003; Teixeira et al. 2008; Guillén et al. 2007; Yen 2006).

To date, all the multi-objective proposals for the RBFN design use the Pittsburgh codification scheme, because of the difficulties in the comparison of partial solutions (as the cooperative–competitive approach requires).

To deal with multiple quality measures, the evolutionary proposals, in the specialized bibliography, combine them in the credit assignment (Rivera et al. 2001) or in the selection procedure (Rivera et al. 2007) among other possibilities.

4 CO^{2}RBFN: an evolutionary cooperative–competitive hybrid algorithm for RBFN design

CO^{2}RBFN (Cooperative Competitive algorithm for RBFN design) combines different soft computing techniques such as neural networks, evolutionary programming with a cooperative–competitive approach and fuzzy rule-based systems. This evolutionary strategy allows us to determine RBFN parameters, and the evolutionary operators' application is decided by means of a FRBS (Fig. 2).

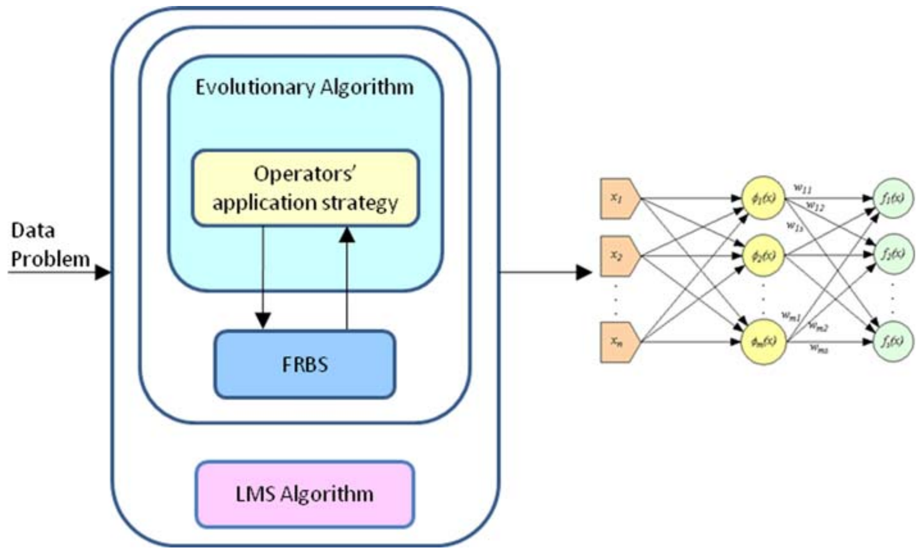

Fig. 2 CO^2RBFN hybrid architecture

The proposed RBFN design algorithm is based on the evolutionary programming paradigm (Fogel et al. 1966) which has the following features:

- there is a phenotypic level evolution
- mutations are the single sources of the modification in feasible solutions
- genetic operators such as crossover or similar are not used
- the selection procedure can be viewed as a tournament between parents and progeny.

As mentioned previously, the objective of our algorithm, CO^2RBFN, is to obtain simple and accurate RBFNs. A simple RBFN is composed of a low number of RBFs, which precisely represent the knowledge about the patterns of their environment and which are correctly located in the space of patterns (with minimum overlapping). In addition, the RBFs must work well together in order to obtain an RBFN with an adequate generalization.

With these aims in mind, our proposal follows the cooperative–competitive evolutionary strategy, where each individual of the population represents an RBF (gaussian function will be considered as RBF) and the entire population is responsible for the final solution. This paradigm provides a framework where an individual of the population represents only a part of the solution, competing to survive (since it will be eliminated if its performance is poor) but at the same time cooperating in order to build the whole RBFN, which adequately represents the knowledge about the problem and achieves a good generalization for new patterns. In this scenario, the local operation (RBFs with local response) and the representation of the majority of the examples (by means of any RBF) is reinforced and the overlapping among RBFs is minimized. These design-guidelines in our algorithm improve the interpretability of the RBFN obtained.

Subsequently, the remaining components of our algorithm are designed: fitness function, evolutionary operators, the distance measure and the fuzzy rule base system, which decides the probability of applying operators.

In this environment, in which the final solution depends on the behaviour of many components, the fitness of each individual is known as credit assignment. In order to measure the credit assignment of an individual, three factors have been proposed to evaluate the role of each RBF in the network (error, contribution and overlapping). These factors reinforce: the individual quality of RBFs (calculating the local error inside the radius of each RBF), their generality (measured by the contribution which promotes RBFs with a high number of patterns inside its radius), and the adequate location of the RBFs in the space of patterns (measured by the overlapping). Together, these three factors enhance the individual role of RBFs, and moreover their cooperative work to build an accurate and simple network. It can be highlighted that the last two factors take into account all the patterns and the behaviour of the rest of the RBFs. In this way, our fitness function combines concepts such as cooperation, specialization and niching (Buchtala et al. 2005).

Our evolutionary operators have been designed in order to achieve an adequate balance between exploration and exploitation of data. Some of them include characteristics of clustering algorithm in order to properly analyze the local environment of the RBFs. Specifically four evolutionary operators have been proposed: an operator which eliminates RBFs, two operators which mutate RBFs, and finally an operator which maintains RBF parameters in order to explore and exploit the search space and to preserve the best RBFs, respectively.

In order to decide the operators' application probability over a certain RBF the algorithm uses an FRBS, which represents expert knowledge in the design of RBFNs. The factors proposed for credit assignment have been used as input parameters for the FRBS, the operators' application probability being the outputs of the FRBS.

Finally, it must be highlighted that the proposal considers a distance measure which deals, without loss of information, with differences between nominal features which are very usual in classification problems.

All these components constitute an algorithm, CO^{2}RBFN, which designs RBFNs for solving classification problems, with an appropriate balance between accuracy and simplicity.

The main steps of CO^{2}RBFN, explained in the following subsections, are shown in the pseudocode in the Fig. 3.

4.1 RBFN initialization

To define the initial network, a simple process is used: a specified number, m, of neurons (i.e. the size of population) is randomly allocated among the different classes of the training set. To do so, each RBF centre, $\vec{c}_i$, is randomly established to a pattern of the training set, taking into account that the RBFs must be distributed equally among the different classes. The RBF widths, d_i, will be set to half of the average distance between the centres. Finally, the RBF weights, w_{ij}, are set to zero.

4.2 RBFN training

During this stage, RBF weights are trained. LMS (Widrow and Lehr 1990) has been used to calculate the RBF weights. This technique exploits the local information that can be obtained from the behaviour of the RBFs.

4.3 RBF evaluation

A credit assignment mechanism is required in order to evaluate the role of each base function in the cooperative–competitive environment. For an RBF ϕ_i, three parameters, a_i, e_i, o_i are defined.

