Neurocomputing 72 (2009) 2731–2742

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Combined projection and kernel basis functions for classification in evolutionary neural networks

P.A. Gutiérrez [a,*], C. Hervás [a], M. Carbonero [b], J.C. Fernández [a]

[a] Department of Computer Science and Numerical Analysis, University of Córdoba, Campus de Rabanales, 14071 Córdoba, Spain
[b] Department of Management and Quantitative Methods, ETEA, Escritor Castilla Aguayo 4, 14005 Córdoba, Spain

ARTICLE INFO

Available online 16 April 2009

Keywords:
Classification
Projection basis functions
Kernel basis functions
Evolutionary neural networks

ABSTRACT

This paper proposes a hybrid neural network model using a possible combination of different transfer projection functions (sigmoidal unit, SU, product unit, PU) and kernel functions (radial basis function, RBF) in the hidden layer of a feed-forward neural network. An evolutionary algorithm is adapted to this model and applied for learning the architecture, weights and node typology. Three different combined basis function models are proposed with all the different pairs that can be obtained with SU, PU and RBF nodes: product–sigmoidal unit (PSU) neural networks, product–radial basis function (PRBF) neural networks, and sigmoidal–radial basis function (SRBF) neural networks; and these are compared to the corresponding pure models: product unit neural network (PUNN), multilayer perceptron (MLP) and the RBF neural network. The proposals are tested using ten benchmark classification problems from well known machine learning problems. Combined functions using projection and kernel functions are found to be better than pure basis functions for the task of classification in several datasets.

1. Introduction

Some features of neural networks (NNs) make them particularly attractive and promising for applications in the modelling and control of nonlinear systems: (1) universal approximation abilities, (2) parallel distributed processing abilities, (3) learning and adaptation, (4) natural fault tolerance and (5) feasibility for hardware implementation [47]. NNs have been an important tool for classification since recent research activities identified them as a promising alternative to conventional classification methods such as linear discriminant analysis or decision trees [31].

Different types of NNs are now being used for classification purposes [31], including, among others: multilayer perceptron (MLP) NNs, where the transfer functions are sigmoidal unit (SU) basis functions; radial basis function (RBF) NNs with kernel functions, where the transfer functions are usually Gaussian [7]; general regression neural networks proposed by Specht [46]; and a class of multiplicative NNs, namely product unit neural networks (PUNNs) [16]. The network is chosen depending on the conditions involved and the prior knowledge about the system under study. The most widely used NN models are MLPs and RBF networks. One distinctive characteristic of all these models is the combination of transfer and activation functions used in the hidden layer or the hidden layers of the NN. This pair of functions (transfer and activation function) will be jointly referred as the basis function further on this paper.

The combination of different basis functions in the hidden layer of a NN has been proposed as an alternative to traditional NNs. For example, Donoho demonstrated that any continuous function can be decomposed into two mutually exclusive type of functions [13], one associated with projection functions (PU or SU) and the other associated with kernel functions (RBF).

This paper aims to obtain empirically a response to the following question: Is the combination of several projection-based and kernel-based functions appropriate for optimizing the accuracy of a NN classifier? In order to answer this question, this paper compares the performance of pure basis function models to the performance of combined basis function models. Three kinds of hidden nodes are considered (SU, PU and RBF) and the three different associated pairs of combined basis function models are proposed: product–sigmoidal unit (PSU) NNs, product–radial basis function (PRBF) NNs, and sigmoidal–radial basis function (SRBF) NNs; these are compared to the corresponding pure models (PUNNs, MLPs and RBF NNs). A hybrid neural network model is proposed, maintaining the linear structure between the hidden and the output layer. Therefore, although the model is complex (it is composed of two different types of basis functions), it keeps a linear structure. In this way, this idea is similar to that proposed by Friedman and Stuetzle [20].

* Corresponding author. Tel.: +34 957 218 349; fax: +34 957 218 630.
E-mail addresses: pagutierrez@uco.es (P.A. Gutiérrez), chervas@uco.es (C. Hervás), mariano@etea.com (M. Carbonero), jcfernandez@uco.es (J.C. Fernández).

0925-2312/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.09.020

The problem of finding a suitable architecture and the corresponding weights for a neural network is a very complex task (for an interesting review of this subject, the reader can consult the review of Yao [53]). This complexity increases significantly when considering different basis functions, which justifies the use of an evolutionary algorithm (EA) to design the structure and training of the weights. This EA is presented in this paper with the aim of optimizing simultaneously the weights and the structure of the two types of basis functions considered in the hidden layer.

Consequently, the objectives of the study are the following:

- To develop an EA capable of overcoming the difficulties associated with the training of NNs containing combinations of the different basis functions considered in a single hidden layer.
- To experimentally check the duality of kernel and projection basis functions introduced by Donoho [13] and determine the most adequate projection function (PU or SU) for this hybridization.
- To check if the hybridization of two projection basis functions (PSU) is suitable for building a classifier.
- To compare the performance and the network size of the combined basis function models to their corresponding pure networks.

The rest of the paper is structured as follows. Section 2 analyzes previous works which are related to the training of projection based NNs, kernel based NNs or to the hybridization of both neural network basis functions. Section 3 formally presents the combined basis function model for classification considered in this work, and analyzes some important properties that have to be considered when combining the different basis functions proposed. The main characteristics of the algorithm used for training the hybrid models are described in Section 4. Section 5 presents the experiments carried out and discusses the results obtained. Finally, Section 6 completes the paper with the main conclusions and future directions suggested by this study.

2. Related works

In this section, different works related to the optimization and the designing of neural network models are presented. First, the main projection basis function neural network methods are described; next, the kernel basis function neural networks and the associated methods are introduced; and finally, the analysis of some works that combine both functional models is performed.

2.1. Neural networks based on projection basis functions

The MLP with the developed efficient back-propagation training algorithm [42] is probably the most frequently used type of neural network model in practical applications. However, due to its multilayer structure and the limitations of the back-propagation algorithm, the training process often settles in undesirable local minima of the error surface or converges too slowly. Adaptation or learning is a major focus of MLP research that provides the NN model with a degree of robustness. Additionally, building a MLP model is complex due to the presence of many training factors. Training factors traditionally involved in building a MLP model may include: the number of hidden nodes, training tolerance, initial weight distribution and the function gradient.

Product unit neural networks (PUNNs) introduced by Durbin and Rumelhart [16] are another promising alternative [25,34]. PUNN models are a class of high-order neural networks which only have unidirectional interlayer connections. In contrast to sigmoidal neural networks, PUNNs are based on multiplicative nodes instead of additive ones. PUs in the hidden layer of PUNNs provide a powerful mechanism for these models to efficiently learn higher-order combinations of inputs. Durbin and Rumelhart determined empirically that the information capacity of PUs (measured by their capacity for learning random Boolean patterns) is approximately $3N$, compared to the $2N$ of a network with additive units for a single threshold logic function, where N denotes the number of inputs to the network [16]. Besides that, it is possible to obtain the upper bounds of the Vapnik Chervonenkis (VC) dimension [50] in PUNNs, these bounds being similar to those obtained in sigmoidal neural networks [44]. It is a straightforward consequence of the Stone–Weierstrass theorem to prove that PUNNs are universal approximators [33] (observe that polynomial functions in several variables are a subset of PU models).

Despite these advantages, product-unit based networks have a major drawback. Networks based on PUs have more local minima and a higher probability of becoming trapped in them [33]. The main reason for this difficulty is that small changes in the exponents can cause large changes in the total error surface. Because of this reason, their training is more difficult than the training of standard MLPs. For example, it is well known that back-propagation is not efficient for the training of PUNNs [41]. Several efforts have been made to carry out learning methods for PUs: Janson and Frenzel [25] developed a genetic algorithm for evolving the weights of a PUNN with a predefined architecture. The major problem with this kind of algorithm is how to obtain the optimal architecture beforehand. Leerink et al. [27] tested different local and global optimization methods for PUNNs. Ismail and Engelbrecht [22] applied four different optimization methods to train PUNNs: random search, particle swarm optimization, genetic algorithms, and LeapFrog optimization. They concluded that random search is not efficient to train this type of network, and that the other three methods show an acceptable performance in three problems of function approximation with low dimensionality.

2.2. Neural networks based on kernel basis functions

RBF NNs have been used in the most varied domains, from function approximation to pattern classification, time series prediction, data mining, signals processing, and nonlinear system modelling and control. They have some useful properties which render them suitable for modelling and control. First, such networks are universal approximators [40]. In addition, they belong to a class of linearly parameterized networks where the network output is connected to tuneable weights in a linear manner. The RBF uses, in general, hyper-ellipsoids to split the pattern space. This is different from MLP networks which build their classifications on hyper-planes, defined by a weighted sum. Due to their functional approximation capabilities, RBF networks have been seen as a good solution for interpolation problems. RBF NNs can be considered a local average procedure, and the improvement in both its approximation ability as well as in the construction of its architecture has been note-worthy [6].

Their simple architecture allows the application of fast and efficient learning algorithms [38]. However, they are only applied for some problems in the theory of interpolation, because strict interpolation may sometimes not be a good strategy for the training of RBF networks due to possibly poor resulting generalization ability. In addition, if the size of the training set is

too large, and/or if there is a large number of redundant data (linearly dependent vectors) in this set, the likelihood of obtaining an ill-conditioned correlation matrix is higher and hinders the training procedure. Moreover, the constraint of having as many RBFs as data points makes the problem over-determined. To overcome these computational difficulties, the complexity of the network has to be reduced, requiring an approximation to a regularized solution [18,37]. This approach involves the search for a suboptimal solution in a lower dimensional space [8].

