Available online at www.sciencedirect.com

ScienceDirect

Neurocomputing 69 (2006) 2268–2283

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Evolutionary system for automatically constructing and adapting radial basis function networks

Daniel Manrique*, Juan Ríos, Alfonso Rodríguez-Patón

Department Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, 28660 Boadilla del Monte, Madrid, Spain

Received 31 October 2004; received in revised form 2 June 2005; accepted 15 June 2005
Available online 20 December 2005
Communicated by H. Degaris

Abstract

This article presents a new system for automatically constructing and training radial basis function networks based on original evolutionary computing methods. This system, called Genetic Algorithm Radial Basis Function Networks (GARBFN), is based on two cooperating genetic algorithms. The first algorithm uses a new binary coding, called basic architecture coding, to get the neural architecture that best solves the problem. The second, which uses real coding, takes its inspiration from mathematical morphology theory and trains the architectures output by the binary genetic algorithm. This system has been applied to a laboratory problem and to breast cancer diagnosis. The results of these evaluations show that the overall performance of GARBFN is better than other related approaches, whether or not they are based on evolutionary techniques.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Genetic algorithms; Radial basis function networks; Self-adaptive intelligent system; Breast cancer diagnosis

1. Introduction

One of the main fields of artificial intelligence research is the development of self-adaptive systems capable of transforming to solve different problem types [17]. On account of their pattern-based learning, data generalization and noise filtering capabilities [18,23], neural networks are commonly used within artificial intelligence to perform real-world tasks.

Radial basis function networks (RBFN) are a type of network that is very useful for pattern classification problems [27]. This is because, unlike the multilayer perceptron (MLP) [23] whose output is generated by the action of all the neurons in the network weighted by the weights of its connections, the output of a RBFN is mainly influenced by the hidden layer neuron, whose centre is closer to the input pattern. Therefore, RBFNs are local approximators, whereas MLPs are global approximators [13].

The benefits of using a RBFN are: (i) local data approximation uses few hidden units for any input; (ii) the hidden and output layer parameters can be trained separately using a hybrid algorithm, and (iii) only one, non-linear, hidden layer is used, whereas the output is linear. As only one hidden layer is used, they converge faster than a MLP with several hidden layers [27].

Despite the advantages of RBFNs, the design of an optimal architecture to solve a particular problem is far from being a straightforward matter [15]. Additionally, RBFNs trained according conventional gradient descent methods have been shown to be more likely to get trapped in local optima, and they are, therefore, less accurate than when applied to MLPs [4]. This is because, as far as RBFNs are concerned, apart from finding a set of weights for connections that best solve the problem, the centres of the radial basis functions of the hidden layer neurons have to be searched.

Several research papers have, therefore, focused on designing new approaches based on different search and optimization techniques to choose the best neural architecture to solve a given problem and to speed up the training process. Some of these studies deal with the

*Corresponding author. Tel.: +34 91 336 6907; fax: +34 352 4819.
E-mail addresses: dmanrique@fi.upm.es (D. Manrique), jrios@fi.upm.es (J. Ríos), arpaton@fi.upm.es (A. Rodríguez-Patón).

0925-2312/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2005.06.018

so-called incremental algorithms [26], which start from a predefined neural architecture and then dynamically add and remove neural connections during the training process. The performance of these algorithms is low and they tend to converge prematurely depending on the original architecture chosen. They cannot, therefore, guarantee a good solution [22].

To surmount the problem of trapping in local optima and improve the accuracy of the results of applying gradient backpropagation to RBFN, research has been conducted aimed at building hybrid training algorithms [7]. These algorithms combine an unsupervised search of the hidden layer neuron centres using clustering algorithms, such as k-means, or improved versions, like moving k-means, with gradient backpropagation to get the weights of the connections [19]. However, these approaches are still beset by the very same weakness of local optima trapping, because they use derivatives-based optimization methods.

Other more promising studies originate from the identification of synergies between evolutionary algorithms and artificial neural networks that can be combined in various ways. These include works related to the genetic adaptation of the internal structure of the network [3,16,25]. Genetic algorithms have also been used to partially replace the network learning method, first applying genetic algorithms to accomplish global search until a point near the solution is reached and then running a local search with classical gradient descent methods to get the optimum solution [5,27]. Other numerical approaches, like regularized orthogonal least squares can also be used [6] instead of gradient descent methods. The snag with all these approaches is that it is not known what is the best time to switch from global to local search.

For any chosen evolutionary optimization approach, the way in which the neural networks that make up the search space are encoded is a crucial step in automatic network design [14]. Therefore, several approaches have been developed to produce efficient codifications of artificial neural networks, and specifically RBFNs. The first of these is the direct binary encoding of network configuration [10], where each bit determines the presence or absence of a single connection. There are two major problems with this approach. First, convergence performance is degraded as the number of neurons in the hidden layer increases because the search space is much larger. Second, direct encoding methods cannot prevent illegal points in the search space (architectures that are not permitted for a RBFN). At the other end of the scale from the direct encoding methods are other approaches to RBFN codification, where there is no direct correspondence between each bit of the string and each connection of the neural architecture. These are called indirect encoding methods, of which the graph generation system is the method for which the best results have been reported [16]. This approach is based on the binary codification of grammars that describe the architecture of the network and prevent the codification of illegal neural architectures. The problem here is that a one-bit variation in a string results in a totally different network. This degrades the convergence process of the genetic algorithm that uses this codification.

Training RBFNs can be seen as an optimization problem, where the mean square error has to be minimized by adjusting the values of the weights of the connections and the centres of the neurons in the hidden layer. Evolutionary algorithms are therefore a suitable option for dealing with this problem. Michalewicz states that if a problem is real-valued in nature, then a real number genetic algorithm is faster and more precise than a binary encoded genetic algorithm [20]. Therefore, genetic algorithms using real number codification to represent the weights of the neural network can be expected to yield the best results. There is a variety of techniques for handling real-coded genetic algorithms. Radcliffe's flat crossover chooses parameters for an offspring by uniformly picking parameter values from (inclusively) the two parents' parameter values [24]. BLX-α was then proposed [11] to work around the premature convergence problems of this operator. BLX-α uniformly picks values that lie between two points that contain the two parents and may extend equally on either side of the interval defined by the parents. This new method, however, is very slow at approximating to the optimum because the extension of the interval defined by the parents is determined by a static, user-specified parameter α set at the start of the run. Another important crossover technique for real-coded genetic algorithms is UNDX [21], which can optimize functions by generating offspring using the normal distribution defined by three parents. The problem here is the high computational cost required to calculate the normal distribution. Other research combines statistical methods, pruning and real-coded genetic algorithms [13], although the problem is, again, the computational cost of calculating the Bayesian regularization on which this algorithm is based.

This paper presents a new evolutionary system, called Genetic Algorithm Radial Basis Function Networks (GARBFN), which automatically designs and trains RBFNs to solve a given problem stated as a set of training patterns. The work presented here is the basis for automatically building RBFN-based self-adaptive intelligent systems. GARBFN consists of a binary-coded genetic algorithm that searches for neural architectures in combination with a hybrid training method that employs a k-mean clustering algorithm to ascertain the centres of the neurons in the hidden layer, and a real-coded genetic algorithm rather than any of the other conventional methods to adjust the weights of the connections.

The binary-coded genetic algorithm employs the basic architectures codification method [3], which has been adapted to work on radial basis function architectures. A specialized binary crossover operator—the RBFN crossover (RBFN-X)—has also been designed to work with the proposed codification method. This operator outperforms other conventional binary crossover operators in searching for radial basis function architectures.

The real-coded genetic algorithm is an extension and adaptation of RBFNs from the Mathematical Morphology Crossover Operator (MMX) [2]. It employs a specific crossover operation (known as real crossover operator) that is based on mathematical morphology theory [8,9] and typically works for image processing. Specifically, this operator is based on the morphological gradient, adapted to give a measure of genetic diversity. This feature allows the genetic algorithm to dynamically focus or generalize the search for the optimum, thereby achieving balanced exploitation and exploration capabilities that make trapping in local optima less likely, while getting a high convergence speed.

Experimental lab tests have been run on the basic architectures codification method applied to RBFN, working together with the RBFN crossover. Results are compared to other codification methods and binary crossover operators, showing that performance is better for the proposed approach. Another battery of tests were chosen to show that the real crossover operator is able to train faster converging RBFNs that are less likely to get trapped in local optima than other related crossover operators and the gradient descent method.

Both designs, binary and real-coded genetic algorithms, were deployed to work together as a system that automatically constructs self-adaptive intelligent systems with RBFNs from a set of training patterns. This system has been successfully applied to a real-world problem: breast cancer diagnosis, consisting on analysing suspicious masses and microcalcifications in breast tissue suspected of being carcinomas. The training and test patterns have been extracted from a database of real patients stored at a Madrid hospital.

1.1. *Radial basis function networks: theoretical background*

A RBFN [12] is composed of three connected layers that do not necessarily have to be but generally are fully connected. The first layer does not process the input data at all, it just sends them to the hidden layer. These data are mapped non-linearly by a radial basis function in the hidden layer. Finally, the data are sent to the output layer through weighted connections. The H neurons of this layer have a linear threshold activation function that provides the network output as follows:

$$y_i = \sum_{j=0}^{H} w_{ij} \cdot \varphi_j(x) \quad \text{for} \quad \begin{cases} w_{i0} = \theta_i, \\ \varphi_0(x) = -1, \end{cases}$$

where θ_i is the threshold for i, w_{ij} is the connection weight between the hidden neuron j and the output neuron i and $\varphi_j(x)$ is the radial basis function applied at neuron j of the hidden layer to input x.