- The contribution, a_i, of the RBF ϕ_i, $i = 1 \ldots m$, is determined by considering the weight, w_i, and the number of patterns of the training set inside its width, pi_i. An RBF with a low weight and few patterns inside its width will have a low contribution:

$$a_i = \begin{cases} |w_i| & \text{if } pi_i > q \\ |w_i| * (pi_i/q) & \text{otherwise} \end{cases} \quad (8)$$

where q is the average of the pi_i values minus the standard deviation of the pi_i values.
- The error measure, e_i, for each RBF ϕ_i, is obtained by counting the wrongly classified patterns inside its radius:

$$e_i = \frac{pibc_i}{pi_i} \quad (9)$$

where $pibc_i$ and pi_i are the number of wrongly classified patterns and the number of all patterns inside the RBF width, respectively.
- The overlapping of the RBF ϕ_i and the other RBFs is quantified by using the parameter o_i. This parameter is calculated by taking into account the *fitness sharing* (Goldberg and Richardson 1987) methodology, whose aim is to maintain the diversity in the population. This factor is expressed as:

$$o_i = \sum_{j=1}^{m} o_{ij}$$
$$o_{ij} = \begin{cases} \left(1 - \|\phi_i - \phi_j\|/d_i\right) & \text{if} \|\phi_i - \phi_j\| < d_i \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where o_{ij} measures the overlapping of the RBF ϕ_i and ϕ_j, $j = 1 \ldots m$

```
1. Initialize RBFN
2. Train RBFN
3. Evaluate RBFs
4. Apply operators to RBFs
5. Substitute the RBFs that were eliminated
6. Apply replacement strategy
7. If the stop-condition is not verified go to step 2
```

Fig. 3 Main steps of CO^{2}RBFN

4.4 Applying operators to RBFs

In this algorithm, four operators have been defined in order to be applied to the RBFs.

- Operator *Remove*: eliminates an RBF.
- Operator *Random Mutation*: modifies the centre and width of an RBF. The width is altered with a probability inversely proportional to the number of features of the classification problem (n), in a percentage below 50% of the old width. The coordinates of the centre are modified as follows: if the coordinate is a real value, it is increased or decreased in a percentage below 50% of the width. If the coordinate is a nominal value, it mutates to another one, among all the possible values of the attribute or feature, with a probability inversely proportional to the HVDM distance from the original value. The number of coordinates to be mutated is randomly obtained and it is a number below 25% of the total number of features.
- Operator *Biased Mutation*: modifies the width and all coordinates of the centre using local information of the

RBF environment. A clustering-based technique for training centres has been used which modifies the RBF centre, $\vec{c}_i$, as follows:

$$c'_{ij} = c_{ij} \pm h \quad \forall j = 1 \ldots n \tag{11}$$

The increase or decrease of the old centre is decided by means of a random number h ($h \leq 0.5\, d_i$). The centre is varied in order to approximate it to the average of the patterns belonging to the RBF class and inside its RBF width.

The objective of the width training is that most of the patterns belonging to the RBF class will be inside the RBF width. In this way, the RBF width is modified as follows:

$$\begin{cases} d' = d - h & \text{if } u \leq D \\ d' = d + h & \text{otherwise} \end{cases} \quad D = \frac{\text{npnci}}{\text{npci}} \quad A = \frac{\text{npci2}}{\text{npnci2}} \tag{12}$$

where h is a random number ($h \leq 0.5 d_i$); u is a random number ($u \leq D + A$); npnci is the number of patterns not belonging to the RBF class inside the RBF width; npci is the number of patterns belonging to the RBF class inside the RBF width; npci2 is the number of patterns belonging to the RBF class inside twice RBF width and npnci2, is the number of patterns not belonging to the RBF class inside twice RBF width.

- Operator *Null*: in this case, all the parameters of the RBF are maintained.

These mutation operators allow us to obtain an appropriate balance between exploitation and exploration, which is a desirable feature in every evolutionary algorithm. Biased mutations use local information from the RBF environment in order to achieve an optimal adaptation. On the other hand, random mutations carry out alterations that lead to the exploration of the environment and thus avoid local optimums.

The operators are applied to the whole population of RBFs. The probability for choosing an operator is determined by means of a Mandani-type fuzzy system (Mandani and Assilian 1975). The inputs of this system are parameters a_i, e_i and o_i used for defining the credit assignment of the RBF ϕ_i. These inputs are considered as linguistic variables va_i, ve_i and vo_i. The outputs, p_{remove}, p_{rm}, p_{bm} and p_{null}, represent the probability of applying remove, random mutation, biased mutation and null operators, respectively.

The number of linguistic labels has been empirically determined and the fuzzy sets have been defined according to their meaning. There are three linguistic labels, Low, Medium and High, to define each input. Four linguistic labels are considered for the outputs: Low, Medium–Low, Medium–High and High. Figure 4 shows the membership functions for the input and output variables, respectively.

Table 1 shows the rule base used to relate the described antecedents and consequents. The rule base represents expert knowledge in the design of RBFNs. It was developed taking into account the fact that an RBF is worse if its contribution (a_i) is low, its error (e_i) is high and its overlapping (o_i) is also high. On the other hand, an RBF is better when its contribution is high, its error is low and its overlapping is also low. Therefore, as the probability of eliminating a basis function increases, the associated RBF becomes worse. However, as the probability of not modifying an RBF increases, the associated basis function improves. The probabilities of mutation usually have a high value in order to promote a parsimonious evolution.

4.5 Introduction of new RBFs

In this step of the algorithm, the eliminated RBFs are substituted by new RBFs. The new RBF is located on a badly classified pattern or on a randomly chosen pattern with a probability of 0.5, respectively.

In the first instance, the RBF is located on the first badly classified pattern outside of any RBF width found. The width of the new RBF will be set to the average of the RBFs in the population plus half of the minimum distance to the nearest RBF. Its weights are set to zero.

If it is chosen randomly, the RBF is located on the first pattern found outside of any RBF width. The width of the new RBF is set to the average of the RBFs in the population and its weights are set to zero.

4.6 Replacement strategy

After applying the mutation operators, new RBFs appear. The algorithm uses the replacement scheme to determine which new RBFs will be included in the new population. To do so, the role of the mutated RBF in the network is

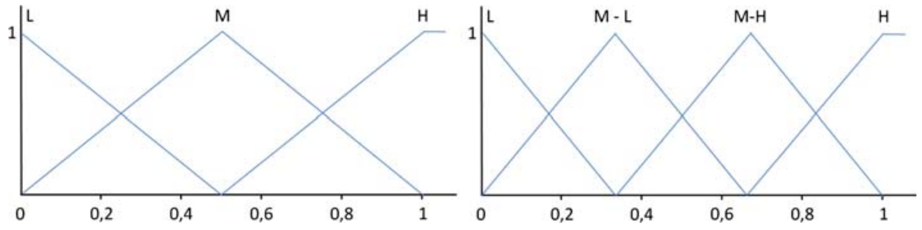

Fig. 4 *Left* input variables membership functions for the FRBS. *Right* output variables membership function

Table 1 Fuzzy rule base representing expert knowledge in the design of RBFNs

	Antecedents			Consequents			
	v_a	v_e	v_o	p_{remove}	p_{rm}	p_{bm}	p_{null}
R1	L			M–H	M–H	L	L
R2	M			M–L	M–H	M–L	M–L
R3	H			L	M–H	M–H	M–H
R4		L		L	M–H	M–H	M–H
R5		M		M–L	M–H	M–L	M–L
R6		H		M–H	M–H	L	L
R7			L	L	M–H	M–H	M–H
R8			M	M–L	M–H	M–L	M–L
R9			H	M–H	M–H	L	L

compared with the original one in order to determine the RBF with the best behaviour in order to include it in the population.

