Logistic Regression by Means of Evolutionary Radial Basis Function Neural Networks

Pedro Antonio Gutiérrez, *Member, IEEE*, César Hervás-Martínez, *Member, IEEE*, and Francisco J. Martínez-Estudillo, *Member, IEEE*

***Abstract*— This paper proposes a hybrid multilogistic methodology, named logistic regression using initial and radial basis function (RBF) covariates. The process for obtaining the coefficients is carried out in three steps. First, an evolutionary programming (EP) algorithm is applied, in order to produce an RBF neural network (RBFNN) with a reduced number of RBF transformations and the simplest structure possible. Then, the initial attribute space (or, as commonly known as in logistic regression literature, the covariate space) is transformed by adding the nonlinear transformations of the input variables given by the RBFs of the best individual in the final generation. Finally, a maximum likelihood optimization method determines the coefficients associated with a multilogistic regression model built in this augmented covariate space. In this final step, two different multilogistic regression algorithms are applied: one considers all initial and RBF covariates (multilogistic initial-RBF regression) and the other one incrementally constructs the model and applies cross validation, resulting in an automatic covariate selection [simplelogistic initial-RBF regression (SLIRBF)]. Both methods include a regularization parameter, which has been also optimized. The methodology proposed is tested using 18 benchmark classification problems from well-known machine learning problems and two real agronomical problems. The results are compared with the corresponding multilogistic regression methods applied to the initial covariate space, to the RBFNNs obtained by the EP algorithm, and to other probabilistic classifiers, including different RBFNN design methods [e.g., relaxed variable kernel density estimation, support vector machines, a sparse classifier (sparse multinomial logistic regression)] and a procedure similar to SLIRBF but using product unit basis functions. The SLIRBF models are found to be competitive when compared with the corresponding multilogistic regression methods and the RBFEP method. A measure of statistical significance is used, which indicates that SLIRBF reaches the state of the art.**

***Index Terms*— Artificial neural networks, classification, evolutionary algorithms, evolutionary programming, logistic regression, radial basis function neural networks.**

I. Introduction

THE pace at which pattern classification algorithms have been successfully applied to new problems has increased exponentially during the last few years [1]. Examples of application come from all branches of science, technology, and medicine (e.g., medical diagnosis, handwritten character recognition, dynamic signature verification, or satellite image analysis), and in many regards classification is the most important statistical problem around [2].

The traditional statistical approach to pattern recognition is a natural application of Bayesian decision theory, and the linear logistic regression model is one of its main representatives. As suggested by Hastie and Tibshirani [2], an obvious way to generalize the linear logistic regression model is to replace the linear predictors with a nonparametric version of them, such as an arbitrary regression surface, a more structured high-dimensional regression surface estimated using procedures as Friedman's multivariate adaptive regression [3], or even more structured nonparametric models such as an additive model of basis function. This paper aims to develop this last idea, presenting a competitive study in multiclass learning which combines different statistical and soft computing elements such as multilogistic regression, radial basis function neural networks (RBFNNs), and evolutionary algorithms (EAs). We broaden the ideas introduced in a recently proposed combination of NNs and logistic regression [4], [5]. This recent methodology [multilogistic regression linear product-unit neural networks (MRLPU)] allows the generation of hybrid linear/nonlinear classification surfaces and the identification of possible strong interactions that may exist between the attributes (also known as covariates in the logistic regression literature) which define the classification problem.

This paper presents an extension of this methodology, since we combine a linear model with a RBFNN nonlinear model instead of a product-unit NN (PUNN) model. The coefficients are then estimated using logistic regression (similar to [6]). The approach, called logistic regression using initial and radial basis function (LIRBF) covariates, consists of a multilogistic regression model built on the combination of the initial covariates and the RBFs of a RBFNN (local approximator). RBFNNs generally use hyper-ellipsoids to split the pattern space, which is different from multilayer perceptrons that build their classifications on hyperplanes, defined by a weighted sum. It is important to point out that, in our method, a true multiclass formulation is introduced based on multinomial logistic regression. Strictly speaking, a multinomial logistic regression formulation for multiclass classification is not new (see [5], [7]), and it has been previously applied to the training of RBFNNs [6]. However, it is not usually employed in pattern recognition and machine learning literature.

Manuscript received January 27, 2009; revised September 17, 2010; accepted November 10, 2010. Date of publication December 6, 2010; date of current version February 9, 2011. This work was supported in part by the Spanish Inter-Ministerial Commission of Science and Technology under Project TIN 2008-06681-C06-03, the European Regional Development fund, and the "Junta de Andalucía," Spain, under Project P08-TIC-3745.

P. A. Gutiérrez and C. Hervás-Martínez are with the Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba 14004, Spain (e-mail: pagutierrez@uco.es; chervas@uco.es).

F. J. Martínez-Estudillo is with the Department of Management and Quantitative Methods, Faculty of Economics and Business Sciences, University of Córdoba, Córdoba 14004, Spain (e-mail: fjmestud@etea.com).

Digital Object Identifier 10.1109/TNN.2010.2093537

Although logistic regression is a simple and useful procedure, it poses problems when applied to real classification problems, where we cannot usually automatically assume that covariates will have additive and purely linear effects [8]. Thus, our technique overcomes these difficulties by augmenting the input vector with new RBF variables. On the other hand, adding linear terms to a RBFNN yields simpler models that are easier to interpret. Specifically, the linear terms reduce both the variance associated with the overall modeling procedure and the likelihood of ending up with unnecessary RBFs. In addition, if a covariate only appears linearly in the final model, then the interpretation of this covariate is easier than that of a covariate which appears in a RBF transformation. Logistic regression models are usually fitted by maximum likelihood, where the iteratively reweighted least square (IRLS) algorithm is the traditional way to estimate the maximum likelihood parameters. The algorithm usually converges since the log-likelihood is concave. However, in our approach, the nonlinearity of the Gaussian RBFs with respect to the centers and radii implies that the corresponding Hessian matrix is generally indefinite and the likelihood function could have local maxima (although the Hessian for the coefficients is still positive semidefinite). These reasons justify, in our opinion, the use of an alternative heuristic procedure such as an EA to estimate the centers and radii of the model.

The estimation of the coefficients is carried out in three steps. In a first step, an evolutionary programming (EP) algorithm determines the number of Gaussian RBFs in the model and their corresponding centers and radii. The algorithm aims to produce a reduced number of RBF transformations with the simplest structure possible, i.e., tries to select the most important input variables for the estimation of the RBFs. This step can be seen as a heuristic search in the coefficients' model space. Once the basis functions have been determined by the EP algorithm, a transformation of the covariate space is considered by adding the nonlinear transformations of the input variables given by the RBFs of the best individual in the final generation. The final model is linear in the set of variables formed by the RBFs and the initial covariates. Now, the Hessian matrix is positive semidefinite, and fitting proceeds with a maximum likelihood optimization method. In this final step, two different multilogistic regression algorithms are applied: 1) multilogistic (MLogistic), which considers all initial and RBF covariates, and 2) simplelogistic (SLogistic), which constructs the model incrementally and applies cross validation, resulting in an automatic covariate selection. Both methods include a regularization parameter, which has also been optimized. This results in two different models: multilogistic initial-RBF regression (MLIRBF) and simplelogistic initial-RBF regression (SLIRBF). We evaluate the performance of our methodology in 18 datasets taken from the UC Irvine repository [9] and 2 datasets corresponding to a real agronomical problem of precision farming.

The results are compared with the corresponding multilogistic regression methods applied to the initial covariate space, to the RBFNNs obtained by the EP algorithm (RBFEP), and to other probabilistic classifiers, including different RBFNN design methods [e.g., relaxed variable kernel density estimation (RVKDE)], support vector machines (SVMs), a sparse classifier [sparse multinomial logistic regression (SMLR)], and a procedure similar to SLIRBF but using product unit basis functions (SLIPU). The SLIRBF method is found to obtain better results than pure SLogistic and RBFEP in almost all the datasets considered. A measure of statistical significance is used, which indicates that SLIRBF reaches the state of the art.

This paper is organized as follows: Section II is devoted to a brief analysis of some works related to the models proposed; the description of the LIRBF models is carried out in Section III; Section IV describes the LIRBF learning algorithm; Section V contains the experimental results; and finally, Section VI summarizes the conclusions of our work.

II. Related Works

This section presents an overview of some research related to the models proposed from different points of view. We especially highlight the learning procedure and the framework of the model. There have been quite a few learning algorithms proposed for RBFNNs. A successful construction of a RBFNN model depends on three factors: the design of a proper kernel (basis) function, the selection of proper centers of kernels, and the determination of a proper weight structure. Basically, there are two categories of learning algorithms proposed for RBFNNs. The first category of learning algorithms simply places one RBF at each sample [10]. If all the training samples are selected as hidden centers, the generalization capability of the network will become so poor that many noised or deformed samples will not be recognized. However, this can be avoided by properly regularizing the model. On the other hand, the second category of learning algorithms attempts to reduce the number of hidden units in the network, or, equivalently, the number of RBFs in the model [11]. One primary motivation behind the design of the second category of algorithms is to improve the efficiency of the learning process. Alternatively, there are many approaches to determine hidden centers. For instance, the number and position of the RBFs may be fixed and defined *a priori* [12], or they may be determined by conducting a clustering analysis of the training dataset and allocate one hidden unit for each cluster. Algorithms differ by the clustering algorithm employed: k-means clustering, fuzzy k-means clustering, hierarchical clustering and self-organizing map NNs, [13], or also input–output clustering [14].

An interesting alternative is to evolve RBFNNs using EAs. A very complete state of the art of the different approaches and characteristics of a wide range of EA and RBFNN combinations is given in [15]. For example, RBFNN design has been approached by using evolutionary clustering techniques with local search [16], hybrid algorithms [17], [18], or multiobjective algorithms [19], or by evolving only the basis functions [20], [21] or space-filling curves to generate the RBF centers [22].

On the other hand, many linear parametric models can be recast into an equivalent dual representation in which the

predictions are based on a linear combination of a kernel function evaluated at the training data points. From a structural point of view, RBFNNs and the LIRBF models proposed in this paper are closely related to direct kernel methods [23]. The basis function kernel model provides a general mechanism for constructing nonlinear generalizations of a wide range of conventional linear statistical methods, in which the result is a family of kernel learning methods [24], [25]. The SVM [26], [27] is perhaps the most common kernel learning method for statistical pattern recognition. The parameters of a kernel model are typically given by the solution of a convex optimization problem, so there is a single global optimum.

The predictions of SVM are not probabilistic and estimate the optimal decision boundary separating examples belonging to each class directly (indirect probabilistic discriminative approach), rather than estimating the *a posteriori* probability of class membership and subsequently establishing the decision boundary at some fixed threshold probability (direct probabilistic generative approach). A possible way to obtain these probabilities is the use of a confidence estimate, e.g., by using a pairwise ("1-versus-1") approach and estimating the probabilities from the pairwise probabilities [28]. It is important to point out that something very similar to the use of initial covariates in the final predictor function can be achieved for kernel methods by using a weighted sum of RBFs and linear kernel functions.

An extension of the SVM in order to offer a natural estimate of the probability of class membership and to be generalized to the multiclass case is the kernel logistic regression (KLR) [29], i.e., replacing the loss function of the SVM by the negative log-likelihood (NLL) function. Although KLR compromises the hinge loss function of the SVM, there are some sparse KLR models such as the informative vector machine [30].

An alternative sparse kernel technique, known as relevance vector machines (RVMs) [31], is based on a Bayesian formulation of the kernel model introducing a prior to the model weights governed by a set of hyperparameters, one associated with each weight, whose most probable values are iteratively estimated from the data. In contrast to SVM, the kernel is not required to satisfy the Mercer condition and RVM provides natural posterior probability outputs. Probabilistic classification vector machines (PCVMs) [32] modify RVM by introducing a signed and truncated Gaussian prior over every weight, where the sign of the prior is determined by the class label. Both RVM and PCVM are Bayesian sparse kernel techniques that share many characteristics of SVM while avoiding its principal limitations without degrading the general performance.