The number and positions of the basis functions, which correspond to the nodes in the hidden layer of the network, have an important influence on the performance of the RBF neural net. 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 [12]; or they can be evolved using evolutionary algorithms [8,29] or a hybrid algorithm [55].

In order to solve the problem of selecting a proper size for the hidden layer, a new category of algorithms has been introduced to automatically determine the structure of the network. These include the orthogonal least squares algorithm [9]; constructive methods, where the structure of the network is built incrementally [56]; pruning methods that start with an initial selection of a large number of hidden units which is reduced as the algorithm proceeds [36]; and the simultaneous selection of network structure and parameters by employing optimization methods based on genetic algorithms [5].

2.3. Neural networks based on hybrid basis functions

Hybrid models have also been proposed, where different activation/transfer functions are used for the nodes in the hidden layer. Several authors have proposed the hybridization of different basis functions, using either one single hybrid hidden layer or several connected pure layers.

According to Duch and Jankowski [15], mixed transfer functions within one network may be introduced in two ways. In the first way, a constructive method selects the most promising function from a pool of candidates in RBF-like architecture and add it to the network [14]. In the second approach, starting from a network that already contains several types of functions (such as Gaussian and sigmoidal functions), pruning or regularization techniques are used to reduce the number of functions [15].

Optimal transfer function networks were presented as a method for selecting appropriate functions for a given problem [24], creating architectures that are well matched for some given data and resulting in a very small number of hidden nodes.

In the functional link networks of Pao [39], a combination of various functions, such as polynomial, periodic, sigmoidal and Gaussian functions is used. The basic idea behind a functional link network is the use of functional links, adding nonlinear transformation of the input variables to the original set of variables, and suppressing the hidden layer, only performing a linear transformation of this derived input space. In this way, the first nonlinear mapping is fixed and the second linear mapping is adaptive.

A more complex approach considers several layers or models, each one containing a basis function structure, resulting in a modular system. For example, Iulian proposes a methodology including three distinct modules implementing a hybrid feed-forward neural network, namely a Gaussian type RBF network, a principal component analysis (PCA) process, and a MLP NN [23]. Another proposal of Lehtokangas and Saarinen considers two hidden layers in the model [28], the first one composed of Gaussian functions and the second one made up of SU basis functions.

Neural networks using different transfer functions should use fewer nodes, enabling the function performed by the network to be more transparent. For example, one hyperplane may be used to divide the input space into two classes and one additional Gaussian function to account for local anomaly. Analysis of the mapping performed by an MLP network trained on the same data will not be so simple.

In this context, it is worth emphasizing the paper by Cohen and Intrator [11], which is based on the duality and complementary properties of projection-based functions (SU and PU) and kernel typology (RBF). This hybridization of models has been justified theoretically by Donoho [13], who demonstrated that any continuous function can be decomposed into two mutually exclusive functions, such as radial (kernel functions) and crest ones (based on the projection). Although theoretically this decomposition is justified, in practice it is difficult to apply gradient methods to separate the different locations of a function (in order to adjust them by means of a combination of RBFs) and then to estimate the residual function by means of a functional approach based on projections, all without getting trapped in local optima in the procedure of error minimization [19].

Recently, Wedge and collaborators [52] have presented a hybrid RBF and sigmoidal neural network using a three step training algorithm for function approximation, aiming to achieve an identification of the aspects of a relationship that are universally expressed separately from those that vary only within particular regions of the input space.

3. Combined basis function model for classification

The combined basis function (CBF) model used in the classification process is proposed in this section and it is represented by means of a neural network structure. First of all, it is important to formally define the classification task. 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 dataset $D=\{(\mathbf{x}_n,\mathbf{y}_n); n=1,2,\ldots,N\}$ is available, where $\mathbf{x}_n=(x_{1n},\ldots,x_{kn})$ is the random vector of measurements taking values in $\Omega\subset R^k$, and $\mathbf{y}_n$ is the class level of the n-th individual. The representation of the class level is performed using the common technique of a "1-of-J" encoding vector, $\mathbf{y}=(y^{(1)}, y^{(1)}\ldots,y^{(J)})$, and the correctly classified rate (*CCR*) is used as an accuracy measure defined by $CCR=\frac{1}{N}\sum_{n=1}^{N} I(C(\mathbf{x}_n)=\mathbf{y}_n)$, where $I(.)$ is the zero–one loss function. A good classifier tries to achieve the highest possible *CCR* in a given problem.

In order to tackle this classification problem, the outputs of the CBF model have been interpreted from the point of view of probability through the use of the softmax activation function [7], which is given by

$$g_l(\mathbf{x},\boldsymbol{\theta}_l)=\frac{\exp f_l(\mathbf{x},\boldsymbol{\theta}_l)}{\sum_{j=1}^{J}\exp f_j(\mathbf{x},\boldsymbol{\theta}_j)},\quad l=1,2,\ldots,J \tag{1}$$

where J is the number of classes in the problem, $f_l(\mathbf{x},\boldsymbol{\theta}_l)$ is t he output of the l output node for pattern $\mathbf{x}$ and $g_l(\mathbf{x},\boldsymbol{\theta}_l)$ the probability a pattern $\mathbf{x}$ has of belonging to class l. The hybrid CBF model to estimate the function $f_l(\mathbf{x},\boldsymbol{\theta}_l)$ is given by a weighted combination of a pair of basis functions

$$f_l(\mathbf{x},\boldsymbol{\theta}_l)=\beta_0^l+\sum_{j=1}^{m_1}\beta_j^{l,1}B_j^1(\mathbf{x},\mathbf{w}_j^1)+\sum_{j=1}^{m_2}\beta_j^{l,2}B_j^2(\mathbf{x},\mathbf{w}_j^2),\quad l=1,2,\ldots,J \tag{2}$$

where $\boldsymbol{\theta}_l = (\boldsymbol{\beta}^l, \mathbf{w}_1^1, \ldots, \mathbf{w}_{m_1}^1, \mathbf{w}_1^2, \ldots, \mathbf{w}_{m_2}^2)$, $\boldsymbol{\beta}^l = (\beta_0^l, \beta_1^{l,1}, \ldots, \beta_{m_1}^{l,1}, \beta_1^{l,2}, \ldots, \beta_{m_2}^{l,2})$ and $\mathbf{w}_j^i$ is the weight vector of the connections between the input layer and the j-th hidden node of type i. In order to simplify the following expressions, $B_j(\mathbf{x}, \mathbf{w}_j)$ will be used in a general way to substitute both $B_j^1(\mathbf{x}, \mathbf{w}_j^1)$ or $B_j^2(\mathbf{x}, \mathbf{w}_j^2)$. Two types of projection basis functions are considered in (2) to replace B_j^1 or B_j^2:

- Sigmoidal units in the form:

$$B_j(\mathbf{x}, \mathbf{w}_j) = \frac{1}{1 + \exp(-w_{j0} - \sum_{i=1}^{k} w_{ji} x_i)}$$

  where $\mathbf{w}_j = (w_{j0}, w_{j1}, \ldots, w_{jk})$, w_{j0} being the bias.
- Product units have the following expression:

$$B_j(\mathbf{x}, \mathbf{w}_j) = \prod_{i=1}^{k} x_i^{w_{ji}}$$

  where $\mathbf{w}_j = (w_{j1}, \ldots, w_{jk})$, not considering bias in the inputs.
- Kernel basis functions replacing B_j^1 or B_j^2 in (2) are RBFs, in the form:

$$B_j(\mathbf{x}; (\mathbf{c}_j | r_j)) = \exp\left(-\frac{|\mathbf{x} - \mathbf{c}_j|^2}{2r_j^2} \right)$$

  where $\mathbf{w}_j = (w_{j0}, w_{j1}, \ldots, w_{jk})$, and $\mathbf{c}_j = (w_{j1}, \ldots, w_{jk})$ and $r_j = w_{j0}$ are, respectively, the centre and width of the Gaussian basis function of the j-th hidden unit.

Using the softmax activation function presented in Eq. (1), the class predicted by the neural net 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:

$$C(\mathbf{x}) = \hat{l} \quad \text{where } \hat{l} = \arg\max_l g_l(\mathbf{x}, \hat{\boldsymbol{\theta}}) \text{ for } l = 1, 2, \ldots, J \tag{3}$$

The function used to evaluate the CBF model is the function of cross-entropy error and is given by the following expression for J classes:

$$l(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{l=1}^{J} y_n^{(l)} \log g_l(\mathbf{x}_n, \boldsymbol{\theta}_l) \tag{4}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J)$. Moreover, because of the normalization condition

$$\sum_{l=1}^{J} g_l(\mathbf{x}, \boldsymbol{\theta}_l) = 1,$$

the probability for one of the classes does not need to be estimated. There is a redundancy in the functions $f_l(\mathbf{x}, \boldsymbol{\theta}_l)$, since adding an arbitrary $h(\mathbf{x})$ to each output leaves the decision (3) unchanged. Traditionally one of them is set to zero ($f_J(\mathbf{x}, \boldsymbol{\theta}_J) = 0$) which reduces the number of parameters to estimate. With all these considerations, Fig. 1 represents the general scheme of the CBF model, not considering bias in the input layer.