There are important differences between the cooperative competitive method developed by the authors (Rivera et al. 2007) and CO^{2}RBFN. In summary, the evolutionary process has been deeply remodelled in order to increase its exploration and exploitation features, and the RBFN design is optimized. The main differences are highlighted here:

- In order to address nominal attributes a new dissimilarity measure, HVDM, has been introduced.
- The number of network outputs in Rivera et al. (2007) was set to one because it solved regression problem. In CO^{2}RBFN, the network has one output for each class in the addressed dataset.
- New and simpler credit assignment parameters to measure the contribution and the error are defined in CO^{2}RBFN in order to increase the efficiency and to only penalize RBFs which represent a small set of patterns.
- In Rivera et al. (2007) operators were applied to the worst RBFs, but in CO^{2}RBFN, operators are applied to the whole population. Therefore, a high level of exploration–exploitation is promoted.
- Regarding the operators, in Rivera et al. (2007), two operators were considered: an RBF deleting operator, and a biased mutation operator based on gradient error techniques for tuning the RBF centre and radius. CO^{2}RBFN considers four operators: an RBF deleting operator; a random mutation operator which modifies RBF width and centre and so improves the exploration features; a new biased operator which carries out a more specialized analysis of the local environment information, based on clustering techniques, so the exploitation characteristics are improved; finally, in CO^{2}RBFN, a new operator that maintains the RBF parameters is introduced in order to achieve a parsimonious evolution of the population.
- A new fuzzy rule base is developed for CO^{2}RBFN taking into account new operators.
- Beside the specialized method for introducing RBFs in the population, a new random introduction RBF method is considered in order to improve the balance between exploration and exploitation.
- Finally, the replacement mechanism is also modified. In Rivera et al. (2007), progeny always substitute parents, but in CO^{2}RBFN, this mechanism is defined as a tournament between parents and progeny. Again, the equilibrium between exploration and exploitation of the evolutionary process is enhanced.

5 Experimental framework

In this study, CO^{2}RBFN is applied to 11 data sets and is compared with five different soft computing methods. A complete statistical study has been carried out to test the efficiency and complexity of the compared methods.

The collection of data sets used in this section was obtained from the UCI Repository of Machine Learning Database (Asuncion and Newman 2007).

Table 2 shows the features of the Datasets.

Car, Credit, Hepatitis and Wbcd datasets have missing values and therefore a pre-processing stage needs to be performed. In the case of numerical attributes, any missing value of an attribute in a given instance is substituted for the average of the values for this attribute in the remaining instances of the same class. When the attribute is nominal, the mode is used instead of the average.

In order to estimate the precision, we use a tenfold cross validation approach, that is, ten partitions for training and

Table 2 Dataset features

Dataset	Instances	Numerical features	Nominal features	Classes
Car	1,728	0	6	4
Credit	690	6	9	2
Glass	114	9	0	7
Hepatitis	155	6	13	2
Ionosphere	351	34	0	2
Iris	150	4	0	3
Pima	768	8	0	2
Sonar	208	60	0	2
Wbcd	699	9	0	2
Vehicle	846	18	0	4
Wine	178	13	0	3

test sets, 90% for training and 10% for testing, where the ten test partitions form the whole set. For each dataset, we consider the average results of the ten partitions.

CO^{2}RBFN has been compared with several methods covering a wide range within the machine learning field: alternative paradigms in the RBFN design field, another neural network model (multilayer perceptron), and a decision tree model. Specifically:

- GeneticRBFN: algorithm for RBFNs design based on the Pittsburgh scheme where each individual is a whole network. Implementation designed by the authors (see Appendix 2 for a wide description).
- C4.5: algorithm that creates classification rules in the form of decision trees from a dataset (Quinlan 1993). Implementation obtained from KEEL (Alcalá-Fdez et al. 2009).
- MLP-Back: algorithm for Multilayer Perceptron Networks design which uses the Backpropagation algorithm for learning (Rojas and Feldman 1996). Implementation obtained from KEEL.
- RBFN-Decr: algorithm for RBFNs design based on a decremental scheme (Broomhead and Lowe 1988). Implementation obtained from KEEL.
- RBFN-Incr: algorithm for RBFNs design based on an incremental scheme (Plat 1991). Implementation obtained from KEEL.

As has been said, in CO^{2}RBFN there is only one network, and the number of individuals is the number of RBF nodes. CO^{2}RBFN is run with a number of RBFs from the number of classes in the given dataset to four times the number of these classes. The parameters used for CO^{2}RBFN are shown in Table 3.

The parameters used for GeneticRBFN, C4.5, MLP-Back, RBFN-Decr and RBFN-Incr are set to the values indicated by the authors and they are shown from Table 3.

From Tables 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 and 14, the classification test rate of the methods and their corresponding complexities (number of nodes—RBFs—in the RBFN or number of the rules in the FRBSs) are shown. For the CO^{2}RBFN algorithm, the best efficiency network obtained inside the execution range is chosen (CO^{2}RBFN results for all the execution range are shown in Appendix 1). In the tables, the bold entries highlight the best results.

6 Analysis of the results

A first analysis of the results shows that CO^{2}RBFN obtains RNFNs comparable in the classification rate to other methods (even higher in six datasets) and with low complexity.

Nevertheless, in order to formally analyse the results with regard to the precision and complexity of the methods, we perform series of statistical tests. The objective of these tests is to determine if the differences are significant.

Demšar (2006) illustrates the necessity of using non-parametric statistics due to the failure of evolutionary

Table 3 Parameters used for algorithms

Algorithm	Parameter	Value
CO^{2}RBFN	Generations of the main loop	200
	Number of RBF's	Min = number of classes
		Max = 4 × number of classes
GeneticRBFN	Generations of the main loop	200
	Individuals	40
	Chromosome length	Min = number of classes
		Max = 4 × number of classes
	Crossover probability	0.6
	Mutation probability	0.1
	Mutation widths percent	0.2
	Mutation centres percent	0.2
	Tournament size	3
C4.5	Pruned	True
	Confidence	0.25
	InstancesPerLeaf	2
MLP-Back	Hidden_layer	2
	Hidden_nodes	15
	Transfer	Htan
	Eta	0.15
	Alpha	0.10
	Lambda	0.0
RBFN-Decr	Percent	0.1
	nNeuronsIni	20
	Alpha	0.3
RBFN-Incr	Epsilon	0.1
	Alpha	0.3
	Delta	0.5

Table 4 Results with Car dataset

Algorithm	# Nodes/rules	Classification rate (%)
C4.5	119.4	**91.550**
GeneticRBFN	15.9	80.783
MLP-Back	30.0	49.245
RBFN-Decr	16.0	73.847
RBFN-Incr	1340.6	93.171
CO^{2}RBFN	**5.0**	81.007

Table 5 Results with Credit dataset

Algorithm	# Nodes/rules	Classification rate (%)
C4.5	22.0	**87.101**
GeneticRBFN	7.0	85.507
MLP-Back	30.0	82.609
RBFN-Decr	7.3	61.449
RBFN-Incr	599.9	66.087
CO^{2}RBFN	**2.0**	84.232

Table 6 Results with Glass dataset

Algorithm	# Nodes/rules	Classification rate (%)
C4.5	26.3	67.443
GeneticRBFN	27.0	65.089
MLP-Back	30.0	37.609
RBFN-Decr	15.6	24.438
RBFN-Incr	124.0	54.072
CO^{2}RBFN	**22.0**	**67.710**

Table 7 Results with Hepatitis dataset

Algorithm	# Nodes/rules	Classification rate (%)
C4.5	**7.1**	**89.647**
GeneticRBFN	7.5	86.711
MLP-Back	30.0	71.817
RBFN-Decr	12.6	76.711
RBFN-Incr	120.1	76.637
CO^{2}RBFN	8.0	87.399

Table 8 Results with Ionosphere dataset

Algorithm	# Nodes/rules	Classification rate (%)
C4.5	13.2	90.632
GeneticRBFN	7.5	**92.885**
MLP-Back	30.0	72.948
RBFN-Decr	13.0	82.348
RBFN-Incr	180.5	92.592
CO^{2}RBFN	**8.0**	91.411