Nevertheless, it is significant to highlight that the methods above are specifically designed for two classes. The extension to more than two classes is usually achieved by considering "1-versus-all" or "1-versus-1" approaches [33].

A direct multiclass formulation called SMLR is introduced in [7], where the approach is based on multinomial logistic regression with a sparsity-promoting prior. By combining a bound optimization approach with a component-wise update procedure, the authors derive a fast exact algorithm for learning sparse multiclass classifiers that scale favorably in both the number of training samples and feature dimensionality, making them applicable even to large datasets in high-dimensional feature spaces. There are some recent applications and extensions of the SMLR algorithm [34]–[36]. From a different point of view, direct multiclass approaches have been developed based on the estimation of kernel density for efficient construction of RBFNNs. RVKDE [37] works by constructing one RBF sub-network to approximate the probability density function of each class of objects in the training dataset. The SMLR and RVKDE multiclass approaches have been chosen for comparison with the models proposed in this paper.

Finally, a first proposal of a combination of NNs and logistic regression is given in two recent studies [4], [5]. The model is based on the hybridization of a linear multilogistic regression model and a nonlinear PUNN model for binary and multiclass classification problems. The main differences between these studies and this paper are the following.

1) First of all, this paper considers RBFs (which are *local* approximators) for the nonlinear part of the model, while the MRLPU method is based on PUs (which are *global* approximators).
2) The EA presented in this paper is based on the determination of the RBFNN models, where population initialization (radii and centers of the Gaussian functions) is more complex than that of the PUNN models.
3) The SLogistic algorithm used in this paper (see Section IV-A) is more advanced than the IRLS method used for MRLPU, and performs an automatic structural simplification of the model. This results in more robust maximum-likelihood estimations and, in general, avoids overfitting.
4) A more exhaustive 10-fold experimental design has been performed with 10 repetitions per each fold, since the whole process has been automated.

III. LIRBF Models

In the classification problem, measurements x_i, $i = 1, 2, \ldots, k$, are taken on a single individual (or object), and the individuals are classified into one of the J classes on the basis of these measurements. It is assumed that J is finite, and the measurements x_i are random observations from these classes. Let $D = \{(\mathbf{x}_n, \mathbf{y}_n); n = 1, 2, \ldots, N\}$ be a training dataset, where $\mathbf{x}_n = (x_{1n}, \ldots, x_{kn})$ is the vector of measurements taking values in $\Omega \subset \mathbb{R}^k$, and $\mathbf{y}_n$ is the class level of the nth individual. The common technique for representing class levels using a "1-of-J" encoding vector is adopted, $\mathbf{y} = \left(y^{(1)}, y^{(2)}, \ldots, y^{(J)}\right)$, such that $y^{(l)} = 1$ if $\mathbf{x}$ corresponds to an example belonging to class l and $y^{(l)} = 0$ otherwise. Based on the training samples, we wish to find a classification rule $F : \Omega \rightarrow \{1, 2, \ldots, J\}$ for classifying the individuals. In other words, F provides a partition, say $D_1, D_2, \ldots, D_J$, of Ω, where D_l corresponds to the lth class, $l = 1, 2, \ldots, J$, and measurements belonging to D_l will be classified as coming from the lth class. A misclassification occurs when the decision rule F assigns an individual (based on the measurement vector) to a class j when it is actually coming from a class $l \neq j$.

To evaluate the performance of the classifiers, the correct classification rate (CCR or C) is defined by

$$C = \frac{1}{N}\sum_{n=1}^{N} I(F(\mathbf{x}_n) = \mathbf{y}_n) \tag{1}$$

where $I(\bullet)$ is the zero-one loss function. A good classifier tries to achieve the highest possible C for a given problem. It is usually assumed that the training data are independent and identically distributed samples from an unknown probability distribution. Suppose that the conditional probability that $\mathbf{x}$ belongs to class l verifies: $p\left(y^{(l)} = 1 \middle| \mathbf{x}\right) > 0$, $l = 1, 2, \ldots, J$, $\mathbf{x} \in \Omega$, and sets the function

$$\log \frac{p\left(y^{(l)} = 1 \middle| \mathbf{x}\right)}{p\left(y^{(J)} = 1 \middle| \mathbf{x}\right)} = f_l(\mathbf{x}, \theta_l)$$

where θ_l is the weight vector corresponding to class l, the right-hand side of the equation is the predictor function, and $f_J(\mathbf{x}, \theta_J) = 0$. In this way, the probability for one of the classes (the last one, in our case) does not need be estimated. Under a multinomial logistic regression, the probability that $\mathbf{x}$ belongs to class l is then given by

$$p\left(y^{(l)} = 1 \middle| \mathbf{x}, \theta\right) = \frac{\exp f_l(\mathbf{x}, \theta_l)}{\sum_{j=1}^{J} \exp f_j\left(\mathbf{x}, \theta_j\right)}, \quad l = 1, 2, \ldots, J \tag{2}$$

where $\theta = (\theta_1, \theta_2, \ldots, \theta_{J-1})$.

The classification rule coincides with the optimal Bayes' rule. In other words, an individual should be assigned to the class which has the maximum probability, given the measurement vector $\mathbf{x}$

$$F(\mathbf{x}) = \hat{l}, \quad \text{where } \hat{l} = \arg\max_{l} p\left(y^{(l)} = 1 \middle| \mathbf{x}, \theta\right), \quad l = 1, \ldots, J.$$

The predictor function of our logistic regression model proposal is based on the combination of the standard linear model in the initial covariates and a nonlinear model constructed with RBF transformed covariates, which captures well-defined locations in the covariate space. The general expression of the predictor function is given by

$$f_l(\mathbf{x}, \theta_l) = \alpha_0^l + \sum_{i=1}^{k} \alpha_i^l x_i + \sum_{j=1}^{m} \beta_j^l \exp\left(-\frac{\left\|\mathbf{x} - \mathbf{c}_j\right\|^2}{r_j^2}\right) \tag{3}$$

where $l = 1, 2, \ldots, J-1$, $\theta_l = (\boldsymbol{\alpha}^l, \boldsymbol{\beta}^l, \mathbf{W})$ is the vector of parameters for each predictor function, $\boldsymbol{\alpha}^l = (\alpha_0^l, \alpha_1^l, \ldots, \alpha_k^l)$ and $\boldsymbol{\beta}^l = \left(\beta_1^l, \ldots, \beta_m^l\right)$ are the coefficients of the multilogistic regression model, and $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m)$ are the parameters of the RBF nonlinear transformations, $\mathbf{w}_j = (\mathbf{c}_j, r_j)$, $\mathbf{c}_j = (c_{j1}, c_{j2}, \ldots, c_{jk})$ is the center or average of the jth Gaussian RBF transformation, r_j is the corresponding radius or standard deviation (SD) and $c_{ji}, r_j \in \mathbb{R}$. Thus, the final model is equivalent to a RBFNN with "skip-layer" connections (in standard NN terminology), where the softmax transformation is applied to the outputs of the network. We have selected Gaussians with isotropic covariance matrices [8]. Although this restriction influences our results, the optimization of anisotropic Gaussian RBFs (with different SDs in each direction) would have increased the number of parameters to be adjusted by the EA (associated to the product of the number of characteristics by the number of basis functions), thereby increasing the computational cost.

IV. LIRBF Learning Algorithm

In the supervised learning context, the components of the weight vectors $\theta = (\theta_1, \theta_2, \ldots, \theta_{J-1})$ are estimated from the training dataset D. To perform the maximum likelihood estimation of θ, one can minimize the NLL function

$$\begin{aligned} L(\theta) &= -\frac{1}{N}\sum_{n=1}^{N} \log p\left(\mathbf{y}_n \middle| \mathbf{x}_n, \theta\right) \\ &= \frac{1}{N}\sum_{n=1}^{N}\left[-\sum_{l=1}^{J-1} y_n^{(l)} f_l(\mathbf{x}_n, \theta_l) + \log \sum_{l=1}^{J-1} \exp f_l(\mathbf{x}_n, \theta_l)\right] \end{aligned} \tag{4}$$

where $\theta_l = (\boldsymbol{\alpha}^l, \boldsymbol{\beta}^l, \mathbf{W})$ and $f_l(\mathbf{x}_n, \theta_l)$ corresponds to the LIRBF model defined in (3).

The nonlinearity of the model with respect to the parameters $\mathbf{c}_j$ and r_j of $\mathbf{W}$ and the indefinite character of the associated Hessian matrix of $L(\theta)$ do not recommend the use of gradient-based methods to maximize the log-likelihood function. Moreover, the optimal number of basis functions of the model (i.e., the number of RBF transformations) is unknown. Thus, the estimation of the vector parameter $\hat{\theta}$ is carried out by means of the hybrid procedure described below.

The methodology proposed is based on the combination of an EP algorithm (global explorer) and a standard maximum likelihood optimization method (local exploiter). In a first step, the EP algorithm is applied to design the structure and train the weights of an RBFNN. The evolutionary process determines the number m of RBFs in the model, and the corresponding matrix $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m)$. Once the basis functions have been determined by the EP algorithm, we consider a transformation of the input space by adding the nonlinear transformations of the input variables given by the RBFs of the best individual in the final generation of the EP algorithm (i.e., the overall best individual obtained by the EP algorithm).

The model is now linear in these new variables and the initial covariates. The remaining coefficient vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are calculated by the maximum likelihood optimization method which involves choosing the parameters that maximize the probability of the data points observed. For the multilogistic regression model, there are no closed-form solutions for these estimates. Instead, the numeric optimization algorithms that have to be used are those that approach the maximum likelihood solution iteratively and reach it at the limit. In the next subsection, two algorithms for obtaining this maximum likelihood solution are introduced. Then, the different steps of the LIRBF learning algorithm are described and, in the last subsection, the details of the EP algorithm are given.

A. Algorithms for Multilogistic Regression Maximum Likelihood Optimization

In this paper, two different algorithms have been considered for obtaining the maximum likelihood solution for the multilogistic regression model, both available on the WEKA machine learning workbench [38].

1) MLogistic: It is an algorithm for building a multinomial logistic regression model with a ridge estimator to guard against overfitting by penalizing large coefficients, based on the work by le Cessie and van Houwelingen [39]. For adjusting this hyperparameter, we have performed a 10-fold cross validation with the following range: $\{10^{-2}, 10^{-1.5}, \ldots, 10^{2}\}$.

In order to find the coefficient matrices $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ for which $L(\theta)$ in (4) is minimized, a quasi-Newton method is used. Specifically, the method used is the active sets method with the Broyden–Fletcher–Goldfarb–Shanno (BFGS) update [40].

2) SLogistic: This algorithm builds multinomial logistic regression models by using the LogitBoost algorithm [41], which was proposed by Friedman *et al*. for fitting *additive logistic regression models* by maximum likelihood. These models are a generalization of the (linear) logistic regression models.

LogitBoost is a boosting algorithm that performs forward stagewise fitting: in every iteration i, it computes "response variables" that encode the error of the model currently fit on the training examples (in terms of probability estimates), and then tries to improve the model by adding a new function $f_{ij}(\mathbf{x})$ to the committee of functions $F_j(\mathbf{x})$, $1 \leq j \leq J$.

As shown by Friedman *et al*. [41], this amounts to performing a quasi-Newton step in every iteration, where the Hessian matrix is approximated by its diagonal. In the special case that the $f_{ij}(\mathbf{x})$ and so the $F_j(\mathbf{x})$ are linear functions of the input covariates, the additive logistic regression model is equivalent to the linear logistic model. Assuming that $F_j(\mathbf{x}) = \boldsymbol{\alpha}_j^T \mathbf{x}$, the equivalence of the two models is established by setting $\boldsymbol{\alpha}_j = \boldsymbol{\beta}_j - \boldsymbol{\beta}_J$ for $j = 1, \ldots, J-1$ and $\boldsymbol{\alpha}_J = \boldsymbol{\beta}_J$. This means that LogitBoost can be used for learning multilogistic regression models by fitting a standard least-squares regression function as the f_{ij}.

