International Journal of Hybrid Intelligent Systems 7 (2010) 1–10
DOI 10.3233/HIS-2010-0117
IOS Press

Classification by evolutionary generalised radial basis functions

Adiel Castaño[b,*], Francisco Fernández-Navarro[a], César Hervás-Martínez[a], M.M. García[c] and Pedro Antonio Gutiérrez[a]
[a]*Department of Computer Science and Numerical Analysis, University of Córdoba, Campus de Rabanales, Albert Einstein building, 3rd floor, 14071 – Córdoba, Spain*
[b]*Department of Informatics, University of Pinar del Rio, Pinar del Rio, Cuba*
[c]*Department of Computer Science, University of Las Villas, Santa Clara, Cuba*

Abstract. This paper proposes a Neural Network model using Generalised kernel functions for the hidden layer of a feed forward network. These functions are Generalised Radial Basis Functions (GRBF), and the architecture, weights and node topology are learned through an evolutionary algorithm. The proposed model is compared with the corresponding standard hidden-node models: Product Unit (PU) neural networks, Multilayer Perceptrons (MLP) with Sigmoidal Units (SUs) and the RBF neural networks. The proposed methodology is tested using twelve benchmark classification datasets from well-known machine learning problems. GRBFs are found to perform better than other standard basis functions at the classification task.

Keywords: Classification, Neural Networks, Generalized Radial Basis Functions, Evolutionary Algorithm, Radial Basis Functions

1. Introduction

The simplest classification method generates a class label based on observations via linear functions of the input variables. This process of model fitting is quite stable, resulting in low variance but a potentially high bias. Frequently, in real-classification problem, we cannot make the stringent assumption of additive and purely linear effects of the input variables. A traditional technique to overcome these difficulties is to augment or replace the variables of the input vector with new variables, the basis functions. These, are transformations of the input variables, and a linear model can be used in this new space of derived input features.

Depending on the basis function used in the hidden nodes, we can distinguish, among other, the following Neural Networks (NNs) [1]: 1) NNs based on projection basis functions, such as the Multilayer Perceptron (MLP) NNs where the transfer functions are Sigmoidal Units (SUs) or the Product Unit (PU) NN [2,3], and 2) NNs based on kernel basis functions, such as the Radial Basis Function (RBF) NN [4].

The RBF NN can be considered a local averaginge procedure. Its improved approximation capability and its architecture have drawn a lot of attention. Bishop [4] concluded that an RBF NN can provide a fast linear algorithm capable of representing complex nonlinear mappings. A detailed discussion on various applications of the RBF NN can be found in [5]. The RBF NN has been proven to be a universal approximator [6]. In the literature, it has been compared to other universal approximators such as MLPs [5,7] and fuzzy systems [8,9].

Training of RBFs can be classified into two categories: quick learning and full learning. A Quick learning usually involves a two-step process. First, the parameters governing the basis functions are determined by a relatively fast, unsupervised clustering or vector

*Corresponding author: Adiel Castaño. Department of Informatics. University of Pinar del Rio. Pinar del Rio. Cuba. Tel./Fax: +53 48726803; E-mail: adiel@info.upr.edu.cu.

quantisation (VQ) approach. Next, the weights of the basis functions are determined using linear optimisation techniques. One drawback of quick learning is that the clustering/VQ methods are usually unsupervised and the target outputs are totally ignored during learning [10]. In addition, the linear optimisation problem may be ill-condited in some cases [11]. A full learning scheme optimises all of the parameters in a supervised mode [10,12,13]. Compared to quick learning, full learning usually provides more accurate results because it trains the model using both input and target variables.

In high-dimensional space, all pairwise distances between patterns seem to be very similar, i.e., the distances to the nearest and furthest neighbours look nearly identical. In this kind of problem, the distances are concentrated, and the Gaussian kernel loses its interpretation in terms of locality around its centre [14]. Therefore, the widely-used Gaussian kernel and Euclidean distance are not necessarily appropriate functions to quantify similarity in high dimensional spaces. Despite this problem, the use of the Euclidean distance in high-dimensional spaces has not been questioned by the machine learning community, because it corresponds to the distance as we define for our three-dimensional world.

This work evaluates the accuracy obtained by a special class of RBF NNs, namely Generalised Radial Basis Function (GRBF) NNs. This work aims to alleviate the problem associated with the high dimensionality of the input space. It is important to observe that the GRBF NN is not an alternative to the classic Gaussian function but is, a parametric generalisation of Gaussian function, with a new parameter, τ, that relaxes or contracts the basis function (Fig. 1). The training of these networks is performed by a specific evolutionary algorithm, in which the principal issue is the establishment of a method to choose adequate values for the principal parameters of the GRBF NNs, which involves fitting a new parameter τ. The key of this proposal is constituted by the relationship between this new parameter and deviations of distances or the width of the RBF, r.

This paper is organised as follows. Section 2 introduces RBF Neural Networks and specifically, the GRBF model. Section 3 formally presents the GRBF model considered in this work, which is adapted to the classification problem. In section 4, the main characteristics of the algorithm used for training the model are described. Section 5 presents the experiments carried out and discusses the results obtained. Finally, Section 6 presents the main conclusions and future directions suggested by this study.