Table 9 Results with Iris dataset

Algorithm	# Nodes/rules	Classification rate (%)
C4.5	**4.8**	94.000
GeneticRBFN	10.4	95.067
MLP-Back	30.0	64.667
RBFN-Decr	9.6	94.000
RBFN-Incr	29.8	94.667
CO^{2}RBFN	6.0	**96.267**

Table 10 Results with Pima dataset

Algorithm	# Nodes/rules	Classification rate (%)
C4.5	18.3	73.972
GeneticRBFN	7.6	75.615
MLP-Back	30.0	70.819
RBFN-Decr	7.6	72.014
RBFN-Incr	671.4	67.728
CO^{2}RBFN	**4.0**	**75.950**

Table 11 Results with Sonar dataset

Algorithm	# Nodes/rules	Classification rate (%)
C4.5	14.3	71.071
GeneticRBFN	**7.7**	73.305
MLP-Back	30.0	67.762
RBFN-Decr	13.8	59.143
RBFN-Incr	159.8	74.976
CO^{2}RBFN	8.0	**75.086**

Table 12 Results with Vehicle dataset

Algorithm	# Nodes/rules	Classification rate (%)
C4.5	71.8	**71.034**
GeneticRBFN	16.0	67.043
MLP-Back	30.0	38.066
RBFN-Decr	**10.8**	45.046
RBFN-Incr	750.7	56.259
CO^{2}RBFN	16.0	69.192

algorithms for continuous optimization problems to verify the various hypotheses necessary for the use of parametric tests.

We will apply the following non-parametric tests:

- In order to see whether there were differences among the algorithms, we use Friedman's test and Iman and Davenport's test.
- For pair wise comparison, Wilcoxon's signed-ranks test is used.

The tests are applied using 0.05 as level of confidence (α). A wider description of these tests is shown in Appendix 3.

In the next two subsections, these tests are applied considering classification rate and model complexity (number of nodes or rules), respectively.

6.1 Classification rate analysis

In this section, the classification accuracy of the methods is analysed. First, Friedman's test is applied and a ranking of

Table 13 Results with Wbcd dataset

Algorithm	# Nodes/rules	Classification rate (%)
C4.5	12.4	94.995
GeneticRBFN	6.2	96.713
MLP-Back	30.0	87.722
RBFN-Decr	10.4	92.119
RBFN-Incr	319.9	94.303
CO^{2}RBFN	**5.0**	**97.083**

Table 14 Results with Wine dataset

Algorithm	# Nodes/rules	Classification rate (%)
C4.5	**5.1**	94.902
GeneticRBFN	10.4	95.275
MLP-Back	30.0	93.301
RBFN-Decr	8.3	68.562
RBFN-Incr	125.3	74.739
CO^{2}RBFN	7.0	**96.739**

Table 15 Rankings obtained for the algorithms taking into account classification rate

Algorithm	Ranking
C4.5	2.591
CO^{2}RBFN	**1.727**
GeneticRBFN	2.455
MLP-Back	5.364
RBFN-Decr	5.136
RBFN-Incr	3.727

methods is obtained. This ranking is used by Friedman and Iman and Davenport's tests in order to determine whether there are significant differences between the methods.

Table 15 shows the average rankings (computed by the Friedman's test) of the algorithms. A lower value in the ranking represents a better algorithm. It can be seen how the best algorithm in the ranking is CO^{2}RBFN. Table 16 shows the results of applying both tests (Friedman and Iman and Davenport).

The Friedman statistic is 35.403. The critical value of χ^2 distribution with 5 degrees of freedom is lower than the Friedman statistic, which implies that there are significant differences between the algorithms.

The Iman and Davenport statistic is 18.065. The critical value of *F-distribution* with 5 and 50 degrees of freedom is lower than the Iman and Davenport value, which implies there are significant differences between the algorithms.

Next, Wilcoxon's test is applied in order to detect significant differences between the behaviour of pairs of

Table 16 Result of Friedman and Iman and Davenport's tests taking into account classification rate

Test	Statistic	Critical values	Significant differences?
Friedman	35.403	11.071	Yes
Iman and Davenport	18.065	2.400	Yes

Table 17 Results of Wilcoxon's test taking into account classification rate

R^+ CO^{2}RBFN	R^-		Critical value	Significant differences?
35.0	C4.5	31.0	10	No
52.0	GeneticRBFN	14.0	10	No
66.0	MLP-Back	0.0	10	Yes
66.0	RBFN-Decr	0.0	10	Yes
57.0	RBFN-Incr	9.0	10	Yes

Table 18 Rankings obtained for the algorithms taking into account complexity models

Algorithm	Ranking
C4.5	3.273
Co^{2}RBFN	**1.773**
GeneticRBFN	2.5
MLP-Back	4.909
RBFN-Decr	2.636
RBFN-Incr	5.909

algorithms. In Table 17, results of Wilcoxon's test are shown. As can be seen, in these tables there are significant differences between CO^{2}RBFN and MLP-Back, RBFN-Decr and RBFN-Incr.

6.2 Model complexity analysis

The model complexity is measured based on the number of rules, in the case of C4.5 methods, and on the number of nodes in the case of CO^{2}RBFN, GeneticRBFN, MLP-Back, RBFN-Decr and RBFN-Incr methods.

Friedman's test is applied and the ranking obtained is shown in Table 18. Again, the best algorithm in the ranking is CO^{2}RBFN. Table 19 shows the results of Friedman and Iman and Davenport's tests.

The Friedman statistic is 39.506. The critical value χ^2 distribution with 5 degrees of freedom is lower than the Friedman statistic, which implies significant differences between the algorithms.

The Iman and Davenport statistic is 25.499. The critical value of *F-distribution* with 5 and 50 degrees of freedom is

Table 19 Results of Friedman and Iman and Daverport's tests taking into account complexity models

Test	Statistic	Critical values	Significant differences?
Friedman	39.506	11.071	Yes
Iman and Davenport	25.499	2.400	Yes

Table 20 Results of Wilcoxon's test taking into account complexity models

R^+ CO^{2}RBFN	R^-		Critical value	Significant differences?
60.0	C4.5	6.0	10	Yes
56.5	GeneticRBFN	9.5	10	Yes
66.0	MLP-Back	0.0	10	Yes
50.0	RBFN-Decr	16.0	10	No
66.0	RBFN-Incr	0.0	10	Yes

lower than the Iman and Davenport statistic, which implies significant differences between the algorithms.

The results of applying Wilcoxon's test are shown in Table 20. As can be seen, in these tables there are significant differences between CO^{2}RBFN and C4.5, GeneticRBFN, MLP-Back and RBFN-Incr.

6.3 Summary

The Friedman's test ranking shows that CO^{2}RBFN is the best algorithm both in terms of accuracy and complexity (see Tables 15, 18) where minimum value implies best algorithm, in precision and simplicity, respectively).

Friedman and Iman and Davenport statistics demonstrate that there are significant differences between the methods analysed (see Tables 16, 19).

In order to determine pair differences, Wilcoxon's signed-ranks test is applied. This statistical analysis shows that there are significant differences in accuracy between CO^{2}RBFN and MLP-Back, RBFN-Incr and RBFN-Decr, and there are no significant differences between C4.5 and GeneticRBFN. Nevertheless, regarding complexity CO^{2}RBFN shows significant differences between C4.5 and GeneticRBFN, as well as MLP-Back and RBFN-Incr.

Taking into account the above, CO^{2}RBFN outperforms in complexity, with statistical significant differences, all the methods considered except RBFN-Decr (but CO^{2}RBFN outperform, with statistical significant differences, RBFN-Decr in accuracy).

Regarding accuracy, CO^{2}RBFN outperforms all the proposals studied except C4.5 and GeneticRBFN, always with significant differences. It must be highlighted that CO^{2}RBFN improves C4.5 and GeneticRBFN, with statistical significant differences.