However, it is also possible to use even simpler functions for the f_{ij}, simple regression functions that perform a regression on only one covariate present in the training data. In this way, fitting simple regression by least-squared error means fitting a simple regression function to each covariate in the data using least squares as the error criterion, and then selecting the covariate that gives the smallest squared error

$$f_{ij} = \beta_l x_l + \gamma_l, \qquad l = \arg\min_l E_{lj}, \qquad l = 1, \ldots, k \tag{5}$$

where i is the number of iteration of the LogitBoost algorithm, j is the class considered, k is the number of inputs, β_l is the slope of the simple linear regression, γ_l is the corresponding intercept, and E_{lj} is the weighted sum of squared error of the linear regression considering only the lth covariate with respect to the expected class label Y_j. The final model found by LogitBoost will be the same because quasi-Newton stepping is guaranteed to actually find the maximum likelihood solution if the likelihood function is convex. Using simple regression instead of multiple ones will basically slow down the process, but, if it is stopped before it converges, this will result in automatic covariate selection, because the model will only include the most relevant covariates present in data. The *SLogistic* algorithm is based on applying LogitBoost with simple regression functions and determining the optimum number of iterations by cross validation. In this paper, we have slightly modified the criterion for the selection of the covariate in iteration i by adding a regularization term

$$f_{ij} = \beta_l x_l + \gamma_l, \qquad l = \arg\min_l (E_{lj} + \lambda |\beta_l|), \qquad l = 1, \ldots, k \tag{6}$$

where λ is a regularization hyperparameter. This results in selecting not only the covariate that obtains the minimum error, but also taking into account its coefficient absolute value. Consequently, there are two hyperparameters that have to be adjusted for applying this modified version of SLogistic, λ and the maximum number of iterations. The number of iterations has been obtained by using a fivefold cross validation, with a maximum number of 500 iterations. For adjusting the λ hyperparameter, we have performed a 10-fold cross validation with the following range: $\lambda \in \{10^{-2}, 10^{-1.5}, \ldots, 10^{2}\}$. Once λ and the maximum number of iterations are obtained, the SLogistic algorithm is run again with the whole training set. Further details about the algorithm can be found in [42].

B. Estimation of the Model Coefficients

In this subsection, the steps of the LIRBF learning algorithm are described in detail. The process is structured in three steps.

Step 1: For determining the best RBFNN model, we apply an EP algorithm to find the basis functions

$$\mathbf{B}(\mathbf{x}, \mathbf{W}) = \{B_1(\mathbf{x}, \mathbf{w}_1), B_2(\mathbf{x}, \mathbf{w}_2), \ldots, B_m(\mathbf{x}, \mathbf{w}_m)\}$$

corresponding to the nonlinear part of $f_l(\mathbf{x}, \theta_l)$ in (3). We have to determine the number of basis functions m and the weight matrix $\mathbf{W}$. To apply evolutionary NN techniques, we consider a RBFNN with softmax outputs and the standard structure: an input layer with a node for every input variable; a hidden layer with several RBFs; and an output layer with $J-1$ nodes, where J is the number of classes. There are no connections between the nodes of a layer and none between the input and output layers either. A scheme of these models is given in Fig. 1.

The activation function of the jth node in the hidden layer is given by

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

where $\mathbf{c}_j = (c_{j1}, \ldots, c_{jk})$ and c_{ji} is the weight of the connection between the ith input node and the jth RBF.

The activation function of the lth output node is given by

$$g_l(\mathbf{x}, \boldsymbol{\beta}^l, \mathbf{W}) = \beta_0^l + \sum_{j=1}^{m} \beta_j^l B_j(\mathbf{x}, \mathbf{w}_j) \tag{7}$$

where β_j^l is the weight of the connection between the jth RBF and the lth output node and β_0^l is the bias of the lth output

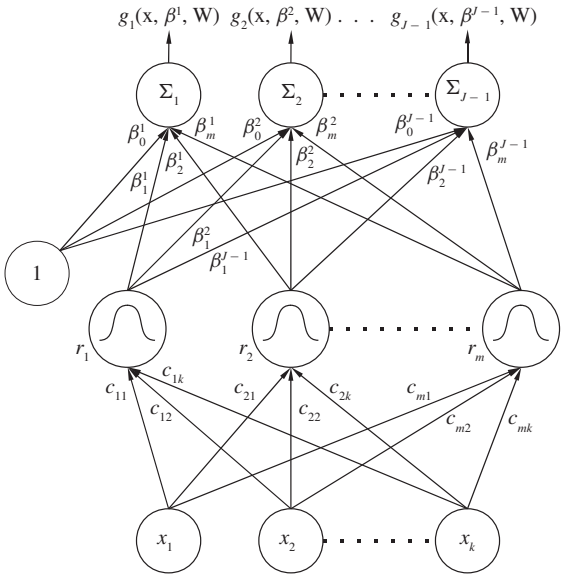

Fig. 1. Structure of RBFNNs: an input layer with k input variables, a hidden layer with m RBFs, and an output layer with $J-1$ nodes.

node. The transfer function of all output nodes is the identity function.

The weight matrix $\mathbf{W}$ is estimated by means of an evolutionary NN algorithm (detailed in Section IV-C) that optimizes the error function given by the NLL for N observations associated with the RBFNN model

$$L^*(\boldsymbol{\beta}, \mathbf{W}) = \frac{1}{N}\sum_{n=1}^{N}\left[-\sum_{l=1}^{J-1} y_n^{(l)} g_l(\mathbf{x}_n, \boldsymbol{\beta}^l, \mathbf{W}) + \log\sum_{l=1}^{J-1}\exp g_l(\mathbf{x}_n, \boldsymbol{\beta}^l, \mathbf{W})\right]. \quad (8)$$

Although in this step the evolutionary process obtains a concrete value for the $\boldsymbol{\beta}$ vector, we only consider the estimated weight vector $\hat{\mathbf{W}} = (\hat{\mathbf{w}}_1, \hat{\mathbf{w}}_2, \ldots, \hat{\mathbf{w}}_m)$, which builds the basis functions. The RBFs are appended to the initial covariates space and the values for the $\boldsymbol{\beta}$ vector need to be re-estimated together with the coefficients of the initial covariates ($\boldsymbol{\alpha}$ coefficient vector), since we consider them to be new covariates of the problem.

Step 2: The following transformation of the input space is introduced by including the nonlinear basis functions obtained by the EP algorithm in Step 1 for the best model of the final generation (i.e., the overall best individual obtained by the EP algorithm, given that the algorithm is elitist)

$$H : \mathbb{R}^k \to \mathbb{R}^{k+m},$$
$$(x_1, x_2, \ldots, x_k) \to (x_1, x_2, \ldots, x_k, z_1, z_2, \ldots, z_m)$$

where $z_1 = B_1(\mathbf{x}, \hat{\mathbf{w}}_1)$, $z_2 = B_2(\mathbf{x}, \hat{\mathbf{w}}_2)$, ..., $z_m = B_m(\mathbf{x}, \hat{\mathbf{w}}_m)$.

Step 3: Using a matrix notation, the model of (3) can be expressed as

$$f_l(\mathbf{x}, \theta_l) = \boldsymbol{\alpha}^l\mathbf{x} + \boldsymbol{\beta}^l\mathbf{z}, \qquad l = 1, 2, \ldots, J-1$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ and $\mathbf{z} = (z_1, z_2, \ldots, z_m)$.

We minimize the NLL function of (4) with respect to the parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ of θ

$$L(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{N}\sum_{n=1}^{N}\left[-\sum_{l=1}^{J-1} y_n^{(l)}(\boldsymbol{\alpha}^l\mathbf{x}_n + \boldsymbol{\beta}^l\mathbf{z}_n) + \log\sum_{l=1}^{J-1}\exp(\boldsymbol{\alpha}^l\mathbf{x}_n + \boldsymbol{\beta}^l\mathbf{z}_n)\right]$$

where $\mathbf{x}_n = (1, x_{1n}, \ldots, x_{kn})$. Now, the Hessian matrix of the NLL in the new variables $x_1, x_2, \ldots, x_k, z_1, z_2, \ldots, z_m$ is semidefinite positive.

In this final step, both algorithms presented in Section IV-A have been applied to obtain the parameter matrix θ. This results in two different models: one with all $x_1, x_2, \ldots, x_k, z_1, z_2, \ldots, z_m$ covariates present in the model (*MLogistic* algorithm), and the other with only those variables selected by the *SLogistic* algorithm (see Section IV-A). These two approaches will be called MLogistic initial-RBF regression (MLIRBF) and SLogistic initial-RBF regression (SLIRBF), respectively.

For comparison purposes, we have also considered the multilogistic regression models that are obtained with these two algorithms but constructed only from the nonlinear transformations given by the RBFNN of the EP algorithm, i.e., $z_1, z_2, \ldots, z_m$. This results in two other approaches, which we will call MLogistic RBF regression (MLRBF) and SLogistic RBF regression (SLRBF).

Finally, Fig. 2 presents a schematic view of the steps of the methodology and the different associated models. The methods are represented in a double squared box. The best RBFNN individual obtained in Step 1 by the EP algorithm is also evaluated (RBFEP method).

C. Evolutionary Acquisition of the RBF Nonlinear Transformations

This subsection presents the EA used for obtaining the RBF nonlinear transformations. The algorithm is aimed to produce a reduced number of RBF transformations with the simplest structure possible, i.e., trying to select the most important input variables for the estimation of the RBFs.

From the different paradigms of evolutionary computation, we have chosen EP because we are evolving artificial NNs. Therefore, crossover is not used because of its potential disadvantages in evolving artificial networks [43]. The population-based EA for architectural design and the estimation of the real coefficients have points in common with other EAs in the literature [43]–[46]. The search begins with an initial population, and after each iteration the population is updated using a population-update algorithm, applying parametric and structural mutation to the best 10% and the best 90% of the population, respectively. The algorithm is elitist with respect to the best individual, and the best 10% of individuals replace the worst 10% of individuals in each generation. The stop condition is defined as follows: the variance of the fitness of the best 10% of the population is less than a user-defined value, or if a maximum number of generations is reached.