From a statistical point of view, with the softmax activation function defined in (1) and the cross-entropy error defined in (4), the neural network model can be seen as a multilogistic regression model. Nevertheless, the nonlinearity of the model with respect to parameters $\boldsymbol{\theta}$, and the indefinite character of the associated Hessian matrix do not recommend the use of gradient-based methods to minimize the negative log-likelihood function (for example, Iteratively Reweighted Least Squares, which is commonly used in the optimization of log-likelihood in linear multinomial logistic regression). Thus, an evolutionary algorithm

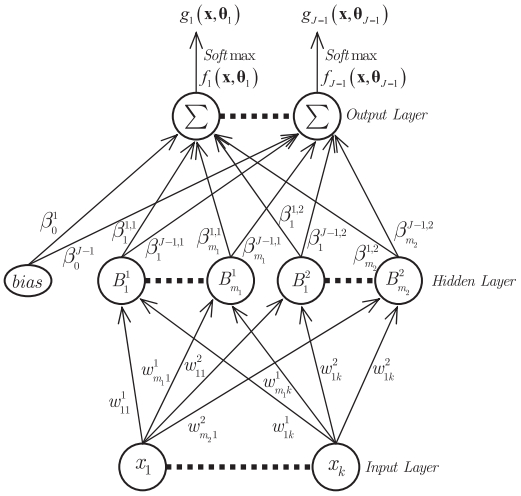

Fig. 1. Combined basis function model for classification.

is used to determine both the number of nodes in the hidden layer as well as the weights of the nets.

3.1. Basis function complexity and generalization capability

The first consideration when using NNs in a classification problem is the type of projection or kernel basis function to be selected. When the underlying cross-entropy function is unbounded, the PU basis functions are very adequate for classification, because, if the whole space R^k is considered to be the input domain, an MLP basis function model has the following upper and lower bounds:

$$|f_{MLP}(\mathbf{x})| = \left| \sum_{j=1}^{m} \beta_j \frac{1}{1 + e^{-\langle \mathbf{x}, \mathbf{w} \rangle}} \right| \leqslant \sum_{j=1}^{m} |\beta_j|$$

where $\langle , \rangle$ is the inner product.

In the same way, a Gaussian RBF has the following bounds:

$$|f_{RBF}(\mathbf{x})| = \left| \sum_{j=1}^{m} \beta_j e^{-\|\mathbf{x} - \mathbf{c}_j\| / r_j} \right| \leqslant \sum_{j=1}^{m} |\beta_j|$$

where $\| \ \|$ is the norm.

However a PU basis function model is unbounded, as if $f_{PU}(\mathbf{x}) = \sum_{j=1}^{m} \beta_j \prod_{i=1}^{p} x_i^{w_{ji}}$ then:

$$\lim_{\|x\| \to +\infty} f_{PU}(\mathbf{x}) = \lim_{\|x\| \to +\infty} \sum_{j=1}^{m} \beta_j \prod_{i=1}^{p} x_i^{w_{ji}} = \infty$$

considering certain values for the parameters of the model.

In general, the training data available for solving a classification problem belongs to a compact set, which is the set where the parameters of the model are adjusted. In this set, the MLP models can always be applied. However, the bounded functions do not show a good generalization capability for data which do not belong to the training compact set. Consequently, the behaviour of the basis functions at the training set domain borders is a characteristic which influences the decision about which basis functions are the most adequate.

The second consideration to take into account is the generalization capability of the combined basis function models. For this purpose, the combined models formed by PU projection basis functions and kernel RBFs (PRBF models) and those formed by SU basis functions and RBFs (SRBF models) are considered first. If a family of functions is an average of projection and kernel basis function, this structure enhances the capability to reduce

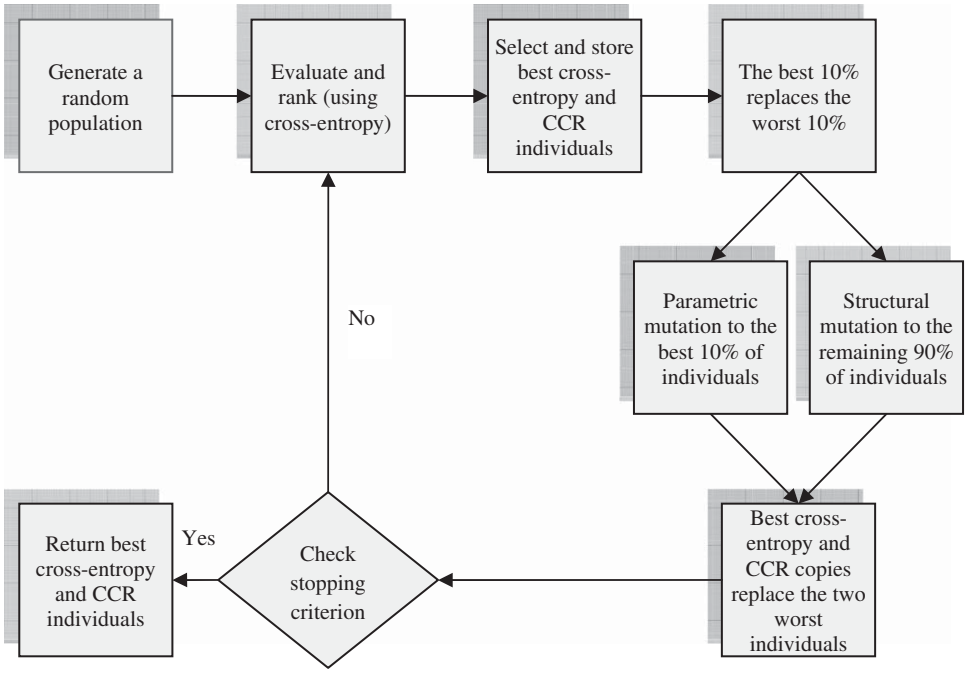

Fig. 2. Combined basis function evolutionary programming scheme.

empirical risk, that is, the error obtained with the generalization patterns. The reason for this reduction is that this kind of model can be consider to be an ensemble of the two models it is made up of, a simple average of the outputs obtained from two individual classifiers using two different structures, PU or SU, and RBF. Most of the existing ensemble learning algorithms [4,49] can be interpreted to be building diverse base classifiers implicitly. When projection basis functions and kernel basis functions are used, the ensemble results in a high degree of diversity because the individual PU or SU submodel contributes a global recognition structure and will disagree with the RBF submodel in the composed network, as this model contributes a local recognition structure. Using this kind of models, generalization capability is related to the VC dimension [50]. Particularly, when a model is of high complexity, the VC dimension is higher and its generalization capacity decreases. Schmitt [45] established a super-linear lower bound on the VC dimension for a RBF neural network with W parameters, and one hidden layer of k RBF nodes, of at least $(W/12)\log(k/8)$. On the other hand, MLP neural networks have a VC dimension that is also super-linear in the number of network parameters W, and when there is one hidden layer with k MLP nodes, the VC dimension is at least $(W/18)\log(k/4)$ [43]. Thus, the cooperative network effect enhancing the computational power of sigmoidal networks is also confirmed for RBF networks, and, consequently, for SRBF and PRBF hybrid models. The latter occurs since the upper bounds of the VC dimension in a PUNN are similar to those obtained for an MLP [34].

The capacity for generalization of this ensemble-like typology using basis functions with sufficiently diverse global-local discriminator characteristics (i.e. SRBF or PRBF models) is similar to or greater than the generalization capability of pure PU, SU or RBF models, which is proven empirically in this study.

The next point is the generalization capacity of the combined models formed by product and sigmoidal projection basis functions (PSU models). In general and considering the bias-variance trade-off, the higher the complexity obtained by the models, the lower their generalization capability. In this respect, it is interesting to point out that Schmitt [44] obtains the upper bounds of VC dimension using networks where SUs and PUs are combined, and that these upper bounds are similar to those obtained for pure PU or SU neural networks. This fact guarantees good generalization capability for the PSU model, this generalization being similar to the capability of MLPs or PUNNs. However both basis functions have a projection characteristic which prevents their global discrimination characteristics from contributing sufficient diversity in some datasets, the generalization capability of these models being similar to that obtained with their corresponding pure models. This will be analyzed in the experimental section.

4. The combined basis function evolutionary programming algorithm

In this section, the evolutionary algorithm used to estimate the parameters and the structure (including the number of hidden nodes of each basis function type) of the CBF models is presented.

The objective of the algorithm is to design a CBF neural network with optimal structure and weights for each classification problem tackled. The population is subject to the operations of replication and mutation, and crossover is not used due to its potential disadvantages in evolving artificial networks [2,54]. With these features the algorithm falls into the class of evolutionary programming (EP) [17]. It is an extension of the neural net evolutionary programming (NNEP) algorithm proposed in previous works [33,34], adding the necessary functionalities to consider CBF neural networks with different basis functions in the hidden layer and resulting in what we have called *combined basis function evolutionary programming* (CBFEP). NNEP is a software package developed in JAVA by the authors, as an extension of the JCLEC[1] framework [51] and it is available in the noncommercial JAVA tool named KEEL[2] [1].

The general framework of CBFEP is shown in Algorithm 1. The CBFEP algorithm starts by generating a random population of CBF

[1] Java Class Library for Evolutionary Computation (http://jclec.sourcepforge.net).

[2] Knowledge Extraction base on Evolutionary Learning (http://www.keel.es).

