

MELM-GRBF: A modified version of the extreme learning machine for generalized radial basis function neural networks

Francisco Fernández-Navarro*, César Hervás-Martínez, Javier Sanchez-Monedero, Pedro Antonio Gutiérrez

Department of Computer Science and Numerical Analysis, University of Córdoba, Campus de Rabanales, Albert Einstein Building, 3rd floor, 14074 Córdoba, Spain

ARTICLE INFO

Available online 20 May 2011

Keywords:

Generalized radial basis functions neural networks
 Extreme learning machine
 Multi-classification
 Generalized Gaussian distribution

ABSTRACT

In this paper, we propose a methodology for training a new model of artificial neural network called the generalized radial basis function (GRBF) neural network. This model is based on generalized Gaussian distribution, which parametrizes the Gaussian distribution by adding a new parameter τ . The generalized radial basis function allows different radial basis functions to be represented by updating the new parameter τ . For example, when GRBF takes a value of $\tau = 2$, it represents the standard Gaussian radial basis function. The model parameters are optimized through a modified version of the extreme learning machine (ELM) algorithm. In the methodology proposed (MELM-GRBF), the centers of each GRBF were taken randomly from the patterns of the training set and the radius and τ values were determined analytically, taking into account that the model must fulfil two constraints: locality and coverage. An thorough experimental study is presented to test its overall performance. Fifteen datasets were considered, including binary and multi-class problems, all of them taken from the UCI repository. The MELM-GRBF was compared to ELM with sigmoidal, hard-limit, triangular basis and radial basis functions in the hidden layer and to the ELM-RBF methodology proposed by Huang et al. (2004) [1]. The MELM-GRBF obtained better results in accuracy than the corresponding sigmoidal, hard-limit, triangular basis and radial basis functions for almost all datasets, producing the highest mean accuracy rank when compared with these other basis functions for all datasets.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Artificial neural networks (ANN) are largely used in applications involving classification or function approximation. Lately, it has been proved that several classes of ANN are universal function approximators [2–4]. Among them, we find radial basis function neural networks (RBFNNs) [5,6], multi-layer perceptrons (MLPs) [7] or product unit neural networks (PUNNs) [8,9]. All are multi-layered networks and can be considered as connectionist models. RBFNNs use, in general, hyper-ellipsoids to split the pattern space. This is different from MLPs which build their classifications on pseudo-hyper-planes, defined by a weighted sum [10].

RBFNNs use the value of the distance to estimate the response value, being functions of two arguments, \mathbf{x} and \mathbf{c} , where $\mathbf{x} = (x_1, x_2, \dots, x_K)^T$ is the vector of co-ordinates of a pattern of the dataset and $\mathbf{c} = (c_1, c_2, \dots, c_K)^T$ are the location parameters to determine kernel positions. The characteristic feature of local

RBFNNs is the fact that their response value decreases monotonically with the distance from the center \mathbf{c} of the radial function.

RBFNNs are parametrized by a width denoted here by r . If the distance between \mathbf{x} and \mathbf{c} is small compared to the width of the kernel, the kernel value will be close to one. Large distances by contrast are mapped to values close to zero. The width of the RBFNNs in kernel-based methods must produce a correct balance between *covering*: the sum of all RBFs must have a high value in all patterns of the dataset; and *locality*: the RBF should provide a high value (close to one) for patterns that are close to the environment where the RBF is located, and low values (near zero) for patterns that are not located in the region of space where the RBF is centered.

The Gaussian RBFs are based on the Gaussian density function and are defined by a “center” position and a “width” parameter. The Gaussian function gives the highest output when the incoming variables are closest to the center position and decreases monotonically as the distance from the center increases. Gaussian distribution can be parametrized by a real parameter τ , resulting in generalized Gaussian distribution (GGD). The GGD may represent different forms of distribution function by changing a real parameter τ . We can highlight the impulsive, Laplacian, Gaussian and uniform distributions.

* Corresponding author. Tel.: +34 957 21 83 49; fax: +34 957 21 83 60.
 E-mail address: i22fenaf@uco.es (F. Fernández-Navarro).

Based on this probability distribution, we propose the generalized radial basis function (GRBF) by removing the constraints of a probability function. In this way, the generalized radial basis function (GRBF) is defined as

$$\phi(\mathbf{x}; \mathbf{c}, r, \tau) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}\|^\tau}{r^\tau}\right). \tag{1}$$

Training of RBFNNs can be classified into two categories: quick learning and full learning. Quick learning usually involves a two-step process. First, the parameters governing the basis functions are determined by a relatively fast, unsupervised clustering [11] or vector quantization approach [12]. Next, the weights of the basis functions are determined using linear optimization techniques. A full learning scheme (for instance gradient-descent-based methods) optimizes all of the parameters in a supervised mode [6,13,14].

Gradient-descent-based algorithms may converge very slowly to the solution of the given problem if the learning rate is small. However, if the learning rate is large, they can be unstable or divergent. They may also easily get over-fitting or be stuck in local optima [15,16]. Moreover, most of the training algorithms based on gradient descent are still slow due to the many iterative steps required in the learning process. That is the reason why our proposal will be based on the first approach.

Recently, Huang et al. showed that a single hidden layer feedforward neural network (SLFN) can learn distinct observations with an arbitrary small error margin if the activation function is chosen properly [17–19]. An effective training algorithm for SLFNs called extreme learning machine (ELM) was also proposed by Huang et al. [20,21]. In ELM, the input weights of the hidden nodes are randomly chosen, and the output weights of SLFNs can be determined through the pseudo-inverse operation of the output matrix in the hidden layer. This algorithm can avoid many of the problems which occur in gradient-descent-based learning methods. For that reason, the GRBF proposed in this paper was trained by means of a modified version of the ELM algorithm (MELM-GRBF).

The main novelty introduced by the MELM-GRBF is in the determination of the GRBFs. While in the ELM-RBF algorithm [1], the centers and the radii of the RBFs are selected randomly, in the MELM-GRBF algorithm proposed, the centers are initialized by randomly selecting some patterns in the training dataset. The values of the radius and τ are determined analytically by solving two equations that ensure that the model fulfils two constraints: locality and coverage.

This paper is organized as follows: a brief analysis of the generalized Gaussian distribution is given in Section 2. The single layer feedforward GRBFNN is presented in Section 3. A methodology to optimize the GRBFNN parameters based on ELM is presented in Section 4. Section 5 explains the experiments that were carried out. Finally, Section 6 summarizes the conclusions of our work.

2. Generalized Gaussian distribution

In order to cope with some limitations of the Gaussian RBF [22–24], we need to use another model that can describe the statistical behaviors of the object and background classes in a multiclassification problem in the best possible way. A possible solution is to adopt a more general parametric model that should satisfy two main properties: (i) flexibility (i.e., it should be capable of modeling a large variety of statistical behaviors) and (ii) stability (i.e., it should not require the estimation of a large number of parameters). Motivated by the above observations,

the present study proposes a new class of RBFs based on generalized Gaussian distribution (GGD).