In summary, we can conclude that CO^{2}RBFN is the best proposal in the accuracy-complexity balance for the RBFN design.

7 Conclusions

In this paper, we propose CO^{2}RBFN, a new hybrid evolutionary cooperative–competitive algorithm for RBFN design in the field of pattern classification.

The aim of our algorithm is to obtain simple and accurate networks and both the design of the general evolutionary paradigm and the design of the rest of the components have this objective.

An important key point of our proposal is the identification of the role (credit assignment) of each basis function in the whole network. In order to evaluate this value for a given RBF three factors are defined and used: the RBF contribution to the network's output, a_i (promoting the generality of the RBF); the error in the basis function radius, e_i (reinforcing the quality of the individual); and the degree of overlapping among RBFs, o_i (promoting the good location of the RBFs). In order to drive the cooperative–competitive process, with an adequate balance between exploration and exploitation, four operators are used: *Remove*, *Random Mutation*, *Biased Mutation* (based on clustering) and *Null*. The application of these operators is determined by a fuzzy rule-based system which represents expert knowledge of the RBFN design. The inputs of this system are the three parameters used for credit assignment: a_i, e_i, and o_i.

It must be highlighted that the proposal considers a distance measure, HVDM, which deals, without loss of information, with differences between nominal features which are very usual in classification problems.

Our approach has been evaluated using eleven well-known datasets, and their results have been compared with those obtained by five other soft computing methods. These methods cover a wide range within the machine-learning field including alternative paradigms in RBFN design, another neural network model and a decision tree model.

The accuracy and complexity of the models have been formally analysed using a series of statistical tests. This study shows that CO^{2}RBFN obtains RBFNs with an appropriate balance between accuracy and simplicity, outperforming the other methods considered.

Acknowledgments This work has been partially supported by the CICYT Spanish Projects TIN2005-04386-C05-03, TIN2007-60587 and the Andalusian Research Plan TIC-3928.

Appendix 1: Detailed results obtained for CO^{2}RBFN

Car dataset

# Nodes	Error training	Standard dev	Error test	Stand dev	% Training	% Test
4	0.129	0.007	0.207	0.061	87.104	79.350
5	**0.122**	**0.007**	**0.190**	**0.048**	**87.823**	**81.007**
6	0.116	0.008	0.204	0.045	88.354	79.572
7	0.112	0.007	0.198	0.046	88.753	80.197
8	0.108	0.009	0.198	0.051	89.186	80.241
9	0.102	0.009	0.200	0.056	89.784	79.998
10	0.098	0.009	0.207	0.053	90.203	79.270
11	0.095	0.009	0.191	0.052	90.543	80.925
12	0.093	0.011	0.200	0.040	90.678	80.011
13	0.089	0.008	0.202	0.047	91.075	79.804
14	0.087	0.010	0.207	0.049	91.279	79.261
15	0.079	0.008	0.220	0.061	92.085	77.974
16	0.079	0.010	0.199	0.045	92.103	80.127

Credit dataset

# Nodes	Error training	Standard dev	Error test	Stand dev	% Training	% Test
2	**0.123**	**0.006**	**0.158**	**0.065**	**87.655**	**84.232**
3	0.120	0.004	0.158	0.081	87.974	84.203
4	0.118	0.004	0.177	0.100	88.235	82.261
5	0.116	0.005	0.175	0.096	88.432	82.522
6	0.115	0.003	0.172	0.088	88.470	82.812
7	0.114	0.004	0.167	0.084	88.577	83.275
8	0.113	0.004	0.179	0.094	88.686	82.116

Glass dataset

# Nodes	Error training	Standard dev	Error test	Stand dev	% Training	% Test
7	0.328	0.020	0.358	0.113	67.223	64.216
8	0.319	0.016	0.373	0.103	68.145	62.699
9	0.310	0.017	0.354	0.104	68.987	64.575
10	0.296	0.014	0.360	0.120	70.410	63.990
11	0.290	0.015	0.354	0.111	71.034	64.635
12	0.282	0.017	0.333	0.105	71.812	66.694
13	0.277	0.014	0.332	0.111	72.299	66.778
14	0.275	0.016	0.356	0.116	72.547	64.389
15	0.266	0.015	0.330	0.109	73.399	66.976
16	0.262	0.015	0.343	0.107	73.826	65.654
17	0.258	0.015	0.346	0.104	74.230	65.425
18	0.255	0.016	0.335	0.103	74.490	66.487
19	0.251	0.014	0.340	0.117	74.925	65.980
20	0.250	0.017	0.349	0.109	74.977	65.086
21	0.248	0.014	0.354	0.118	75.164	64.602
22	**0.248**	**0.014**	**0.323**	**0.111**	**75.175**	**67.710**
23	0.242	0.016	0.328	0.114	75.798	67.223
24	0.240	0.015	0.332	0.118	75.997	66.763
25	0.236	0.014	0.342	0.116	76.389	65.820
26	0.233	0.015	0.326	0.102	76.700	67.391
27	0.234	0.019	0.329	0.115	76.587	67.065
28	0.235	0.015	0.337	0.107	76.536	66.332

Hepatitis dataset

# Nodes	Error training	Standard dev	Error test	Stand dev	% Training	% Test
2	0.087	0.016	0.168	0.145	91.270	83.228
3	0.074	0.009	0.168	0.138	92.632	83.157
4	0.067	0.011	0.151	0.094	93.261	84.905
5	0.065	0.011	0.151	0.071	93.548	84.914
6	0.064	0.008	0.128	0.107	93.563	87.187
7	0.063	0.010	0.139	0.075	93.749	86.137
8	**0.057**	**0.007**	**0.126**	**0.074**	**94.280**	**87.399**

Ionosphere dataset

# Nodes	Error training	Standard dev	Error test	Stand dev	% Training	% Test
2	0.149	0.017	0.171	0.053	85.078	82.926
3	0.134	0.020	0.160	0.049	86.648	84.003
4	0.105	0.015	0.134	0.050	89.485	86.579
5	0.097	0.017	0.119	0.049	90.294	88.069
6	0.087	0.014	0.111	0.043	91.270	88.907
7	0.074	0.013	0.099	0.048	92.643	90.111
8	**0.065**	**0.011**	**0.086**	**0.039**	**93.485**	**91.411**

Iris dataset

# Nodes	Error training	Standard dev	Error test	Stand dev	% Training	% Test
3	0.017	0.008	0.045	0.047	98.252	95.467
4	0.012	0.004	0.048	0.055	98.770	95.200
5	0.011	0.004	0.045	0.052	98.919	95.467
6	**0.010**	**0.004**	**0.037**	**0.042**	**99.007**	**96.267**
7	0.010	0.005	0.037	0.050	99.007	96.267
8	0.009	0.004	0.044	0.054	99.067	95.600
9	0.009	0.004	0.052	0.050	99.081	94.800
10	0.009	0.004	0.044	0.049	99.141	95.600
11	0.009	0.004	0.048	0.050	99.111	95.200
12	0.008	0.004	0.040	0.046	99.230	96.000

Pima dataset

# Nodes	Error training	Standard dev	Error test	Stand dev	% Training	% Test
2	0.239	0.024	0.260	0.051	76.068	74.001
3	0.227	0.010	0.248	0.043	77.341	75.218
4	**0.221**	**0.007**	**0.240**	**0.049**	**77.873**	**75.950**
5	0.218	0.006	0.247	0.047	78.224	75.252
6	0.215	0.006	0.243	0.047	78.516	75.716
7	0.213	0.006	0.244	0.045	78.675	75.638
8	0.212	0.006	0.242	0.044	78.814	75.796