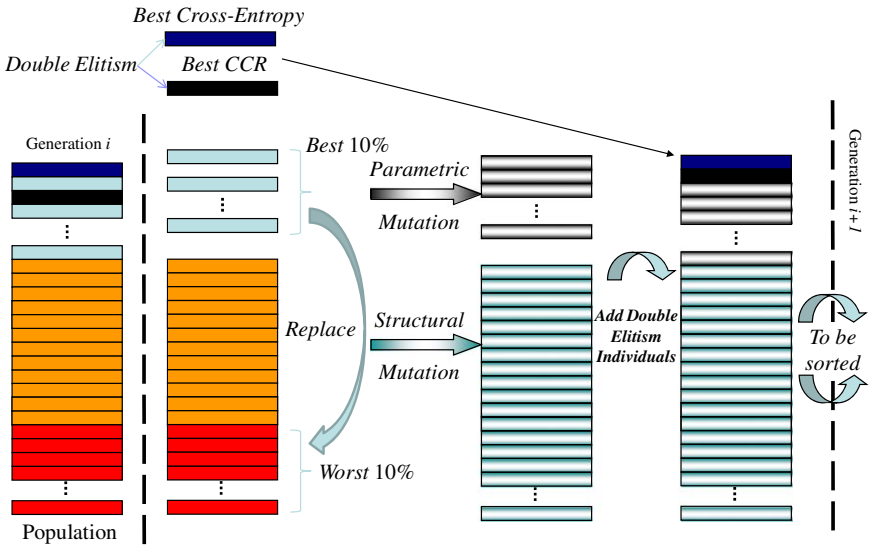

Fig. 3. Double elitism procedure scheme.

models and makes them evolve by applying different mutation and replication operations. A graphical representation of the algorithm has been included in Fig. 2, where the different stages of the algorithm are more clearly differentiated.

Algorithm 1. General framework of combined basis function evolutionary programming algorithm.

(1) Generate a random population of size N_P, where each individual presents a combined basis function structure.
(2) Repeat until the stopping criterion is fulfilled.
  (2.1) Calculate the fitness (decreasing transformation of cross-entropy error) of every individual in the population and rank the individuals with respect to their cross-entropy error.
  (2.2) Select and store best cross-entropy individual and best *CCR* individual (*double elitism*).
  (2.3) The best 10% of population individuals are replicated and substitute the worst 10% of individuals.
  (2.4) Apply mutations:
    (2.4.1) Parametric mutation to the best 10% of individuals.
    (2.4.2) Structural mutation to the remaining 90% of individuals, using a modified add node mutation in order to preserve the combined basis function structure.
  (2.5) Add best cross-entropy individual and best CCR individual from previous generation and substitute the two worst individuals.
(3) Select the best *CCR* individual and the best cross-entropy individual in the final population and consider both as possible solutions.

The fitness measure is a strictly decreasing transformation of the cross-entropy error $l(\boldsymbol{\theta})$ given by $A(g) = 1/1 + l(\boldsymbol{\theta})$, where g is a CBF model given by the multivaluated function $g(\mathbf{x}, \boldsymbol{\theta}) = (g_1(\mathbf{x}, \boldsymbol{\theta}_1), \ldots, g_J(\mathbf{x}, \boldsymbol{\theta}_J))$ and $l(\boldsymbol{\theta})$ is the cross-entropy error, defined in (4). The severity of mutations depends on the temperature $T(g)$ of the neural network model, defined by $T(g) = 1 - A(g)$, $0 \leqslant T(g) \leqslant 1$. For further details about the general characteristics of the NNEP algorithm, the reader can consult previous works [32–34].

One specific characteristic of this algorithm is the *double elitism*. As can be observed, the proposed algorithm returns both the best cross-entropy and the best *CCR* individuals as feasible solutions (step 3, Algorithm 1). In general, the relationship between *CCR* and cross-entropy error strongly depends on the dataset structure. Hence, regarding experimental results, using cross-entropy elitism is more suitable for some datasets in order to result in higher generalization accuracy, but using *CCR* elitism can be more appropriate for some other datasets. For this reason, the algorithm returns the best *CCR* and the best cross-entropy individuals as solutions, the best approach for each problem being difficult to ascertain a priori. In generation i, both the best cross-entropy individual and the best *CCR* individual are stored (step 2.2, Algorithm 1). Then, the selection and the mutations are applied (steps 2.3 and 2.4, Algorithm 1). Finally, the population of the next generation is formed merging the double elitism individuals and the mutated individuals (step 2.5, Algorithm 1) and they will be sorted in the next generation by using the cross-entropy error function (step 2.1, Algorithm 1). In Fig. 3, it can be observed graphically how the *double elitism* procedure works and how the different steps are applied.

Next, there is an explanation of the specific details of the CBFEP algorithm which concern to the optimization of the CBF structure. We will refer to the two types of hidden nodes considered in CBF topology as t_1 and t_2.

In order to define the topology of neural networks generated in the evolution process, three parameters are considered: m, M_E and M_I. They correspond to the minimum and the maximum number of hidden nodes in the whole evolutionary process and the maximum number of hidden nodes in the initialization process, respectively. In order to obtain an initial population formed by models simpler than the most complex model possible, parameters must fulfil the condition $m \leqslant M_I \leqslant M_E$.

An initial population of size $10N_P$ is generated, where $N_P = 1000$ is the size of the population during the evolutionary process. Then the best N_P neural networks are selected (step 1, Algorithm 1). For the generation of a network, the number of nodes in the hidden layer is taken from a uniform distribution in the interval $[m, M_I]$. Once the number of hidden nodes is decided, each hidden node is generated using a probability of 0.5 to decide if the node corresponds to t_1 or t_2. For PU and SU hidden nodes, the number of connections between each node in the hidden layer and the input nodes is determined from a uniform distribution in

Table 1
Main characteristics of each dataset tested and non-common parameter values.

Dataset	# Instances	# Inputs	Distribution	# Classes	# Gen.	m	M_I	M_E
Balance	625	4	(288,49,288)	3	500	3	4	5
Card	690	51	(307,383)	2	50	1	2	3
German	1000	61	(700,300)	2	300	2	3	4
Glass	214	9	(17,76,13,29,70,9)	6	500	7	8	9
Heart	270	13	(150,120)	2	100	1	1	2
Ionosphere	351	34	(126,225)	2	300	3	4	5
Newthyroid	215	5	(150,35,30)	3	100	1	1	4
Pima	768	8	(500,268)	2	160	1	2	3
Vote	435	16	(267,168)	2	10	1	1	2
Zoo	101	16	(41,20,5,13,4,8,10)	7	400	2	3	3

the interval $(0,k]$, where k is the number of independent variables. For RBF hidden nodes, the number of connections is always k, since these connections represent the coordinates of the centre of the node. The number of connections between each hidden node and the output layer is determined from a uniform distribution in the interval $(0,J-1]$.

Weights are initialized differently depending on the type of hidden node generated. For PU and SU hidden nodes, weights are assigned using a uniform distribution defined throughout two intervals: $[-5,5]$ for connections between the input layer and hidden layer and, for all kinds of nodes, $[-10,10]$ for connections between the hidden layer and the output layer. For RBF hidden nodes, the connections between the input layer and hidden layer represent the centre of the associated Gaussian distribution and these centres are initialized using a clustering algorithm, so the EA can start the evolutionary process with well positioned centres. The main idea is to cluster input data in k groups, k being the number of hidden RBF nodes. In this way, each hidden RBF node can be positioned in the centroid of its corresponding cluster. Finally, the radius of the RBF hidden node is calculated as the geometric mean of the distance to the two closest centroids. The clustering technique chosen (similar to that proposed by Cohen and Intrator [10]) is a modification of the classic k-means algorithm, in which the initial centroids are calculated using a specific initialization algorithm that avoids local minima, increasing the probability that the initial k cluster centres not be from a smaller number of clusters.

Parametric mutation (step 2.4.1, Algorithm 1) is accomplished for PU, SU and RBF hidden nodes by adding a Gaussian noise represented by a one-dimensional normally distributed random variable with mean 0 and adaptive variance, where variances are updated throughout the evolution of the algorithm. Radius r_i of RBF hidden nodes are mutated using a similar procedure.

On the other hand, structural mutation (step 2.4.2, Algorithm 1) implies a modification in the neural network structure and allows the exploration in different regions of the search space while helping to keep up the diversity of the population. There are different structural mutations, similar to the mutations in the GNARL model [2], including connection deletion, connection addition, node deletion, node addition and node fusion. There is no reason for connection deletion and connection addition if the node mutated is a RBF. Therefore, these mutations are not taken into consideration with this kind of nodes.

In node fusion, two randomly selected hidden nodes, a and b, are replaced by a new node c, which is a combination of the two. a and b must be of the same type in order to accomplish this mutation. The connections that are common to both nodes are kept and the connections that are not shared by the nodes are inherited by c with a probability of 0.5, their weight being unchanged. The weight resulting from the fusion of common connections depends again on the type of hidden node chosen. For PU and SU nodes, the resulting weights are given by

$$\beta_c^l = \beta_a^l + \beta_b^l, \quad w_{jc} = \left(\frac{w_{ja} + w_{jb}}{2}\right)$$

while for RBF nodes, the fact that RBFs can be interpreted as circumferences with a perfectly identified center and radius obliges the resulting weights to be considered as

$$\mathbf{c}_c = \frac{r_a}{r_a + r_b}\mathbf{c}_a + \frac{r_b}{r_a + r_b}\mathbf{c}_b, \quad r_c = \frac{r_a + r_b}{2}$$

$\mathbf{c}_j$ being the center defined by the hidden node j, which is $\mathbf{c}_j = (w_{j1}, w_{j2}, \ldots, w_{jk})$, and r_j being its radius.

In node addition, once the number of hidden nodes to be added has been decided, each hidden node is generated using a probability of 0.5 to assign its type (t_1 or t_2). If the number of hidden nodes in the neural net is M_E, no hidden nodes are added.