2. RBF Neural Networks and GRBF model

RBF NNs [15] have been used in extremely varied domains, including function approximation, pattern classification, time series prediction, data mining, signals processing, and nonlinear system modelling and control. RBF NNs have some useful properties that render them suitable for modelling and control. From a structural viewpoint, RBFNN are closely related to direct kernel methods [16] and Support Vector Machines (SVM) with Gaussian kernels functions [17–19].

The RBF NN learning process, i.e., the optimisation of adjustable parameters, includes determining centre vectors, radii (or widths) of the distributions, and linear output weights connecting the RBF hidden nodes to the output nodes. Another important issue is the determination of the network structure or the number of RBF hidden nodes based on the parsimonious principle [20, 21]. The number and positions of the basis functions, which correspond to the neurons in the hidden layer of the network, have an important influence on the performance of the RBF NN. Both problems have been tackled using a variety of approaches. For instance, the number and position of the RBFs may be fixed and defined a priori [21]; they may be determined by unsupervised clustering algorithms [22]; they may be determined through a supervised learning scheme that includes growing and pruning procedures [23]; or they can be evolved using evolutionary [24,25], or hybrid algorithms [26].

One of the most common RBF models is represented by a Gaussian function where the output depends on the distance between the instance and the centre of the RBF. This distance can be formulated in different ways. The most common formulation is the Euclidean distance, but when dimensionality grows and/or when data are concentrated in boundaries of the K dimensional space, standard Gaussian basis functions do not perform well. To prevent the effects observed for standard Gaussian RBFs, these basis functions can be generalised by including a new parameter τ which can relax or contract the basis functions. In this way, the Generalized Radial Basis Function (GRBF) is defined using the following expression [14]:

$$B(\mathbf{x}, \mathbf{w}_j) = exp\left(-\frac{\|\mathbf{x} - \mathbf{c_j}\|^{\tau_j}}{r_j^{\tau_j}}\right) \quad (1)$$

where K is the number of inputs, $\mathbf{x}_i = (x_{1i}, \ldots, x_{Ki})$ is the random vector of measurements and $\mathbf{w}_j = (w_{j0}, w_{j1}, \ldots, w_{jK}, w_{j(K+1)})$, with $r_j = w_{j0}$, $\mathbf{c}_j = (w_{j1}, \ldots, w_{jK})$ and $\tau_j = w_{j(K+1)}$. In this way, the

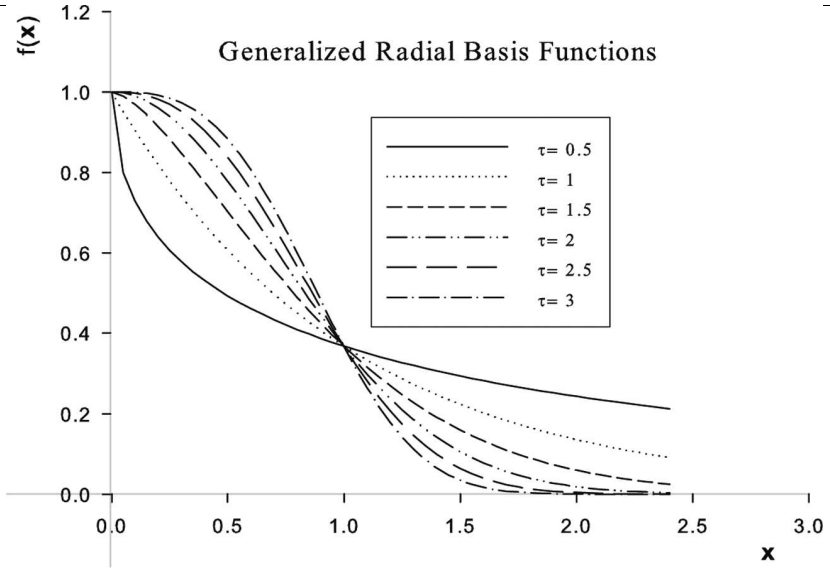

Fig. 1. Generalised Gaussian with $r = 1$, and different τ values.

weight vector $\mathbf{w}_j$ includes, respectively, the width, the centre and the exponent of the j-th GRBF. These basis functions allow a 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 r (see Fig. 1).

3. Generalized Radial Basis Functions for Classification

In a classification problem, measurements x_i, $i = 1, 2, \ldots, K$, of a single individual (or object) are taken, and the individuals are to be classified into one of the J classes based on these measurements. A training sample $D = \{(\mathbf{x}_n, \mathbf{y}_n); n = 1, 2, \ldots, N\}$ is available, where $\mathbf{x}_n = (x_{1n}, \ldots, x_{kn})$ is the random vector of measurements taking values in $\Omega \subset \mathbb{R}^K$, and $\mathbf{y}_n$ is the class level of the n-th individual, where the common technique of representing class levels using a "1-of-J" encoding vector is adopted, $\mathbf{y} = (y^{(1)}, y^{(2)}, \ldots, y^{(J)})$, and the Correctly Classified Rate or accuracy of the classifier is defined by $C = \frac{1}{N}\sum_{n=1}^{N} I(C(\mathbf{x}_n) = \mathbf{y}_n)$, where $I(.)$ is the zero-one lost function. A good classifier tries to achieve the highest possible C in a given problem.