Although there are several radial basis functions, the Gaussian function provides the best results for pattern classification [25]:

$$\varphi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \quad \text{for some } \sigma > 0 \text{ and } \mathrm{r} \in \Re,$$

where σ is the standard deviation, which represents how wide the function $\varphi(x)$ is, and $r = \|x - c\|$, with $\|\cdot\|$ denotes a generally Euclidean norm, in which x is the network input variable and c is the centre (mean) of the Gaussian function.

2. Training radial basis function networks with genetic algorithms

First, the k-means clustering algorithm is employed to find the centres of the neurons in the hidden layer. Then, a specifically designed genetic system to employ real number encoding is used to train the RBFN. The individuals (chromosomes) of the population are strings of real numbers that represent the weights of the connections between the hidden and output layers, and their respective thresholds. Given a fully connected RBFN with H neurons in the hidden layer and O neurons in the output layer, the chromosome contains, first, the weights of the connections between these two layers and, second, the thresholds of the output neurons. Fig. 1 shows an individual in the population with the meaning of each of its genes: W_{ij}, $i = 1, \ldots, O$; $j = 1, \ldots, H$ represents the weight of the connection between the jth neuron of the hidden layer and the ith neuron of the output layer and $\theta_1, \theta_2, \ldots, \theta_O$ are the thresholds of the output neurons.

The genetic algorithm that uses this code employs the roulette-wheel method as the selection algorithm. Mutation involves replacing one of the individual's genes chosen at random by another real number that is also picked at random. The new individuals of the offspring engendered by the crossover operator replace the worst individuals of the population (a particular implementation of the ready-state genetic algorithm). This algorithm applies the *real crossover operator*, especially designed for real-number encoding to train the radial basis networks. So, the goal of the real-coded genetic algorithm is to find a combination of weights and thresholds (an individual) that minimizes the mean square error for the set of training patterns.

W_{11}	W_{21}	...	W_{O1}	W_{12}	W_{22}	...	W_{O2}	...	W_{1H}	W_{2H}	...	W_{OH}	θ_1	θ_2	...	θ_O

Fig. 1. Genotype of the individuals in the real-coded genetic algorithm.

2.1. The real crossover operator

Let $D_{\Re}$ be a point in the search space, which is represented by the string $s=(a_0, a_1, \ldots, a_{l-1})$, where $a_i \in \Re$. The real crossover operator works on each of the parents' genes independently to get the respective gene for the two descendants. To do the crossover, an odd number n of strings are taken from the population. These form the progenitor matrix X, of dimensions n by l, where l is the chromosome length. X is defined as

$$X = \begin{pmatrix} a_{10} & a_{11} & \cdots & a_{1l-1} \\ a_{20} & a_{21} & \cdots & a_{2l-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n0} & a_{n1} & \cdots & a_{nl-1} \end{pmatrix},$$

where $(a_{i0}, a_{i1}, \ldots, a_{i,l-1}) = s_i$, for $i=1,\ldots,n$.

Each of the columns $f_i=(a_{1i}, a_{2i}, \ldots, a_{ni})$ of matrix X is processed to get genes o_i and o'_i. So, the two descendants engendered from the application of this operator are $o=(o_0, o_1, \ldots, o_{l-1})$ and $o'=(o'_0, o'_1, \ldots, o'_{l-1})$, where o and $o' \in D_{\Re}$. The calculation of these two descendants is a three-step process:

1. *Calculate the genetic diversity g_i, for each gene a_i*:
The morphological gradient operator, $g_b(f_i)$: $D_{fi} \to \Re$, is applied to each vector (column) $f_i, i=0,1,\ldots,l-1$, with a structuring element b: $D_b \to \Re$, defined as $b(x)=0, \quad \forall x \in D_b, \quad D_b=\{-E(n/2), -E(n/2)-1, \ldots, 0, \ldots, E(n/2)\}$, and $E(x)$ being the integer part of x. With these premises, g_i is calculated as: $g_i = g_b(fi)(E(n/2)+1) i \in \{0,1,,l-1\}$. So, the genetic diversity of the ith gene is the value of the morphological gradient applied to the component located in the centre of the column vector f_i of the progenitor matrix.
The results yielded by the morphological gradient have been reinterpreted from its usual application to digitalized images. In this case, it provides a measure of "similarity" among the components of vector f_i. If these components are similar, then g_i is near to zero. On the other hand, if this value is higher (the maximum value is one), the components of f_i are scattered. As the individuals within matrix X are chosen from the actual population, f_i can be considered as a sample of the values for the ith gene in the population. Therefore, g_i is taken to be a measure of the heterogeneity of the ith gene in the population. If this value is high, then the population is scattered, and more exploration capabilities are needed to find the solution. If g_i is near to zero, then this gene is converging, and more exploration capabilities may be needed to make trapping in local optima less likely. The morphological gradient is interpreted as an on-line measure of the genetic diversity of the population at very low computational cost.
2. *Calculate the crossover interval*:
Let φ be a function defined in the range [0,1] as φ: $\Re \to \Re$. Let $g_{i\max}$ be the maximum gene, defined as $g_{i\max} = \max(f_i) - \varphi(g_i)$. And let $g_{i\min}$ be the minimum gene, defined as $g_{i\min} = \min(f_i) + \varphi(g_i)$. The maximum and minimum genes determine the bounds of the crossover interval: $C_i = [g_{i\min}, g_{i\max}]$, from which the ith genes o_i and o'_i will be taken in step 3. The function φ dynamically (at genetic algorithm runtime) controls the range of the crossover interval C_i, depending on the genetic diversity g_i. This function is designed to reduce the range of the crossover interval with respect to the reference values $\min(f_i)$ and $\max(f_i)$ when the individuals to be crossed are diverse (for high values of g_i). This way, the genetic algorithm focuses the search for the optimum within the crossover interval, speeding up this process.
On the other hand, the range of the crossover interval is wider than the reference interval $[\min(f_i), \max(f_i)]$ when genetic diversity falls, because the genetic algorithm is converging. Therefore, the ith genes of the offspring o_i and o'_i could fall outside the reference interval. This way, the reference interval can be fully explored, making trapping in local optima less likely.
The function φ is defined as two straight-line equations according to four parameters a, b, c and d:

$$\begin{cases} \frac{g_i}{c} = \frac{\varphi(g_i)-a}{b-a}; & \text{if } g_i \leqslant c, \\ \frac{g_i-c}{1-c} = \frac{\varphi(g_i)}{d}; & \text{if } g_i > c, \end{cases} \text{equivalent to} : \varphi(g_i) = \begin{cases} \frac{(b-a)\cdot g_i}{c} + a; & \text{if } g_i \leqslant c, \\ \frac{d \cdot g_i - c \cdot d}{1-c}; & \text{if } g_i > c. \end{cases}$$

Parameters a, b, c and d define the behaviour of function φ and, therefore, depend on the problem type. However, a binary genetic algorithm was used to find the best values for these parameters to achieve the highest convergence speed, while making trapping in local optima less likely, for the actual task of training radial basis networks to solve some benchmark tests. The best results were achieved for $a=-0.001$, $b=-0.133$, $c=0.54$ and $d=0.226$. Consequently, the analytical expression for function φ is as follows:

$$\varphi(g_i) = \begin{cases} -(0.25 \cdot g_i) - 0.001 & \text{if } g_i \leqslant 0.54, \\ (0.5 \cdot g_i) - 0.265 & \text{if } g_i > 0.54. \end{cases}$$

This function does only one multiplication per gene. The crossover operator is, therefore, very efficient, as it only needs l multiplications to generate two new descendants (the length of the individuals).
3. *Generate offspring*:
For each crossover interval C_i calculated in the previous step, two new descendants, $o=(o_0, \ldots, o_{l-1})$ and $o'=(o'_0, \ldots, o'_{l-1})$, are generated as follows:

- Each gene o_i belonging to o is randomly chosen from within C_i.
- o'_i, belonging to o' is calculated from o_i using the following formula:
- $o'_i = \min(f_i) + \max(f_i) - o_i$.

These two new genes, o_i and o'_i, are symmetric with respect to the central point of the crossover interval, which is a desirable rule:

$$o_i + o'_i = \min(f_i) + \max(f_i) = g_{i\min} + g_{i\max}.$$

3. Designing radial basis function networks with genetic algorithms

The automatic design of radial basis neural architectures is accomplished by a genetic algorithm that employs a new binary encoding for networks like these. The goal of this genetic algorithm is to search for the best architecture for solving a classification problem given a set of training patterns. The RBFN crossover operator has been specifically designed to work with this encoding. The RBFN-X is based on the same principles as the real crossover operator, although, this time, applied to binary encoding. Therefore, it shares the benefits of this operator.

3.1. Encoding RBF architectures

Definition 1. A RBFN architecture is considered as a set of input, hidden and output neurons and a set of connections between input and hidden neurons or hidden and output neurons. Formally, a RBFN architecture r with I input neurons, H units in only one hidden layer and O output neurons is defined as $r \subset (\bar{I} \times \bar{H}) \cup (\bar{H} \times \bar{O})$, where $\bar{I} = \{i_1, i_2, \ldots, i_I\}$ denotes the set of I input neurons, $H = \{h_1, h_2, \ldots, h_H\}$ is the set of H units in the hidden layer and $\bar{O} = \{o_1, o_2, \ldots o_O\}$ is the set of O output neurons. If $(a,b) \in r$, then the neuron a is connected to the neuron b. The Cartesian product of input and hidden neurons is $\bar{I} \times \bar{H}$, which represents the set of all possible connections from the input layer to the hidden layer and $\bar{H} \times \bar{O}$ is the set of all connections from the hidden layer to the output.