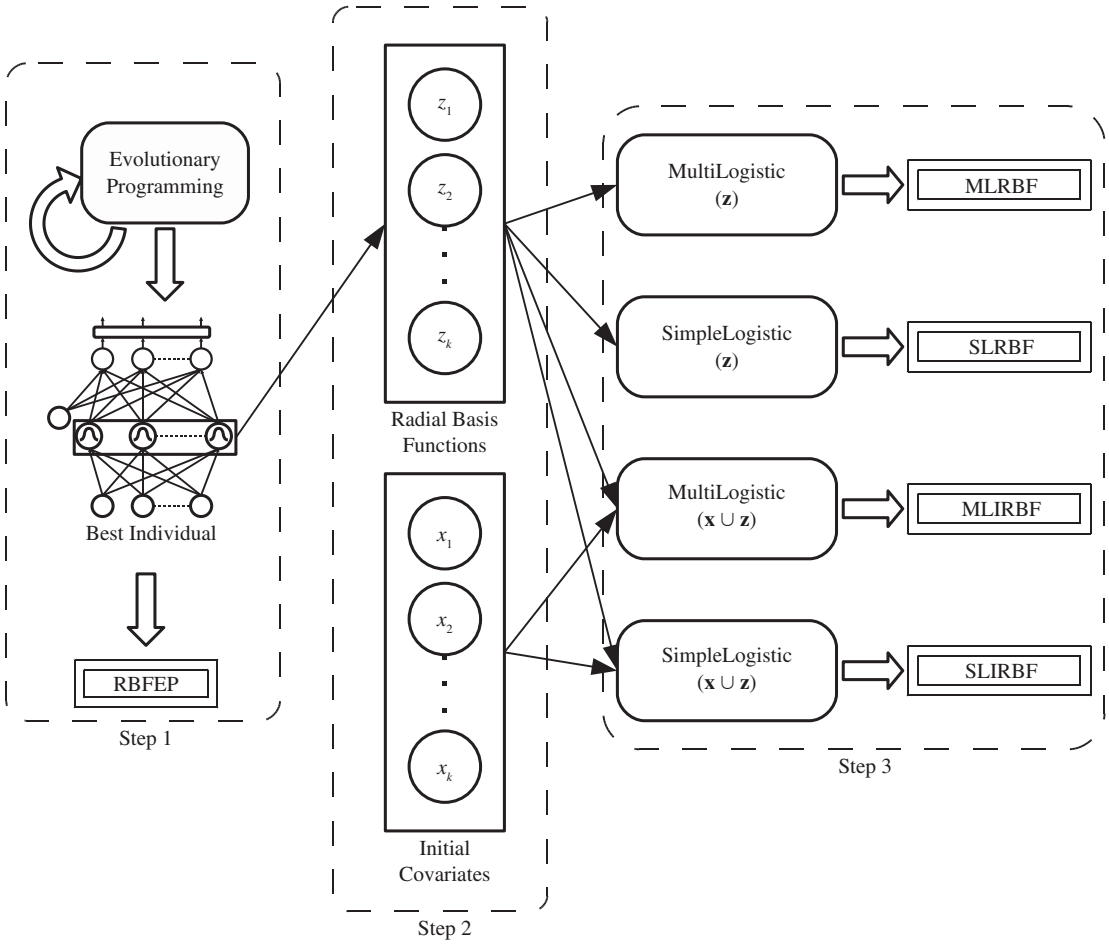

Fig. 2. Different steps in the LIRBF methodology. The different models associated with this methodology are presented in a double squared box.

The RBFEP algorithm is detailed in Fig. 3, where p^B is the best optimized RBFNN returned by the algorithm. The main characteristics of the algorithm are the following.

1) Representation of the Individuals: The algorithm evolves architectures and connection weights simultaneously, each individual being a fully specified RBFNN. The NNs are represented using an object-oriented approach and the algorithm deals directly with the RBFNN phenotype. Each connection is specified by a binary value, indicating whether the connection exists, and a real value representing its weight. As the crossover is not considered, this object-oriented representation does not assume a fixed order between the different RBFs.

In order to define the topology of the NNs, two parameters are considered: $M_{\min}$ and $M_{\max}$. They correspond, respectively, to the minimum and maximum number of RBFs in the whole evolutionary process.

2) Error and Fitness Functions: We consider $L^*(\boldsymbol{\beta}, \mathbf{W})$ defined in (8) as the error function of an individual $g(\mathbf{x}, \boldsymbol{\beta}, \mathbf{W})$ in the population. Observe that g is an RBFNN and can be seen as a multivaluated function

$$g(\mathbf{x}, \boldsymbol{\beta}, \mathbf{W}) = \left(g_1(\mathbf{x}, \boldsymbol{\beta}^1, \mathbf{W}), \ldots, g_{J-1}(\mathbf{x}, \boldsymbol{\beta}^{J-1}, \mathbf{W})\right)$$

where $g_l(\mathbf{x}, \boldsymbol{\beta}^l, \mathbf{W})$ is defined in (7).

The fitness measure needed for evaluating the individuals (Fig. 3, Steps 2, 7, and 16) is a strictly decreasing transformation of the error function $L^*(\boldsymbol{\beta}, \mathbf{W})$ given by $A(g) = 1/(1 + L^*(\boldsymbol{\beta}, \mathbf{W}))$, where $0 < A(g) \leq 1$.

3) Initialization of the Population: The initial population is generated by trying to obtain RBFNNs with the maximum possible fitness. The necessity of initial well-positioned centers was experimentally checked, making the RBFEP algorithm obtain a better performance than purely randomly generated centers. A classical k-means clustering process is applied to obtain these initial centers. The increase in the selective pressure balances out with the large population size of the algorithm and the use of highly disruptive structural mutators.

First, 5000 random RBFNNs are generated (Fig. 3, Step 1), where the number of RBFs m is a random value in the interval $[M_{\min}, M_{\max}]$. The number of connections between all RBFs of an individual and the input layer is a random value in the interval $[1, k]$ and all of them are connected with the same randomly chosen input variables. In this way, all the RBFs of each individual are initialized using the same random subset of the input variables (this is necessary to later apply the k-means algorithm). A random value in the $[-I, I]$ interval is assigned for the weights between the input layer and the hidden layer and in the $[-O, O]$ interval for those between the hidden layer

RBFEP Algorithm:

Input: Training dataset (D)

Output: Best optimized RBFNN (p^B)

1: $P^I \leftarrow \{p_1^I, ..., p_{5000}^I\}$ // p_i^I is a randomly generated RBFNN

2: $\forall p_i^I \in P^I$, $f_i^I \leftarrow A(p_i^I)$ // Evaluate fitness

3: $P \leftarrow \{\{p_{(1)}, ..., p_{(5000)}\}, \quad (p_{(i)} \prec p_{(j)}) \iff (f_i^I > f_j^I)$ // Sort individuals in P^I by increasing f_i^I

4: $P \leftarrow \{p_{(1)}, ..., p_{(500)}\}$ // Retain the best 500 RBFNNs

5: $\forall p_{(i)} \in P, p_{(i)} \leftarrow k$–means$(p_{(i)})$ // Improve individuals' centres

6: **while not** Stop Condition **do**

7: $\forall p_i \in P$, $f_i \leftarrow A(p_i)$ // Evaluate fitness

8: $P \leftarrow \{p_{(1)}, ..., p_{(500)}\}, \quad (p_{(i)} \prec p_{(j)}) \iff (f_i > f_j)$ // Sort individuals in P by increasing f_i

9: $p^B \leftarrow p_{(1)}$ // Store Best Individual

10: $P^P \leftarrow \{p_{(1)}, ..., p_{(50)}\}$ // Parametric mutation parents (best 10% of individuals)

11: $P^S \leftarrow \{p_{(1)}, ..., p_{(449)}\}$ // Structural mutation parents (best 90% of individuals minus one)

12: $\forall p_{(i)}^P \in P^P$, $p_{(i)}^P \leftarrow$ parametricMutation$(p_{(i)}^P)$ // Apply parametric mutation

13: $\forall p_{(i)}^S \in P^S$, $p_{(i)}^S \leftarrow$ structuralMutation$(p_{(i)}^S)$ // Apply structural mutation

14: $P \leftarrow P^P \cup P^S \cup \{p^B\}$ // Offspring including the elite

15: **end while**

16: $\forall p_i \in P$, $f_i \leftarrow A(p_i)$ // Evaluate fitness

17: $P \leftarrow \{p_{(1)}, ..., p_{(500)}\}, \quad (p_{(i)} \prec p_{(j)}) \iff (f_i > f_j)$ // Sort individuals in P by increasing f_i

18: $p^B \leftarrow p_{(1)}$

19: **return** p^B

Fig. 3. RBFEP training algorithm framework.

and the output layer. The individuals obtained are evaluated using the fitness function, and the initial population is finally obtained by selecting the best 500 RBFNNs (Fig. 3, Steps 2–4).

In order to improve the randomly generated centers, the standard k-means clustering algorithm [47] is applied using these random centers as the initial centroids for the algorithm and a maximum number of iterations of 100 (Fig. 3, Step 5).

4) Parametric Mutation: Parametric mutation (Fig. 3, Step 10) alters the value of the coefficients of the model. It is accomplished for each coefficient $w \in (\boldsymbol{\beta}, \mathbf{W})$ of the $g(\mathbf{x}, \boldsymbol{\beta}, \mathbf{W})$ individual adding a Gaussian noise, $w(t+1) = w(t) + \xi(t)$, where $\xi(t) \in N(0, \alpha(t))$ and $N(0, \alpha(t))$ represent a 1-D normally distributed random variable with mean 0 and variance $\alpha(t)$. The weights are sequentially mutated, hidden node after hidden node, and a standard simulated annealing process [48] is applied to accept or reject the modifications in each RBF. Thus, if ΔA is the difference in the fitness function after and before the random step, the criterion is as follows: if $\Delta A \geq 0$, the step is accepted, and if $\Delta A < 0$, the step is accepted with a probability $\exp(\Delta A / T(g))$, where the temperature $T(g)$ of an individual g is given by $T(g) = 1 - A(g)$, $0 \leq T(g) < 1$.

The variance $\alpha(t)$ is updated throughout the evolution. There are different methods to update the variance. We use one of the simplest methods, the $1/5$ success rule of Rechenberg [49]. This rule states that the ratio of successful mutations should be $1/5$. Therefore, if the ratio of successful mutations is greater than $1/5$, the mutation deviation should increase, otherwise, the deviation should decrease. Thus

$$\alpha(t+s) = \begin{cases} (1+\lambda)\alpha(t), & \text{if } s_g > 1/5 \\ (1-\lambda)\alpha(t), & \text{if } s_g < 1/5 \\ \alpha(t), & \text{if } s_g = 1/5 \end{cases}$$

where s_g is the frequency of successful mutations over s generations and λ gives the magnitude of the updating. The adaptation tries to avoid being trapped in local minima and to speed up the evolutionary process when searching conditions are suitable.

5) Structural Mutation: Structural mutation (Fig. 3, Step 11) implies a modification in the structure of the RBFNNs and allows the exploration of different regions in the search space, helping to keep the diversity of the population. There are five different structural mutations: node addition, node deletion, connection addition, connection deletion, and node fusion. These five mutations are applied sequentially to each network, each one with a specific probability. The node mutations are performed as follows.

a) Node addition. One or more RBFs are added to the hidden layer. The origin nodes of the connections from the input layer are chosen randomly and have a random value in the interval $[-I, I]$. The origin nodes of the connections to the output layer are chosen randomly and their values are random values in the interval $[-O, O]$.

b) Node deletion. One or more RBFs, together with their connections, are randomly selected and deleted.

c) Node fusion. Two randomly selected RBFs of the mutated individual, a and b, are replaced by a new node c, which is a combination of the two. The connections common to both basis functions are kept, with a weight given by

$$c_{ci} = \frac{r_a}{r_a + r_b} c_{ai} + \frac{r_b}{r_a + r_b} c_{bi}, \qquad \beta_c^i = \frac{(\beta_a^i + \beta_b^i)}{2}$$
$$r_c = \frac{(r_a + r_b)}{2}.$$

Those connections not shared by the basis functions are inherited by c with a probability of 0.5 and their weight is unchanged.

The number of nodes added or deleted in node addition, node deletion, and node fusion mutations is calculated as $\Delta_{\min} + uT(g)[\Delta_{\max} - \Delta_{\min}]$, u being a random uniform

variable in the interval [0, 1], $T(g) = 1 - A(g)$ the temperature of the neural net, and $\Delta_{\min}$ and $\Delta_{\max}$ a minimum and maximum number of nodes specified as parameters.

The structural connection mutations are performed as follows.

a) Connection addition. Connection addition mutations are first performed in the hidden layer and then in the output layer. When adding a connection from the input layer to the hidden layer, a node from each layer is randomly selected, and then the connection is added with a random weight. A similar procedure is performed from the hidden layer to the output layer.
b) Connection deletion. In the same way, connection deletion mutation is first performed in the hidden layer and then in the output layer, choosing the origin node randomly from the previous layer and the target node from the mutated layer.