The GGD requires only one additional parameter to be estimated compared to the Gaussian distribution, and it can approximate a large class of statistical distributions (e.g., impulsive, Laplacian, Gaussian, and uniform distributions). The analytical equation of the probability density function of the GGD is given by

$$p(\mathbf{x}; \mathbf{c}, r, \tau) = \frac{\tau}{2r\Gamma(1/\tau)} \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}\|^\tau}{r^\tau}\right), \tag{2}$$

where \mathbf{c} , $r > 0$ and $\tau > 0$ are the parameters of the mean, the scale or width and the shape of the distribution, respectively. $\Gamma(z)$ is the Gamma function, an extension of the factorial function, which is defined as $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t} dt$, for $z > 0$. The scale parameter r that expresses the width of the distribution is related to the normal standard deviation by the equation:

$$r = \sigma \sqrt{\frac{\Gamma(1/\tau)}{\Gamma(3/\tau)}}, \tag{3}$$

where σ is the normal standard deviation. The shape parameter τ refines the decay rate of the density function. It is worth noting that $\tau = 2$ yields Gaussian density and $\tau = 1$ results in Laplacian density distribution. As limit cases, for $\tau \rightarrow 0$, the distribution becomes impulsive, whereas for $\tau \rightarrow \infty$ it approaches uniform distribution (Fig. 1). Then, the scale parameter models the width of the GGD peak and the shape parameter is inversely proportional to the decreasing rate of the peak.

The GGD model is intrinsically stable, since it is characterized by few parameters to be estimated. Compared to Gaussian distribution, thanks to an additional statistical parameter (i.e., the shape parameter), it is more flexible and can approximate a large class of statistical distributions.

In this paper, based on this probability distribution, we define a novel RBF, by removing the constraints of a probability function, called generalized radial basis function (GRBF) which is defined using the following expression (for a k -dimensional input space):

$$\phi_j(\mathbf{x}; \mathbf{c}_j, r_j, \tau_j) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_j\|^{\tau_j}}{r_j^{\tau_j}}\right), \tag{4}$$

where $\mathbf{x}_i = (x_{i1}, \dots, x_{ik})^T$ is the vector of measurements, k is the number of inputs, r_j the width of the GRBF, $\mathbf{c}_j = (c_{j1}, \dots, c_{jk})^T$ the center and τ_j the shape parameter of the j -th GRBF.

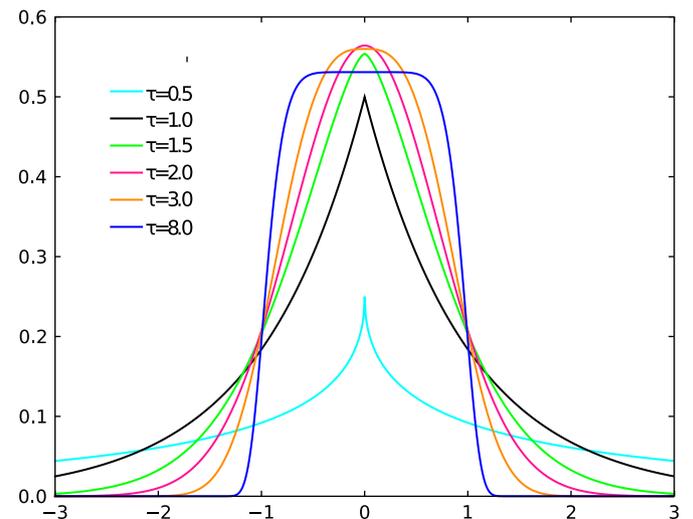


Fig. 1. Probability density function of the generalized Gaussian distribution (GGD) with different values of τ , $c=0$ and $r=1$.

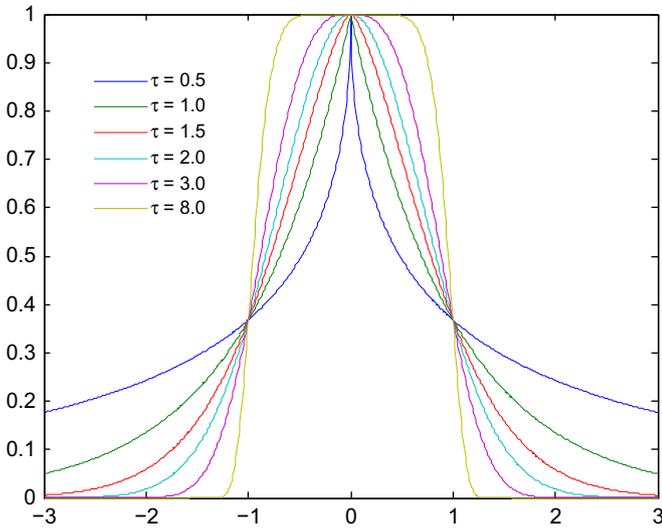


Fig. 2. Radial unit activation in one-dimensional space with $c=0$ and $r=1$ for the generalized RBF (GRBF) with different values of τ .

This basis function allows better matching between the shape of the kernel and the distribution of the distances, since the τ parameter provokes concavity or convexity around the point where the distance is the radii of the kernel, r . Fig. 2 presents the radial unit activation for the GRBF for different values of τ .

3. Single layer feedforward generalized radial basis function neural network

A scheme of single layer feedforward GRBFNN models is given in Fig. 3, where J is the number of classes and m is the number of hidden nodes or GRBFs of the ANN.

Suppose that there are n training patterns $(\mathbf{x}_i, \mathbf{t}_i)$, $i = 1, 2, \dots, n$, where $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{ik})^T$ and $\mathbf{t}_i = (t_{i1}, t_{i2}, \dots, t_{ij})^T$ are the i -th input pattern and its target, respectively. Let us denote that $\mathbf{c}_l = (c_{l1}, c_{l2}, \dots, c_{lk})^T$ is the center vector connecting the input units to the l -th hidden unit, r_l is the width of the l -th GRBF and τ_l is the shape parameter, $l = 1, \dots, m$. Finally, $\beta^j = (\beta_1^j, \beta_2^j, \dots, \beta_m^j)^T$ is the weight vector connecting the hidden nodes to the j -th output node. The main goal of training process is to determine the optimized parameters: \mathbf{c}_l , r_l , τ_l , and β^j , so that they minimize the squared error (SE) function defined by

$$SE = \sum_{i=1}^n \sum_{f=1}^J (o_{if} - t_{if})^2, \quad (5)$$

where o_{if} is the estimated output corresponding to the i -th input pattern and the f -th class.

A popular training algorithm for solving this problem has been backpropagation (BP) [25] in which the ANN parameters are tuned based on gradient descent with error propagation from the output layer to the input layer. Gradient-descent-based methods (like BP [25] or iRprop+ algorithms [26,24]) may easily get over-fitting or be stuck in local optima. Furthermore, it is well known that gradient-descent-based methods are computationally very costly. For that reason, the parameters of the single layer generalized radial basis function neural network are optimized with a modified version of the extreme learning machine (ELM) algorithm [17].