The set of all RBF architectures with a *maximum* of I input neurons, H hidden units and O output units is denoted by $R_{I,H,O}$. There is a special case, the null architecture, defined as $n = \varnothing$, where there are no connected neurons.

Definition 2. Of the set of RBFN architectures, we are concerned with the *valid* neural networks: if there exists a connection between an input and hidden neuron, then there has to be a connection from this hidden neuron to any output neuron. So, a RBFN architecture $v \in R_{I,H,O}$ with $v \subset (\bar{I} \times \bar{H}) \cup (\bar{H} \times \bar{O})$ is called a valid RBFN, and hence $v \in VR_{I,H,O}$, if and only if for all $(i_r, h_s) \in \bar{I} \times \bar{H} \cap v$, there exists $o_p \in \bar{O}$ such that $(h_s, o_p) \in \bar{H} \times \bar{O} \cap v$ and, reciprocally, for all $(h_s, o_p) \in \bar{H} \times \bar{O} \cap v$, there exists $i_r \in I$ such that $(i_r, h_s) \in \bar{I} \times \bar{H} \cap v$. From this it follows that the null architecture $n \in VR_{I,H,O}$ is also a valid RBFN architecture.

Definition 3. Let v, $v' \in VR_{I,H,O}$, then the superimposition operation between v and v' is defined as $v \oplus v' = v \cup v'$.

Thanks to the above definitions, the set $VR_{I,H,O}$ of valid RBF architectures, with the superimposition operation $\oplus$ (internal composition law in $VR_{I,H,O}$), can be said to form the algebraic structure of Abelian semi-group with neutral element (the null radial basis architecture), denoted as $(VR_{I,H,O,}\oplus)$. Hence, the result of the superimposition operation between two valid RFBN architectures is another valid RFBN architecture composed of all the neurons and connections there are in the original two.

Definition 4. A neural network within the set of valid RBFNs is also called *basic* if it is the null net or has exactly two connections: one from an input to a hidden neuron and the other from this hidden neuron to any output neuron. Formally, a valid RBF architecture $b \in VR_{I,H,O}$ is called *basic radial basis function architecture*, and hence, $b \in BR_{I,H,O}$, if and only if $\#b = 2$ or $\#b = 0$. $\#b$ denotes the cardinal of the set b. If $\#b = 2$, then $b = \{(i_r, h_s), (h_s, o_p)\}$ with $(i_r, h_s) \in \bar{I} \times \bar{H}$ and $(h_s, o_p) \in \bar{H} \times \bar{O}$. If $\#b = 0$, then $b = \varnothing$, which is the null RBF architecture. The subset $BR_{I,H,O} \subseteq VR_{I,H,O}$, composed of all the basic RBF architectures, has the property of being able to define any valid RBF architecture from the superimposition of these simple structures.

Definition 5. Let $v \in VR_{I,H,O}$ and $B = \{b_1, , b_k\} \subseteq BR_{I,H,O}$. If $v = b_1 \oplus \cdots \oplus b_k$ then B is called *decomposition* of v, thus the decomposition of a given network is the set of basic RBFNs needed to produce this network using the superimposition operation.

Theorem 1. *$\forall v \in VR_{I,H,O}$ there exists at least one subset $B = \{b_1, , b_k\} \subseteq BR_{I,H,O}$ such that B is a decomposition of v. That is, any valid RBF architecture is composed of the superimposition of elements of $BR_{I,H,O}$.*

Proof. Let $v \in VR_{I,H,O}$ with $v \subset (\bar{I} \times \bar{H}) \cup (\bar{H} \times \bar{O})$. As shown in the definition of valid RBF architectures (Definition 2), each pair of sets $b = \{(i_r, h_s), (h_s, o_p)\}$ is also a basic RBF architecture with $\#b = 2$. The decomposition of the null RBF architecture $v = \varnothing$ is itself. Therefore, v may be expressed as the superimposition of all basic RBF architectures as $\{(i_r, h_s), (h_s, o_p)\}$ such that $(i_r, h_s) \in \bar{I} \times \bar{H} \cap v$ and $(h_s, o_p) \in \bar{H} \times \bar{O} \cap v$. $\square$

Corollary 1. *If $B = \{b_1, , b_k\} \subseteq BR_{I,H,O}$ and $B' = \{b'_1, , b'_{k'}\} \subseteq BR_{I,H,O}$ are decompositions of $v \in VR_{I,H,O}$, then $B \cup B'$ is another decomposition of v.*

Definition 6. $\forall v \in VR_{I,H,O}$, the decomposition $M \subseteq BR_{I,H,O}$ is called *maximum decomposition*, if and only if $M = B_1 \cup B_2 \cup \cdots \cup B_n$, $B_1, , B_n \subseteq BR_{I,H,O}$ are possible decompositions of v.

Remark 1. According to the above definition, we find that $\forall v \in VR_{I,H,O}$ and there exists one and only one maximum decomposition of v.

3.2. *The cardinal of set of basic RBF architectures*

To encode the valid RBF architectures $VR_{I,H,O}$, the cardinal of the set $BR_{I,H,O}$, $\#BR_{I,H,O}$, subset of $VR_{I,H,O}$, first needs to be calculated. If $\#b = 2$, then there are connections between the input and hidden neurons and between the hidden neurons and the output neurons. Therefore, there are $I \cdot H \cdot O$ possible combinations. Taking into account that the null architecture is also basic, $\#b = 0$, the cardinal of $BR_{I,H,O}$ is $\#BR_{I,H,O} = I \cdot H \cdot O + 1$.

3.3. *Binary encoding of the set of valid RBF architectures*

It has been shown that any valid RBF architecture belonging to $VR_{I,H,O}$ can be built from the set of basic architectures $BR_{I,H,O}$ that contains exactly $I \cdot H \cdot O + 1$ elements. As the null architecture cannot build other valid architectures, there are $I \cdot H \cdot O$ basic RBF architectures that can be combined to build more complex architectures. The set of basic RBF architectures, excluding the null net is denoted by $BR = \{b_0, \ldots, b_{I \cdot H \cdot O}\}$. These architectures can be combined in $2^{I \cdot H \cdot O}$ different ways to build valid RBF networks. It follows from the above that there exists a one-to-one correspondence between the set of all possible decompositions of all valid RBF architectures and the set $VR^b_{I,H,O}$, composed of all binary strings of length $I \cdot H \cdot O$ that encode each decomposition of valid architectures.

From all these premises, it follows that the encoding of all points in the search space $VR_{I,H,O}$ is based on the codification of all basic RBF architectures $BR_{I,H,O} = \{b_0, b_1, b_2, \ldots, b_i, \ldots, b_{I \cdot H \cdot O}\}$, with binary strings of length $I \cdot H \cdot O$ bits. Table 1 shows the correspondence table between binary strings and basic RBF architectures.

The elements $b_1, b_2, \ldots, b_i, \ldots, b_{I \cdot H \cdot O}$ can be ordered in any way, taking into account, however, that the same ordering always has to be used and that the null network is always encoded as a string of $I \cdot H \cdot O$ zeros. Fig. 2 shows the chosen ordering for the basic RBF architectures that encode the set $VR_{2,2,1}$.

Table 1
Correspondence table: binary encoding of the basic RBFN architectures of length $I \cdot H \cdot O$ bits

Binary codification	Notes
$b_0 \to 0, 0, \ldots, 0, \ldots, 0$	Null RBFN architecture
$b_1 \to 1, 0, \ldots, 0, \ldots, 0$	
$b_2 \to 0, 1, \ldots, 0, \ldots, 0$	
…	
$b_i \to 0, 0, \ldots, 1, \ldots, 0$	The 1 is set in the ith position
…	
$b_{I \cdot H \cdot O} \to 0, 0, \ldots, 0, \ldots, 1$	

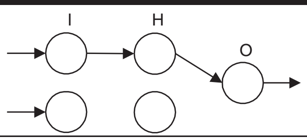
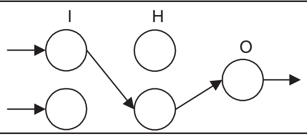
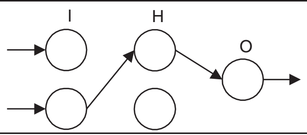
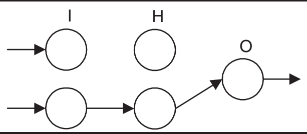
Binary encoding	Basic RBFN architecture
$b_1 \to 1, 0, 0, 0$	
$b_2 \to 0, 1, 0, 0$	
$b_3 \to 0, 0, 1, 0$	
$b_4 \to 0, 0, 0, 1$	

Fig. 2. Binary encoding for $BR_{2,2,1}$.

Having encoded the basic architectures, any valid RBF network in $VR^b_{I,H,O}$ can be built by applying the binary operator $OR\ (\vee)$ to the encoded basic nets, yielding binary strings that belong to the set $VR^b_{I,H,O}$. Therefore, the binary encoding of any element v of the set $VR_{I,H,O}$ can be output straightforwardly by calculating one of the possible decompositions of v and, starting from a string of zeros, changing the ith bit of the string to 1 if the basic RBF architecture appears in the ith position of the correspondence table and in the decomposition of v. Fig. 3 shows an example of a valid RBF architecture encoding.

The process of decoding a string of bits in an RBF architecture is also straightforward: superimposition operations should be applied to the basic architectures represented by 1 s in the ith positions of the string of bits to be decoded. Fig. 4 shows an example of how to decode a string in a neural architecture.