In order to tackle this classification problem, the outputs of the GRBF model have been interpreted from the point of view of probability through the use of the softmax activation function [2], which is given by:

$$g_l(\mathbf{x}, \theta_l) = \frac{\exp f_l(\mathbf{x}, \theta_l)}{\sum_{j=1}^{J} \exp f_j(\mathbf{x}, \theta_j)}, l = 1, 2, \ldots, J \quad (2)$$

where J is the number of classes in the problem, $f_j(\mathbf{x}, \theta_l)$ is the output of the j output neuron for pattern $\mathbf{x}$, $\theta_l = (\beta_0^l, \cdots, \beta_m^l, \mathbf{w}_1, \cdots, \mathbf{w}_m)$, m is the number of GRBFs and $g_l(\mathbf{x}, \theta_l)$ is the probability a pattern $\mathbf{x}$ has of belonging to class j. The model to estimate the function $f_l(\mathbf{x}, \theta_l)$ is defined by the following equation:

$$f_l(\mathbf{x}, \theta_l) = \beta_0^l + \sum_{j=1}^{m} \beta_j^l \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^{\tau_j}}{r_j^{\tau_j}}\right) \quad (3)$$

Using the softmax activation function presented in Eq. (2), the class predicted by the NN corresponds to the node in the output layer whose output value is the greatest. In this way, the optimum classification rule $C(\mathbf{x})$ is the following:

$$C(\mathbf{x}) = \widehat{l}, \text{ where } \widehat{l} = \text{argmax}_l\ g_l(\mathbf{x}, \theta_l), \quad \text{for } l = 1, 2, \ldots, J \quad (4)$$

The function used to evaluate a GRBF NN is the function of cross-entropy error and it is given by the following expression:

$$l(\theta) = \sum_{n=1}^{N}[-\sum_{l=1}^{J} y_n^{(l)} f_l(\mathbf{x}_n, \theta_l) + log \sum_{l=1}^{J} \exp f_l(\mathbf{x}_n, \theta_l)] \quad (5)$$

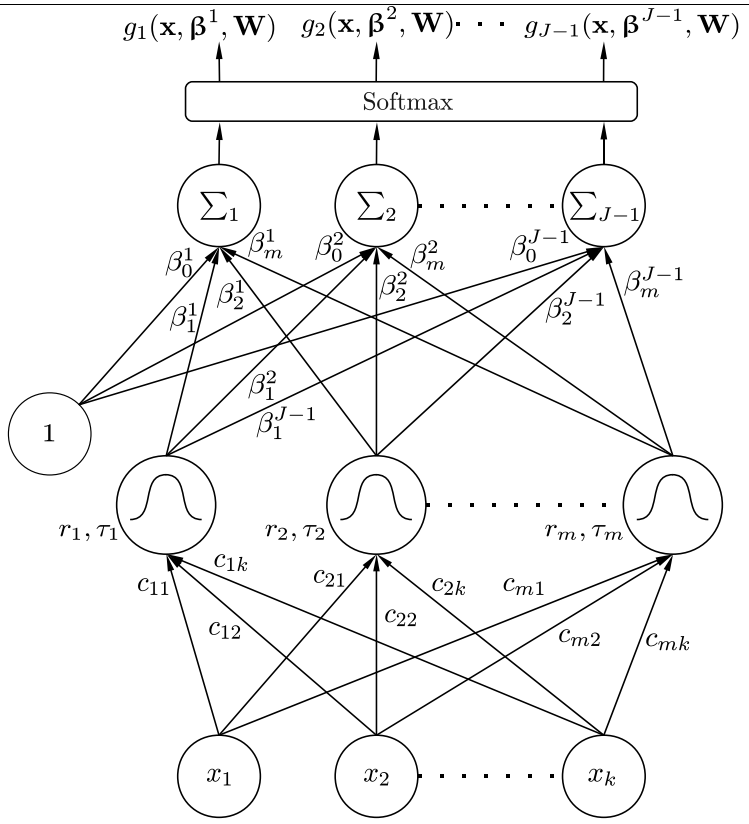

Fig. 2. Structure of Generalised Radial Basis Function Neural Networks: an input layer with k input variables, a hidden layer with m RBFs and an output layer with $J-1$ nodes.

where $\theta = (\theta_1, \ldots, \theta_J)$. The proposed algorithm returns the best cross-entropy individuals as feasible solutions. Finally, because of the normalization condition:

$$\sum_{l=1}^{J} g_l(\mathbf{x}, \theta_l) = 1 \tag{6}$$

and the probability for one of the classes does not need to be estimated. For that reason, the GRBF NN models proposed have $J-1$ outputs nodes instead of J output nodes since the $g_J(\mathbf{x}, \theta_J) = 1 - \sum_{i=0}^{J-1} g_i(\mathbf{x}, \theta_i)$. This was used to reduce the number of output nodes in the GRBF NN and, consequently, the complexity of the model. A scheme of these models is given in Fig. 2, where J is the number of classes and m is the number of hidden nodes or RBFs of the neural net.

The error surface associated with the model is very convoluted with numerous local optima and the Hessian matrix of the error function $l(\theta)$ is, in general, indefinite. Moreover, the optimal number of basis functions in the model (i.e. the number of hidden nodes in the neural network) is unknown, and, in this case, the GRBFs are not Mercer's kernels, i.e., they are not positive semi-definite for all values of the τ parameter [14]. This means that the optimisation problem will generally not be convex. Thus, we estimate the parameters θ by means of an evolutionary algorithm. This kind of metaheuristics has been proof to be very effective when optimizing neural network models [27–29].