Sonar dataset

# Nodes	Error training	Standard dev	Error test	Stand dev	% Training	% Test
2	0.247	0.020	0.282	0.111	75.299	71.762
3	0.235	0.019	0.261	0.114	76.517	73.910
4	0.222	0.015	0.283	0.097	77.756	71.705
5	0.219	0.020	0.285	0.092	78.120	71.514
6	0.210	0.013	0.279	0.099	78.952	72.114
7	0.206	0.016	0.271	0.090	79.412	72.948
8	**0.201**	**0.012**	**0.249**	**0.098**	**79.882**	**75.086**

Vehicle dataset

# Nodes	Error training	Standard dev	Error test	Stand dev	% Training	% Test
4	0.432	0.021	0.446	0.048	56.782	55.391
5	0.400	0.019	0.415	0.054	60.032	58.489
6	0.389	0.021	0.405	0.054	61.135	59.520
7	0.369	0.026	0.381	0.045	63.092	61.912
8	0.357	0.019	0.386	0.043	64.303	61.390
9	0.343	0.016	0.370	0.050	65.721	62.954
10	0.332	0.016	0.350	0.047	66.840	64.979
11	0.318	0.015	0.353	0.046	68.201	64.703
12	0.311	0.013	0.334	0.045	68.873	66.645
13	0.307	0.010	0.326	0.041	69.320	67.414
14	0.296	0.014	0.318	0.038	70.433	68.201
15	0.292	0.014	0.316	0.044	70.846	68.433
16	**0.287**	**0.012**	**0.308**	**0.047**	**71.253**	**69.192**

Wbcd dataset

# Nodes	Error training	Standard dev	Error test	Stand dev	% Training	% Test
2	0.024	0.004	0.032	0.023	97.581	96.769
3	0.023	0.002	0.033	0.021	97.743	96.741
4	0.023	0.002	0.033	0.021	97.749	96.739
5	**0.022**	**0.002**	**0.029**	**0.018**	**97.797**	**97.083**
6	0.022	0.002	0.032	0.019	97.794	96.792
7	0.022	0.002	0.033	0.020	97.803	96.740
8	0.022	0.002	0.029	0.020	97.813	97.054

Wine dataset

# Nodes	Error training	Standard dev	Error test	Stand dev	% Training	% Test
3	0.008	0.006	0.051	0.060	99.189	94.915
4	0.004	0.004	0.057	0.066	99.575	94.261
5	0.003	0.003	0.043	0.061	99.738	95.732
6	0.002	0.003	0.038	0.045	99.813	96.157
7	**0.001**	**0.002**	**0.033**	**0.046**	**99.913**	**96.739**
8	0.001	0.002	0.036	0.040	99.950	96.412
9	0.000	0.001	0.037	0.046	99.975	96.281
10	0.000	0.000	0.045	0.054	100.000	95.484
11	0.000	0.001	0.038	0.047	99.975	96.196
12	0.000	0.000	0.038	0.050	100.000	96.190

Appendix 2: GeneticRBFN description

To test our cooperative–competitive method against a Pittsburgh based proposal, a typical genetic algorithm for the RBFN design, GeneticRBFN, has been developed. The design lines of GeneticRBFN are the classical ones for these kinds of algorithms (Harpham et al. 2004). In order to establish similar operating conditions certain characteristics of CO^{2}RBFN have been introduced in GeneticRBFN, like analogies in the operators and the HVDM dissimilarity measure.

This method follows the traditional Pittsburgh evolutionary approach for the design of RBFNs: each individual is a whole network. The objective of the evolutionary process is to minimise the classification error. The main steps of this algorithm are shown in Fig. 5.

The main components of GeneticRBFN algorithm are described below.

Initialization

The initialization stage is the same as that in CO^{2}RBFN. So the RBFs will be centred, in an equidistributed way, for each RBFN/individual.

Genetic operators: selection, recombination and mutation

With the crossover operator, two individuals/RBFNs parents are randomly chosen to obtain an RBFN offspring.

```
1. RBFN Initialization
2. Selection
3. Recombination
4. Mutation
5. RBFN Training/Evaluation
6. If the stop-condition is not verified go to step 2.
```

Fig. 5 Main steps of GeneticRBFN algorithm

The number of RBFs of the new individual will be delimited between a minimum and a maximum value. The minimum value is set to the number of RBFs of the parent with fewer RBFs. In the same way, the maximum value is set to the number of RBFs of the parent with more RBFs. In order to generate the offspring RBFs will be chosen in a random way from the parents.

Six mutation operators, usually considered in the specialised bibliography (Harpham et al. 2004) have been implemented. They can be classified as random operators or biased operators. The random operators are:

- *DelRandRBFs*: randomly eliminates k RBFs, where k is a *pm* percent of the total number of RBFs in the RBFN.
- *InsRandRBFs*: randomly aggregates k RBFs, where k is a *pm* percent of the total number of RBFs in the RBFN.
- *ModCentRBFs*: randomly modifies the centre of k RBFs, where k is a *pm* percent of the total number of RBFs in the RBFN. The centre of the basis function will be modified in a *pr* percent of its width.
- *ModWidtRBFs*: randomly modifies the centre of k RBFs, where k is a *pm* percent of the total number of RBFs in the RBFN. The width of the basis function will be modified in a *pr* percent of its width.

Biased operators, which exploit local information are:

- *DelInfRBFs*: deletes the k RBFs of the RBFN with a lower weight. k is a *pm* percent of the total number of RBFs in the RBFN.
- *InsInfRBFs*: inserts the k RBFs in the RBFN outside the width of any RBF present in the RBFN. k is a *pm* percent of the total number of RBFs in the RBFN.

An intermediate population with the parents and the offspring is considered and a tournament selection mechanism is used to determine the new population. The diversity of the population is promoted by using a low value for the tournament size ($k = 3$).

Training weights

In order to train the weights, the LMS algorithm is used. Its parameters are set to their standard values.

Individual evaluation

The fitness defined for each individual/RBFN is its classification error for the given problem.

In order to increase the efficiency of the GA, the search space of this method has been drastically reduced. As is well known, in Pittsburgh GAs, where the only objective to optimise is the classification error, the complexity of the individuals (i.e. number of RBFs) grows in an uncontrolled way (because normally an RBFN with more RBFs gives a lower error percentage than an RBFN with few RBFs). In this experimentation, the search space has been reduced by fixing the maximum complexity (and so chromosome size) between a minimum and a maximum number of RBFs. The minimum number of RBFs has been set to the number of classes for the problem and the maximum to four times this number.

Appendix 3: Statistical methods

Non-parametric methods are often referred to as *distribution free* methods, as they do not rely on assumptions that the data are drawn from a given probability distribution.

Non-parametric methods are widely used for studying populations which take a ranked order (such as movie reviews receiving one to four stars). The use of non-parametric methods may be necessary when data has a ranking but no clear numerical interpretation, such as when assessing preferences.

As non-parametric methods make fewer assumptions, their applicability is much wider than the corresponding parametric methods. In particular, they may be used in situations where less is known about the application in question. In addition, due to the reliance on fewer assumptions, non-parametric methods are more robust.

Another justification for the use of non-parametric methods is simplicity. In certain cases, even when the use of parametric methods is justified, non-parametric methods may be easier to use. Due both to this simplicity and to their greater robustness, non-parametric methods are seen by some statisticians as leaving less room for improper use and misunderstanding.