This encoding has three important advantages. First, any string of length $I \cdot H \cdot O$ bits represents a possible solution to the problem, as it is a RBF architecture that belongs to the set $VR_{I,H,O}$. Therefore, there are no encodings with illegal architectures. Second, according to Theorem 1, there are different ways of decomposing a valid architecture into basic architectures. Therefore, there are several encodings for the same RBF architecture, which means that there is more than one solution in the genetic algorithm search space to the same problem. Consequently, the solution is more likely to be found, and the process is, therefore, faster. Third, the result of changing just one bit of a binary string is another string that represents a RBF architecture that is very similar to the original one. The only difference between the two is the presence or absence of one basic RBF architecture. Therefore, similar genotypes represent similar phenotypes, which is a very interesting feature that improves the chances of local search success.

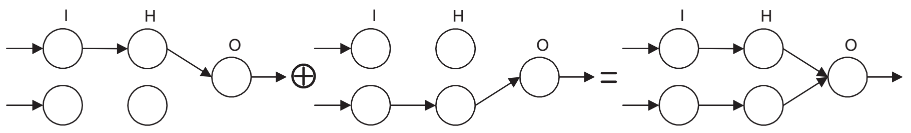


Fig. 3. Binary encoding of a RBF architecture $\in VR_{2,2,1}$. $b_1 = 1000 \vee b_2 = 0001 = v = 1001$.

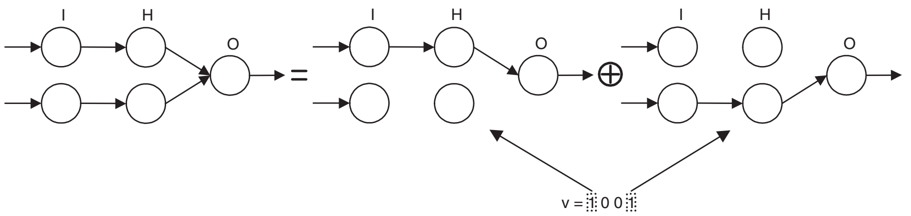


Fig. 4. Decoding of the RBFN architecture $1001 \in VR_{2,2,1}$. $v = 1001 = b_1 = 1000 \vee b_2 = 0001$.

3.4. The radial basis function network crossover

This crossover operator has been developed to work in conjunction with the basic architectures binary encoding method to enhance the three above-mentioned benefits. Like the real crossover operator, RBFN-X calculates the genetic diversity of the actual population, working in this case with binary strings. If genetic diversity is low, that is, individuals are similar, the crossover operator provides a greater probability of the offspring being less like their progenitors. On the other hand, if diversity is high, the offspring generated by the operator will be more like their parents, encouraging a reduction in diversity in the forthcoming iterations.

RBFN-X calculates the genetic diversity employing the concept of Hamming distance d_H. Given two binary strings $s = (a_0, a_1, \ldots, a_{l-1})$ and $s' = (a'_0, a'_1, \ldots, a'_{l-1})$ of length l and a_i, $a'_i \in \{0, 1\}$, the Hamming distance $d_H(s,s')$ is given as

$$d_H(s, s') = \sum_{i=0}^{l-1} a_i \otimes a'_i,$$

where $\otimes$ represents the binary exclusive-OR function.

Given the set $G = \{s_1, s_2, \ldots, s_\lambda\}$, composed of λ progenitor strings picked at random from the population, three steps are applied to get the offspring o and o'. Let us suppose, by way of an example to better illustrate the steps taken by the proposed crossover operator, that the set G is composed of four binary strings of length 7, each corresponding to a RBFN:

$$G = \{s_1 = (1,0,0,0,0,0,0),\ s_2 = (0,1,1,1,0,0,0),\ s_3 = (0,1,1,1,1,0,0),\ s_4 = (0,1,1,1,1,1,0)\}.$$

1. Calculate the maximum Hamming distance h between the progenitor strings of G: if $s_{\min}$ and $s_{\max} \in G$ are such that $d_H(s_{\min}, s_{\max}) \geqslant d_H(s_i, s_j)$ $\forall s_i, s_j \in G$, then $h = d_H(s_{\min}, s_{\max})$.
   In this example, the maximum Hamming distance is $h = 6$ between s_1 and s_4, so $s_{\min} = (1,0,0,0,0,0,0)$ and $s_{\max} = (0,1,1,1,1,1,0)$.
2. Calculate the measure of genetic diversity g of the population: $g = h/l$, where $g \in [0, 1]$. The value of g is 6/7 for this example. In this case, the genetic diversity can be said to be high.
3. Generate offspring: the operator acts adaptively according to the calculated measure of genetic diversity g. If the values of g are close to zero, the population is converging, for which reason the operator increases genetic diversity to make trapping in local optima less likely. On the other hand, if the values of g are high, indicating that the population is very scattered (this is the case illustrated in the example), the operator generates offspring that encourage a reduction in diversity, thereby increasing the local search capability. As in the real crossover operator, the function φ: $\Re \to \Re$ is employed to do this. The function φ provides the maximum number of bits, n, that have to be modified in the two offspring as follows:
   $n = E[\varphi(g) \cdot l]$, where $E[x]$ is the integer part of x. Using the function φ defined in Section 2.1, $n = 1$ for the proposed example.
   Given $h = d_H(s_{\min}, s_{\max})$, the minimum strings set $G_{\min}$ is defined as the set of binary strings at a Hamming distance $|n|$ from $s_{\min}$ and h–n of $s_{\max}$:

$$G_{\min} = \{s_1, \ldots, s_\theta\}, \quad d_H(s_{\min}, s_i) = |n| y,$$
$$d_H(s_{\max}, s_i) = h - n, \quad \forall s_i \in G_{\min}.$$

In our example, $G_{\min}$ is composed of all the binary strings at a Hamming distance 1 from (1,0,0,0,0,0,0) and 5 from (0,1,1,1,1,1,0), which is satisfied by two strings:

$$G_{\min} = \{(0,0,0,0,0,0,0), (1,0,0,0,0,1,0)\}.$$

Similarly, the maximum strings set, denoted as $G_{\max}$, is defined as the set of binary strings at a Hamming distance $|n|$ from $s_{\max}$ and $h-n$ of $s_{\min}$:

$$G_{\max} = \{s'_1, \ldots, s'_\theta\}, \quad d_H(s_{\max}, s'_i) = +|n|y,$$
$$d_H(s_{\min}, s'_i) = h - n, \quad \forall s'_i \in G_{\max}.$$

In the proposed example: $G_{\max} = \{(0,1,1,1,1,0,0), (1,1,1,1,1,1,0)\}$. The crossover interval is bounded by $G_{\min}$ and $G_{\max}$, and the offspring cannot be outside this interval (Hamming distance).

The offspring set $O_m = \{o_1, \ldots, o_p\}$, where $m \in \{0, \ldots, h-2n\}$, is defined as the set of binary strings such as

$$\forall s \in G_{\min}, \quad \forall s' \in G_{\min} : d_H(o_i, s) = m,$$
$$d_H(o_i, s') = h - 2n - m, \text{ with } o_i \in O_m, \text{ or}$$
$$\forall s \in G_{\min}, \quad \forall s' \in G_{\min} : d_H(o_i, s) = h - 2n - m,$$
$$d_H(o_i, s') = m, \text{ with } o_i \in O_m.$$

$m \in \{0,1,2,3,4\}$ for our example. If $m = 0$, then O_0 is composed of strings at a Hamming distance 0 from any of the strings within $G_{\min}$ and 4 from any of the strings within $G_{\max}$ or vice versa. Taking the first option, O_0 is the offspring set that contains the strings within just one of the bounds of the crossover interval: $O_0 = G_{\min}$. Taking the second option, O_0 is the opposite bound $G_{\max}$.

Having calculated the offspring set $O_m = \{o_1, \ldots, o_p\}$, the symmetric offspring set, denoted as $O'_m = \{o'_1, \ldots, o'_q\}$, is defined by the set of binary strings such that

If for any string $s'' \in G_{\min}$, $d_H(o_i, s'') = m$, with $o_i \in O_m$, then $\forall s \in G_{\min}$, $\forall s' \in G_{\max}$:$d_H(o'_i, s) = h-2n-m$, $d_H(o'_i, s') = m$, $\forall o'_i \in O'_m$.

If for any string $s \in G_{\min}$, $d_H(o_i, s) = h-2n-m$, with $o_i \in O_m$, then $\forall s \in G_{\min}$, $\forall s' \in G_{\max}$: $d_H(o'_i, s) = m$, $d_H(o'_i, s') = h-2n-m$, $\forall o'_i \in O'_m$.

If $O_0 = G_{\min}$ is calculated in the previous step, then the Hamming distance between the strings from O_0 and $G_{\min}$ is $m = 0$, which is how the first equation of this step is calculated. O'_0 is calculated as the strings at a Hamming distance 4 from $G_{\min}$ and 0 from $G_{\max}$. Hence, $O'_0 = G_{\max}$, which is the exact opposite bound to $G_{\min}$. Actually, $G_{\min}$ and $G_{\max}$ contain symmetric points, as they are the crossover interval bounds. If $O_0 = G_{\max}$ is calculated, then we would get $O'_0 = G_{\min}$.

Finally, the operator picks a string from the set O_m as the first offspring at random. The second string is also chosen at random from the symmetric set O'_m.

Fig. 5 shows how the crossover intervals widen or narrow depending on the genetic diversity in the Hamming crossover operator to generate the offspring. For clarity's sake, this is illustrated on the straight line of real numbers.