4. Evolutionary Algorithm

A population-based Evolutionary Algorithm (EA) is used for the architectural design and the estimation of the real coefficients of the GRBF NN. The search begins with an initial population of GRBF NNs, and, in each iteration, the population is updated using a population-update algorithm. The population is subject to the operations of replication and mutation. The general structure of the EA is similar to the structure of the one presented in [30,31], but with several significant modifications. In the current approach, $l(\theta)$ is the error function for an individual g of the population, where g is a

GRBF NN, which is given by the multivaluated function $\mathbf{g}(\mathbf{x},\theta) = \{g_1(\mathbf{x},\theta_1),\ldots,g_l(\mathbf{x},\theta_{J-1})\}$ [since the number of outputs can be reduced because of the normalization condition, see Eq. (6)]. The fitness measure is a strictly decreasing transformation of the entropy error $l(\theta)$ given by $A(g) = \frac{1}{1+l(\theta)}$. The severity of mutations depends on the temperature $T(g)$ of the GRBF NN model, defined by $T(g) = 1 - A(g), 0 \leqslant T(g) \leqslant 1$.

To define the topology of the NNs generated during the evolution process, we consider three parameters: m, M_{E} and M_{I}. The parameters m and M_{E} correspond to the minimum and maximum number of hidden nodes in the whole evolutionary process and M_{I} corresponds to the maximum number of hidden nodes in the initialisation process. To obtain an initial population formed by models simpler than the most complex model possible, the parameters must fulfil the condition $m \leqslant M_{\mathrm{I}} \leqslant M_{\mathrm{E}}$.

We generate $10N_{\mathrm{p}}$ networks, where $N_{\mathrm{p}} = 200$ is the number of population networks during the evolutionary process. Then we select the best N_{p} neural networks. To generate a network, the number of nodes in the hidden layer is taken from a uniform distribution in the interval $[m, M_{\mathrm{I}}]$. For hidden nodes, the number of connections is always $K+2$, because these connections represent, respectively, the centre, the width and the exponent of each generalised radial basis function. The number of connections between each output node and the hidden layer is determined from a uniform distribution in the interval $(1, J-1)$, and the connection to the bias node is always present.

For the GRBF hidden nodes, the connections between the input layer and hidden layer are initialized using a clustering algorithm, so the EA can start the evolutionary process with well positioned centers. The main idea is to cluster input data in M groups, M being the number of GRBF hidden nodes. Therefore, each hidden GRBF neuron can be positioned at the centroid of its corresponding cluster.

The first algorithm modification consists of initialisation of the GRBF radii, where the determination of the initial τ and r values are intimately related to the distribution of the distances and can be set according to the specificities of that distribution. The method to choose adequate values for τ and r is based on the following criterion: the largest or "furthest" distances (d_{F}, the 5-th percentile of the distribution) must be mapped to lower values of probability. Then, τ and r can be calculated as follows by solving two different equations [32]:

$$\begin{aligned} exp\left(-\left(\frac{d_{\mathrm{F}}}{r}\right)^{\tau}\right) &= 0.05; \\ exp\left(-\left(\frac{\mu}{r}\right)^{\tau}\right) &= 0.5 \end{aligned} \tag{7}$$

where μ is the mean of point's distances of the members of the cluster to the centroid and d_{F} is a value representing "farest" distances. Indeed, the most critical part of these equations is determined by the μ and d_F values. For this reason, we use an estimator of d_F associated to the statistical distribution of the distances between the centroids and the individuals in the cluster: d_F values can be approximately calculated, under normality hypothesis, as $\mu + 1.645\sigma$, where σ is the standard deviation of these distances.

In every generation, a parametric mutation is accomplished for each coefficient w_{ji} or β_j^l of the model by adding Gaussian noise, where the variances of the Normal distribution are updated throughout the evolution of the algorithm. Once the mutation is performed, the fitness of the individual is recalculated and a conventional simulated annealing process is applied. First, the link weights are mutated by adding a value $\varepsilon \in N(0, \alpha \cdot T(g))$ where α is the learning rate and $T(g)$ is the network temperature. The radii r_j of each GRBF hidden node is mutated in the same way, adding another value $\eta \in N(0, \alpha \cdot T(g))$. The learning rate α is updated throughout the evolution of the algorithm. There are different methods to update the variance. We use one of the simplest methods: the 1/5 success rule of Rechenberg [33]. The modification of GRBFs is very sensitive to τ variation. Indeed, when τ is near to the interval $[0, 2.5]$, a τ variation drastically changes the contraction of the GRBF basis function. On the other hand, when $\tau \gg 2.5$, the same τ variation does not drastically change the generalised Gaussian. Due to this behaviour, the τ modification value must depend on the desired effect (see Fig. 3). To define this desired effect, the τ mutation is formulated as:

$$\tau_{n+1} = \frac{e \cdot r(\tau_n - e \cdot r \cdot tan(\Delta))}{e \cdot r + \tau_n tan(\Delta)} \tag{8}$$

where e is the Euler's constant, τ_n, τ_{n+1} are the values of τ in the generation t and $t+1$ and Δ is the angle's variation that must be produced on the tangent of the curve associated to the generalized function at the point where the radii is r (see Fig. 3). Further details about the τ mutation and the mathematical derivation of the formula can be found in the Appendix.