5. Experiments

In order to analyze the performance of the CBF model and the corresponding training algorithm CBFEP, 10 datasets in the UCI repository have been tested [3]. The experimental design was conducted using a holdout cross-validation procedure with $3n/4$ instances for the training set and $n/4$ instances for the generalization set, where the size of the dataset is n. All parameters of the CBFEP algorithm are common to these ten problems, except the m, M_I, M_E and the number of generations (#Gen.) values, which are represented in Table 1 together with the main characteristics of each dataset. An analysis of the results obtained for all pure (SU, PU and RBF) or combined (PSU, PRBF and SRBF) basis functions used in the neural network model is performed for every dataset.

Table 2 shows the mean value and standard deviation (mean $\pm$ SD) results of the correctly classified rate for the training (CCR_T) and generalization (CCR_G) sets and the mean and standard deviation of the number of net connections (#links) of the best models obtained in 30 runs of all the experiments performed. As the algorithm returns both the best CCR and cross-entropy individuals, two averages and standard deviations are obtained for each dataset. However, in order to reduce the extension of the table, the results included in Table 2 correspond only to the best performing variant in each CBF model, since the suitability of each elitism depends on the characteristics of the dataset and the basis function structure applied. It can be observed that the combination of basis functions produces good results with respect to CCR_G. In fact, from a purely descriptive point of view, the best result is obtained using CBF models in six out of the ten datasets analyzed, and the second best result is obtained in three out of the four remaining datasets. Moreover, in two of the 10 datasets analyzed, the first and second best results have been obtained using combined basis functions.

Table 2
Statistical results (mean ± SD) of the correctly classified rate for 10 datasets and for the training (CCR_T) and generalization (CCR_G) sets and number of connections (#links) for 30 executions of the CBFEP algorithm using pure or combined basis functions and CCR_T (C) or cross-entropy (E) elitism.

Dataset	Func.	Elit.	CCR_T Mean ± SD	CCR_G Mean ± SD	# Links Mean ± SD	Dataset	Func.	Elit.	CCR_T Mean ± SD	CCR_G Mean ± SD	# Links Mean ± SD
Balance	PU	C	99.30 ± 0.34	96.45 ± 1.28	23.1 ± 2.6	Card	PU	E	84.40 ± 2.02	87.50 ± 2.75	24.6 ± 14.7
	SU	C	96.89 ± 1.49	95.11 ± 1.58	31.9 ± 1.8		*SU*	E	86.82 ± 0.94	*87.71 ± 1.42*	52.1 ± 10.8
	RBF	C	90.67 ± 0.61	90.77 ± 1.31	29.7 ± 3.0		RBF	C	78.51 ± 1.90	76.69 ± 3.33	124.2 ± 24.1
	PSU	C	99.44 ± 0.61	**98.01 ± 0.96**	29.3 ± 4.5		PSU	E	86.71 ± 1.00	87.38 ± 1.17	43.3 ± 15.9
	PRBF	C	98.74 ± 0.69	*97.41 ± 1.11*	26.6 ± 3.9		PRBF	C	84.40 ± 2.17	86.43 ± 4.09	61.4 ± 32.5
	SRBF	C	93.83 ± 1.85	92.91 ± 1.61	32.6 ± 2.0		**SRBF**	E	86.16 ± 0.93	**88.02 ± 1.01**	62.2 ± 21.0
German	PU	C	74.13 ± 1.37	71.24 ± 1.52	47.9 ± 19.5	Glass	PU	C	75.90 ± 4.74	65.16 ± 4.17	62.4 ± 7.1
	SU	C	81.21 ± 1.39	73.07 ± 1.64	93.9 ± 21.8		**SU**	E	75.22 ± 2.52	**67.67 ± 3.49**	83.8 ± 6.0
	RBF	C	73.71 ± 0.88	71.69 ± 1.32	213.0 ± 30.4		RBF	C	66.29 ± 2.81	64.91 ± 4.74	108.1 ± 9.4
	PSU	E	79.40 ± 1.60	*73.12 ± 1.71*	86.0 ± 21.9		PSU	E	75.78 ± 2.13	66.23 ± 3.91	82.8 ± 6.5
	PRBF	E	72.60 ± 1.65	71.25 ± 1.45	119.6 ± 43.9		PRBF	C	74.76 ± 2.96	65.03 ± 3.96	89.3 ± 9.5
	SRBF	E	79.11 ± 2.05	**73.44 ± 1.61**	105.7 ± 34.0		*SRBF*	C	74.35 ± 2.53	*67.17 ± 4.36*	97.9 ± 7.5
Heart	PU	C	86.44 ± 1.57	83.58 ± 2.15	11.0 ± 2.6	Ionos	PU	E	96.79 ± 1.13	91.15 ± 2.20	39.1 ± 9.8
	SU	E	86.06 ± 0.90	**86.91 ± 2.06**	17.2 ± 2.0		*SU*	E	98.83 ± 0.75	*92.61 ± 1.56*	73.9 ± 10.2
	RBF	C	85.78 ± 0.79	81.37 ± 2.54	27.6 ± 4.2		RBF	C	91.39 ± 1.27	90.42 ± 2.60	158.5 ± 18.9
	PSU	C	88.27 ± 0.95	*85.93 ± 2.27*	16.9 ± 2.6		PSU	C	99.07 ± 0.55	92.11 ± 1.88	67.9 ± 12.6
	PRBF	E	84.14 ± 1.64	82.79 ± 2.57	20.4 ± 4.4		PRBF	E	94.29 ± 2.18	91.34 ± 2.41	93.9 ± 16.4
	SRBF	E	85.86 ± 1.63	85.49 ± 1.96	18.9 ± 4.3		**SRBF**	C	98.13 ± 0.88	**93.22 ± 1.61**	100.2 ± 16.6
Newth	*PU*	C	99.25 ± 0.55	*96.85 ± 2.71*	16.4 ± 3.2	Pima	PU	E	77.27 ± 0.65	78.45 ± 1.29	12.5 ± 2.1
	SU	C	98.72 ± 0.65	94.88 ± 2.26	22.1 ± 3.6		**SU**	E	76.88 ± 0.60	**79.98 ± 1.53**	18.6 ± 2.0
	RBF	C	95.67 ± 0.62	95.00 ± 2.01	24.2 ± 3.8		RBF	C	74.81 ± 1.54	75.66 ± 2.56	26.8 ± 3.1
	PSU	E	99.36 ± 0.60	96.36 ± 2.77	20.3 ± 3.7		PSU	E	76.95 ± 0.77	78.89 ± 1.87	17.0 ± 2.8
	PRBF	E	98.63 ± 0.97	**97.96 ± 2.45**	19.7 ± 3.8		PRBF	E	77.46 ± 0.51	78.54 ± 1.44	17.0 ± 1.5
	SRBF	C	94.70 ± 1.38	95.62 ± 2.20	23.5 ± 3.3		*SRBF*	E	77.40 ± 0.48	*79.64 ± 1.29*	22.6 ± 3.0
Vote	*PU*	E	95.14 ± 0.97	*95.52 ± 2.26*	5.9 ± 2.4	Zoo	**PU**	C	98.20 ± 1.74	**94.80 ± 4.48**	29.7 ± 2.8
	SU	E	95.88 ± 0.68	94.26 ± 1.91	14.0 ± 4.5		*SU*	C	99.34 ± 1.02	*92.67 ± 4.34*	49.9 ± 4.5
	RBF	C	91.55 ± 1.52	87.50 ± 2.77	30.4 ± 7.6		RBF	C	78.46 ± 2.73	75.07 ± 5.00	66.0 ± 1.5
	PSU	E	95.59 ± 0.63	94.57 ± 1.96	12.8 ± 4.3		PSU	C	98.46 ± 1.80	92.13 ± 5.09	42.2 ± 6.8
	PRBF	C	95.88 ± 0.45	**96.02 ± 0.88**	13.3 ± 8.2		PRBF	E	95.44 ± 3.60	91.33 ± 5.95	35.3 ± 6.0
	SRBF	C	96.26 ± 0.42	94.54 ± 2.23	16.0 ± 7.3		SRBF	E	97.02 ± 3.86	90.40 ± 4.77	48.6 ± 5.2

Elit.: Elitism; the represented results correspond only to the best performing final mean (E: best training cross-entropy individual; C: best CCR_T individual). The best result in the generalization set has been represented in bold face and the second best result in italic face.

To ascertain the statistical significance of the differences observed in each dataset performance, an analysis of variance (ANOVA) test [35] with the CCR_G of the best models as the test variable has been carried out (previously evaluating if the CCR_G values follow normal distribution, using a Kolmogorov–Smirnov test). Based on the hypothesis of normality, ANOVA examines the effects of some quantitative or qualitative variables (called factors) on one quantitative response. Here, the objective of this analysis is to determine if the influence of the basis function used in the hidden layer of the neural nets is significant in mean with respect to the CCR_G obtained by the CBFEP algorithm. Thus, the linear model for CCR_G has the form

$$CCR_{G_{ij}} = \mu + B_i + e_{ij}$$

for $i = 1,\ldots,6$ and $j = 1,2,\ldots,30$. Factor B_i analyzes the effect on the CCR_G in the i-th level of that factor, where B_i represents the typology of the basis functions used in the hidden layer of the net, with levels: ($i = 1$) for pure PUs; ($i = 2$) for pure SUs; ($i = 3$) for pure RBFs; ($i = 4$) for combined PSU; ($i = 5$) for combined PRBF; ($i = 6$) for combined SRBF. The term μ is the fixed effect that is common to all the populations. The term e_{ij} is the influence on the result of everything that could not be otherwise assigned, or of random factors.