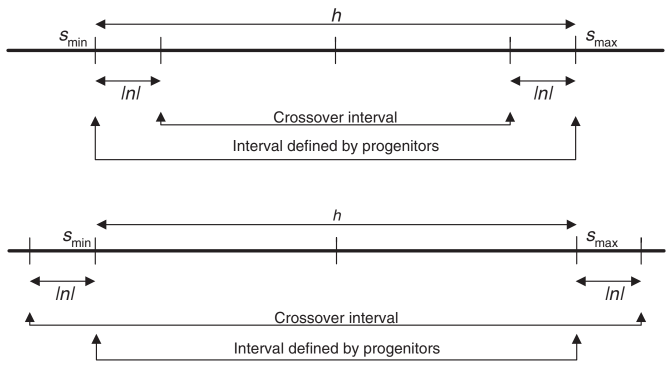


Fig. 5. Crossover intervals generated with the Hamming crossover operator ($n \geq 0$ and $n < 0$, respectively).

4. Cooperation between binary and real-coded genetic algorithms to design and train radial basis function networks

Traditionally, the neural architecture was designed by trial and error taking a set of training patterns. It was then trained to find out whether the training algorithm converged and, if so, whether the mean square error was low enough to solve the problem, taking special care not to choose an architecture that was so big as to lose the neural network's generalization capability.

The binary and real-coded genetic algorithms presented above can cooperate to work together as a system that automatically constructs self-adaptive intelligent systems with RBFNs. This system is called Genetic Algorithm Radial Basis Function Networks (GARBFN = Binary RBFN Encoding + RBFN-X + Real Crossover Operator). As in the traditional case, the minimum RBF architecture that gets a mean square error less than a previously established value can be generated taking the set of training patterns that adequately describes the problem, in this case automatically, without external intervention.

The structure of GARBFN is illustrated in Fig. 6. It consists of two subsystems that work in parallel. The binary genetic algorithm subsystem employs the modified version of the basic architectures codification, adapted to RBFNs as presented in this paper and combined with the RBFN-X crossover operator. This subsystem is responsible for generating RBF architectures in search of the optimum architecture that is capable of solving the problem. The real genetic algorithm subsystem receives a binary-coded RBF architecture with I input, H hidden and O output neurons, generated by the binary genetic algorithm subsystem, for evaluation after a decoding process. The real genetic algorithm encodes the weights and biases of the neuron connections between hidden and output layers using a chromosome of length $O \cdot H + O$ real numbers (after the k-means clustering algorithm has been run to find the centres of the neurons in the hidden layer). This genetic algorithm employs the real crossover operator to evolve the population in search of the best combination of weights and biases that minimizes the mean square error. The mean

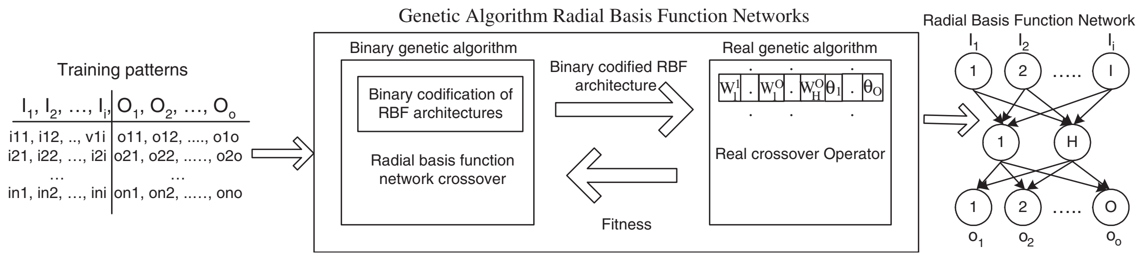

Genetic Algorithm Radial Basis Function Networks
Training patterns
I1, I2, ..., Ii, O1, O2, ..., Oo
i11, i12, .., v1i | o11, o12, ..., o1o
i21, i22, ..., i2i | o21, o22, ..., o2o
in1, in2, ..., ini | on1, on2, ..., ono
Binary genetic algorithm
Binary codification of RBF architectures
Radial basis function network crossover
Binary codified RBF architecture
Fitness
Real genetic algorithm
Real crossover Operator
Radial Basis Function Network

Fig. 6. GARBFN structure.

square error is employed to calculate the fitness (f) of the RBF architecture using the following expression: $f = \mathrm{MSE}_{\mathrm{it}}(C_a/C_t)$, where C_a is the number of connections existing in the actual neural network and C_t is the maximum number of connections allowed by the codification. $\mathrm{MSE}_{\mathrm{it}}$ is the mean square error output by the network after running "it" real genetic iterations. This fitness is returned to the binary genetic algorithm for use in handling and evolving the population in search of a better RBF architecture.

5. Results

The results of the experiment run to test the GARBFN system are related to convergence speed, size of the networks built, as well as the accuracy of the output of the network solution. We ran two types of test. The first is a laboratory example based on pattern classification. The second is composed of two tests for diagnosing breast cancer from two types of breast lesion: microcalcifications and masses. This second test examines specificity, sensitivity and accuracy, which are very commonly used in medicine for diagnosing the two lesions. Also the results yielded by the RBFN output by GARBFN are compared with the diagnoses given by two expert radiologists called Doctor A and B.

The lab and the breast cancer diagnosis tests run are both analysed for comparison with other important related approaches. These analyses demonstrate the performance and accuracy of the real crossover operator for training RBFNs, as well as the quality of the problem-solving architectures and the speed of convergence of the proposed binary RBFN codification together with the RBFN crossover operator.

The real crossover operator proposed for use in the GARBFN system to train the RBFNs is compared with the blend crossover, BLX-α, as well as the traditional method for training this type of neural networks, the backpropagation algorithm. These comparative tests show the mean convergence speed of each algorithm after 100 runs each. In all the tests run, the weights of the connections and biases are bound within the interval [−25,25]. As the computational cost of one iteration of a genetic algorithm using the real crossover operator and of one backpropagation iteration is not the same, convergence speed is measured as a function of the number of floating point operations (FLOPS) required by each algorithm for the RBFN being trained to achieve a given mean square error for a given training set.

Supposing that the real crossover operator works with strings of length l, l floating point multiplications (FLOMS) for each algorithm iteration, plus $S = l \cdot (2n + 6)$ floating point additions, where n is the number of progenitors selected by the operator, the total number of FLOPS would be:

$$\mathrm{FLOPS} = l + l \cdot (2n + 6). \tag{1}$$

The number of FLOPS in the BLX-α case would be:

$$\mathrm{FLOPS} = l + S_{\mathrm{BLX}-\alpha} = l + l. \tag{2}$$

Finally, the number of FLOPS employed in each backpropagation iteration would be as follows:

$$\mathrm{FLOPS} = (6 \cdot P + 3) \cdot l + l, \tag{3}$$

P being the number of training patterns.

When the cardinal of the set of training patterns is greater than 17, the number of sums employed by each backpropagation iteration is always greater than the sums used by the real and BLX-α crossover operators. As RBFNs are usually trained with a lot more than 17 patterns and because the computational load for doing a FLOM is greater than for doing a sum, the algorithms are compared as to the number of FLOMS only, neglecting the number of floating comma sums. Table 2 shows the number of FLOMS employed in each iteration of each of the three algorithms.

The genetic algorithm population is composed of 30 individuals. The real crossover operator employs five progenitors, and factor α is set at 0.1 for BLX-α. A learning rate of 0.3 is used for the backpropagation algorithm, while its momentum factor is set at 0.5. All these values are calculated empirically and are the ones that yield the best results for each algorithm. The fitness of each individual (the set of values for all the weights and biases of the RBFN to be trained) is equal to the mean square error given by the RBFN for all the patterns in the training set.

The tests run to build the RBFNs with the GARBFN system are designed to compare the results yielded by employing the proposed encoding of RBF architectures with the direct and grammar encoding methods, widely used for such problems. The tests run take into account the use of these three different encoding methods in combination with the RBFN crossover and generalized crossover [1] operators. The results output by the direct encoding method and generalized crossover operator, however, are not shown because this combination tend to generate illegal individuals, leading to non-convergence of the genetic algorithm.

Each test was run 100 times, examining the mean convergence speed for each combination of different crossover operators and encoding methods. The final mean size of the neural architectures output as solutions is calculated.

The different neural architectures that are output by the genetic algorithms as possible solutions are trained using the real crossover operator within a real-coded genetic algorithm, which is, as we found, the one that yields the best results and is used by GARBFN. Again in this test set, the weights and thresholds are established within the [−25, 25] interval.

The fitness of the individuals in the binary genetic algorithm is calculated by means of the following evaluation function:

$$\text{Fitness} = (1 - w) \cdot \frac{C_A}{C_{MAX}} + w \cdot \text{Error}, \quad (4)$$

where w is the efficiency/effectiveness ratio of the network, whose value is between 0 and 1. C_A is the number of connections there are in the network, C_{MAX} is the maximum number of possible connections in the network and Error is defined by

$$\text{Error} = \frac{\text{MSE}}{O},$$

MSE being the mean square error output in the network training process and O is the number of output neurons that it has.

Table 2
FLOMS employed in each training algorithm iteration

Training algorithm	FLOMS
Real crossover	l
BLX-α	l
Backpropagation	$(6 \cdot P + 2) \cdot l$

5.1. Laboratory test

This test involves building and training RBFNs to classify linearly inseparable patterns (points in 3D). Five groups composed of 52 points are proposed as shown in Fig. 7. Each pattern consists of a vector composed of an input, namely, the point's three coordinates, and an output, which is the cluster to which the point belongs.

5.1.1. Training radial basis function networks

The RBFN to be trained is fully connected and has three neurons in the input layer (one per each coordinate of the point to be classified), five neurons in the hidden layer (one per group), and one neuron in the output layer.

From the above architecture, we get that the length l of the individuals in the genetic algorithms is six. Table 3 shows the number of FLOMS per iteration of each training algorithm of this network in the knowledge that there is a training set of $P = 52$ patterns (3D points).