The angle's variation Δ is updated in each generation since it depends of the learning rate α and it is defined as:

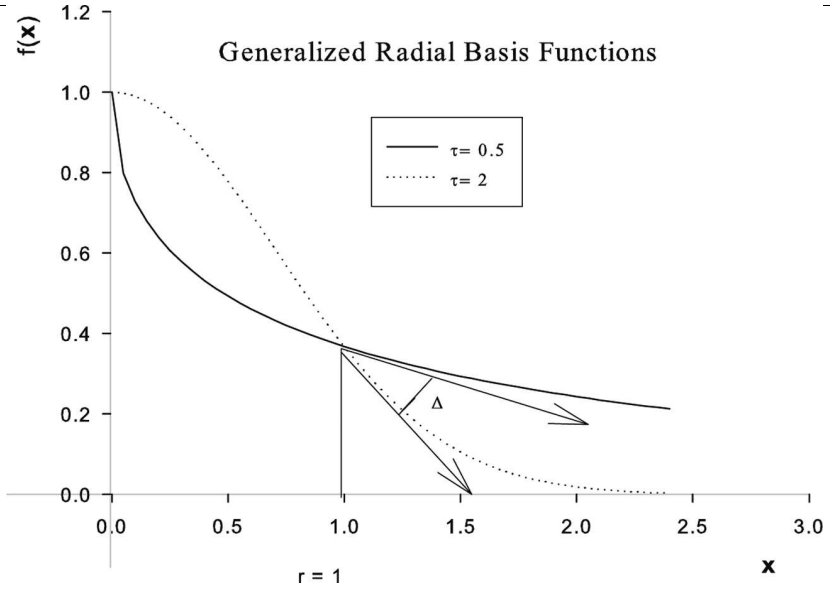

Fig. 3. Parametric Mutation for the GRBFs models based on the Δ value.

$$\Delta = a \cdot \frac{\pi}{30} \cdot \left(1 - \frac{1}{1 - e^{\alpha - 10.5}}\right) \quad (9)$$

where a is an integer value that can take values of 1 or -1 with probability 0.5.

Structural mutation implies a modification in the neural network structure and allows explorations in different regions of the search space while promoting the diversity of the population. There are four structural mutations: connection deletion, connection addition, node deletion and node addition. These mutations are applied sequentially to each network. The reader can consult [31] for more details about structural mutations.

The stop criterion is reached if one of the following conditions is fulfilled: a maximum number of generations is reached or the variance of the fitness for the best ten percent of the population is less than 10^{-4}.

5. Experiments

In the first subsection, the the datasets and the experimental configuration are described. In the subsequent subsections, the main experiments are presented.

5.1. Description of the datasets and the experimental design

The proposed methodologies are applied to twelve datasets taken from the UCI repository [34], to test their overall performance when compared among themselves. To analyse the performance of the GRBF method, its results are compared to those obtained with the same evolutionary algorithm and parameters but using other basis functions commonly used in NNs for classification: RBFs with Gaussian transfer functions, Sigmoidal Units (SUs) and Product Units (PUs).

The selected datasets include seven binary problems and five multi-class problems. They present different numbers of instances, features and classes (see Table 1). The minimum and maximum number of hidden nodes were obtained from the best result of a preliminary experimental design ANOVA I, considering a small, medium and high value [35] for the parameters $m, M_{\mathrm{I}}, M_{\mathrm{E}}$. On each dataset, we performed 10 different simulations (out of the 27 possible configuration that satisfy the constraint $m < M_{\mathrm{I}} < M_{\mathrm{E}}$). The Tukey post-hoc test was applied to analyse the results of the simulations [36]. If there were no statistically significant differences among the 30 C_T values (accuracy in the training set) for each configuration, we selected the configuration corresponding to the minimum values of m, M_{I} and M_{E}. Otherwise, we selected the configuration that achieved the best results. After that, we ran the EA using the configuration previously selected for each dataset. This value is also included in Table 1.

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 generalisation dataset. To evaluate the stability of the methods, the evolutionary algorithm was run 30

Table 1
Characteristics of the thirteen 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.), per-class distribution of the instances (Distribution), minimum and maximum number of hidden nodes used for each dataset ($[m, M_{\mathrm{I}}, M_{\mathrm{E}}]$) and number of generations (# Gen.)

Dataset	Size	R	B	N	# In.	# Out.	*Distribution*	$[m, M_{\mathrm{I}}, M_{\mathrm{E}}]$	# Gen.
Labor	57	8	3	5	29	2	$(30, 27)$	$[1, 2, 3]$	25
Hepatitis	155	6	13	–	19	2	$(32, 123)$	$[1, 2, 3]$	25
Sonar	208	60	–	–	60	2	$(98, 110)$	$[1, 2, 3]$	300
BreastC	286	4	3	2	15	2	$(201, 85)$	$[1, 2, 3]$	50
Ionos	351	33	1	–	34	2	$(126, 225)$	$[3, 4, 5]$	300
Card	690	6	4	5	51	2	$(307, 308)$	$[1, 2, 3]$	50
German	1000	6	3	11	61	2	$(700, 300)$	$[2, 3, 4]$	300
Newthyroid	215	5	–	–	5	3	$(150, 35, 30)$	$[1, 1, 4]$	100
Post-Op	90	1	–	7	20	3	$(2, 24, 64)$	$[1, 2, 3]$	100
Glass	214	9	–	–	9	6	$(70, 76, 17, 13, 9, 29)$	$[7, 8, 9]$	500
Zoo	101	1	15	–	16	7	$(41, 20, 5, 13, 4, 8, 10)$	$[4, 7, 9]$	300
Ecoli	336	7	–	–	7	8	$(143, 77, 52, 35, 20, 5, 2, 2)$	$[4, 7, 9]$	300

All nominal variables are transformed to binary variables.
BreastC: Breast-Cancer.

times. The performance of each method was evaluated using the correct classification rate (C).