We apply connection mutations sequentially for each mutated neural net, first adding (or deleting) $1 + u[\Delta_o n_o]$ connections from the hidden layer to the output layer, and then adding (or deleting) $1 + u[\Delta_h n_h]$ connections from the input layer to the hidden layer. u is a random uniform variable in the interval [0, 1], Δ_o and Δ_h are previously defined ratios of number of connections in the hidden and the output layer, and n_o and n_h are the current number of connections in the output and the hidden layers.

Instead of applying repair mechanisms, the mutators are applied only if the net generated is valid, otherwise, a new mutation is randomly chosen. Parsimony is also encouraged in evolving the networks by attempting the five structural mutations sequentially, where node or connection deletion and node fusion are always attempted before addition. Moreover, the deletion and fusion operations are made with higher probability ($T(g)$ for deletion and fusion mutations, and $T^2(g)$ for addition ones). If a deletion or fusion mutation is successful, no other mutation will be made. If the probability does not select any mutation, one of the mutations is chosen at random and applied.

6) Parameters of the Algorithm: All the parameters used in the EA have the same heuristic values in all the problems analyzed below. The use of an EA, which dynamically adapts to the problem evaluated, results in a performance which is negligibly affected by minor changes in these parameters.

We have done a simple linear rescaling of the input variables in the interval $[-2, 2]$, X_i^* being the transformed variables. The centers c_{ji} are initialized in this interval (i.e., $[-I, I] = [-2, 2]$), and the coefficients β_j^l are initialized in the $[-5, 5]$ interval (i.e., $[-O, O] = [-5, 5]$). The initial value of the radii r_j is obtained as a random value in the interval $(0, d_{\max}]$, where $d_{\max}$ is the maximum distance between two training input examples.

The size of the population is $N = 500$. We have considered $\alpha(0) = 0.5$, $\lambda = 0.1$, and $s = 5$ for the parametric mutations. For the structural mutations, the number of RBFs that can be added or removed in a structural mutation is within the $[\Delta_{\min}, \Delta_{\max}] = [1, 2]$ interval. The ratio of the number of connections to add or delete in the hidden and the output layer during structural mutations is $\Delta_o = 0.05$ and $\Delta_h = 0.3$. A wide enough range for the number of hidden nodes is used ($M_{\min} = 1$ and $M_{\max} = 14$).

The stop criterion is reached whenever one of the following two conditions is fulfilled: the variance of the fitness of the best 10% of the population is less than 10^{-4}; or 500 generations are completed.

V. Experiments

The proposed methodology is applied to 18 datasets taken from the UCI repository [9], to test its overall performance when compared to other methods. Two additional datasets described in Section V-C have been included, which correspond to a real agronomical problem of discriminating cover crops in olive orchards. The datasets have been included on a public website.[1]

The first subsection defines the experimental design and the next one describes the performance evaluation and the model selection for the different methods compared. Characteristics of the real agronomical problems are given in the next subsection. Then, the comparison of the proposed models to the SLogistic, MLogistic, and RBFEP methods is presented, and the next subsection is devoted to a comparison to other related probabilistic classifiers. The last two subsections include a study of two of the best models obtained and the computation time required by the different algorithms, respectively.

A. Experimental Design

The selection of the methods used for the comparison is based on their similarities to the model and the methodology proposed. In this way, the proposed methods (MLRBF, SLRBF, MLIRBF, and SLIRBF) are compared to the following algorithms.

1) Multilogistic regression methods, including the SLogistic and MLogistic algorithms applied to the initial covariate space (see Section IV-A). As our models are logistic regression models, it is necessary to compare their performance to standard logistic regression algorithms.
2) The RBFEP method. As our models are built from the RBFs of the best RBFNN obtained by the EP algorithm, it is necessary to compare their performance to the original RBFEP method.
3) Some high-performance probabilistic classifiers.
   a) A Gaussian RBF network (RBFN) [6], deriving the centers and width of hidden units using k-means and combining the outputs obtained from the hidden layer using logistic regression. k-means is applied separately to each class to derive k clusters for each class. The structure and learning process of this RBFN is similar to that of the MLRBF method.
   b) AdaBoost.M1 [50], using RBFN as the base learner and the maximum number of iterations set to 100 iterations [Ada100(RBFN)].

[1] Available at http://www.uco.es/ayrna/index.php?lang=en ("Datasets" section).

TABLE I

CHARACTERISTICS OF THE 20 DATASETS USED FOR THE EXPERIMENTS: NUMBER OF INSTANCES (SIZE), NUMBER OF REAL (R), BINARY (B), AND NOMINAL (N) INPUT VARIABLES, TOTAL NUMBER OF INPUTS (#IN.), NUMBER OF CLASSES (#OUT), AND PER-CLASS DISTRIBUTION OF THE INSTANCES (DISTRIBUTION)

Dataset	Size	R	B	N	#In.	#Out.	Distribution
Hepatitis	155	6	13	–	19	2	(32, 123)
Glass(G2)	163	9	–	–	9	2	(87, 76)
Sonar	208	60	–	–	60	2	(97, 111)
Heart-c	302	6	3	4	26	2	(164, 138)
Ionosphere	351	33	1	–	34	2	(126, 225)
Vote	435	–	16	–	16	2	(267, 168)
Australian	690	6	4	5	51	2	(307, 383)
Breast-w	699	9	–	–	9	2	(458, 241)
German	1000	6	3	11	61	2	(700, 300)
Post-Op.	90	–	1	6	20	3	(2, 24, 64)
Iris	150	4	–	–	4	3	(50, 50, 50)
Newthyroid	215	5	–	–	5	3	(150, 35, 30)
Balance	625	4	–	–	4	3	(288, 49, 288)
Cortijo(Sp)	80	7	–	–	7	4	(40, 20, 10, 10)
Cortijo(Su)	50	7	–	–	7	4	(10, 20, 10, 10)
Lymph.	148	3	9	6	38	4	(2, 81, 61, 4)
Anneal	898	6	14	18	59	5	(8, 99, 684, 67, 40)
Glass	214	9	–	–	9	6	(70, 76, 17, 13, 9, 29)
Zoo	101	1	15	–	16	7	(41, 20, 5, 13, 4, 8, 10)
Audiology	226	–	61	8	96	24	(1, 1, 57, 22, 1, 2, 20, 48, 2, 6, 2, 4, 2, 2, 9, 3, 1, 2, 22, 4, 1, 4, 8, 2)

All nominal variables are transformed into binary variables.

Heart-c: Heart disease (Cleveland); Lymph.: Lymphography; Post-Op.: Post-operative.

c) $C-$SVM [51] with RBF kernels (SVM). From a structural point of view, the SVMs are related to RBFNNs and the LIRBF models proposed in this paper and they have become one of the most popular and developed methods nowadays. In order to deal with the multiclass case, a "1-versus-1" approach has been considered, following the recommendations of Hsu and Lin [33].

d) RVKDE [37]. This is a very interesting learning alternative for the efficient construction of RBFNNs.

e) SMLR [7]. This method has been selected as a good representative of recently developed sparse classifiers (RVM, PCVM, etc.) and, as far as we are concerned, the only genuine multinomial classifier in this context. The SMLR method was run by using Gaussian kernels centered on all the training points (following the SVM configuration).

f) SLIPU. This algorithm consists of applying the MRLPU methodology proposed in two previous papers [4], [5] but using the SLogistic algorithm instead of the IRLS method. The differences between MRLPU and LIRBF were analyzed in Section II. The SLogistic algorithm has been used for the comparisons because we want to analyze the effect of using RBFs instead of PUs, and not the effect of the maximum-likelihood optimization algorithm.

Thorough experiments on several datasets have been performed. The selected datasets present different numbers of instances, features, and classes (see Table I).

The RBFEP algorithm was implemented in JAVA using the evolutionary computation framework JCLEC[2] [52]. For the MLRBF, SLRBF, MLIRBF, SLIRBF, and SLIPU methods, the RBFEP algorithm was modified slightly, applying the SLogistic and MLogistic algorithms from WEKA [38]. We also used "libsvm" [53] to obtain the results of the SVM method, and WEKA to obtain the results of the RBFN and the Ada100(RBFN) methods. The RVKDE algorithm is implemented and available in the "rvkde" software package,[3] and the SMLR method is also widely available in a Java package.[4]

Regarding the parameters of the RBFEP algorithm, the values for all the parameters are common for all datasets and they were introduced in Section IV-C.6. In order to perform a fair comparison, the parameter values for the EP algorithm used for obtaining the SLIPU models are the same as those used for the RBFEP algorithm.

B. Performance Evaluation and Model Selection

Performance evaluation and model selection were carried out in the following way.

1) For the RBFEP, MLRBF, MLIRBF, SLRBF, SLIRBF, and SLIPU methods, a 10-fold cross validation was considered for performance evaluation. Ten runs of the RBFEP algorithm were run per each fold, in order to take into account the randomness of the method,

[2]Available at http://jclec.sourceforge.net.

[3]Available at http://mbi.ee.ncku.edu.tw/rvkde/.

[4]Available at http://www.cs.duke.edu/∼amink/software/smlr/.

resulting in a total of 100 runs. Per each run, the model selection involved the adjustment of the different hyperparameters of the logistic regression methods. Once the RBFs were obtained by the RBFEP algorithm, the adjustment was performed using a nested cross validation applied over the corresponding training set, selecting the value for the parameter that resulted in the lowest cross-validated error, and repeating the training with the selected value and the complete training set. Concretely, the following hyperparameters were optimized.

a) The ridge parameter of the MLRBF and MLIRBF methods was adjusted using a 10-fold cross validation with the following range: $\lambda \in \{10^{-2}, 10^{-1.5}, \ldots, 10^{2}\}$.

b) The λ regularization parameter of the SLRBF, SLIRBF, and SLIPU methods was adjusted using a 10-fold cross validation with the following range: $\lambda \in \{10^{-2}, 10^{-1.5}, \ldots, 10^{2}\}$. Per each λ value, the maximum number of iterations has to be also adjusted, and, in this case, a 5-fold cross validation with a maximum of 500 iterations was used: the corresponding training set was stratified in five folds, SLogistic was run on every training set up to a maximum number of 500 iterations and the classification error on the respective test set was logged. Afterwards, SLogistic was run again on all data using the number of iterations that gave the smallest error on the test set averaged over the five folds. The reason why a five fold was considered in this case is that we experimentally checked that this hyperparameter is less sensitive than the regularization parameter, and the additional computational cost of a 10-fold cross validation was not really necessary.

2) For all the other methods (SLogistic, MLogistic, RBFN, Ada100RBFN, SVM, RVKDE, and SMLR), the results were obtained performing 10 times a 10-fold cross validation, because they are all deterministic methods, i.e., they are not based on random values and they return the same result for each execution. Although this is not a mandatory methodology for deterministic methods, it can be found in recent works [42]. Per each training fold of the 100 runs, the different hyperparameters involved in each method were adjusted using also a nested cross validation applied over the corresponding training set, selecting the value for the parameter that resulted in the lowest cross-validated error, and repeating the training with the selected value and the complete training set. Concretely, the following hyperparameters were optimized.

a) For the selection of the SVM hyperparameters (regularization parameter C, and width of the Gaussian functions γ), a grid search algorithm was applied with a 10-fold cross validation, using the following ranges: $C \in \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and $\gamma \in \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$.

b) The RVKDE algorithm includes three hyperparameters (β, k_1, and k_2, see [37] for more details) that were also optimized using a grid search with 10-fold cross validation and the following ranges: $\beta \in \{1, 1.5, \ldots, 5\}$, $k_1 \in \{1, 2, \ldots, 30\}$ and $k_2 \in \{1, 2, \ldots, 30\}$.

c) The main hyperparameter for the SMLR algorithm is the λ regularization parameter, which was selected by 10-fold cross validation with the range $\lambda \in \{10^{-2}, 10^{-1.5}, \ldots, 10^{2}\}$. Kernel width parameter was selected by using the result of the cross validation for the SVM: the best width for SVM was then used for SMLR. This was done as suggested by Krishnapuram *et al*. [7].