Fig. 8 is a graph plotting the mean convergence speed of each of the three RBFN training algorithms, measuring the fall in the mean square error of the network depending on the number of iterations used up to iteration 2500.

Table 4 contains the mean number of FLOMS required by each training algorithm to achieve different mean

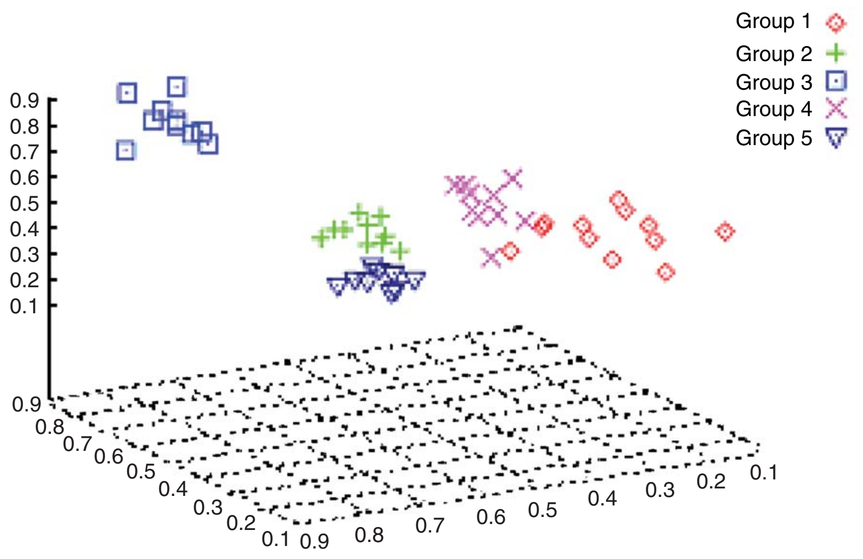


Fig. 7. Training patterns for the lab test.

Table 3
FLOMS employed in each training algorithm iteration for the 3D-point classification problem

Training algorithm	FLOMS
Real crossover	6
BLX-α	6
Backpropagation	1884

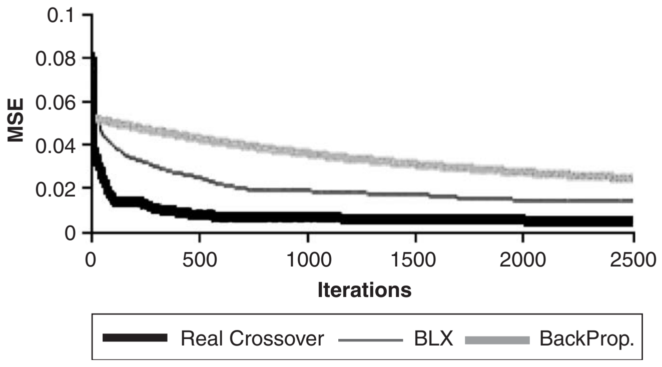


Fig. 8. Convergence speed for each training method.

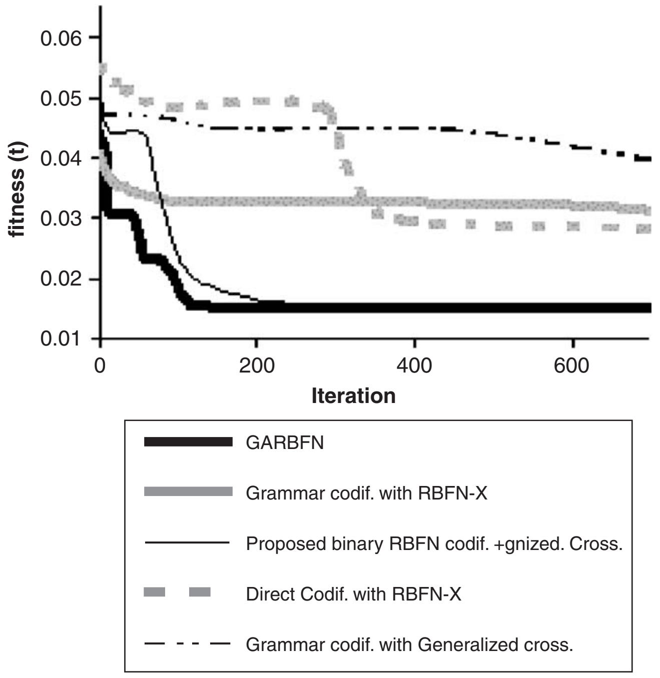


Fig. 9. Mean convergence speed for searching RBFNs.

Table 4
FLOMS employed by each training algorithm to achieve different mean square errors

Training algorithm	MSE	FLOMS	Success rate (%)
Real crossover	6×10^{-3}	15,000	76.92
BLX-α	1.75×10^{-2}	15,000	55.77
Backpropagation	3.5×10^{-2}	4.71×10^{6}	21.15

square errors and the success rate for the classification problem that the trained network has to achieve.

From Fig. 8 and Table 4, it is clear that the real crossover has a greater speed of convergence than the other methods, where backpropagation is the method that yields the worst results. The highest success rate is for the real crossover operator, and the backpropagation method again performs worst here. Likewise, the backpropagation method has need of a very high number of FLOMS to train the RBFN for this problem, although it achieves the lowest MSE. For the same number of FLOMS, the real crossover operator yields a much lower MSE than BLX-α.

5.1.2. Constructing radial basis function networks with GARBFN

This test involves automatically generating RBFNs with three input neurons, one output neuron and a maximum of 10 neurons in the hidden layer to classify 52 points in 3D, as shown in Fig. 7. To run these tests, we take a value of $w = 0.9$ to calculate the fitness (formula 4) of the neural networks that the genetic algorithm generates. The MSE of the neural architecture is calculated after 10,000 iterations of the real-coded genetic algorithm.

Fig. 9 shows the mean convergence speed of the combinations of encoding methods and crossover operators after having executed each one 100 times.

As we can see from Fig. 9, the grammar encoding method combined with the generalized or RBFN crossover operators yields much lower mean convergence speeds than GARBFN or the proposed binary RBFN encoding method combined with the generalized crossover. This is because there is no correspondence between the encoding type used and the crossover operator that handles the progenitor genotypes to engender offspring whose phenotypes should be somewhat similar to their parents'. This leads to a notable expansion in the search space and makes the genetic algorithm exploration capabilities too powerful and convergence less likely. On the other hand, being based on the Hamming distance, the RBFNs crossover generates two offspring whose phenotypes bear a resemblance to their progenitors'. The direct encoding method plus the RBFNs crossover does not achieve a high convergence speed either. This is because this encoding type does not rule out illegal architectures whose fitness can be of no guidance for the genetic algorithm in the search for the best solutions. Additionally, this type of codification handles binary strings of length greater than 91 bits, as opposed to the 30 bits used by the proposed encoding method.

The only two encoding and crossover operator combinations that achieve an MSE of 0.015 for this problem are: GARBFN and the proposed encoding method plus the generalized crossover. The first takes 182 iterations, whereas as the second needs 912. In both cases the RBFN solution is a not fully connected architecture with eight hidden neurons.

5.2. Breast cancer diagnosis

GARBFN is employed to diagnose masses or microcalcifications present in breast tissue suspected of being carcinomas. This application is part of a larger project for automatically detecting and diagnosing breast pathologies. The input for the system built so far is a complete set of views of digitalized mammograms from both patients' breasts, which it uses to search for microcalcifications and suspicious masses, the two main abnormalities that can be detected by mammography. The output is a set of characteristics for each abnormality found. These characteristics are stored in a database containing the lesions of real 25- to 79-year-old patients at a Madrid hospital. All these cases are diagnosed by two expert radiologists, the results of which are compared with the results of the biopsy, which is also available. These clinical data is used to run the tests, designing one set of training and test patterns to diagnose microcalcifications and another for masses.

For both types of lesions, microcalcifications and masses, diagnosis involves classifying the abnormality as either: benign or malignant. The following characteristics are used for diagnosis: patient's *age*; *breast* in which the lesion is located, whose possible values are right and left; *location* of the lesion, whose values, central, bilateral, subareolar and auxiliary tail, match each of the sections into which the breast is divided lengthways; and, finally, *depth*, which matches each of the cross-sections of the breast and its permitted values are central, middle, posterior and subareolar.

For the case of microcalcifications diagnosis, apart from the above-mentioned characteristics, we take into account the following:

- *Size of the lesion*. Microcalcifications usually appear as clusters. This characteristic is established by the widest diameter of the cluster, whose value ranges from 1 to 77 mm.
- *Number of microcalcifications that appear in the cluster*. Its possible values are: from 1 to 5, from 6 to 10 and over 10.
- *Distribution*. This represents how the microcalcifications are grouped: diffused, grouped, linear, regional or segmental.
- *Type*.This represents the appearance of the microcalcifications after visual examination. This characteristic has three possible values: typically benign, intermediate and typically malignant. Typically benign microcalcifications may have the following values: coarse, eggshell or rim, large rodlike, milk of calcium, lucent centred, punctate, round, or void if the microcalcifications is not typically benign. The possible values for intermediate microcalcifications are as follows: amorphous, indistinct or void. Finally, the possible values for typically malignant microcalcifications are as follows: fine branching, fine linear, heterogeneous or void.

To build the set of training and test patterns, 184 cases in which microcalcifications were present were selected at random, half of which were malignant and the other half benign cases. Each pattern is composed of a total of eight inputs—age, breast, location, depth, size, number, distribution and type—and just one output—the diagnosis with two possible values: benign or malignant.