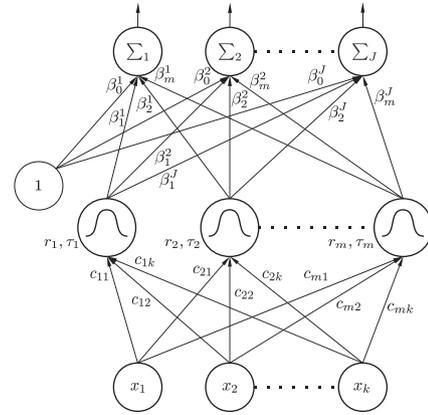


Fig. 3. Structure of generalized radial basis function neural networks.

4. Selection of parameters of the GRBFs via ELM: MELM-GRBF

Recently, an efficient learning algorithm, called extreme learning machine (ELM), for single layer networks (SLFNs) has been proposed by Huang et al. [27]. The minimization process of squared error function in the ELM is performed by using a linear system:

$$\mathbf{H}\beta = \mathbf{T}, \quad (6)$$

where \mathbf{H} is called as the hidden layer output matrix of the SLFN and defined as

$$\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_m) = \begin{pmatrix} \phi_1(\mathbf{x}_1; \mathbf{c}_1, r_1, \tau_1) & \dots & \phi_m(\mathbf{x}_1; \mathbf{c}_m, r_m, \tau_m) \\ \dots & \dots & \dots \\ \phi_1(\mathbf{x}_n; \mathbf{c}_1, r_1, \tau_1) & \dots & \phi_m(\mathbf{x}_n; \mathbf{c}_m, r_m, \tau_m) \end{pmatrix}_{n \times m}, \quad (7)$$

$$\mathbf{T} = (\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n)^T_{n \times J}, \quad (8)$$

and

$$\beta = (\beta^1, \beta^2, \dots, \beta^J)_{m \times J}. \quad (9)$$

While in the original ELM-RBF [1], the centers of the RBFs are initialized randomly (as the radius value) in MELM-GRBF, the centers of the GRBFs are taken randomly from patterns in the training set. For the i -th hidden node, an integer random k is selected in $\{1, \dots, n\}$, where n is the number of patterns in the training set. After that, the value of the center of the i -th hidden node is assigned to the value of the k -th pattern of the training set, i.e., $\mathbf{c}_i \leftarrow \mathbf{x}_k$, $k \in U(\{1, \dots, n\})$ and $U(\{1, \dots, n\})$ represents a one-dimensional uniformly distributed discrete random variable, where n is the number of patterns in the training set.

Once the centers of the GRBFs have been located, the value of the width and the shape parameter τ of each one have to be defined. Typically two alternatives are considered in order to determine the value of the width parameter. The first one consists in taking the widths r_j equal to a constant for all Gaussian functions [28,29]. In [29], for example, the widths are fixed as follows:

$$r = \frac{d_{max}}{\sqrt{2M}}, \quad (10)$$

where M is the number of centers and d_{max} is the maximum distance between those centers. Such a procedure fixes the degree of overlapping of the Gaussian kernels. This choice would be close to the optimal solution if the data were uniformly distributed in the input space, leading to a uniform distribution of the centroids. The second option consists in estimating the width of each RBF

independently, computing the width factors r_j by the k -nearest neighbor heuristic [30–32]:

$$r_j = \frac{1}{k} \sqrt{\left(\sum_{i=1}^k \|\mathbf{c}_j - \mathbf{c}_i\|^2 \right)} \quad (11)$$

where the \mathbf{c}_i are the k -nearest neighbors of centroid \mathbf{c}_j . A suggested value given in [30] for k is 2. This second class of methods offers the advantage of taking the distribution variations of the data into account.

In MELM-GRBF, the widths of the GRBF, r , and the shape parameters, τ , have been fitted according to the characteristics of the distribution of the distances because these parameters are intimately related to them.

Two requirements are to be fulfilled: (i) smallest distances in the distribution (which we note d_N) are to be mapped to high values (for instance, 0.95) of the basis function and (ii) largest distances (which we note d_F) must be mapped to lower values (for instance, 0.05). These high and low values will be determined by a user-defined λ parameter (e.g., if $\lambda = 0.05$ high values will be 0.95 and low values 0.05).

In the case of GRBFNN, τ and r can be calculated as follows by solving two different equations:

$$\exp\left(-\left(\frac{d_F}{r}\right)^\tau\right) = \lambda, \quad (12)$$

$$\exp\left(-\left(\frac{d_N}{r}\right)^\tau\right) = 1 - \lambda. \quad (13)$$

From these equations, the τ and r parameters are defined as

$$\tau = \frac{\ln\left(\frac{\ln(\lambda)}{\ln(1-\lambda)}\right)}{\ln\frac{d_F}{d_N}}, \quad (14)$$

$$r = \frac{d_N}{(-\ln(1-\lambda))^{1/\tau}} = \frac{d_F}{(-\ln(\lambda))^{1/\tau}}. \quad (15)$$

The most critical part in choosing the parameter of the basis function is in choosing the values of “far” distances (d_F). In our proposal, the d_F parameter is chosen as the distance from the center of the hidden node whose τ value is being adjusted to the nearest center, i.e., for the i -th hidden node, the d_F parameter is set as $d_F = \|\mathbf{c}_i - \mathbf{c}_j\|$, where j is the nearest hidden node to hidden node i . Analyzing various forms of GRBFs with different values of τ , we observed that the main difference between them lies in the tail, so the d_N value could be considered equal for all GRBF. Therefore, the d_N parameter is set to the square root of a sum of squared, random, small residuals along all dimensions, i.e., for the i -th hidden node, the d_N parameter is set as $d_N = \sqrt{(\delta^2) \times k}$, where k is the number of inputs and δ represents a small residual distance in each dimension, which in our case is taken as 0.001 (it is necessary to anticipate that the input data will be normalized in the interval $[-2,2]$).

Then, we show an example of how to calculate d_N and d_F parameters for an example with three hidden nodes and five input variables. Eq. (16) shows the pairwise distances between the three nodes in the hidden layer. As an example, for the second hidden node, the d_N parameter is calculated as $d_N = \sqrt{(\delta^2) \times k} = \sqrt{(0.001^2) \times 5} = 0.002$, and the d_F parameter is calculated as $d_F = \|\mathbf{c}_i - \mathbf{c}_j\| = \|\mathbf{c}_2 - \mathbf{c}_3\| = 0.031$:

$$\text{Pairwise distances} = \begin{pmatrix} 0.000 & 0.054 & 0.073 \\ 0.054 & 0.000 & 0.031 \\ 0.073 & 0.031 & 0.000 \end{pmatrix}. \quad (16)$$

By using these expressions for d_F and d_N , the only free parameter is λ , which determines the τ and r values for a given combination of d_F and d_N . The λ value which determines what is considered a high value and what is considered a low value was experimentally determined by a cross-validation procedure using the values $\{0.01, 0.02, \dots, 0.10\}$.

Once the respective definitions of d_N and d_F are set, τ is found by Eq. (14) and then r is determined using Eq. (15).

Finally, the output weights are determined by

$$\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}, \quad (17)$$

where \mathbf{H}^\dagger is the pseudo-inverse of \mathbf{H} . In summary, the MELM-GRBF algorithm can be described as follows (Fig. 4).

1: MELM-GRBF (λ, m):

2: Randomly take centers of the GRBFs from the patterns of the training set.