All parameters of the EA are common for all problems, except m, M_I, M_E and the number of generations values, which are represented in Table 1. We have carried out a simple linear rescaling of the input variables to the interval $[-2, 2]$, X_i^* representing the transformed variables. The connections between the hidden and output layers are initialised in the $[-5, 5]$ interval. The initial value of the radii r_j are obtained in the interval $(0, d_{max}]$, where d_{max} is the maximum distance between two training input examples.

The size of the population is $N = 200$. For the structural mutation, the number of nodes that can be added or removed is within the $[1, 2]$ interval, and the number of connections to add or delete in the hidden and the output layer during structural mutations is within the $[1, 7]$ interval.

5.2. Results of the GRBF NN

Table 2 shows the mean and the standard deviation of the correct classification rate in the generalisation set (C_{G}) for each dataset and the GRBF, RBF, SU and PU models. All of the algorithms were run 30 times. Based on the mean C_{G} of the 30 executions, we obtained the ranking for each model on each dataset ($R = 1$ for the best performing model and $R = 4$ for the worst one). Table 2 also includes the mean accuracy ($\overline{C}_{\mathrm{G}}$) and the mean ranking ($\overline{R}_{C_{\mathrm{G}}}$) for all datasets. From these results, we conclude from a purely descriptive point of view, that the GRBF models obtained the best C_{G} results for nine datasets and the RBF models yield the highest performance for three datasets. Furthermore, the GRBF models yield the best mean ($\overline{C}_{\mathrm{G}} = 83.26\%$) and ranking ($\overline{R}_{C_{\mathrm{G}}} = 1.33$) in C_{G}. As can be observed, the GRBF models are especially accurate when dealing with high dimensionality problems (e.g., see the German, Card, Ionos or Labor datasets), because the τ parameter contracts or relaxes the shape of the Gaussian, adapting to the statistical distribution of the distance.

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 [37] using the C_{G} rank of of the best models as the test variable. A previous evaluation of the C_{G} values resulted in rejecting the normality and the equality of variances' hypothesis. The Friedman test shows that the effect of the method used for classification is statistically significant with a significance level of 5%, as the confidence interval is $C_0 = (0, F_{0.05} = 2.89)$ and the F-distribution statistical values are $F^* = 6.50 \notin C_0$. 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 classifiers 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):

$$CD = q\sqrt{\frac{K(K+1)}{6D}} \tag{10}$$

where K and D are the number of classifiers and datasets, and the q value is derived from the studentized range statistic divided by $\sqrt{2}$ [38,39]. However,

Table 2
Statistical results of the evolutionary algorithm using different basis functions: Mean and Standard Deviation (SD) of the accuracy in the generalisation set ($C_{\mathrm{G}}(\%)$), mean accuracy ($\overline{C}_{\mathrm{G}}(\%)$) and mean ranking ($\overline{R}$)

	Method ($C_{\mathrm{G}}(\%)$)			
	GRBF	RBF	PU	SU
Labor	**91.19 ± 8.32**	**90.71** ± *6.26*	83.33 ± 10.15	85.71 ± 9.56
Hepatitis	*85.96 ± 3.21*	**86.84** ± *3.16*	85.08 ± 5.04	85.52 ± 5.01
Sonar	*74.10 ± 3.60*	**73.68 ± 3.53**	75.25 ± 4.87	68.71 ± 4.08
BreastC	**68.87 ± 2.10**	68.45 ± 1.87	67.65 ± 2.49	*68.68 ± 2.14*
Ionos	**93.75 ± 1.76**	90.42 ± 2.60	91.15 ± 2.20	*92.61 ± 2.75*
Card	**87.94 ± 1.38**	76.69 ± 3.33	87.50 ± 2.75	*87.71 ± 1.42*
German	**74.33** ± 2.77	71.69 ± 1.32	71.24 ± 1.24	*73.07 ± 1.64*
Newthyroid	**97.10** ± 1.93	95.00 ± 2.01	*96.85 ± 2.71*	94.88 ± 2.26
Post-Op	**79.09** ± 8.06	*78.93 ± 7.21*	78.93 ± 8.39	76.51 ± 7.17
Glass	**68.93 ± 5.19**	64.91 ± 4.74	65.16 ± 4.17	*67.67 ± 3.49*
Zoo	**94.93 ± 2.77**	75.07 ± 5.00	*94.80 ± 4.48*	92.67 ± 4.34
Ecoli	82.98 ± 3.82	**84.44 ± 2.92**	80.30 ± 5.20	*83.73 ± 2.45*
$\overline{C}_{\mathrm{G}}(\%)$	**83.26**	79.73	81.43	*81.45*
$\overline{R}$	**1.33**	*2.87*	3.04	2.75

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

Table 3
Statistical analysis of the evolutionary algorithm using different basis functions: Critical Difference (CD) values and differences of rankings of the Bonferroni-Dunn and Nemenyi tests, using GRBF as the control method