Now we will explain the non-parametric methods used:

Friedman's test

This is a non-parametric equivalent of the test of repeated-measures ANOVA. It computes the ranking of the observed results for algorithm (r_j for the algorithm j with k algorithms) for each data-set, assigning to the best the ranking 1, and to the worst the ranking k. Under the null hypothesis, formed from supposing that the results of the algorithms are equivalents and, therefore, their rankings are also similar, Friedman's statistic

$$\chi_F^2 = \frac{12N_{ds}}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right],$$

is distributed according to χ_F^2 with $k-1$ degrees or freedom, being $R_j = \frac{1}{N_{ds}}\sum_i r_i^2$, and N_{ds} the number of data-set. The critical values for Friedman's statistic coincide with those established in the χ^2 distribution when $N_{\text{ds}} > 10$ and $k > 5$.

Iman and Davenport's test

This is a metric derived from Friedman's statistic, given that this last metric produces a conservative undesirable effect. The statistic is:

$$F_F = \frac{(N_{\text{ds}} - 1)\chi_F^2}{N_{\text{ds}}(k-1) - \chi_F^2},$$

and it is distributed according to a F-distribution with $k - 1$ and $(k - 1)(N_{\text{ds}} - 1)$ degrees of freedom.

Wilcoxon's signed-rank test

This is analogous to the paired t test in non-parametrical statistical procedures; therefore, it is a pair test that aims to detect significant differences between the behaviour of two algorithms.

Let d_i be the difference between the performance score of the two classifiers on ith out of N_{ds} data-sets. The differences are ranked according to their absolute values; average ranks are assigned in case of ties. Let R^+ be the sum of ranks for the data-sets is which the first algorithm outperformed the second, and R^- the sum of ranks for the opposite. For ranks of $d_i = 0$ are split evenly the sums; if there is an odd number of them, one is ignored:

$$R^+ = \sum_{d_i > 0} \text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \text{rank}(d_i),$$

$$R^- = \sum_{d_i < 0} \text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \text{rank}(d_i).$$

Let T be the smallest of the sums, $T = \min(R^+, R^-)$. If T is less than or equal to the value of the distribution of Wilcoxon for N_{ds} degrees of freedom, the null hypothesis of equality of means is rejected.

References

Alcalá-Fdez J, Sánchez L, García S, Del Jesus MJ, Ventura S, Garrell JM, Otero J, Romero C, Bacardit J, Rivas VM, Fernández JC, Herrera F (2009) KEEL: a software tool to assess evolutionary algorithms to data mining problems. Soft Comput 13(3): 307–318

Ampazis N, Perantonis SJ (2002) Two highly efficient second-order algorithms for training feedforwards networks. IEEE Trans Neural Netw 13(3):1064–1074

Asuncion A, Newman DJ (2007) UCI machine learning repository. School of Information and Computer Science, University of California, Irvine, CA. http://www.ics.uci.edu/~mlearn/MLRepository.html

Bäck T, Hammel U, Schwefel H (1997) Evolutionary computation: comments on the history and current state. IEEE Trans Evol Comput 1(1):3–17

Balakrishnan K, Honavar V (1995) Evolutionary design of neural architecture—a preliminary taxonomy and guide to literature. Technical report, AI Research Group, CS-TR 95–01

Billings SA, Zheng GL (1995) Radial basis function network configuration using genetic algorithms. Neural Netw 8(6):877–890

Broomhead D, Lowe D (1988) Multivariable functional interpolation and adaptive networks. Complex Syst 2:321–355

Buchtala O, Klimek M, Sick B (2005) Evolutionary optimization of radial basis function classifiers for data mining applications. IEEE Trans Syst Man Cybern B 35(5):928–947

Burdsall B, Giraud-Carrier C (1997) GA-RBF: a self optimising RBF network. In: Proceedings of conference on artificial neural networks and genetic algorithms. Springer, Berlin

Chaiyaratana N, Zalzala AMS (1998) Evolving hybrid RBF-MLP networks using combined genetic/unsupervised/supervised learning. In: Proceedings of UKACC international conference on control, Swansea, UK

Chen S, Billings SA, Cowan CFN, Grant PW (1990) Practical identification of narmax models using radial basis functions. Int J Control 52(6):1327–1350

Chen S, Cowan C, Grant P (1991) Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans Neural Netw 2:302–309

Chen S, Wu Y, Luk BL (1999) Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks. IEEE Trans Neural Netw 10(5):1239–1243

Cheng V, Li CH, Kwok JT, Li CK (2004) Dissimilarity learning for nominal data. Pattern Recogn 37:1471–1477

Dawson CW, Wilby RL, Harpham C, Brown MR, Cranston, E, Darby EJ (2000) Modelling Ranunculus presence in the Rivers test and Itchen using artificial neural networks. In: Proceedings of international conference on geocomputation, Greenwich, UK

Deb K (2001) Multi-objective optimization using evolutionary algorithms, 1st edn. Wiley, New York

Demšar J (2006) Statistical Comparisons of Classifiers over Multiple Data Sets. J Mach Learn Res 7:1–30

Du H, Zhang N (2008) Time series prediction using evolving radial basis function networks with new enconding scheme. Neurocomputing 71:1388–1400

Er MJ, Chen W, Wu S (2005) High-speed face recognition base on discrete cosine transform and RBF neural networks. IEEE Trans Neural Netw 16(3):679–691

Esposito F, Malerba D, Tamma V, Bock HH (2000a) Classical resemblance measures. In: Bock H-H, Diday E (eds) Analysis of symbolic data. Exploratory methods for extracting statistical information from complex data, Series: studies in classification, data analysis, and knowledge organization, vol 15. Springer, Berlin, pp 139–152

Esposito A, Marinaro M, Oricchio D, Scarpetta S (2000b) Approximation of continuous and discontinuous mappings by a growing neural RBF-based algorithm. Neural Netw 13(6):651–665

Ferreira PM, Ruano AE, Fonseca CM (2003) Genetic assisted selection of RBF model structures for greenhouse inside air temperature prediction. IEEE Control Appl 1:576–581

Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial Intelligence through simulation evolution. Wiley, New York

Fu X, Wang L (2002) A GA-based novel RBF classifier with class-dependent features. Proc Congr Evol Comput 2:1964–1969

Fu X, Wang L (2003) Data dimensionality reduction with application to simplifying RBF network structure and improving classification performance. IEEE Trans Syst Man Cybern B 33(3):399–409

Goldberg D (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading

Goldberg D, Richardson J (1987) Genetic algorithms with sharing for multimodal function optimization. In: Grefenstette (ed) Proceedings of second international conference on genetic algorithms. Lawrence Erlbaum Associates, pp 41–49

Golub G, Van Loan C (1996) Matrix computations, 3rd edn. J. Hopkins University Press, Baltimore

González J, Rojas I, Ortega J, Pomares H, Fernández FJ, Díaz AF (2003) Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation. IEEE Trans Neural Netw 14(6): 1478–1495

Guillén A, Pomares H, Rojas I, González J, Herrera LJ, Rojas F, Valenzuela O (2007) Output value-based initialization for radial basis function neural networks. Neural Process Lett. doi: 10.1007/s11063-007-9039-8

Harpham C, Dawson C, Brown M (2004) A review of genetic algorithms applied to training radial basis function networks. Neural Comput Appl 13:193–201

Holcomb T, Morari M (1991) Local training for radial basis function networks: towards solving the hidden unit problem. In: Proceedings of American control conference, Boston

Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press

Huang SN, Tan KK, Lee TH (2008) Adaptive neural network algorithm for control design of rigid-link electrically driven robots. Neurocomputing 71(4–6):885–894

Jang JSR, Sun CT (1993) Functional equivalence between radial basis functions and fuzzy inference systems. IEEE Trans Neural Netw 4:156–158