C. Description of the "Cortijo(Sp)" and "Cortijo(Su)" Datasets

Both datasets correspond to a real agronomical problem of precision farming, which consists of discriminating cover crops in olive orchards as affected by their phenological stage, using a high-resolution field spectroradiometer.

Olives are the main perennial Spanish crop, and soil management in olive orchards is mainly based on intensive land tillage operations, which are quite relevant to the increase in atmospheric CO_2, desertification, erosion, and land degradation. Due to this negative environmental impact, the European Union only subsidizes crop systems that alter the natural soil as little as possible and protect it with cover crops. Current methods to estimate the cover crop soil coverage consist of sampling and ground visits to only 1% of the total olive orchards. Remotely sensed data may offer the ability to efficiently identify and map crops and cropping methods over large areas [54]. To detect and map olive trees and cover crops, suitable differences must exist in spectral reflectance between them and bare soil. Moreover, the spectral signature of any vegetation type is different depending on the phenological stage in which it is obtained [55]. This analysis will help to program the suitable wavelengths of airborne hyperspectral sensors. In this way, hyperspectral measurements obtained using a spectroradiometer were used to discriminate four classes: live cover crops, dead cover crops, bare soil, and olive trees.

This study was conducted in Andalusia, southern Spain, in a location named "Cortijo del Rey" in early spring [Cortijo(Sp)] and early summer [Cortijo(Su)]. Forty spectral signatures of live cover crop, 20 of dead cover crops, 10 of olive trees, and 10 of bare soil were taken in spring in 2007. The same number of samples was acquired in the summer but only 10 samples of live cover crop could be obtained. The hyperspectral range was between 400 and 900 nm. However, preliminary experiments suggested that using only seven wavelengths was enough to characterize the original spectral curves and so the datasets include only these wavelengths.

D. Comparison to SLogistic, MLogistic, and RBFEP

First of all, an analysis is performed on the accuracy of all the methods proposed (MLRBF, SLRBF, MLIRBF, and SLIRBF) when compared to the SLogistic, MLogistic,

TABLE II

COMPARISON OF THE PROPOSED METHODS TO SLOGISTIC, MLOGISTIC, AND RBFEP: MEAN AND SD OF THE ACCURACY RESULTS (C_G) FROM 100 EXECUTIONS OF A 10-FOLD CROSS VALIDATION, MEAN ACCURACY ($\overline{C_G}$), AND MEAN RANKING ($\overline{R}$)

	Method ($C_G(\%)$)						
	SLogistic	MLogistic	RBFEP	MLRBF	SLRBF	MLIRBF	SLIRBF
Dataset	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD
Hepatitis	84.70 ± 8.56	**86.44 ± 7.46**	82.43 ± 6.76	82.08 ± 7.48	81.53 ± 7.31	85.15 ± 6.78	*85.38 ± 6.60*
Glass(G2)	76.04 ± 9.97	70.52 ± 9.76	*78.28 ± 9.49*	77.36 ± 9.48	77.28 ± 10.09	77.12 ± 9.26	**78.34 ± 9.63**
Sonar	74.93 ± 9.74	74.97 ± 9.12	80.21 ± 8.08	80.55 ± 8.31	80.46 ± 8.48	*80.88 ± 6.49*	**80.90 ± 8.23**
Heart-c	83.29 ± 6.33	84.10 ± 6.59	*84.55 ± 4.07*	83.72 ± 5.23	83.88 ± 3.96	84.01 ± 4.46	**84.70 ± 4.51**
Ionosphere	87.78 ± 5.08	87.78 ± 5.39	**94.26 ± 3.87**	93.92 ± 4.20	93.86 ± 4.09	93.61 ± 4.15	*93.95 ± 4.11*
Vote	95.98 ± 2.70	95.95 ± 2.57	95.77 ± 2.73	95.70 ± 2.73	95.77 ± 2.73	*96.09 ± 2.40*	**96.23 ± 2.74**
Australian	85.62 ± 4.05	85.42 ± 3.90	*85.74 ± 4.32*	85.64 ± 4.23	85.55 ± 4.20	85.45 ± 4.46	**85.84 ± 3.56**
Breast-w	96.24 ± 2.00	*96.52 ± 1.99*	96.01 ± 2.40	95.90 ± 2.45	95.96 ± 2.51	96.26 ± 2.51	**96.61 ± 2.48**
German	75.07 ± 3.77	**75.79 ± 3.63**	75.40 ± 3.29	74.72 ± 3.25	74.88 ± 3.25	74.77 ± 3.91	*75.15 ± 3.73*
Post-Op.	*70.89 ± 5.86*	70.11 ± 6.45	70.22 ± 7.22	70.44 ± 7.27	69.78 ± 7.25	70.56 ± 6.39	**71.00 ± 5.67**
Iris	96.00 ± 5.36	96.53 ± 4.69	95.93 ± 5.83	96.20 ± 5.21	95.60 ± 5.62	*96.67 ± 4.59*	**96.80 ± 4.69**
Newthyroid	**97.36 ± 3.23**	96.57 ± 3.45	96.49 ± 2.96	95.44 ± 7.50	96.36 ± 3.14	95.39 ± 7.46	*96.59 ± 3.11*
Balance	87.12 ± 3.02	88.53 ± 2.75	91.15 ± 2.09	93.89 ± 3.04	93.93 ± 3.11	*94.13 ± 2.42*	**94.24 ± 2.52**
Cortijo(Sp)	95.63 ± 7.41	95.63 ± 7.19	85.50 ± 4.94	90.13 ± 10.10	90.63 ± 9.30	**97.50 ± 5.89**	*96.38 ± 6.23*
Cortijo(Su)	91.40 ± 11.81	**94.00 ± 10.44**	92.20 ± 11.68	93.40 ± 11.03	92.80 ± 10.06	*93.60 ± 10.20*	93.00 ± 9.59
Lymph.	83.85 ± 10.34	**84.97 ± 8.68**	82.13 ± 10.32	81.04 ± 10.33	81.44 ± 10.23	82.70 ± 8.42	*84.19 ± 10.96*
Anneal	99.18 ± 1.16	*99.34 ± 0.87*	94.56 ± 2.28	96.77 ± 4.48	97.51 ± 1.76	98.73 ± 4.90	**99.38 ± 0.84**
Glass	66.26 ± 8.54	63.88 ± 8.90	68.42 ± 9.44	68.72 ± 8.68	68.74 ± 10.32	*68.87 ± 9.72*	**70.02 ± 8.64**
Zoo	94.75 ± 6.77	**96.23 ± 5.94**	94.86 ± 6.39	95.25 ± 6.39	94.15 ± 7.37	*95.65 ± 6.04*	95.35 ± 5.35
Audiology	**85.11 ± 6.86**	81.90 ± 7.45	29.58 ± 8.06	41.95 ± 14.56	42.18 ± 14.16	81.81 ± 9.88	*84.53 ± 8.63*
$\overline{C_G}(\%)$	86.36	86.26	83.68	84.64	84.61	*87.45*	**87.93**
$\overline{R}$	4.50	3.70	4.58	4.95	5.18	*3.45*	**1.65**

The best result is in boldface and the second best result in italics.

and RBFEP methods. This is essential because the proposed methods are based on different combinations of the RBFs of an RBFNN obtained using the RBFEP method and both SimpleLogistic and MultiLogistic logistic regression algorithms. Consequently, the different proposals must achieve a better performance than all these methods in order to justify the additional complexity of these combinations.

In Table II, the mean and the SD of the correct classification rate in the generalization set (C_G) is shown for each dataset and a total of 100 executions. Based on the mean C_G, the ranking of each method in each dataset ($R = 1$ for the best performing method and $R = 7$ for the worst one) is obtained, and the mean accuracy ($\overline{C_G}$) and mean ranking ($\overline{R}$) are also included in Table II. From the analysis of the results, it can be concluded, from a purely descriptive point of view, that the SLIRBF method obtains the best result for 11 datasets, the MLogistic method yields the highest performance for 5 datasets, SLogistic for 2 datasets and, RBFEP and MLIRBF only for 1 dataset. Furthermore, the SLIRBF method obtains the best mean ranking ($\overline{R} = 1.65$), followed by the MLIRBF method ($\overline{R} = 3.45$), and reports the highest mean accuracy ($\overline{C_G} = 87.93\%$), followed also by MLIRBF ($\overline{C_G} = 87.45\%$). It can be observed that SLRBF never yields better results than SLIRBF. When comparing the MLIRBF and MLRBF methods, there are four cases where MLRBF outperforms MLIRBF [Glass(G2), Ionosphere, Australian, and Newthyroid], although the differences are really low and probably due to the stochastic nature of the methods (it is important to observe the SD of the results).

To quantify whether a statistical difference exists between any of these algorithms, a procedure for comparing multiple classifiers over multiple datasets is employed [56]. This procedure begins with the Friedman test [57], using the C_G ranking of the best models as the test variable. This test is a nonparametric equivalent to the repeated-measures ANOVA test, and in our case it is applied since a previous evaluation of the C_G values results in rejecting the normality and the equality of variance hypothesis. Given the ranks of the K classifiers averaged over the D datasets, R_k, $k = 1, \ldots, K$, and under the null hypothesis that all classifiers are equivalent, the Friedman statistic is

$$\chi_F^2 = \frac{12D}{K(K+1)}\left[\sum_k R_k^2 - \frac{K(K+1)^2}{4}\right]$$

which is χ^2-distributed with $(K-1)$ degrees of freedom. This statistic has been shown to be overly conservative, and some have recommended the use of a more sensitive one

$$F_F = \frac{(D-1)\chi_{\mathrm{F}}^2}{D(K-1)-\chi_{\mathrm{F}}^2}$$

which is F distributed with $(K-1)$ and $(K-1)(D-1)$ degrees of freedom. Applying this test to the average ranks at the bottom of Table II, the test shows that the effect of the method used for classification is statistically significant at a significance level of 5%, because the confidence interval is $C_0 = (0, \chi^2_{F(\alpha=0.05)} = 18.55)$ and $\chi_F^2 = 37.62 \notin C_0$, and also, for the more sensitive statistic, the confidence interval is

TABLE III
CRITICAL DIFFERENCE (CD) VALUES AND DIFFERENCES IN RANKINGS OF THE NEMENYI TEST

Nemenyi test							
	Method(j)						
Method(i)	SLogistic	MLogistic	RBFEP	MLRBF	SLRBF	MLIRBF	SLIRBF
SLogistic	–	0.80	0.08	0.45	0.68	1.05	$2.85^{+}_{\bullet}$
MLogistic	–	–	0.88	1.25	1.48	0.25	$2.05^{+}_{\bullet}$
RBFEP	–	–	–	0.37	0.60	1.13	$2.93^{+}_{\bullet}$
MLRBF	–	–	–	–	0.23	1.50	$3.30^{+}_{\bullet}$
SLRBF	–	–	–	–	–	1.73	$3.53^{+}_{\bullet}$
MLIRBF	–	–	–	–	–	–	1.80
$CD_{(\alpha=0.05)} = 2.01$ $CD_{(\alpha=0.1)} = 1.84$							

•: Statistically significant differences with $\alpha = 0.05$.
+: The difference is in favor of Method(j).