Thus, 180 simulations were carried out for each dataset, corresponding to the 30 runs of each of the six levels for the factor. First, a Levene test (L) [30] is performed to evaluate the equality of variances (if $\alpha < p$-value, then the variances of CCR_G are equal) and then the Snedecor's F tests is performed for assessing if the influence of the basis function used in the hidden layer of the neural nets is significant in mean with respect to the CCR_G. The p-value of both tests are presented in Table 3. The F test shows that the effect of basis function typology used in the hidden layer is statistically significant in all datasets for the CCR_G and the # links at a significance level of 5%, i.e., $\alpha > p$-value in all datasets (see third column in Table 3).

Table 3
p-value of the Levene test (L) and the F test of ANOVA I applied to the CCR_G and # links values.

Dataset	p-Value			
	CCR_G		#links	
	L test	F test	L test	F test
Balance	0.048*	0.000*	0.008*	0.000*
Card	0.000*	0.000*	0.000*	0.000*
German	0.935	0.000*	0.000*	0.000*
Glass	0.948	0.033*	0.013*	0.000*
Ionos	0.033*	0.000*	0.001*	0.000*
Newthyroid	0.033*	0.000*	0.538	0.000*
Vote	0.000*	0.000*	0.000*	0.000*
Pima	0.000*	0.000*	0.013*	0.000*
Heart	0.591	0.000*	0.004*	0.000*
Zoo	0.425	0.000*	0.000*	0.000*

* Statistically significant differences with p-value < 0.05.

After carrying out these tests, a post-hoc multiple comparison test has been performed on the average CCR_G obtained. If the hypothesis that the variances are equal is accepted (using the Levene test presented in Table 3), a Tukey test [35] is carried out,

Table 4
p-value of the Tamhane and Tukey tests for CCR_G and ranking of the different basis function proposal based on this multiple comparison test.

$H_0 \equiv \hat{\mu}_{(I)} = \hat{\mu}_{(J)}$		p-Value									
(I)	(J)	Balance Tam.	Card Tam.	German Tukey	Glass Tukey	Heart Tukey	Ionos. Tam.	Newth. Tam.	Pima Tam.	Vote Tam.	Zoo Tukey
PU	SU	0.009*	1.000	0.000*	0.176	0.000*	0.110	+0*	0.002*	0.289	0.558
	RBF	0.000*	0.000*	0.866	1.000	0.003*	0.986	0.059	0.000*	0.000*	0.000*
	PSU	0.000*	1.000	0.000*	0.916	0.001*	0.690	1.000	0.995	0.735	0.303
	PRBF	0.043*	0.984	1.000	1.000	0.763	1.000	0.799	1.000	0.991	0.080
	SRBF	0.000*	0.998	0.000*	0.412	0.017*	0.002*	0.592	0.012*	0.771	0.010*
SU	PU	0.009*	1.000	0.000*	0.176	0.000*	0.110	0.050*	0.002*	0.289	0.558
	RBF	0.000*	0.000*	0.009*	0.103	0.000*	0.007*	1.000	0.000*	0.000*	0.000*
	PSU	0.000*	0.998	1.000	0.752	0.552	0.999	0.340	0.218	1.000	0.998
	PRBF	0.000*	0.839	0.000*	0.136	0.000*	0.371	0.000*	0.006*	0.001*	0.904
	SRBF	0.000*	0.998	0.937	0.997	0.153	0.676	0.967	0.998	1.000	0.489
RBF	PU	0.000*	0.000*	0.866	1.000	0.003*	0.986	0.059	0.000*	0.000*	0.000*
	SU	0.000*	0.000*	0.009*	0.103	0.000*	0.007*	1.000	0.000*	0.000*	0.000*
	PSU	0.000*	0.000*	0.006*	0.816	0.000*	0.083	0.409	0.000*	0.000*	0.000*
	PRBF	0.000*	0.000*	0.880	1.000	0.153	0.928	0.000*	0.000*	0.000*	0.000*
	SRBF	0.000*	0.000*	0.000*	0.279	0.000*	0.000*	0.989	0.000*	0.000*	0.000*
PSU	PU	0.000*	1.000	0.000*	0.916	0.001*	0.690	1.000	0.995	0.735	0.303
	SU	0.000*	0.998	1.000	0.752	0.552	0.999	0.340	0.218	1.000	0.998
	RBF	0.000*	0.000*	0.006*	0.816	0.000*	0.083	0.409	0.000*	0.000*	0.000*
	PRBF	0.360	0.980	0.000*	0.872	0.000*	0.944	0.271	1.000	0.010*	0.989
	SRBF	0.000*	0.341	0.967	0.949	0.975	0.226	0.988	0.701	1.000	0.755
PRBF	PU	0.043*	0.984	1.000	1.000	0.763	1.000	0.799	1.000	0.991	0.080
	SU	0.000*	0.839	0.000*	0.136	0.000*	0.371	0.000*	0.006*	0.001*	0.904
	RBF	0.000*	0.000*	0.880	1.000	0.153	0.928	0.000*	0.000*	0.000*	0.000*
	PSU	0.360	0.980	0.000*	0.872	0.000*	0.944	0.271	1.000	0.010*	0.989
	SRBF	0.000*	0.515	0.000*	0.342	0.000*	0.013*	0.004*	0.043*	0.025*	0.978*
SRBF	PU	0.000*	0.998	0.000*	0.412	0.017*	0.002*	0.592	0.012*	0.771	0.010*
	SU	0.000*	0.998	0.937	0.997	0.153	0.676	0.967	0.998	1.000	0.489
	RBF	0.000*	0.000*	0.000*	0.279	0.000*	0.000*	0.989	0.000*	0.000*	0.000*
	PSU	0.000*	0.341	0.967	0.949	0.975	0.226	0.988	0.701	1.000	0.755
	PRBF	0.000*	0.515	0.000*	0.342	0.000*	0.013*	0.004*	0.043*	0.025*	0.978

Dataset	Means ranking of the CCR_G
Balance	$\boldsymbol{\mu_{PSU}} \geqslant \mu_{PRBF} \geqslant \mu_{PU} > \mu_{SU} > \mu_{SRBF} > \mu_{RBF};\ \mu_{PSU} > \mu_{PU};$
Card	$\boldsymbol{\mu_{SRBF}} \geqslant \mu_{SU} \geqslant \mu_{PU} \geqslant \mu_{PSU} \geqslant \mu_{PRBF} > \mu_{RBF}$
German	$\boldsymbol{\mu_{SRBF}} \geqslant \mu_{PSU} \geqslant \mu_{SU} > \mu_{RBF} \geqslant \mu_{PRBF} \geqslant \mu_{PRBF}$
Glass	$\boldsymbol{\mu_{SU}} \geqslant \mu_{SRBF} \geqslant \mu_{PSU} \geqslant \mu_{PU} \geqslant \mu_{PRBF} \geqslant \mu_{RBF}$
Heart	$\boldsymbol{\mu_{SU}} \geqslant \mu_{PSU} \geqslant \mu_{SRBF} > \mu_{PU} \geqslant \mu_{PRBF} \geqslant \mu_{RBF};\ \mu_{PU} > \mu_{RBF}$
Ionos.	$\boldsymbol{\mu_{SRBF}} \geqslant \mu_{SU} \geqslant \mu_{PSU} \geqslant \mu_{PRBF} \geqslant \mu_{PU} \geqslant \mu_{RBF};\ \mu_{SRBF} > \mu_{PRBF};\ \mu_{SU} > \mu_{RBF}$
Newth.	$\boldsymbol{\mu_{PRBF}} \geqslant \mu_{PU} \geqslant \mu_{PSU} \geqslant \mu_{SRBF} \geqslant \mu_{RBF} \geqslant \mu_{SU};\ \mu_{PRBF} > \mu_{SRBF};\ \mu_{PU} > \mu_{RBF}$
Pima	$\boldsymbol{\mu_{SU}} \geqslant \mu_{SRBF} \geqslant \mu_{PSU} \geqslant \mu_{PRBF} \geqslant \mu_{PU} > \mu_{RBF};\ \mu_{SU} > \mu_{PRBF}$
Vote	$\boldsymbol{\mu_{PRBF}} \geqslant \mu_{PU} \geqslant \mu_{PSU} \geqslant \mu_{SRBF} \geqslant \mu_{SU} > \mu_{RBF};\ \mu_{PRBF} > \mu_{PSU}$
Zoo	$\boldsymbol{\mu_{PU}} \geqslant \mu_{SU} \geqslant \mu_{PSU} \geqslant \mu_{PRBF} \geqslant \mu_{SRBF} > \mu_{RBF};\ \mu_{PU} > \mu_{SRBF}$

* Statistically significant differences with p-value < 0.05; Tam.: Tamhane test; Tukey: Tukey test. $\mu_A \geqslant \mu_B$: topology A yields better results than topology B, but the differences are not significant; $\mu_A > \mu_B$: topology A yields better results than topology B with significant differences. The binary relation $\geqslant$ is not transitive.

and if not, a Tamhane test [48] is done. Both tests aim to rank the average of each level in each factor, in order to locate the level whose mean CCR_G is significantly better than the mean CCR_G of all the other levels.

Table 4 shows the results obtained following the above methodology, including the test performed (Tukey or Tamhane) and the performance ranking of the different basis functions. For the Balance dataset, the best result is obtained for PSU or PRBF combined basis functions. That is, the average CCR_G obtained with PSU is better than the averages obtained with the other combined models, except PRBF. For Card, German and Ionos, the most descriptive result is obtained with the combined model SRBF, although it is not significantly better than all the others in any case: it results in a multiple draw with four of the remaining methods for the Card dataset, and with two methods for German and Ionos. Very similar conclusions can be obtained from the analysis of almost all the other datasets, except for the Zoo dataset, where the procedures based on a single basis function type result in significantly better results than those based on CBF models.