Nemenyi test (C_{G})				
	Method (j)			
Method(i)	RBF	PU	SU	GRBF
RBF	–	0.16	0.12	$1.54^{+}_{\bullet}$
PU	–	–	0.25	$1.70^{+}_{\bullet}$
SU	–	–	–	$1.41^{+}_{\bullet}$
Bonferroni-Dunn test (C_{G})				
	Compared Method			
Control Method	RBF	PU	SU	GRBF
GRBF	$1.54^{+}_{\bullet}$	$1.70^{+}_{\bullet}$	$1.41^{+}_{\bullet}$	–

Nemenyi Test: $CD_{(\alpha=0.1)} = 1.20$, $CD_{(\alpha=0.05)} = 1.35$;
Bonferroni-Dunn Test: $CD_{(\alpha=0.1)} = 1.12$, $CD_{(\alpha=0.05)} = 1.26$;
$\bullet$, $\circ$: Statistically difference with $\alpha = 0.05$ ($\bullet$) and $\alpha = 0.1$ ($\circ$);
+: The difference is in favour of the Control Method or Method(j).

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 (a control method). One approach to this latter type of comparison is the Bonferroni-Dunn test. This test can be computed using Eq. (10) with appropriate adjusted values of q [39].

The results of the Bonferroni-Dunn and Nemenyi tests for $\alpha = 0.10$ and $\alpha = 0.05$ can be seen in Table 3, using the corresponding critical values. From the results of these tests, it can be concluded that the GRBF method obtains a significantly higher C_{G} ranking when compared to all of the other basis functions. This result justifies the proposal.

6. Conclusions

The proposed models, formed using Generalised Radial Basis Functions (GRBF) as transfer functions, are a viable alternative to existing methods and obtained more accurate classifications than other methods we tested. These models were designed with an evolutionary algorithm constructed specifically to account for the characteristics of this kernel model. The evaluation of both the model and the algorithm for the twelve datasets we considered showed results that are better than those using other basis function neural network models. The post-hoc Bonferroni-Dunn test was applied using C_{G} as the test variable and the results show that the GRBF method obtained a significantly higher C_{G} ranking when compared with the rest of the basis functions.

Appendix: Derivation of the formula used for the mutation of the τ parameter

Firstly, the Generalized Radial Basis Function (RBF) is presented:

$$f(x) = e^{-(\frac{x}{r})^{\tau}} \tag{11}$$

Then we evaluate its derivative for $x = r$. This value is equal to the the line's slope which is tangent to the GRBF in this point:

$$f'(x) = -\frac{e^{-(\frac{x}{r})^{\tau}} \tau (\frac{x}{r})^{\tau-1}}{r}$$

$$\tan(\beta) = -\frac{\tau}{e * r} \tag{12}$$

Using the Tangent's Sum Theorem:

$$\tan(\beta + \Delta) = -\frac{\tau_{n+1}}{e * r} \quad (13)$$

$$\tan(\beta + \Delta) = \frac{\tan(\beta) + \tan(\Delta)}{1 - \tan(\beta) * \tan(\Delta)} \quad (14)$$

Finally a system compounded by Eqs (13) and (14) is solved and the τ_{n+1} value is obtained

$$\tau_{n+1} = \frac{e \cdot r(\tau_n - e \cdot r \cdot tan(\Delta))}{e \cdot r + \tau_n tan(\Delta)} \quad (15)$$

Acknowledgement

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 Andalucia" Predoctoral Program, grant reference P08-TIC-3745.

References

[1] R. Lippmann, Pattern classification using neural networks, *IEEE Communications Magazine* **27** (1989), 47–64.

[2] D. Richard and E.R. David, Product units: a computationally powerful and biologically plausible extension to backpropagation networks, *Neural Comput* **1**(1) (1989), 133–142, 78412.

[3] C. Hervás-Martínez, F.J. Martínez-Estudillo and P.A. Gutiérrez, Classification by means of evolutionary product-unit neural networks, in: Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN'06), Vancouver, Canada, 2006, pp. 2834–2842.

[4] C.M. Bishop, Neural networks for pattern recognition, Oxford University Press, Oxford, UK, 1996.

[5] C.G. Looney, Radial basis functional link nets and fuzzy reasoning, *Neurocomputing* **48** (1–4).

[6] E.J. Hartman, J.D. Keeler and J.M. Kowalski, Layered neural networks with Gaussian hidden units as universal approximations, *Neural Computation* **2**(2) (1990), 210–215.

[7] T. Poggio and F. Girosi, Networks for approximating and learning, Vol. 78, Proc. IEEE, 1990, pp. 1481–1497.

[8] K.B. Cho and B.H. Wang, Radial basis function based adaptive fuzzy systems and their applications to system adaptive fuzzy systems and their applications to system, *Fuzzy Sets and Systems* **83** (1996), 325–339.

[9] A. Gegov, Complexity management in fuzzy systems: a rule base compression approach, *International Journal of Hybrid Intelligent Systems* **5**(1) (2008), 55.

[10] A. Staiano, R. Tagliaferri and W. Pedrycz, Improving RBF networks performance in regression tasks by means of a supervised fuzzy clustering, *Neurocomputing* **69**(13–15) (2006), 1570–1581.