3: Calculate the d_N values by $d_N = \sqrt{(\delta^2) \times k}$.

4: Calculate the d_F values by $d_F = \|\mathbf{c}_i - \mathbf{c}_j\|$.

5: Compute the τ values by $\tau = \frac{\ln\left(\frac{\ln(\lambda)}{\ln(1-\lambda)}\right)}{\ln\frac{d_F}{d_N}}$.

6: Determine the r values by $r = \frac{d_N}{(-\ln(1-\lambda))^{1/\tau}} = \frac{d_F}{(-\ln(\lambda))^{1/\tau}}$.

7: Compute the hidden layer output matrix \mathbf{H} .

8: Calculate the output weights β by $\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$.

Fig. 4. MELM-GRBF framework.

5. Experiments

The proposed methodology was applied to 15 datasets taken from the UCI repository [33]. The selected datasets include binary problems and multi-class problems and present different numbers of instances, features and classes (see Table 1). The datasets with their corresponding partitions have been included on a public website.¹

In the first subsection, a description of the datasets and the experimental configuration is given. Then, there is a presentation of the proposed model compared to other models with different basis functions in the hidden layer. The influence of the λ and the number of hidden unit parameters is then evaluated and, finally, the values of τ are analyzed in order to find some hints explaining the difference of performance of the GRBF model.

5.1. Experimental design

The proposed method (MELM-GRBF) is compared with the ELM algorithm using different ANN models. The ANNs selected for comparison purposes differ in the activation function used in the hidden layer nodes. Thus, there is a comparison with models that have the following activation functions in the hidden layer nodes:

- Sigmoidal function (Sig). In this context, sigmoidal function refers to the special case of the logistic function, defined by the formula:

$$\text{sig}(n) = \frac{1}{1 - \exp(-n)}, \quad (18)$$

where n is the weighted sum of the inputs.

- Hard-limit transfer function (Hardlim). This transfer function returns zero if the argument of the function is less than zero and returns one if the argument is greater than or equal to

¹ <http://www.uco.es/grupos/ayrna/index.php?lang=en> (“Datasets” section).

Table 1
Characteristics of the 15 datasets used for the experiments: number of instances (Size), number of real (R), binary (B) and nominal (N) input variables, total number of inputs (In.), number of classes (Out.), and per-class distribution of the instances (Distribution).

Dataset	Size	R	B	N	In.	Out.	Distribution
Hepatitis	155	6	13	–	19	2	(32, 123)
Heart	270	13	–	–	13	2	(150, 120)
Haberman	306	3	–	–	3	2	(225, 81)
Card	690	6	4	5	51	2	(307, 383)
Pima	768	8	–	–	8	2	(500, 268)
German	1000	6	3	11	61	2	(700, 300)
Newthyroid	215	5	–	–	5	3	(150, 35, 30)
Balance	625	4	–	–	4	3	(288, 49, 288)
Gene	3175	–	–	60	120	3	(765, 765, 1648)
Lymph	148	3	9	6	38	4	(2, 81, 61, 4)
Anneal	898	6	14	18	59	5	(8, 99, 684, 67, 40)
Glass	214	9	–	–	9	6	(70, 76, 17, 13, 9, 29)
Ecoli	336	7	–	–	7	8	(143, 77, 52, 35, 20, 5, 2, 2)
Vowel	990	10	–	–	10	11	(90, 90, 90, 90, 90, 90, 90, 90, 90, 90)
Yeast	1484	8	–	–	8	10	(463, 429, 30, 163, 51, 44, 35, 244, 20, 5)

All nominal variables are transformed to binary variables.

zero. Hardlim is defined as follows:

$$\text{hardlim}(n) = 1 \text{ if } n \geq 0; = 0, \text{ otherwise.} \quad (19)$$

- Triangular basis function (Tribas). Tribas function is a piecewise linear (PWL) function and it calculates its output according to

$$\text{tribas}(n) = 1 - \text{abs}(n) \text{ if } -1 \leq n \leq 1; = 0, \text{ otherwise.} \quad (20)$$

- Radial basis function (Radbias). Radial basis function for additive type of SLFNs instead of RBF type of SLFNs. Radbias is defined as

$$\text{radbias}(n) = \exp(-n^2). \quad (21)$$

Finally, we also compare MELM-GRBF to the ELM-RBF proposed by Huang et al. [1], where the centroids and radius of the basis functions are selected randomly and the connections between hidden and output layers are determined by solving Eq. (17). The main difference between ELM-RBF and Radbias is that Radbias applies a standard linear combination of input variables and the connections between the input and hidden layers, and ELM-RBF measures the distance of each pattern to its centroid, weighting the final output by its radius.

The experimental design was conducted using a holdout cross-validation procedure with $3/4 \cdot n$ instances for the training dataset and $n/4$ instances for the generalization dataset. To evaluate the stability of the methods, the ELM algorithm was run 30 times for each problem. The performance of each model was evaluated using the correct classification rate (C) in the generalization set and the average time needed to train the model at each iteration, measured in seconds (T).

As previously stated, the λ value for MELM-GRBF was experimentally determined by a cross-validation procedure applied to the training set, using the values $\{0.01, 0.02, \dots, 0.10\}$. For all the approaches, the number of hidden nodes was also adjusted in a similar way, gradually increasing its value by an interval of 5 ($\{5, 10, 15, 20\}$) and then selecting the nearly optimal number of nodes based on a cross-validation method. Table 2 includes the optimal number of hidden nodes selected for each basis functions in each dataset.

Furthermore, a simple linear rescaling of the input variables was carried out over the interval $[-2, 2]$, with X_i^* being the transformed variables. Finally, all the simulations were carried out in MATLAB 2009 (R2009a) environment running in an Intel Core i5, 2.27 GHz CPU. The source code in MATLAB of the MELM-GRBF methodology is freely available upon request to the authors.

Table 2

Number of hidden nodes used for each basis function in the 15 datasets.

Dataset	ELM-Sig	ELM-Hardlim	ELM-Tribas	ELM-Radbias	ELM-RBF	MELM-GRBF
Hepatitis	5	20	15	20	20	15
Heart	20	20	15	20	20	15
Haberman	5	5	5	5	20	5
Pima	20	15	20	20	20	20
Newthyroid	20	20	20	20	20	10
Balance	20	20	20	20	20	15
Glass	15	20	20	20	20	20
Ecoli	15	20	20	20	20	10

In Card, German, Gene, Lymph, Anneal, Yeast, and Vowel, all basis functions were performed with 20 hidden nodes.

5.2. Results of MELM-GRBF

In this subsection, the MELM-GRBF method is compared to ELM with different basis functions (described in Section 5.1). The aim of this section is to show that by using the MELM-GRBF algorithm, the ELM methodology can improve its performance.