Jiang N, Zhao ZY, Ren LQ (2003) Design of structural modular neural networks with genetic algorithm. Adv Eng Soft 1:17–24

Jin Y, Sendhoff B (2003) Extracting interpretable fuzzy rules from RBF networks. Neural Process Lett 17(2):149–164

Lacerda E, Carvalho A, Braga A, Ludermir T (2005) Evolutionary radial functions for credit assessment. Appl Intell 22:167–181

Lee S, Kil RM (1991) A Gaussian potential function network with hierarchically seft-organising learning. Neural Netw 4:207–224

Leung H, Dubash N, Xie N (2002) Detection of small objects in clutter using a GA-RBF neural network. IEEE Trans Aero Electr Sys 38(1):98–118

Li M, Tian J, Chen F (2008) Improving multiclass pattern recognition with a co-evolutionary RBFNN. Pattern Recogn Lett 29(4):392–406

Maglogiannis I, Sarimveis H, Kiranoudis CT, Chatziioannou AA, Oikonomou N, Aidinis V (2008) Radial basis function neural networks classification for the recognition of idiopathic pulmonary fibrosis in microscopic images. IEEE Trans Inf Technol B 12(1):42–54

Mandani E, Assilian S (1975) An experiment in linguistic synthesis with a fuzzy logic controller. Int J Man Mach Stud 7(1):1–13

Marcos JV, Hornero R, Álvarez D, Del Campo F, López M, Zamarrón C (2008) Radial basis function classifiers to help in the diagnosis of the obstructive sleep apnoea syndrome from nocturnal oximetry. Med Biol Eng Comput 46:323–332

Moechtar M, Farag AS, Hu L, Cheng TC (1999) Combined genetic algorithms and neural network approach for power system transient stability evaluation. Europ Trans Elect Power 9(2):115–122

Moody J, Darken CJ (1989) Fast learning in networks of locally-tuned processing units. Neural Comput 1:281–294

Musavi MT, Ahmed W, Chan KH, Faris KB, Hummels DM (1992) On the training of radial basis function classifiers. Neural Netw 5:595–603

Neruda R, Kudová P (2005) Learning methods for radial basis function networks. Future Gener Comp Sy 21(7):1131–1142

Orr MJL (1995) Regularization on the selection of radial basis function centers. Neural Comput 7:606–623

Park J, Sandberg I (1991) Universal approximation using radial-basis function networks. Neural Comput 3:246–257

Park J, Sandberg I (1993) Universal approximation and radial basis function network. Neural Comput 5(2):305–316

Pedrycz W (1998) Conditional fuzzy clustering in the design of radial basis function neural networks. IEEE Trans Neural Netw 9(4):601–612

Peng JX, Li K, Huang DS (2006) A hybrid forward algorithm for RBF neural network construction. IEEE Trans Neural Netw 17(6):1439–1451

Pérez-Godoy MD, Rivera AJ, del Jesus MJ, Rojas I (2007) CoEvRBFN: an approach to solving the classification problem with a hybrid cooperative–coevolutive algorithm. In: Proceedings of international workshop on artificial neural networks, pp 324–332

Pérez-Godoy MD, Aguilera JJ, Berlanga FJ, Rivas VM, Rivera AJ (2008) A preliminary study of the effect of feature selection in evolutionary RBFN design. In: Proceedings of information processing and management of uncertainty in knowledge-based system, pp 1151–1158

Plat J (1991) A resource allocating network for function interpolation. Neural Comput 3(2):213–225

Potter M, De Jong K (2000) Cooperative coevolution: an architecture for evolving coadapted subcomponents. Evol Comput 8(1):1–29

Powell M (1985) Radial basis functions for multivariable interpolation: a review. In: IMA Proceedings of conference on algorithms for the approximation of functions and data, pp 143–167

Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kauffman, Menlo Park

Rivas VM, Merelo JJ, Castillo PA, Arenas MG, Castellanos JG (2004) Evolving RBF neural networks for time-series forecasting with EvRBF. Inf Sci 165(3–4):207–220

Rivera AJ, Ortega J, Prieto A (2001) Design of RBF networks by cooperative/competitive evolution of units. In: Proceedings of international conference on artificial neural networks and genetic algorithms (ICANNGA 2001), pp 375–378

Rivera AJ, Rojas I, Ortega J, del Jesus MJ (2007) A new hybrid methodology for cooperative–coevolutionary optimization of radial basis function networks. Soft Comput. doi:10.1007/s00500-006-0128-9

Rojas R, Feldman J (1996) Neural networks: a systematic introduction. Springer, Berlin

Rojas I, Valenzuela O, Prieto A (1997) Statisctical analysis of the main parameters in the definition of radial basis function networks. Lect Notes Comput Sci 1240:882–891

Sanchez VD (2002) A searching for a solution to the automatic RBF network design problem. Neurocomputing 42:147–170

Schaffer JD, Whitley D, Eschleman LJ (1992) Combinations of genetic algorithms and neural networks: a survey of the state of the art. In: Proceedings of international workshop on combinations of genetic algorithms and neural networks

Sergeev SA, Mahotilo KV, Voronovsky GK, Petrashev SN (1998) Genetic algorithm for training dynamical object emulator based on RBF neural network. Int J Appl Electro Mech 9(1):65–74

Sheta AF, De Jong K (2001) Time-series forecasting using GA-tuned radial basis functions. Info Sci 133(3–4):221–228

Stanfill C, Waltz D (1986) Towards memory-based reasoning, Commun. ACM 29(12):1213–1228

Sumathi S, Sivanandam SN, Ravindran R (2001) Design of a soft computing hybrid model classifier for data mining applications. Engin Intell Sys Electr Engin Comm 9(1):33–56

Sun YF, Liang YC, Zhang WL, Lee HP, Lin WZ, Cao LJ (2005) Optimal partition algorithm of the RBF neural network and its application to financial time series forecasting. Neural Comput Appl 14(1):36–44

Sundararajan N, Saratchandran P, Yingwei L (1999) Radial basis function neural network with sequential learning: MRAN and its application. World Scientifics, New York

Teixeira CA, Ruano MG, Ruano AE, Pereira WCA (2008) A soft-computing methodology for noninvasive time-spatial temperature estimation. IEEE Trans Bio Med Eng 55(2):572–580

Topchy A, Lebedko O, Miagkikh V, Kasabov N (1997) Adaptive training of radial basis function networks based on co-operative evolution and evolutionary programming. In: Proceedings of international conference neural information processing (ICONIP), pp 253–258

Vesin JM, Gruter R (1999) Model selection using a simplex reproduction genetic algorithm. Sig Process 78:321–327

Whitehead B, Choate T (1996) Cooperative–competitive genetic evolution of radial basis function centers and widths for time series prediction. IEEE Trans Neural Netw 7(4):869–880

Widrow B, Lehr MA (1990) 30 Years of adaptive neural networks: perceptron, madaline and backpropagation. Proc IEEE 78(9):1415–1442

Wilson DR, Martinez TR (1997) Improved heterogeneous distance functions. J Artif Intell Res 6(1):1–34

Xue Y, Watton J (1998) Dynamics modelling of fluid power systems applying a global error descent algorithm to a selforganising radial basis function network. Mechatronics 8(7):727–745

Yao X (1993) A review of evolutionary artificial neural networks. Int J Intell Syst 8(4):539–567

Yao X (1999) Evolving artificial neural networks. Proc IEEE 87(9):1423–1447

Yen GG (2006) Multi-Objective evolutionary algorithm for radial basis function neural network design. Stud Comput Intell 16:221–239