Table 5 represents the results obtained following a similar methodology with the number of links (# links) of the obtained models, including the test performed (Tukey or Tamhane) and the number of link ranking for different basis functions. The models that have the significantly lowest number of connections are pure PU models, while pure RBF models have the highest number of connections. In this way, when using pure basis function models, the number of PU model connections is always lower than that of all the others, and when using CBF models, the number of connections of those models formed with PUs (PRBF and PSU) is also always lower than the number obtained with SRBF. This fact is due to the properties of PU basis function: PUs have the ability to capture interactions between input variables, which means they do not need so many connections.

In some datasets, the hybridization of projection type basis functions with RBFs yields better generalization results than

Table 5
p-values of the Tamhane and Tukey tests for #links and ranking of the different basis function proposal based on this multiple comparison test.

$H_0 \equiv \hat{\mu}_{(I)} = \hat{\mu}_{(J)}$		*p*-Value									
(I)	(J)	Balance Tam.	Card Tam.	German Tam.	Glass Tam.	Heart Tam.	Ionos. Tam.	Newth. Tukey	Pima Tam.	Vote Tam.	Zoo Tam.
PU	SU	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	RBF	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	PSU	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.001*	0.000*	0.000*	0.000*
	PRBF	0.003*	0.000*	0.000*	0.000*	0.000*	0.000*	0.006*	0.000*	0.001*	0.001*
	SRBF	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
SU	PU	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	RBF	0.025*	0.000*	0.000*	0.000*	0.000*	0.000*	0.224	0.000*	0.000*	0.000*
	PSU	0.102	0.206	0.938	1.000	1.000	0.544	0.392	0.208	0.997	0.000*
	PRBF	0.000*	0.911	0.091	0.151	0.012*	0.000*	0.112	0.016*	1.000	0.000*
	SRBF	0.907	0.312	0.843	0.000*	0.621	0.000*	0.668	0.000*	0.964	0.996
RBF	PU	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	SU	0.025*	0.000*	0.000*	0.000*	0.000*	0.000*	0.224	0.000*	0.000*	0.000*
	PSU	1.000	0.000*	0.000*	0.000*	0.000*	0.000*	0.001*	0.000*	0.000*	0.000*
	PRBF	0.020*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	SRBF	0.001*	0.000*	0.000*	0.000*	0.000*	0.000*	0.976	0.000*	0.000*	0.000*
PSU	PU	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.001*	0.000*	0.000*	0.000*
	SU	0.102	0.206	0.938	1.000	1.000	0.544	0.392	0.208	0.997	0.000*
	RBF	1.000	0.000*	0.000*	0.000*	0.000*	0.000*	0.001*	0.000*	0.000*	0.000*
	PRBF	0.249	0.128	0.008*	0.052	0.008*	0.000*	0.988	1.000*	1.000	0.002*
	SRBF	0.013*	0.004*	0.146	0.000*	0.451	0.000*	0.010	0.000*	0.498	0.003*
PRBF	PU	0.003*	0.000*	0.000*	0.000*	0.000*	0.000*	0.006*	0.000*	0.001*	0.001*
	SU	0.000*	0.911	0.091	0.151	0.012*	0.000*	0.112	0.016	1.000	0.000*
	RBF	0.020*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	PSU	0.249	0.128	0.008*	0.052	0.008*	0.000*	0.988	1.000*	1.000	0.002*
	SRBF	0.000*	1.000	0.945	0.004*	0.948	0.901	0.001*	0.000*	0.946	0.000*
SRBF	PU	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	SU	0.907	0.312	0.843	0.000*	0.621	0.000*	0.668	0.000*	0.964	0.996
	RBF	0.001*	0.000*	0.000*	0.000*	0.000*	0.000*	0.976	0.000*	0.000*	0.000*
	PSU	0.013*	0.004*	0.146	0.000*	0.451	0.000*	0.010	0.000*	0.498	0.003*
	PRBF	0.000*	1.000	0.945	0.004*	0.948	0.901	0.001*	0.000*	0.946	0.000*

Dataset	Means Ranking of the # Links
Balance	$\mu_{PU} < \mu_{PRBF} < \mu_{PSU} \leqslant \mu_{RBF} \leqslant \mu_{SU} \leqslant \mu_{SRBF}$; $\mu_{PSU} < \mu_{SU}$; $\mu_{RBF} < \mu_{SRBF}$
Card	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{SU} \leqslant \mu_{PRBF} \leqslant \mu_{SRBF} < \mu_{RBF}$; $\mu_{PSU} < \mu_{PRBF}$
German	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{SU} \leqslant \mu_{SRBF} \leqslant \mu_{PRBF} < \mu_{RBF}$; $\mu_{SU} < \mu_{PRBF}$
Glass	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{SU} \leqslant \mu_{PRBF} < \mu_{SRBF} < \mu_{RBF}$; $\mu_{PSU} < \mu_{PRBF}$
Heart	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{SU} \leqslant \mu_{SRBF} \leqslant \mu_{PRBF} < \mu_{RBF}$; $\mu_{SU} < \mu_{PRBF}$
Ionos	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{SU} < \mu_{PRBF} \leqslant \mu_{SRBF} < \mu_{RBF}$
Newth.	$\mu_{PU} < \mu_{PRBF} \leqslant \mu_{PSU} \leqslant \mu_{SU} \leqslant \mu_{SRBF} \leqslant \mu_{RBF}$; $\mu_{PSU} < \mu_{SRBF}$
Pima	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{PRBF} \leqslant \mu_{SU} < \mu_{SRBF} < \mu_{RBF}$
Vote	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{PRBF} \leqslant \mu_{SU} \leqslant \mu_{SRBF} < \mu_{RBF}$
Zoo	$\mu_{PU} < \mu_{PRBF} < \mu_{PSU} < \mu_{SRBF} \leqslant \mu_{SU} < \mu_{RBF}$

* Statistically significant differences with p-value < 0.05; Tam.: Tamhane test; Tukey: Tukey test. $\mu_A \geqslant \mu_B$: topology A results in a lower number of connections than topology B, but the differences are not significant; $\mu_A > \mu_B$: topology A results in a lower number of connections than topology B with significant differences. The binary relation $\geqslant$ is not transitive.

corresponding pure RBF models, with a significantly lower number of connections. In general, there is no definite conclusion about the type of basis function applicable to a given dataset, based on the statistical tests. Hybridization enhances generalization accuracy in Balance and Ionosphere datasets, while using pure basis function models are suggested for Heart and Zoo datasets. Moreover, CCR_G values are significantly more homogeneous using hybrid models for the Balance, Card and Vote datasets and there are no significant differences in the remaining datasets.

6. Conclusions

A hybrid CBF neural network model has been proposed, using a possible combination of two different transfer projection functions (SU and PU) and/or kernel functions (RBF) in the hidden layer of a feed-forward neural network. The different CBF models proposed have been designed with an EA (CBFEP) constructed specifically for the typology of a combination of basis functions, taking into account the specific characteristics of each pure and CBF model. The performance in the different datasets has been improved by considering the best individuals both in *CCR* and cross-entropy training errors (*double elitism*). The evaluation of the model an the algorithm for ten datasets has shown results that are comparable to those of other classification techniques found in machine learning literature [26].

The results obtained confirm the theory that a classification task can be performed using a model with two different components [13]: one associated with projection functions (PU or SU) and the other associated with kernel functions (RBF). Determining the most adequate projection function strongly depends on the dataset tackled, but, in general, the SRBF hybrid models have better accuracy than PRBF, especially in those datasets with two classes where only one discriminant function exists. The capacity for generalization using basis functions with

sufficiently diverse global-local discriminator characteristics (i.e. SRBF or PRBF models) is similar to or greater than the generalization capability of pure PU, SU or RBF models. Moreover, the hybridization of the two projection basis functions (PU and SU) did not present better accuracy than the remaining hybridizations but it presented a significantly lower number of connections (allowing smaller and more interpretable models) in the Card, German, Glass, Heart, Ionosphere and Vote datasets. The generalization capability of PSU models is similar to that obtained with their corresponding pure models.

The CBF models and the CBFEP algorithm maintain the performance and the number of links of corresponding pure models in several problems, or they even improve accuracy and/or model sizes. In general, the number of connections of pure models formed by PUs is always lower than that of all the others, and the number of connections of those CBF models using PUs (PRBF and PSU) is also lower than the number obtained with SRBF.

We are currently working on a hypothesis about the case in which a modelling structure using combined PU and RBF basis functions could possibly fit better than SU and RBF pure basis functions, associated to the existence of a big difference between the values of a relatively small number of data at the domain borders and in-domain inner data. This hypothesis is going to be evaluated empirically using a noise level measurement obtained for each dataset, and, in this way, the best combined model (SRBF or PRBF) would be determined according to the characteristics of the dataset in question.

Acknowledgments

This work has been partially subsidized by 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".