Table 3 shows the mean and the standard deviation of the correct classification rate (C_G) in the generalization set for each dataset and the Sig, Hardlim, Tribas, Radbias, RBF and GRBF basis functions and the mean of training time (T) of each basis function for all datasets. Based on the mean C_G and T , the ranking of each method in each dataset ($R=1$ for the best performing method and $R=6$ for the worst one) is obtained and the mean accuracy and training time (\bar{C}_G and \bar{T}) and the mean ranking (\bar{R}_{C_G} and \bar{R}_T) are also included in Table 3. From the analysis of the statistical descriptive results, it can be concluded that the MELM-GRBF method obtained the best results for all datasets in C_G . With regard to the training time T , the GRBF basis function alone was less efficient than other basis functions (Sig, Hardlim, Tribas and Radbias). Furthermore, the MELM-GRBF method yields the best mean ($\bar{C}_G = 77.58\%$) and ranking ($\bar{R}_{C_G} = 1.06$) in C_G .

Another aspect that is important to point out is that the MELM-GRBF method is far more robust than the ELM method with other basis functions, which can be observed on analyzing the values of standard deviation that the different basis functions generated for each dataset.

To determine the statistical significance of the rank differences observed for each method in the different datasets, we have carried out a non-parametric Friedman test [34] with the ranking of C_G and T of the best models as the test variables. The test shows that the effect of the method used for classification is statistically significant

Table 3

Statistical results of the ELM algorithm using different basis functions: Mean and standard deviation (SD) of the accuracy in the generalization set (C_G (%)), mean accuracy (\bar{C}_G (%)), mean accuracy ranking (\bar{R}_{C_G}), mean training time (\bar{T}) and mean training time ranking (\bar{R}_T).

Datasets	Method (C_G (%))					
	ELM-Sig	ELM-Hardlim	ELM-Tribas	ELM-Radbas	ELM-RBF	MELM-GRBF
Hepatitis	76.15 ± 4.24	75.98 ± 4.23	70.34 ± 8.34	74.01 ± 5.07	63.16 ± 12.97	80.32 ± 1.89
Heart	75.78 ± 3.51	77.05 ± 2.69	64.01 ± 4.97	66.37 ± 4.34	61.66 ± 9.81	77.64 ± 1.31
Haberman	73.11 ± 2.22	74.07 ± 1.04	71.00 ± 4.06	73.24 ± 2.07	66.75 ± 10.11	75.26 ± 1.00
Pima	76.92 ± 1.65	70.88 ± 3.15	70.05 ± 3.36	72.62 ± 2.79	51.28 ± 13.30	77.96 ± 1.37
Card	80.65 ± 3.76	76.57 ± 4.63	60.73 ± 4.13	64.85 ± 57	51.36 ± 8.22	90.05 ± 0.85
German	70.18 ± 1.59	70.38 ± 1.96	68.34 ± 1.70	68.58 ± 2.29	60.20 ± 14.50	73.56 ± 1.33
Newthyroid	93.39 ± 2.41	91.60 ± 4.83	93.33 ± 3.02	95.55 ± 3.06	80.43 ± 5.20	95.86 ± 1.61
Balance	90.55 ± 1.06	85.02 ± 3.47	69.46 ± 6.96	77.92 ± 6.67	68.48 ± 6.74	91.25 ± 0.53
Gene	55.19 ± 1.85	56.82 ± 2.21	43.86 ± 1.81	51.16 ± 1.02	38.05 ± 12.30	65.98 ± 2.15
Lymph	73.60 ± 6.72	76.93 ± 5.75	45.49 ± 6.66	52.43 ± 6.83	46.93 ± 13.51	85.58 ± 2.34
Anneal	85.45 ± 3.31	81.48 ± 3.41	64.68 ± 5.16	76.26 ± 2.42	60.74 ± 24.67	90.39 ± 1.38
Glass	67.71 ± 5.23	59.62 ± 6.87	64.46 ± 5.65	65.22 ± 4.69	43.20 ± 10.48	72.64 ± 2.34
Ecoli	87.49 ± 2.56	76.11 ± 4.20	77.45 ± 4.46	84.27 ± 2.66	64.47 ± 9.06	88.50 ± 1.54
Vowel	36.50 ± 3.97	29.83 ± 3.00	23.34 ± 3.57	28.60 ± 4.65	17.77 ± 3.63	48.09 ± 2.70
Yeast	56.56 ± 1.15	45.57 ± 2.66	50.04 ± 2.68	53.92 ± 1.32	29.76 ± 7.85	56.67 ± 1.09
\bar{C}_G (%)	73.25	69.86	62.43	66.99	53.61	77.98
\bar{R}_{C_G}	2.46	3.26	4.80	3.53	5.93	1.00
\bar{T}	0.016	0.014	0.016	0.016	0.055	0.019
\bar{R}_T	3.10	2.96	3.10	3.10	5.50	3.23

The best result is in bold face and the second best result in italics.

Table 4

Comparison of the MELM-GRBF method with other basis functions: critical difference (CD) values and differences of rankings of the Nemenyi and Bonferroni–Dunn tests, using MELM-GRBF as the control method and C_G as the test variable.

Nemenyi test						
Method(i)	Method(j)					
	ELM-Sig	ELM-Hardlim	ELM-Tribas	ELM-Radbas	ELM-RBF	MELM-GRBF
ELM-Sig	–	0.79	2.33	1.06	3.46	1.46
ELM-Hardlim	–	–	1.53	0.26	2.66	2.26 ⁺
ELM-Tribas	–	–	–	1.26	1.13	3.80 ⁺
ELM-Radbas	–	–	–	–	2.40	2.53 ⁺
ELM-RBF	–	–	–	–	–	4.93 ⁺
$CD_{\alpha=0.1} = 1.76, CD_{\alpha=0.05} = 1.94$						
Bonferroni–Dunn test						
Control method	Compared Method					
	ELM-Sig	ELM-Hardlim	ELM-Tribas	ELM-Radbas	ELM-RBF	MELM-GRBF
MELM-GRBF	1.46	2.26 ⁺	3.80 ⁺	2.53 ⁺	4.93 ⁺	–
$CD_{\alpha=0.1} = 1.58, CD_{\alpha=0.05} = 1.75$						

•, ◦: Statistically difference with $\alpha = 0.05$ (•) and $\alpha = 0.1$ (◦).
 +: The difference is in favor of Method(j) (Nemenyi test) or Control method (Bonferroni–Dunn test).

at a significance level of 5%, as the confidence interval is $C_0 = (0, F_{0.05} = 2.34)$ and the F -distribution statistical values are $F^* = 83.39 \notin C_0$ for C_G and $F^* = 5.34 \notin C_0$ for T . Consequently, we reject the null-hypothesis stating that all algorithms perform equally in mean ranking.

Based on this rejection, the Nemenyi post hoc test is used to compare all classifier to each other. This test considers that the performance of any two classifier is deemed significantly different if their mean ranks differ by at least the critical difference (CD):

$$CD = q \sqrt{\frac{K(K+1)}{6D}}, \tag{22}$$

where K and D is the number of classifiers and datasets, and the q value is derived from the studentized range statistic divided by $\sqrt{2}$ [35,36]. However, it has been noted that the approach of

comparing all classifiers to each other in a post hoc test is not as sensitive as the approach of comparing all classifiers to a given classifier. One approach to this latter type of comparison is the Bonferroni–Dunn test. This test can be computed using Eq. (22) with appropriate adjusted values of q [36].