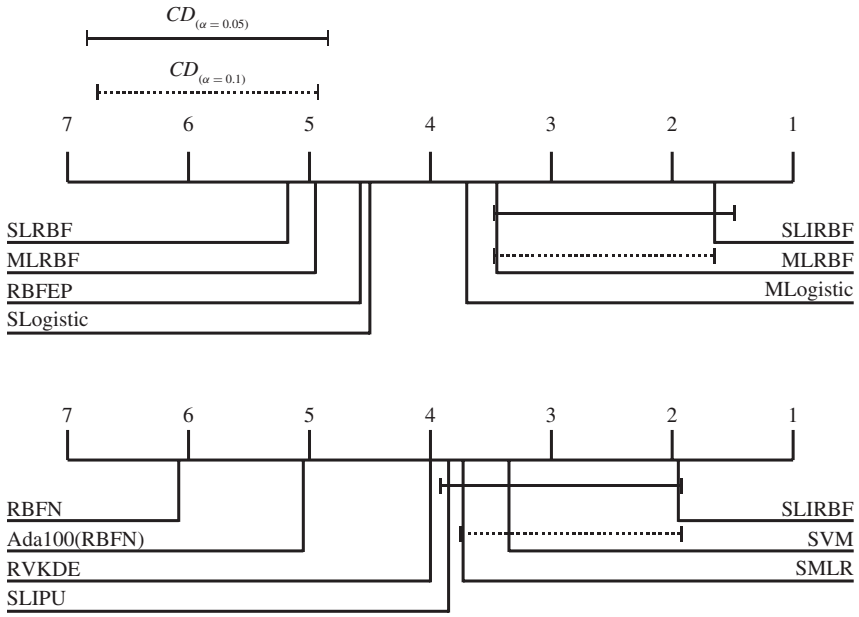

Fig. 4. Nemenyi CD diagrams comparing the different methods considered in this paper.

$C_0 = (0, F_{(\alpha=0.05)} = 2.18)$ and the F-distribution statistical value is $F_F = 8.68 \notin C_0$. Consequently, we reject the null hypothesis stating that all algorithms perform equally well in mean ranking and a *post hoc* test is warranted for further investigation.

On the basis of 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 classifiers is deemed significantly different if their mean ranks differ by at least the CD

$$CD = q\sqrt{\frac{K(K+1)}{6D}} \tag{9}$$

where q is derived from the studentized range statistic divided by $\sqrt{2}$ (the corresponding table for this statistic is available in [56]). The differences in rankings between the different algorithms and the results of the Nemenyi test for $\alpha = 0.1$ and $\alpha = 0.05$ can be seen in Table III, using the corresponding critical values (and also in the Nemenyi CD diagram of Fig. 4). By using this test, it can be seen that the SLIRBF method significantly outperforms all the other methods except MLIRBF for $\alpha = 0.05$, although the differences between MLIRBF and SLIRBF (1.80) are very close to the differences needed for assessing statistically significance at $\alpha = 0.1$ [$CD_{(\alpha=0.1)} = 1.84$]. Consequently, for this subset of datasets, SLIRBF obtains a significantly higher performance than the two methods from which it is derived, i.e., SLogistic and RBFEP, which justifies the proposal.

E. Comparison to Other Related Probabilistic Classifiers

This section presents a comparison of the best performing method of those proposed (SLIRBF). As previously stated, very competitive probabilistic classifiers have been selected along with others that share some characteristics with the different proposals.

Comparison is again performed using the correct classification rate or accuracy in the generalization set (C_G), the mean accuracy ($\overline{C_G}$), and the mean ranking ($\overline{R}$), and the results are included in Table IV. A descriptive analysis of the results leads to the following remarks: the SLIRBF method obtains the best results for 9 out of the 20 datasets, the second best results for

TABLE IV

COMPARISON OF THE PROPOSED METHOD TO OTHER PROBABILISTIC METHODS: MEAN AND SD OF THE ACCURACY RESULTS (C_G) FROM 100 EXECUTIONS, MEAN ACCURACY ($\overline{C_G}$), AND MEAN RANKING ($\overline{R}$)

	Method ($C_G(\%)$)						
	RBFN	Ada100(RBFN)	SVM	RVKDE	SMLR	SLIPU	SLIRBF
Dataset	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD
Hepatitis	82.01 ± 9.71	81.22 ± 8.41	82.71 ± 8.87	83.22 ± 7.53	**85.80 ± 9.13**	84.72 ± 6.49	*85.38 ± 6.60*
Glass(G2)	76.03 ± 11.03	*79.85 ± 8.40*	**80.05 ± 8.97**	79.43 ± 9.02	76.31 ± 10.34	73.51 ± 9.11	78.34 ± 9.63
Sonar	71.99 ± 10.44	*82.88 ± 8.23*	82.27 ± 9.52	**83.25 ± 7.64**	77.64 ± 8.40	75.71 ± 9.87	80.90 ± 8.23
Heart-c	81.92 ± 7.61	80.06 ± 6.97	83.58 ± 6.19	*83.67 ± 6.50*	83.48 ± 7.00	83.34 ± 4.38	**84.70 ± 4.51**
Ionosphere	91.23 ± 5.47	93.71 ± 4.13	93.77 ± 3.60	93.62 ± 3.49	**95.27 ± 3.53**	92.53 ± 4.35	*93.95 ± 4.11*
Vote	94.55 ± 3.56	94.97 ± 3.24	*95.75 ± 2.81*	94.51 ± 3.42	95.56 ± 3.11	95.42 ± 2.41	**96.23 ± 2.74**
Australian	75.54 ± 5.37	81.97 ± 4.76	**86.32 ± 4.40**	85.71 ± 3.66	85.68 ± 4.24	85.35 ± 3.80	*85.84 ± 3.56*
Breast-w	96.14 ± 2.01	95.87 ± 2.08	96.44 ± 2.14	96.42 ± 1.94	96.34 ± 1.89	*96.47 ± 2.15*	**96.61 ± 2.48**
German	70.28 ± 3.58	71.95 ± 4.20	75.12 ± 4.27	72.55 ± 3.67	**75.81 ± 3.77**	74.42 ± 3.81	*75.15 ± 3.73*
Post-Op.	66.78 ± 10.42	58.22 ± 12.35	*70.56 ± 5.99*	67.56 ± 9.30	64.22 ± 10.43	69.89 ± 6.94	**71.00 ± 5.67**
Iris	95.80 ± 4.80	94.73 ± 5.55	95.80 ± 4.70	95.67 ± 4.58	95.53 ± 5.19	*96.67 ± 4.29*	**96.80 ± 4.69**
Newthyroid	*96.79 ± 3.46*	96.17 ± 3.75	95.95 ± 3.79	94.97 ± 4.44	96.29 ± 3.72	**97.02 ± 2.51**	96.59 ± 3.11
Balance	85.85 ± 3.83	86.95 ± 3.04	**99.92 ± 0.48**	89.71 ± 1.84	96.96 ± 2.44	*97.70 ± 1.97*	94.24 ± 2.52
Cortijo(Sp)	82.38 ± 11.53	91.25 ± 9.97	90.13 ± 4.82	94.63 ± 6.94	85.63 ± 6.97	*96.25 ± 5.76*	**96.38 ± 6.23**
Cortijo(Su)	88.00 ± 12.06	89.00 ± 12.51	92.20 ± 9.94	92.60 ± 10.50	90.40 ± 12.22	*92.80 ± 10.06*	**93.00 ± 9.59**
Lymph.	76.27 ± 9.68	81.90 ± 9.93	83.02 ± 8.77	81.61 ± 8.65	**84.96 ± 8.56**	79.77 ± 10.16	*84.19 ± 10.96*
Anneal	90.75 ± 2.18	97.44 ± 2.28	98.42 ± 0.87	98.52 ± 1.40	98.42 ± 1.30	*99.00 ± 1.27*	**99.38 ± 0.84**
Glass	65.60 ± 10.80	**71.10 ± 9.43**	68.12 ± 9.93	69.84 ± 9.25	68.65 ± 8.17	67.33 ± 10.87	*70.02 ± 8.64*
Zoo	94.05 ± 6.62	95.05 ± 6.37	93.98 ± 6.68	**97.01 ± 5.01**	*96.41 ± 5.59*	93.84 ± 6.46	95.35 ± 5.35
Audiology	71.97 ± 8.53	76.60 ± 7.67	80.32 ± 7.00	67.49 ± 8.33	77.37 ± 8.98	*83.74 ± 9.17*	**84.53 ± 8.63**
$\overline{C_G}(\%)$	82.70	85.04	*87.22*	86.10	86.34	86.77	**87.93**
$\overline{R}$	6.08	5.05	*3.35*	4.00	3.73	3.85	**1.95**

The best result is in boldface and the second best result in italics.

TABLE V

CD VALUES AND DIFFERENCES IN RANKINGS OF THE NEMENYI TEST

Nemenyi test							
	Method(j)						
Method(i)	RBFN	Ada100(RBFN)	SVM	RVKDE	SMLR	SLIPU	SLIRBF
RBFN	–	1.03	$2.73^{+}_{\bullet}$	$2.08^{+}_{\bullet}$	$2.35^{+}_{\bullet}$	$2.23^{+}_{\bullet}$	$4.13^{+}_{\bullet}$
Ada100(RBFN)	–	–	1.70	1.05	1.32	1.20	$3.10^{+}_{\bullet}$
SVM	–	–	–	0.65	0.38	0.50	1.40
RVKDE	–	–	–	–	0.27	0.15	$2.05^{+}_{\bullet}$
SMLR	–	–	–	–	–	0.12	1.78
SLIPU	–	–	–	–	–	–	$1.90^{+}_{\circ}$
$CD_{(\alpha=0.05)} = 2.01,\ CD_{(\alpha=0.1)} = 1.84$							

$\bullet$, $\circ$: Statistically significant differences with $\alpha = 0.05$ ($\bullet$) and $\alpha = 0.1$ ($\circ$).

$+$: The difference is in favor of Method(j).

other 6 datasets, and the best mean accuracy ($\overline{C_G} = 87.93\%$); SVM obtains the best results for 3 datasets, the second best results for 2 datasets, and the second best mean accuracy ($\overline{C_G} = 87.22\%$), and SMLR achieves the best performance for 4 datasets and the second best performance for 1 dataset, resulting in the fourth best mean accuracy ($\overline{C_G} = 86.34\%$). These three methods result in the best mean rankings, the best one being obtained by SLIRBF ($\overline{R} = 1.95$), followed by SVM ($\overline{R} = 3.35$) and SMLR ($\overline{R} = 3.73$).

It is necessary again to ascertain whether there are significant differences in the mean ranking of C_G, so a procedure similar to that used in the previous subsection has been applied. The nonparametric Friedman test shows that the effect of the method used for classification is statistically significant at a significance level of 5%, as the confidence interval is the same $C_0 = (0, \chi^2_{F(\alpha=0.05)} = 18.55)$ and $\chi^2_F = 43.42 \notin C_0$, and also, for the more sensitive statistic, the confidence interval is $C_0 = (0, F_{F(\alpha=0.05)} = 2.18)$ and the F-distribution statistical value is $F_F = 10.77 \notin C_0$. Consequently, we reject the null hypothesis stating that all algorithms of this second comparison perform equally in mean ranking.