References

[1] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesús, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Computing 13 (3) (2007) 307–318.
[2] P.J. Angeline, G.M. Saunders, J.B. Pollack, An evolutionary algorithm that constructs recurrent neural networks, IEEE Transactions on Neural Networks 5 (1) (1994) 54–65.
[3] A. Asuncion, D.J. Newman, UCI Maching Learning Repository, 〈http://www.ics.uci.edu/~mlearn/MLRepository.html〉, Irvine, CA: University of California, School of Information and Computer Science, 2007
[4] A.S. Atukorale, T. Downs, P.N. Suganthan, Boosting HONG networks, Neurocomputing 51 (2003) 75–86.
[5] S.A. Billings, G.L. Zheng, Radial basis function network configuration using genetic algorithms, Neural Networks 8 (1995) 877–890.
[6] C.M. Bishop, Improving the generalization properties of radial basis function neural networks, Neural Computation 3 (4) (1991) 579–581.
[7] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.
[8] L.N. Castro, E.R. Hruschka, R.J.G.B. Campello, An evolutionary clustering technique with local search to design RBF neural network classifiers, in: Proceedings of the IEEE International Joint Conference on Neural Networks, 2004, pp. 2083–2088.
[9] S. Chen, S.A. Billings, C.F.N. Cowan, P.W. Grant, Practical identification of NARMAX models using radial basis functions, International Journal of Control 52 (1990) 1327–1350.
[10] S. Coen, N. Intrator, Global optimization of RBF networks, Technical Report, S.o.C. Science, Tel-Aviv University.
[11] S. Cohen, N. Intrator, A hybrid projection-based and radial basis function architecture: initial values and global optimization, Pattern Analysis and Applications 5 (2002) 113–120.
[12] C. Darken, J. Moody, Fast adaptive K-means clustering: some empirical results, in: Proceedings of the IEEE INNS International Joint Conference on Neural Networks, 1990, pp. 233–238.
[13] D. Donoho, Projection based in approximation and a duality with kernel methods, The Annals of Statistics 17 (1989) 58–106.
[14] W. Duch, R. Adamczak, G.H.F. Diercksen, Constructive density estimation network based on several different separable transfer functions, in: Proceedings of the 9th European Symposium on Artificial Neural Networks, Bruges, Belgium, 2001, pp. 107–112.
[15] W. Duch, N. Jankowski, Transfer functions hidden possibilities for better neural networks, in: Proceedings of the 9th European Symposium on Artificial Neural Networks, Bruges, Belgium, 2001, pp. 81–94.
[16] R. Durbin, D. Rumelhart, Products units: a computationally powerful and biologically plausible extension to backpropagation networks, Neural Computation 1 (1989) 133–142.
[17] D.B. Fogel, A.J. Owens, M.J. Wals, Artificial Intelligence Through Simulated Evolution, Wiley, New York, 1966.
[18] J.A.S. Freeman, D. Saad, Learning and generalization in radial basis function networks, Neural Computation 7 (5) (1995) 1000–1020.
[19] J. Friedman, Multivariate adaptive regression splines (with discussion), The Annals of Statistics 19 (1991) 1–141.
[20] J.H. Friedman, W. Stuetzle, Projection pursuit regression, Journal of the American Statistical Association 76 (1981) 817–823.
[21] K.J. Hunt, D. Sbarbaro, R. Zbikowski, P.J. Gawthrop, Neural networks for control system—a survey, Automatica 28 (1992) 1083–1112.
[22] A. Ismail, A.P. Engelbrecht, Global optimization algorithms for training product units neural networks, in: Proceedings of the International Joint Conference on Neural Networks IJCNN'2000, Italy, 2000, pp. 132–137.
[23] B.C. Iulian, Hybrid feedforward neural networks for solving classification problems, Neural Processing Letters 16 (1) (2002) 81–91.
[24] N. Jankowski, W. Duch, Optimal transfer function neural networks, in: Procedings of the 9th European Symposium on Artificial Neural Networks, Bruges, Belgium, 2001, pp. 101–106.
[25] D.J. Janson, J.F. Frenzel, Training product unit neural networks with genetic algorithms, IEEE Expert 8 (5) (1993) 26–33.
[26] N. Landwehr, M. Hall, Logistic model trees, Machine Learning 59 (2005) 161–205.
[27] L.R. Leerink, Learning with products units, Advances in Neural Networks Processing Systems 7 (1995) 537–544.
[28] M. Lehtokangas, J. Saarinen, Centroid based Multilayer Perceptron Networks, Neural Processing Letters 7 (1998) 101–106.
[29] F.H.F. Leung, H.K. Lam, S.H. Ling, 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.
[30] H. Levene, In Contributions to Probability and Statistics, Stanford University Press, 1960.
[31] R.P. Lippmann, Pattern classification using neural networks, IEEE Communications Magazine 27 (1989) 47–64.
[32] A.C. Martínez-Estudillo, C. Hervás-Martínez, F.J. Martínez-Estudillo, N. García, Hybridation of evolutionary algorithms and local search by means of a clustering method, IEEE Transaction on Systems, Man and Cybernetics, Part. B: Cybernetics 36 (3) (2006) 534–546.
[33] 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.
[34] F.J. Martínez-Estudillo, C. Hervás-Martínez, P.A. Gutiérrez, A.C. Martínez-Estudillo, Evolutionary product-unit neural networks classifiers, Neurocomputing 72 (1–3) (2008) 548–561.
[35] R.G. Miller, Beyond ANOVA, Basics of Applied Statistics, Chapman & Hall, London, 1996.
[36] M.T. Musavi, W. Ahmed, K.H. Chan, K.B. Farms, D.M. Hummels, On the training of radial basis function classifiers, Neural Networks 5 (1992) 595–603.
[37] M.J.L. Orr, Regularisation in the selection of radial basis function centres, Neural Computation 7 (3) (1995) 606–623.
[38] C. Panchapakesan, M. Palaniswami, D. Ralph, C. Manzie, Effects of moving the centers in an RBF network, IEEE Transactions on Neural Networks 13 (6) (2002) 1299–1307.
[39] Y.H. Pao, Y. Takefuji, Functional-link net computing: theory, system architecture, and functionalities, IEEE Computer 25 (5) (1992) 76–79.
[40] J. Park, I.W. Sandberg, Universal approximation using radial basis function networks, Neural Computation 3 (2) (1991) 246–257.
[41] R. Reed, Pruning algorithms. A survey, IEEE Transactions on Neural Networks 4 (1993) 740–747.
[42] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by backpropagating errors, Nature 323 (1986) 533–536.
[43] A. Sakurai, Tighter bounds of the VC-dimension of three layer networks, in: Proceedings of the World Congress on Neural Networks, Erlbaum, Hillsdale, New Jersey, 1993, pp. 540–543.
[44] M. Schmitt, On the complexity of computing and learning with multiplicative neural networks, Neural Computation 14 (2001) 241–301.
[45] M. Schmitt, Radial basis function neural networks have superlinear VC dimension, in: Proceedings of the COLT/EuroCOLT2001, Lecture Notes in Artificial Intelligence, vol. 2111, 2001, pp. 14–30.
[46] D.F. Specht, A general regression neural network, IEEE Transactions on Neural Networks 2 (6) (1991) 568–576.
[47] J.A.K. Suykens, J.P.L. Vandewalle, B.L.R.D. Moor, Artificial Neural Networks for Modelling and Control of Non-linear Systems, Kluwer Academic Publishers, USA, 1996.
[48] A.C. Tamhane, D.D. Dunlop, Statistics and Data Analysis, Prentice Hall, Englewood Cliffs, NJ, 2000.

[49] E.K. Tang, P.N. Suganthan, X. Yao, An analysis of diversity measures, Machine Learning 65 (2006) 247–271.
[50] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, Berlin, 1999.
[51] S. Ventura, C. Romero, A. Zafra, J.A. Delgado, C. Hervás-Martínez, JCLEC: a JAVA framework for evolutionary computation, Soft Computing 12 (4) (2007) 381–392.
[52] D. Wedge, D. Ingram, D. McLean, C. Mingham, Z. Bandar, On global-local artificial neural networks for function approximation, IEEE Transactions on Neural Networks 17 (4) (2006) 942–952.
[53] X. Yao, Evolving artificial neural networks, Proceedings of the IEEE 87 (9) (1999) 1423–1447.
[54] X. Yao, Y. Liu, A new evolutionary system for evolving artificial neural networks, IEEE Transactions on Neural Networks 8 (3) (1997) 694–713.
[55] Z.Q. Zhao, 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.
[56] Q. Zhu, Y. Cai, L. Liu, A global learning algorithm for a RBF network, Neural Networks 12 (1999) 527–540.

Pedro A. Gutiérrez-Peña was born in Cordoba, Spain, in 1982. He received his B.S. degree in Computer Science from the University of Seville, Seville, Spain, in 2006. He is currently working toward his Ph.D. degree in the Department of Computer Science and Numerical Analysis (University of Cordoba, Spain), in the area of computer science and artificial intelligence. His current interests include neural networks and their applications, evolutionary computation and hybrid algorithms.

César Hervás-Martínez was born in Cuenca, Spain. He received his B.S. degree in Statistics and Operating Research from the Universidad Complutense, Madrid, Spain, in 1978 and his 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 and an Associate Professor in the Department of Quantitative Methods in the School of Economics. His current research interests include neural networks, evolutionary computation, and the modelling of natural systems.

Mariano Carbonero-Ruz was born in Córdoba, Spain. He received his M.Sc. degree in Mathematics in 1987 and his Ph.D. degree in Mathematics in 1995, speciality Statistics and Operational Research, both from the University of Seville, Seville, Spain. Later, he received his M.Sc. degree in Economics Sciences in 2000 from UNED, Spain. He develops his research in applied statistics in machine learning and Economic Sciences. He is currently a Professor in the Department of Management and Quantitative Methods in ETEA, University of Córdoba, Spain.

Juan C. Fernández-Caballero was born in Peñarroya-Pueblonuevo, Córdoba, Spain, in 1980. He received the B.S. degree in Computer Science from the University of Granada, Spain, in 2005. He is currently a Ph.D. Student in the Department of Computing and Numerical Analysis, University of Córdoba, Spain, in the area of computer science and artificial intelligence. His current areas of interest include neural networks and applications, evolutionary computation and multiobjective optimization.