The results of the Bonferroni–Dunn and Nemenyi tests for $\alpha = 0.10$ and 0.05 using C_G as the test variable can be seen in Table 4. From the results of this test, it can be concluded that MELM-GRBF obtains a significantly better C_G ranking than all the remaining classifiers except for the Sig basis function. Using the training time, T , as the test variable, the Hardlim method does not achieve significantly better results than the MELM-GRBF method.

The training time of the MELM-GRBF method is significantly better than that of the ELM-RBF. The reason for this is that the MELM-GRBF has been implemented taking into account the

benefits that MATLAB provided when using matrix operations instead of iterative loops.

5.3. Influence of the λ and m parameters on the accuracy of MELM-GRBF

As we mentioned above, the λ parameter is determined through a cross-validation process. For that reason, we considered interesting to analyze the influence of the λ parameter on the overall accuracy of the model. As an example, Fig. 5 shows the evolution of the generalization accuracy of the model for different values of the λ parameter in the Anneal, Ecoli, German and Glass datasets. In general, as observed from Fig. 5, the best performance values are obtained with λ values of 0.04–0.06 (which is also truth for the other datasets). Therefore, in order to reduce the complexity of the algorithm, the cross-validation process of the λ parameter could be eliminated. Because the best compromise between locality and coverage is obtained with $\lambda = 0.05$ value, we can consider 0.05 as the default value of the λ parameter.

In the MELM-GRBF methodology, the number of hidden nodes m is also a hyperparameter of the algorithm, together with λ . For

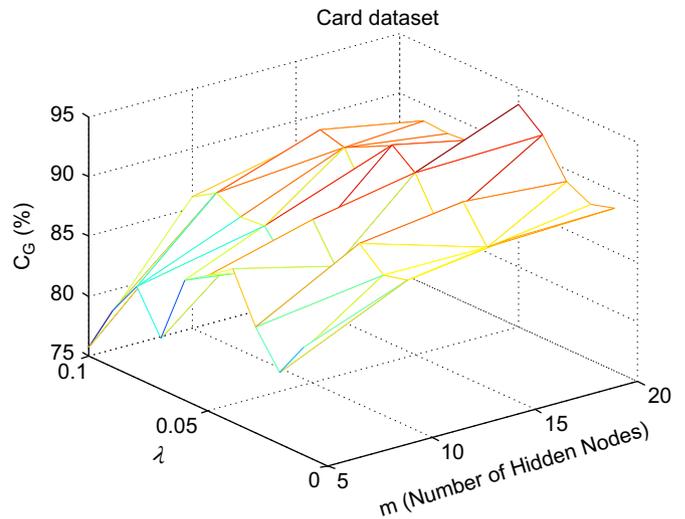


Fig. 6. The performance of MELM-GRBF is not very sensitive to the parameters (λ , m): an example on Card dataset.

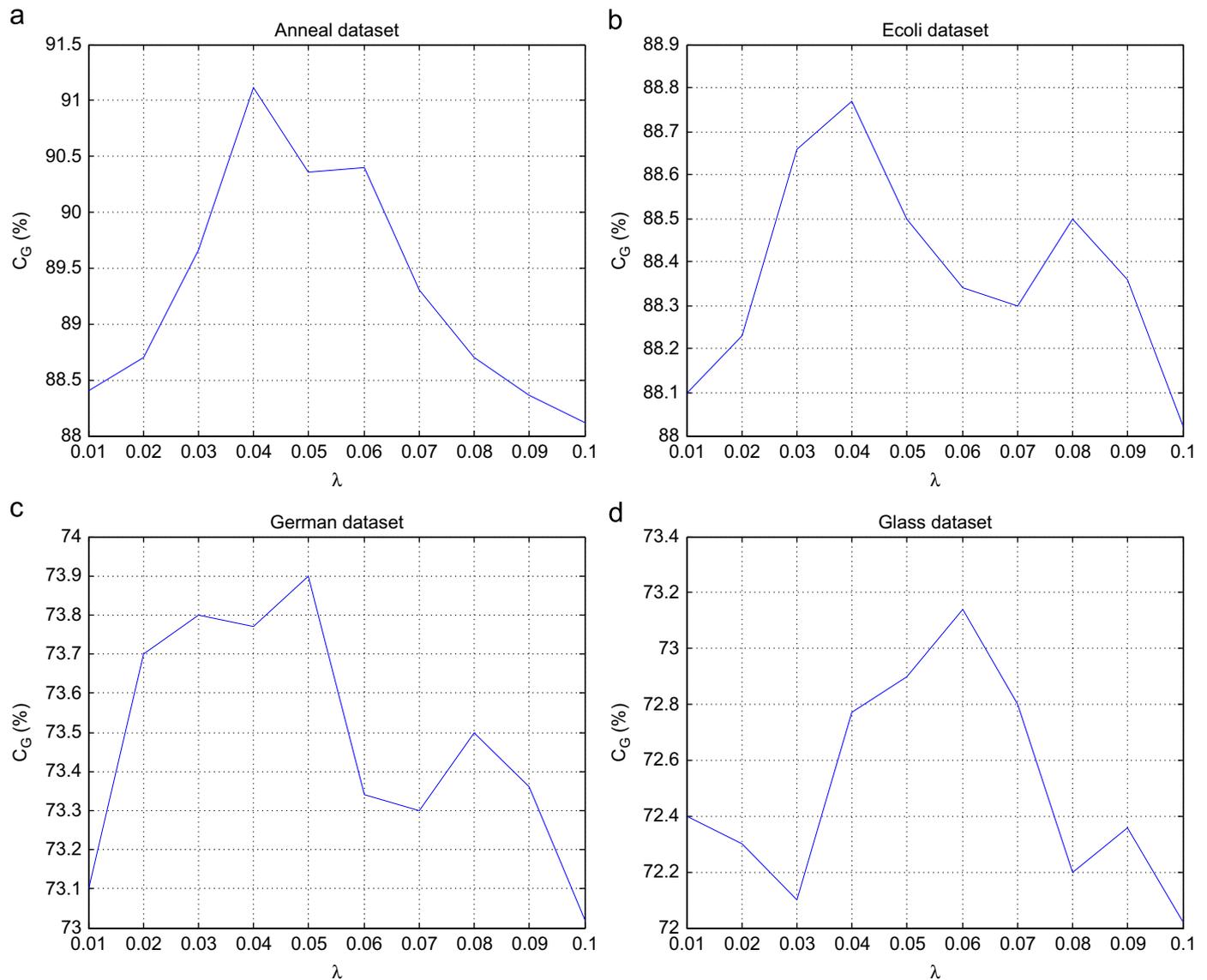


Fig. 5. Influence of the λ parameter on the accuracy of the model.