For the case of masses, the following specific characteristics were taken into account to make the diagnosis:

- *Size*. This represents the size of the mass, measured as the widest diameter in millimetres.
- *Morphology of the mass*. This has four possible values: architectural distortion, lobulated, oval, round and irregular.
- *Margins*. This characteristic describes the external limits of the mass. It has five possible values: circumscribed, ill-defined, microlobulated, spiculated and obscured.
- *Density*. It represents the texture of the tissue inside the mass. The possible values are: high, equal, low and fatty.

In this case, to build the set of training and test patterns, a total of 315 cases are selected in which the breast tissue contains abnormal masses. Of all these, 138 are malignant cases and the other 177 are benign. Each pattern is composed of a total of seven inputs—age, breast, location, depth, size, morphology, margins and density—and again just one output: the diagnosis of the analysed mass.

It is usual practice in the field of medicine to use three criteria to report statistical results and for the purpose of comparisons: accuracy, specificity and sensitivity. *Accuracy* is the percentage of correctly over incorrectly diagnosed cases:

$$\text{Accuracy} = \frac{T_\text{P} + T_\text{N}}{\text{Number of Instances}},$$

where T_P (True Positive) are the cases diagnosed as malignant that really are malignant and T_N (True Negative) are the cases diagnosed as benign that are benign.

Specificity represents the percentage of cases diagnosed as negative, benign lesions, over the total number of negative cases:

$$\text{Specificity} = \frac{T_\text{N}}{T_\text{N} + F_\text{P}},$$

where F_P (False Positive) is the number of cases classed as malignant when they really are benign.

Finally, *sensitivity* is the percentage of cases diagnosed as positive, malignant lesions, over the total number of positive cases:

$$\text{Sensitivity} = \frac{T_\text{P}}{T_\text{P} + F_\text{N}},$$

where F_N (False Negative) is the number of cases that classed as benign when they really are malignant.

5.2.1. *Training radial basis function networks*

To compare the three training methods, we used a RBFN with eight neurons in the input layer (one per lesion characteristic), 15 neurons in the hidden layer and one output neuron to provide the diagnosis for both masses and microcalcifications.

The length of the individuals for the genetic populations is 16. Table 5 shows the number of FLOMS that each training algorithm executes in each iteration. P is the number of training patterns, which is 285 in the case of masses diagnosis and 166 in the case of microcalcifications.

The mean convergence speeds up to iteration 3000 for each of the three RBFN training methods for mass diagnosis are shown on the left-hand side of Fig. 10, whereas the RBFN training for microcalcifications diagnosis is illustrated on the right.

As we can see, both graphs are similar. The training method that has a faster convergence speed is the genetic algorithm with real crossover operator. On the other hand, the convergence speed of the backpropagation method is lower than for the crossover operators.

Tables 6 and 7 list information similar to Table 4 for the mass and microcalcifications diagnoses, respectively.

These results clearly show that not only does the real crossover operator achieve a lower MSE and, therefore, a higher success rate, but, what's more, does so in a shorter time. It can be concluded that this is the best training algorithm for RBFNs. This is why the GARBFN system uses this the training algorithm.

Table 5
FLOMS employed in each training algorithm iteration for the breast cancer diagnosis problem

Training algorithm	FLOMS
Real crossover	120
BLX-α	120
Backpropagation	$96 \cdot P + 32$

5.2.2. *Constructing radial basis function networks with GARBFN*

The binary genetic algorithm search space is composed of a set of radial basis function architectures with a maximum of eight input neurons (one per breast lesion characteristic), a maximum of 15 hidden neurons and a single output neuron. A value of $w = 0.8$ was taken to calculate the fitness (formula 4) of the neural networks to be generated by the binary genetic algorithm To train each RBFN, 1000 iterations of the genetic algorithm with the real crossover operator were run.

The mean convergence speed for searching the radial basis function architecture that best solves the problem of diagnosing mass-type lesions is shown on the left-hand side of Fig. 11, whereas the same graph for microcalcifications is shown on the right.

From the graphs in Fig. 11, we find that the system that yields the best results is GARBFN. It is also noteworthy that the proposed binary RBFN encoding method yields very good results even with other crossover operators. The reason is that the length of the generated individuals is the shortest of the three encoding methods tested: 120 bits, as opposed to 276 bits for direct encoding or 210 bits for the grammar encoding method.

Table 6
FLOMS employed in masses

Training algorithm	MSE	FLOMS	% correct
Real crossover	7.1×10^{-3}	3.6×10^5	77.14
BLX-α	7.8×10^{-3}	3.6×10^5	66.66
Backpropagation	9.2×10^{-3}	8.22×10^7	62.22

Table 7
FLOMS employed in microcalcif

Training algorithm	MSE	FLOMS	% correct
Real crossover	7.3×10^{-3}	3.6×10^5	74.45
BLX-α	8.3×10^{-3}	3.6×10^5	66.84
Backpropagation	9.5×10^{-3}	4.79×10^7	59.78

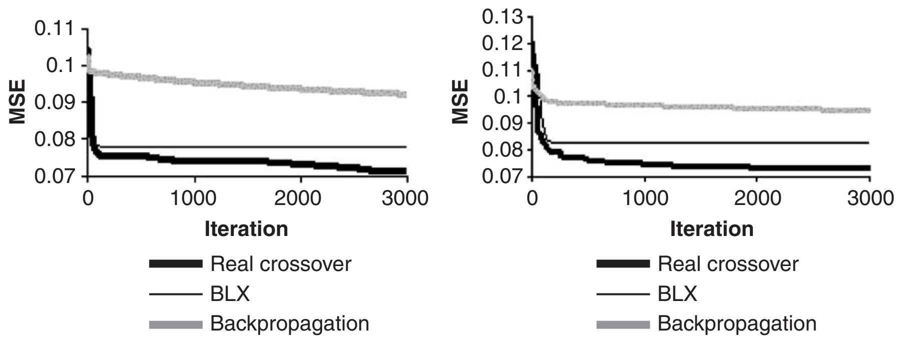


Fig. 10. Convergence speed for the breast cancer diagnosis problem.

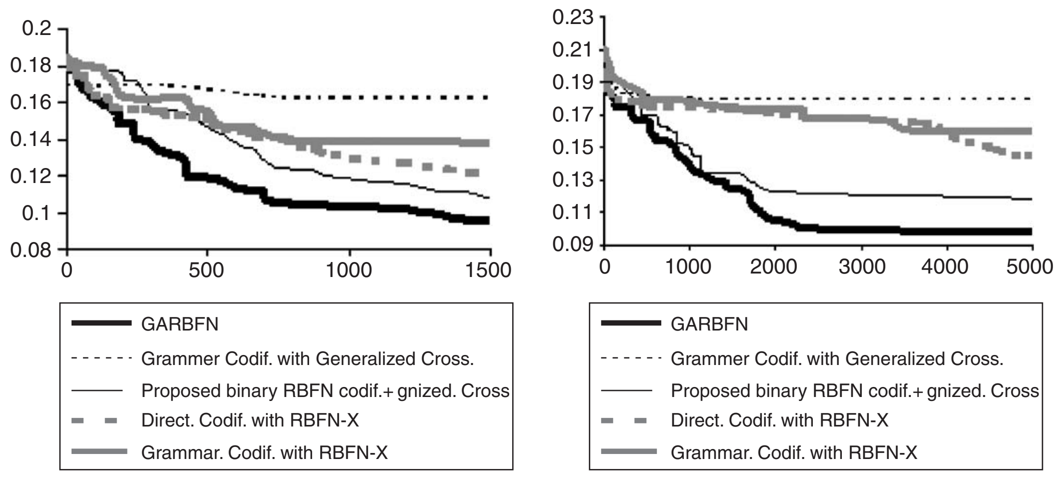


Fig. 11. Mean convergence speed for searching RBFNs.

Table 8
Neural connections for masses

Number of connections	GARBFN (%)	Proposed binary RBFN codif. + gnrzed. cross. (%)
22	72	5
[23–33]	25	15
[33–43]	3	61
>43	0	19

Table 9
Neural connections for microcalcif

Number of connections	GARBFN (%)	Proposed binary RBFN codif. + gnrzed. cross. (%)
22	69	1
[23–33]	20	9
[33–43]	8	58
>43	0	32

Tables 8 and 9 show how many radial basis function architecture connections have been calculated to solve the problem of the masses and microcalcifications diagnosis, respectively. These tables show the results for GARBFN and the combination of the proposed binary RBFN encoding method with the generalized crossover, as they are the ones that yielded the best results in the tests run. From these tables, it follows not only that GARBFN has a higher convergence speed, but also that there is a greater likelihood of getting a smaller-sized radial basis function architecture solution than using the other algorithms. This provides bigger benefits in terms of network generalization capability and network execution speed, as, having fewer connections, it needs to make fewer numerical calculations to output a diagnosis from an input vector of characteristics.

An interesting fact worth mentioning is that, because GARBFN looks for the minimal radial basis function architecture, input neurons can be removed in the final problem-solving architecture. This means that it is implicitly running a principal component analysis, removing inputs that really do not influence the output, because, as we have seen, the MSE of the problem-solving architecture is low enough to solve the problem with the required accuracy.

In the case of the masses diagnosis problem, of the eight original inputs that were taken into consideration to make the diagnosis, GARBFN removes the three variables that are related to the position of the mass in the breast in 72% of the cases (see Table 8): breast (left or right), location and depth. This result is consistent with reality, as masses can appear indistinctly anywhere in the breast. In the case of microcalcifications diagnosis, the inputs removed by GARBFN in 69% of the cases (see Table 9) are the three variables related to the position of the microcalcifications in the breast, plus the number of detected microcalcifications. That the diagnosis does not depend on the number of microcalcifications was corroborated by radiologists.