[11] S. Chen, C.F.N. Cowan and P.M. Grant, Orthogonal least squares learning algorithm for radial basis function networks, *IEEE Trans. on Neural Networks* **2** (1991), 302–309.

[12] L. Xu, RBF nets, mixture experts, and bayesian ying-yang learning, *Neurocomputing* **19**(1–3) (1998), 223–257.

[13] G. Bugmann, Normalized gaussian radial basis function networks, *Neurocomputing* **20**(1–3) (1998), 97–110.

[14] M. Verleysen, D. François, G. Simon and V. Wertz, On the effects of dimensionality on data analysis with neural networks, in: *Artificial Neural Nets Problem solving methods*, J. A. e. J. Mira, ed., Lecture Notes in Computer Science 2687, Springer-Verlag, 2003, pp. II105–II112.

[15] M. Pérez-Godoy, P. Pérez, A. Rivera, M. del Jesus, C. Carmona, M. Frías and M. Parras, Co2rbfn for short-term forecasting of the extra virgin olive oil price in the spanish market, *International Journal of Hybrid Intelligent Systems* **7**(1) (2010), 75–87.

[16] M.J. Embrechts, B. Szymanski and M. Sternickel, Introduction to scientific data mining: Direct kernel methods and applications, in: *Computationally Intelligent Hybrid Systems*, S.J. Ovaska, ed., Wiley Interscience, New York, 2004, Ch. 10, pp. 317–362.

[17] S. Haykin, Neural Networks: A comprehensive Foundation, Prentice Hall, 3rd edition, 2008.

[18] R. Ñanculef, C. Concha, H. Allende, D. Candel and C. Moraga, Ad-svms: A light extension of svms for multicategory classification, *International Journal of Hybrid Intelligent Systems* **6**(2) (2009), 69–79.

[19] C.S. Teh and C.P. Lim, A hybrid som-kmer model for data visualization and classification, *International Journal of Hybrid Intelligent Systems* **2**(3) (2005), 189–203.

[20] C. M. Bishop, Improving the generalization properties of radial basis function neural networks, Neural Computation 3 (4) (1991) 579–581.

[21] K. Hunt, D. Sbarbaro, R. Zbikowski and P. Gawthrop, Neural networks for control system – a survey, *Automatica* **28** (1992), 1083–1112.

[22] C. Darken and J. Moody, Fast adaptive k-means clustering: some empirical results, *Proceedings IEEE INNS International Joint Conference On Neural Networks* **2** (1990), 233–238.

[23] G. Huang, P. Saratchandran and N. Sundararajan, A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation, *IEEE Trans. Neural Networks* **16**(1) (2005), 57–67.

[24] F.H.F. Leung, H.K. Lam, S.H. Ling and P.K.S. Tam, Tuning of the structure and parameters of a neural network using an improved genetic algorithm, *IEEE Transactions on Neural Networks* **14**(1) (2003), 79–88.

[25] L.N. de Castro, E.R. Hruschka and R.J.G.B. Campello, An evolutionary clustering technique with local search to design RBF neural network classifiers, in: *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks (IJCNN'04)*, Vol. 3, 2004, pp. 2083–2088.

[26] Z.Q. Zhao and D.S. Huang, A mended hybrid learning algorithm for radial basis function neural networks to improve generalization capability, *Applied Mathematical Modelling* **31** (2007), 1271–1281.

[27] X. Yao and Y. Liu, A new evolutionary system for evolving artificial neural networks, *IEEE Transactions on Neural Networks* **8**(3) (1997), 694–713.

[28] X. Yao and Y. Liu, Evolving neural network ensembles by minimization of mutual information, *International Journal of Hybrid Intelligent Systems* **1**(1–2) (2004), 12–21.

[29] N.T. Siebel and G. Sommer, Evolutionary reinforcement learning of artificial neural networks, *International Journal of Hybrid Intelligent Systems* **4**(3) (2007), 171–183.

[30] 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.

[31] P.A. Gutiérrez, C. Hervás-Martínez, M. Carbonero and J.C. Fernández, Combined projection and kernel basis functions for classification in evolutionary neural networks, *Neurocomputing* **72**(13–15) (2009), 2731–2742.

[32] D. Francois, High dimentional Data Analisis, From Optimal Metric to Feature Selection, VDM Verlag, Saarbrucken, Germany, 2008, Ch. Seeking on right metric, pp. 54–55.

[33] I. Rechenberg, Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Frommann-Holzboog, Stuttgart, 1973.

[34] A. Asuncion and D. Newman, http://www.ics.uci.edu/∼mlearn/MLRepository.html UCI machine learning repository (2007). http://www.ics.uci.edu/ mlearn/MLRepository.html.

[35] P.A. Castillo-Valdivieso, J.J. Merelo, A. Prieto, I. Rojas and G. Romero, Statistical analysis of the parameters of a neuro-genetic algorithm, *IEEE Transactions on Neural Networks* **13**(6) (2002), 1374–1394.

[36] R.G. Miller, Simultaneous Statistical Inference, 2nd Edition, Wiley, New York, USA, 1981.

[37] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, *Annals of Mathematical Statistics* **11**(1) (1940), 86–92.

[38] O.J. Dunn, Multiple comparisons among means, *Journal of the American Statistical Association* **56** (1961), 52–56.

[39] Y. Hochberg and A. Tamhane, Multiple Comparison Procedures, *John Wiley and Sons*, 1987.