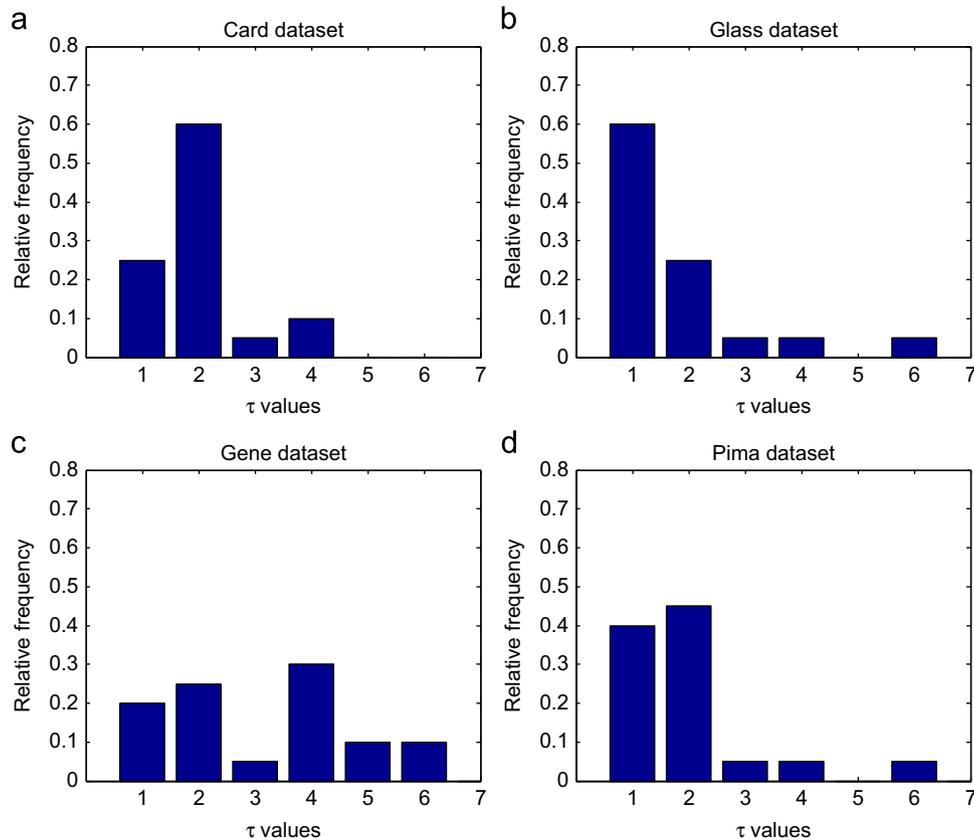


Fig. 7. Relative frequency distribution of τ values of the models provided by the MELM-GRBF.

each problem, we have considered 10 different values of λ and four different values of m resulting in a total of 40 pairs of (λ, m) . The 10 different values of λ are 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, and 0.10 and the four different values of m are 5, 10, 15, 20. After extensive simulations, it is found that the performance of MELM-GRBF is actually not so sensitive to the combination of (λ, m) . Similar to SVM with ELM kernel [37], MELM-GRBF usually can achieve competitive generalization performance in most cases as long as the number of hidden nodes m is large enough. As observed from Fig. 6, the generalization performance of MELM-GRBF tends to monotonically increase with the number of hidden nodes m . One only needs to adjust the λ parameter, although the performance of MELM-GRBF is not very sensitive to λ either. Furthermore, in the previous paragraph, we determined that $\lambda = 0.05$ obtained best compromise between locality and coverage.

5.4. Analysis of τ values

The aim of this subsection is to analyze the values of τ , in order to justify the use of MELM-GRBF instead of ELM-RBF. This can give us some hints about the good performance of the MELM-GRBF model compared to ELM-RBF. Fig. 7 shows the relative frequency distributions of the τ values of the models obtained in the 30 runs for Card, Gene, Glass and Pima datasets. The higher the number of hidden nodes with $\tau \neq 2$, the more different the behavior of the MELM-GRBF with respect to a standard RBFNN.

The RBFNN with GRBF kernels generally presents the best accuracy results in the experiments. These results can be explained because the MELM-GRBF could find good values of the τ parameter for the radial units, which generates a better choice for the shape of the GRBFs. While the final values of τ in the experiments with Pima or Card (Fig. 7(a) and (d)) were close to 2, i.e., the shape of the GRBF was similar to the shape of the Gaussian function, the values of τ

were higher in the experiments with Gene dataset (Fig. 7(c)). The higher flexibility provided by the use of radial units with different RBF shapes (different values of τ) also explains the good performance of the RBFNN with the GRBF kernel.

6. Conclusions

This paper proposes a new approach to determine the optimized parameters for the generalized radial basis functions neural networks. The use of GRBFs has made possible the modification of the shape of the RBF by changing a real parameter τ , and to have radial units with different RBF shapes in the same RBFNN. The coefficients of the model are estimated by means of a modified version of the extreme learning machine (ELM). In this way, the centers of each GRBF are taken randomly from the patterns of the training set and the radius and τ values are determined analytically.

The evaluation of the model and the algorithm for the 15 datasets considered show that the MELM-GRBF is more accurate and robust when compared with the rest of the basis functions used in the ELM (Sig, Hardlim, Tribas, Radbas and ELM-RBF). MELM-GRBF has the highest statistically significant mean ranking of C_G compared to Hardlim, Tribas, Radbas, ELM-RBF and the highest mean ranking of C_G (though not statistically significant) when compared to the Sig basis function. Moreover, MELM-GRBF shows promising values of training time compared to the rest of the basis functions. The influence of the two hyperparameters of the model was also analyzed, concluding that MELM-RBF is not so sensitive to their values.

Acknowledgment

The authors would like to thank Dr. Huang GB who generously provided us with the source code of the ELM-RBF. This work has

been partially subsidized by the TIN 2008-06681-C06-03 project of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P08-TIC-3745 project of the “Junta de Andalucía” (Spain). The research of Francisco Fernández-Navarro has been funded by the “Junta de Andalucía” Predoctoral Program, grant reference P08-TIC-3745.