Tables 10 and 11 show the results in terms of accuracy, specificity and sensitivity of the diagnoses for the set of test patterns of masses and microcalcifications, respectively, comparing the results of GARBFN with the combination of the proposed encoding method and the generalized crossover and two doctors (called A and B) specialized in image diagnosis.

Even though GARBFN uses a smaller radial basis function architecture, it achieves better results than the proposed encoding method combined with the generalized crossover operator. The results yielded by GARBFN are similar to the outcomes for the doctors in terms of accuracy. GARBFN outperforms the doctors as regards specificity and is worse for sensitivity. This is because of how a radiologist makes a diagnosis of a detected breast lesion: only if the doctor is really very sure is the lesion

Table 10
Diagnosis for masses

Combination: codif.–cross.	Accuracy (%)	Specificity (%)	Sensitivity (%)
GARBFN	83	83	81
Proposed binary RBFN codif. + gnrzed. cross.	79	82	76
Doctor A	75	63	89
Doctor B	69	57	87

Table 11
Diagnosis for microcalcif

Combination: codif.–cross.	Accuracy (%)	Specificity (%)	Sensitivity (%)
GARBFN	74	78	70
Proposed binary RBFN codif. + gnrzed. cross.	72	68	76
Doctor A	69	39	78
Doctor B	65	28	77

classed benign; if there is any doubt, the lesion is classed as malignant and a biopsy is done. The GARBFN system has no such prejudices, which means that it yields better specificity results.

6. Conclusions

This article presents GARBFN, capable of automatically building RBFNs to solve a given problem depending on a set of training patterns. GARBFN is based on two genetic algorithms that cooperate with each other. The first is binary; it is responsible for searching radial basis function architectures within a search space created on the basis of a new type of encoding that prevents illegal structures from being encoded and employs short binary strings. This encoding is supplemented by a new crossover operator (RBFN-X) based on the concept of Hamming distance. Thanks to this crossover operator, combined with the proposed encoding, the phenotype of the offspring can inherit similar characteristics to what their parents have.

The calculation of the fitness (*a* parameter to be minimized) of the neural architectures generated with the binary genetic algorithm is inversely proportional to the number of connections and directly proportional to the MSE output during the network training process. A real-coded genetic algorithm based on a new real crossover operator is used to train the RBFNs. This algorithm yields better results in terms of convergence speed and the likelihood of trapping in local optima than the traditional backpropagation training algorithm and other related crossover operators like BLX.

Based on the synergy between the architecture search genetic algorithm and the training genetic algorithm, GARBFN generates a neural network capable of solving the problem, providing a high convergence speed, small-sized neural network and optimal results for solving the classification problem. As GARBFN removes the connections and neurons that are unnecessary for achieving a good MSE, it is capable of doing without some input neurons representing variables that were originally assumed to be useful for the output. What it does, therefore, is to run a sensitivity analysis of the problem in the background.

To demonstrate the above-mentioned properties, we successfully ran a series of classification tests. First, we did laboratory tests involving the classification of 3D points. Second, the system was tested on the diagnosis of breast lesions from a set of patterns. The neural architectures output for this problem have a high success rate and potential for use in the construction of an intelligent system for computer-aided breast cancer diagnosis.

References

[1] D. Barrios, Generalized crossover operator for genetic algorithms (Operador de cruce generalizado en algoritmos genéticos), Ph.D. Thesis, Universidad Politécnica de Madrid, Madrid, 1991.

[2] D. Barrios, A. Carrascal, D. Manrique, J. Ríos, Optimisation with real-coded genetic algorithms based on mathematical morphology, International Journal of Computer Mathematics 80 (3) (2003) 275–293.

[3] D. Barrios, A. Carrascal, D. Manrique, J. Ríos, ADANNET: automatic design of artificial neural networks by evolutionary techniques, in: Proceedings of the 21st 2001 SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence, Brighton, UK, pp. 67–80.

[4] M. Bianchini, P. Frasconi, M. Gori, Learning without local minima in radial basis function networks, IEEE Transactions on Neural Networks 30 (3) (1995) 136–144.

[5] A.D. Brown, H.C. Card, Cooperative coevolution of neural representations, International Journal of Neural Systems 10 (4) (2000) 311–320.

[6] S. Chen, Y. Wu, L. Luk, Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks, IEEE Transactions on Neural Networks 10 (5) (1999) 1239–1243.

[7] S. Cohen, N. Intrator, A hybrid projection-based and radial basis function architecture: initial values and global optimization, Pattern Analysis and Applications 5 (2) (2002) 113–120.

[8] J. Crespo, Morphological connected filters and intra-region smoothing for image segmentation, Ph.D. Thesis, Georgia Institute of Technology, Atlanta, 1993.

[9] L.A. D'alotto, C.R. Giardina, A Unified Signal Algebra Approach to Two-Dimensional Parallel Digital Signal Processing, Marcel Dekker, New York, 1998.

[10] J. Dorado, Cooperative strategies to select automatically training patterns and neural architectures with genetic algorithms, Ph.D. Thesis, University of La Coruña, Spain, 1999.

[11] L.J. Eshelman, J.D. Schaffer, Real-coded genetic algorithms and interval-schemata, Foundations of Genetic Algorithms 2 (1993) 187–202.

[12] S. Haykin, Neural Networks: A Comprehensive Foundation, second ed., Prentice-Hall, Englewood Cliffs, NJ, 1999.

[13] C. Hervas, J.A. Algar, M. Silva, Correction of temperature variations in kinetic-based determinations by use of pruning computational neural networks in conjunction with genetic algorithms, Journal of Chemical Information and Computers Sciences 40 (3) (2000) 724–731.

[14] Y.S. Hwang, S.Y. Bang, An efficient method to construct a radial basis function neural network classifier, Technical Report, Department of Computer Science & Engineering, Pohang University of Science and Technology, Korea, 1997.

[15] H. Kitano, Designing neural networks using genetic algorithms with graph generation system, Complex Systems 4 (1990) 461–476.

[16] A. Konar, Artificial Intelligence and Soft Computing, CRC Press, Boca Raton, FL, 2000.

[17] J. Luo, C. Zhou, Y. Leung, A Knowledge-Integrated RBF Network for Remote Sensing Classification, LREIS, Institute of Geographical Sciences and Natural Resources Research, CAS, Beijin, China, 2001.

[18] M.Y. Mashor, Hybrid training algorithms for RBF network, International Journal of the Computer 8 (2) (2000) 50–65.

[19] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, New York, 1999.

[20] M.T. Musavi, W. Ahmed, K.F. Chan, K.B. Faris, D.M. Hummels, On the training of radial basis function classifiers, Neural Networks 5 (1992) 595–603.

[21] I. Ono, S. Kobayashi, A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover, in: Proceedings of the 1998 Seventh International Conference on Genetic Algorithms, East Lansing, MI, USA, pp. 246–253.

[22] J.C. Principe, N.R. Euliano, W.C. Lefebvre, Neural and Adaptative Systems, Fundamentals Through Simulations, Wiley, New York, 2000.

[23] N. J. Radcliffe, Genetic neural networks on MIMD computers, Ph.D. Thesis, University of Edinburgh, Edinburgh, UK, 1990.

[24] G.E. Robbins, M.D. Plumbley, J.C. Hughes, F. Fallside, R. Pager, Generation and adaptation of neural networks by evolutionary techniques (GANNET), Neural Computing and Applications 1 (1993) 23–31.

[25] F. Sahin, A radial basis function approach to a color image classification problem in a real time industrial application, Ph.D. Thesis, Polytechnic Institute of Virginia, 1997.

[26] A.P. Topchy, O.A. Lebedko, V.V. Miagkikh, N.K. Kasabov, Adaptive training of radial basis function networks based on cooperative evolution and evolutionary programming, in: 1998 Progress in Connectionist-Based Information Systems, Springer, Berlin.

[27] R. Yampolskiy, D. Novikov, L. Reznik, Experimental study of the neural networks for character recognition, Technical Report, Department of Computer Science, Rochester Institute of Technology, Rochester, New York, 2003.

Daniel Manrique received the B.S. in 1997 and the PhD degree in Computing in 2001, both from the Technical University of Madrid (UPM). He was an Associate Professor of artificial intelligence from 2000 to 2003. He is currently a Professor of computing at the UPM's School of Computing. His major fields of study and research are connectionist artificial intelligence, soft computing and e-learning in which he has participated as a researcher in over 15 European or Spanish research projects.

Dr. Manrique is the assistant director of the connectionist artificial intelligence group since 2001 and directs the e-learning work group at UPM since 2002. He has co-authored three books, two book chapters and over 30 papers published in international journals and congresses. At the moment, Dr. Manrique is a member of the International Programme Committee of two International Congresses.

Juan Ríos has a Ph.D. in Civil Engineering and is full Professor at the School of Computer Science of the Technical University of Madrid (UPM). He was a Program Advisory Committee Member of the IV and V International Symposiums on Knowledge Engineering, a member of the expert commission and founder member of the Centre of Computing and Communications Technology Transfer (CETTICO). He is also a member of the International Association of Knowledge Engineering. Professor Rios is de Director of the connectionist artificial intelligence group at UPM. He has authored numerous books about artificial neural networks and over 50 papers published in international journals and congresses related to evolutionary computation, artificial neural networks and knowledge engineering.

Alfonso Rodríguez-Patón is physicist (electronic and computation specialty) and has a Ph.D. in Computer Science. He is professor at the School of Computing of Technical University of Madrid (UPM). He is the director of the Artificial Intelligence Group in the same University. His main research interest lies in the interplay of computer science, biology, and nanotechnology. His topics of research are formal languages and automata theory, DNA/biomolecular computing, membrane computing, and any bio-inspired or unconventional model of computation.