References

- [1] G.B. Huang, C. Siew, Extreme learning machine: RBF network case, in: 8th International Conference on Control, Automation, Robotics and Vision (ICARCV), vol. 2, 2004, pp. 1029–1036.
- [2] J. Park, I.W. Sandberg, Approximation and radial-basis-function networks, *Neural Computation* 5 (2) (1993) 305–316.
- [3] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, UK, 1996.
- [4] C.M. Bishop, Improving the generalization properties of radial basis function neural networks, *Neural Computation* 3 (4) (1991) 579–581.
- [5] D. Manrique, J. Rios, A. Rodríguez-Patón, Evolutionary system for automatically constructing and adapting radial basis function networks, *Neurocomputing* 69 (16–18) (2006) 2268–2283.
- [6] A. Stalano, R. Tagliaferri, W. Pedrycz, Improving RBF networks performance in regression tasks by means of a supervised fuzzy clustering, *Neurocomputing* 69 (13–15) (2006) 1570–1581.
- [7] I.C. Yeh, W. Cheng, First and second order sensitivity analysis of MLP, *Neurocomputing* 73 (10–12) (2010) 2225–2233.
- [8] F.J. Martínez-Estudillo, C. Hervás-Martínez, P.A. Gutiérrez, A.C. Martínez-Estudillo, Evolutionary product-unit neural networks classifiers, *Neurocomputing* 72 (1–2) (2008) 548–561.
- [9] A.C. Martínez-Estudillo, F.J. Martínez-Estudillo, C. Hervás-Martínez, N. García, Evolutionary product unit based neural networks for regression, *Neural Networks* 19 (4) (2006) 477–486.
- [10] P.A. Gutiérrez, C. Hervás-Martínez, M. Carbonero, J.C. Fernández, Combined projection and kernel basis functions for classification in evolutionary neural networks, *Neurocomputing* 72 (13–15) (2009) 2731–2742.
- [11] Z. Uykan, C. Guzelis, Input-output clustering for determining centers of radial basis function network, in: *Proceedings of the 1997 European Conference on Circuit Theory and Design*, vol. 2, Technical University of Budapest, European Circuit Society, Hungary, 1997, pp. 435–439.
- [12] M. Vogt, Combination of radial basis function neural networks with optimized learning vector quantization, in: *Proceedings of IEEE International Conference on Neural Networks (ICNN)*, vol. 3, San Francisco, CA, 1993, pp. 1841–1846.
- [13] L. Xu, RBF nets, mixture experts, and Bayesian Ying-Yang learning, *Neurocomputing* 19 (1–3) (1998) 223–257.
- [14] G. Bugmann, Normalized Gaussian radial basis function networks, *Neurocomputing* 20 (1–3) (1998) 97–110.
- [15] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, in: *Intelligent Signal Processing*, IEEE Press, 2001, pp. 306–351.
- [16] M. Ventresca, H.R. Tizhoosh, Improving gradient-based learning algorithms for large scale feedforward networks, in: *IJCNN'09: Proceedings of the 2009 International Joint Conference on Neural Networks*, IEEE Press, Piscataway, NJ, USA, 2009, pp. 1529–1536.
- [17] G.B. Huang, Q. Zhu, C. Siew, Extreme learning machine: theory and applications, *Neurocomputing* 70 (1–3) (2006) 489–501.
- [18] G.B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, *Neurocomputing* 71 (16–18) (2008) 3460–3468.
- [19] Y. Lan, Y.C. Soh, G. Huang, Ensemble of online sequential extreme learning machine, *Neurocomputing* 72 (13–15) (2009) 3391–3395.
- [20] G. Feng, G.B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, *IEEE Transactions on Neural Networks* 20 (8) (2009) 1352–1357.
- [21] J. Cao, Z. Lin, G.B. Huang, Composite function wavelet neural networks with extreme learning machine, *Neurocomputing* 73 (7–9) (2010) 1405–1416.
- [22] F. Scarselli, A. Chung Tsoi, Universal approximation using feedforward neural networks: a survey of some existing methods, and some new results, *Neural Networks* 11 (1) (1998) 15–37.
- [23] F. Fernández-Navarro, C. Hervás-Martínez, P.A. Gutiérrez, M. Carbonero, Evolutionary q-gaussian radial basis functions neural networks for multi-classification, *Neural Networks*, in press. URL <<http://dx.doi.org/10.1016/j.neunet.2011.03.014>>.
- [24] F. Fernández-Navarro, C. Hervás-Martínez, M. Cruz, P.A. Gutiérrez, A. Valero, Evolutionary q-Gaussian radial basis function neural network to determine the microbial growth/thno growth interface of *Staphylococcus aureus*, *Applied Soft Computing* 11 (3) (2011) 3012–3020.
- [25] Y. Chauvin, D.E. Rumelhart, Backpropagation: Theory, Architectures, and Applications, Lawrence Erlbaum Associates, Inc, Mahwah, NJ, USA, 1995.
- [26] C. Igel, M. Hüsken, Empirical evaluation of the improved Rprop learning algorithms, *Neurocomputing* 50 (6) (2003) 105–123.
- [27] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: *IEEE International Conference on Neural Networks—Conference Proceedings*, vol. 2, 2004, pp. 985–990.
- [28] J. Park, I.W. Sandberg, Universal approximation using radial basis function networks, *Neural Computation* 3 (2) (1991) 246–257.
- [29] S. Haykin, *Neural Networks: A Comprehensive Foundation*, third ed., Prentice-Hall, 2008.
- [30] J. Moody, C. Darken, Fast learning in networks of locally-tuned processing units, *Neural Computation* 1 (2) (1989) 281–294.
- [31] F. Fernández-Navarro, A. Valero, C. Hervás-Martínez, P. Gutiérrez, R. García-Gimeno, G. Zurera-Cosano, Development of a multi-classification neural network model to determine the microbial growth/no growth interface, *International Journal of Food Microbiology* 141 (2010) 203–212.
- [32] F. Fernández-Navarro, C. Hervás-Martínez, P. Gutiérrez, A dynamic over-sampling procedure based on sensitivity for multi-class problems, *Pattern Recognition* 44 (8) (2011) 1821–1833. URL <<http://dx.doi.org/10.1016/j.patcog.2011.02.019>>.
- [33] A. Asuncion, D. Newman, UCI Machine Learning Repository, 2007. URL <<http://www.ics.uci.edu/~mlearn/MLRepository.html>>.
- [34] M. Friedman, A comparison of alternative tests of significance for the problem of *m*rankings, *Annals of Mathematical Statistics* 11 (1) (1940) 86–92.
- [35] O.J. Dunn, Multiple comparisons among means, *Journal of the American Statistical Association* 56 (1961) 52–56.
- [36] Y. Hochberg, A. Tamhane, *Multiple Comparison Procedures*, John Wiley & Sons, 1987.
- [37] G. Huang, X. Ding, H. Zhou, Optimization method based extreme learning machine for classification, *Neurocomputing* 74 (1–3) (2010) 155–163.



Francisco Fernández Navarro was born in Córdoba, Spain, in 1984. He received the B.S. degree in Computer Science from the University of Córdoba, Spain, in 2007. He is currently working toward the Ph.D. degree at the Department of Computer Science and Numerical Analysis. His current interests include radial basis function neural networks, evolutionary computation and hybrid algorithms.



César Hervás Martínez was born in Cuenca, Spain. He received the B.S. degree in Statistics and Operating Research from the Universidad Complutense, Spain, in 1978 and the Ph.D. degree in Mathematics from the University of Seville, Seville, Spain, in 1986. He is a Professor with the University of Córdoba in the Department of Computing and Numerical Analysis, in the area of computer science and artificial intelligence. His current research interests include neural networks, evolutionary computation, and the modelling of natural systems.



Javier Sanchez-Monedero is a Ph.D. Student in the Department of Computer Science and Numerical Analysis, University of Córdoba (Spain). He received the B.S. in Computer Science from the University of Granada, Spain, in 2008 and the M.S. in Multimedia Systems from the University of Granada, Spain, in 2009. His research interests are related to real-time distributed systems middleware and evolutionary algorithms and artificial neural networks for classification.



Pedro Antonio Gutiérrez was born in Córdoba, Spain, in 1982. He received the B.S. degree in Computer Science from the University of Seville, Spain, in 2006 and the Ph.D. degree in Computer Science and Artificial Intelligence from the University of Granada, Granada, Spain, in 2009. He is an Assistant Professor with the University of Córdoba in the Department of Computer Science and Numerical Analysis. His current research interests include neural networks and their applications, evolutionary computation and hybrid algorithms.