On the basis of this rejection, the Nemenyi *post hoc* test is used to compare all classifiers to each other. The differences in rankings between the different algorithms and the results of the Nemenyi test for $\alpha = 0.1$ and $\alpha = 0.05$ can be seen in Table V, using the corresponding critical values (and also in the Nemenyi CD diagram of Fig. 4). This test concludes that, when considering this subset of datasets, the SLIRBF

TABLE VI
BEST SLIRBF MODELS FOR CORTIJO(SP) AND CORTIJO(SU)

Dataset	SLIRBF predictor functions
Cortijo(Sp)	$f_1(\mathbf{x}, \theta_1) = -18.83 - 17.64\lambda^*_{575} - 1.80\lambda^*_{675} + 9.96\lambda^*_{700} + 19.43B_1(\mathbf{x}, \mathbf{w}_1)$
	$f_2(\mathbf{x}, \theta_2) = -16.53 - 10.58\lambda^*_{575} + 4.10\lambda^*_{675} + 9.02\lambda^*_{700} - 4.77\lambda^*_{725} + 4.71B_1(\mathbf{x}, \mathbf{w}_1) + 21.68B_3(\mathbf{x}, \mathbf{w}_3)$
	$f_3(\mathbf{x}, \theta_3) = 7.44 - 4.83\lambda^*_{575} + 9.02\lambda^*_{700} + 4.71B_1(\mathbf{x}, \mathbf{w}_1) - 30.17B_2(\mathbf{x}, \mathbf{w}_2)$
	$B_1(\mathbf{x}, \mathbf{w}_1) = \exp(-0.5 * (((\lambda^*_{725} - 1.18)^2)^{0.5}/(1.28)^2))$
	$B_2(\mathbf{x}, \mathbf{w}_2) = \exp(-0.5 * (((\lambda^*_{625} + 0.67)^2)^{0.5}/(0.93)^2))$
	$B_3(\mathbf{x}, \mathbf{w}_3) = \exp(-0.5 * (((\lambda^*_{625} + 0.39)^2 + (\lambda^*_{650} + 0.21)^2)^{0.5}/(1.13)^2))$
Cortijo(Su)	$f_1(\mathbf{x}, \theta_1) = 0.68 + 1.53\lambda^*_{675} + 1.28\lambda^*_{725}$
	$f_2(\mathbf{x}, \theta_2) = -1.19 + 1.53\lambda^*_{675} + 5.57B_1(\mathbf{x}, \mathbf{w}_1)$
	$f_3(\mathbf{x}, \theta_3) = -1.44 + 1.53\lambda^*_{675} + 4.30B_2(\mathbf{x}, \mathbf{w}_2)$
	$B_1(\mathbf{x}, \mathbf{w}_1) = \exp(-0.5 * (((\lambda^*_{600} + 0.72)^2)^{0.5}/(0.66)^2))$
	$B_2(\mathbf{x}, \mathbf{w}_2) = \exp(-0.5 * (((\lambda^*_{575} - 1.23)^2)^{0.5}/(0.87)^2))$

TABLE VII
COMPUTATION TIME REQUIRED BY THE DIFFERENT ALGORITHMS: MEAN OF THE TIME IN SECONDS (t) USED FOR TRAINING EACH CLASSIFIER FROM THE 100 EXECUTIONS CONSIDERED

	Method ($t(s)$)												
Dataset	SLogistic	MLogistic	RBFEP	MLRBF	SLRBF	MLIRBF	RBFN	Ada100 (RBFN)	SVM	RVKDE	SMLR	SLIPU	SLIRBF
Hepatitis	6.37	1.86	9.46	10.45	14.99	10.15	0.03	1.20	3.74	2.43	0.86	12.93	18.46
Glass(G2)	4.70	1.09	250.92	252.38	261.83	251.65	0.02	0.26	3.96	2.56	1.32	153.78	263.24
Sonar	18.81	9.97	1589.80	1591.44	1600.56	1592.61	0.08	4.29	8.27	3.24	2.90	1276.22	1615.42
Heart-c	14.95	5.13	55.21	56.51	64.96	56.88	0.05	1.72	10.46	4.73	7.35	33.80	74.37
Ionosphere	23.35	7.87	639.40	641.14	656.30	641.90	0.08	2.96	11.54	5.54	8.78	488.29	672.99
Vote	16.15	5.00	47.06	48.51	60.55	48.80	0.07	1.86	11.16	6.64	17.98	37.68	67.99
Australian	57.52	28.16	6697.97	6700.91	6739.42	6705.89	0.20	4.79	50.22	11.11	22.92	2210.11	6808.68
Breast-w	21.34	5.66	72.04	74.56	101.37	74.78	0.10	1.77	11.38	10.02	17.59	54.90	112.70
German	115.67	46.15	2685.01	2688.47	2732.86	2698.39	0.58	13.57	150.21	16.58	48.39	459.50	2854.98
Post-Op.	5.35	2.19	7.78	8.63	12.39	8.53	0.03	0.34	3.16	1.26	0.51	25.58	14.17
Iris	6.32	1.49	8.79	10.09	16.08	9.56	0.03	1.38	2.74	2.07	6.20	12.88	16.24
Newthyroid	9.60	2.40	150.62	152.56	163.52	152.12	0.08	3.37	2.64	2.94	8.47	65.48	165.00
Balance	27.41	6.08	820.02	824.29	892.93	824.37	0.09	0.44	21.36	8.52	102.39	335.17	908.27
Cortijo(Sp)	5.47	1.60	7.53	8.59	13.39	8.30	0.04	1.24	2.05	1.46	0.73	12.15	14.16
Cortijo(Su)	3.43	0.96	6.19	7.34	11.26	6.79	0.03	1.20	1.78	1.04	0.64	8.75	11.01
Lymph.	18.14	18.22	533.59	535.77	545.34	540.28	0.08	3.44	5.16	2.06	5.43	171.27	556.64
Anneal	336.00	40.99	2497.73	2512.54	2612.08	2558.48	34.67	34.67	88.24	12.90	122.27	1110.34	2962.74
Glass	20.84	10.94	1234.93	1239.46	1261.74	1241.14	1.25	3.91	5.51	3.25	6.82	560.21	1270.72
Zoo	12.77	12.63	410.52	414.12	423.29	418.88	0.06	3.11	3.18	1.26	2.64	121.04	428.00
Audiology	346.09	4579.99	570.72	595.78	655.62	8193.35	47.07	144.68	24.07	3.11	17.03	937.25	1144.91

method significantly outperforms RBFN, Ada100(RBFN), and RVKDE for $\alpha = 0.05$ and significantly outperforms SLIPU for $\alpha = 0.1$. Although the mean ranking of SLIRBF is better than that of SMLR and SVM, these differences are not significant. However, it can be observed in Table V that the differences in ranking obtained by SLIRBF take values near 4 for RBFN, while those obtained by SMLR and SVM are not higher than 2.73.

Therefore, we can conclude that the results obtained by SLIRBF make it a competitive method when compared to the probabilistic classifiers previously mentioned.

F. Analysis of the Best SLIRBF Models Obtained for the Real Agronomical Problems

One of the major advantages of the SLIRBF model is the reduced number of features and RBFs included in the final model. This can result in a better interpretability of the model, which is especially important when dealing with real problems. In this way, Table VI includes the best predictor functions of the SLIRBF models obtained for the Cortijo(Sp) and Cortijo(Su) real problems. As discussed in Section V-C, the datasets include seven wavelengths as input variables and the spectral signatures are to be classified into four classes (live cover crop, dead cover crop, olive tree, and bare soil). From these predictor functions, the probability that each pattern $\mathbf{x}$ having to belong to each class can be easily derived by using (2). It can be observed how these models present an optimized structure from different points of view, each of the RBFs uses a subset of the initial covariates (through the use of the RBFEP algorithm) and every predictor function does not include every RBF or initial covariate (through the use of the SLogistic maximum likelihood algorithm).

G. Analysis of the Computation Time of the Different Algorithms

Table VII includes the computation time required by the different algorithms, measured as the mean number of seconds from the 100 executions considered.

The Java platform we used was JDK 1.6.0, under an Ubuntu Server 64 bits environment. The hardware used in the experiments is a computer cluster with eight 2.00-GHz Intel Xeon E5405 CPUs (x86_64 architecture) and 8 GB of RAM per each CPU. It is important to remark that software implementations can affect these computation times.

1) "libsvm" (used for SVM experimentation) is written in C and includes several efficiency improvements.
2) WEKA, JCLEC, SMLR, and the developed RBFEP package are written in Java, making an intensive use of object-oriented programming. Sometimes, this can compromise the computational efficiency.

Regarding the running times of the different algorithms, the largest computer time in the LIRBF methods is employed to obtain the optimized RBFs because of the use of the RBFEP algorithm. However, from the analysis of the results, the application of the second and third steps of the proposed methodology (see Section V-D) results in a significant improvement. From the analysis of the results in Table VII, it is clear that all those methods involving evolutionary optimization demand a considerably higher computational cost. These differences are not so notable for Hepatitis, Post-Op., Iris, Cortijo(Sp), and Cortijo(Su), i.e., those datasets with a fewer instances (see Table I). However, our LIRBF models clearly benefit from the optimized structure learnt by the EA, which leads both to better generalization accuracy and better interpretability, the latter being especially useful for experts to obtain interesting conclusions with respect to the problem under consideration. This interpretability was analyzed in Section V-F.

VI. Conclusion

This paper combined three powerful techniques used in machine learning research: multilogistic regression, EAs, and RBFNNs. The approach carries out an adequate combination of the three elements to solve multiclass problems and follows the proposal of Hastie, Tibshirani, and Friedman of generalizing the linear logistic model using additive models of basis functions [2], [51]. Specifically, this new approach consists of a multilogistic regression model built on the combination of linear covariates and Gaussian RBFs for the predictor function. The process for obtaining the coefficients was carried out in three steps: 1) an EP algorithm aimed to produce a reduced number of RBF transformations with the simplest structure possible (i.e., trying to select the most important input variables for the estimation of these RBFs); 2) a transformation of the input space by adding the nonlinear transformations of the input variables given by the RBFs of the best individual in the final generation; and 3) a maximum likelihood optimization method. In this final step, two different multilogistic regression algorithms were applied, one that considered all initial and RBF covariates (MLIRBF) and another that incrementally constructed the model and applied cross validation, resulting in an automatically covariate selection (SLIRBF).

From the analysis of the results obtained, several conclusions can be drawn. First of all, the covariate selection process incorporated in the SLogistic algorithm of the SLIRBF and SLRBF methods is necessary in some datasets to avoid overfitting. Then, the results reported by the MLIRBF and SLIRBF methods (built on initial and RBF covariates) are better for almost all datasets than those reported by their equivalent standard multilogistic regression methods, i.e., MLogistic and SLogistic (built only on the initial covariates). Finally, SLIRBF is a competitive method, obtaining high accuracy values. A measure of statistical significance is used, which indicates that SLIRBF reaches the state of the art.

Acknowledgment

The authors would like to thank the Precision Farming and Remote Sensing Unit of the Department of Crop Protection, Institute of Sustainable Agriculture (CSIC), Spain, for providing the datasets corresponding to the real agronomical problem of Section V-C. They also wish to thank the anonymous reviewers for their several helpful and valuable comments and suggestions.

References

[1] A. K. Jain, R. P. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 4–37, Jan. 2000.

[2] T. J. Hastie and R. J. Tibshirani, "Nonparametric regression and classification. Part II: Nonparametric classification," in *From Statistics to Neural Networks: Theory and Pattern Recognition Applications* (Computer and System Sciences), vol. 136, V. Cherkassky, J. H. Friedman, and H. Wechsler, Eds. New York: Springer-Verlag, 1996, pp. 70–82.

[3] J. Friedman, "Multivariate adaptive regression splines (with discussion)," *Ann. Stat.*, vol. 19, no. 1, pp. 1–141, 1991.

[4] C. Hervás-Martínez and F. Martínez-Estudillo, "Logistic regression using covariates obtained by product-unit neural network models," *Pattern Recognit.*, vol. 40, no. 1, pp. 52–64, Jan. 2007.

[5] C. Hervás-Martínez, F. J. Martínez-Estudillo, and M. Carbonero-Ruz, "Multilogistic regression by means of evolutionary product-unit neural networks," *Neural Netw.*, vol. 21, no. 7, pp. 951–961, Sep. 2008.

[6] I. T. Nabney, "Efficient training of RBF networks for classification," *Int. J. Neural Syst.*, vol. 14, no. 3, pp. 201–208, Jun. 2004.

[7] B. Krishnapuram, L. Carin, M. A. Figueiredo, and A. J. Hartemink, "Sparse multinomial logistic regression: Fast algorithms and generalization bounds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 957–968, Jun. 2005.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics), 1st ed. New York: Springer-Verlag, 2006.

[9] A. Asuncion and D. Newman. (2007). *UCI Machine Learning Repository* [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[10] M. J. Orr, "Optimising the widths of radial basis functions," in *Proc. 5th Brazilian Symp. Neural Netw.*, Belo Horizonte, Brazil, Dec. 1998, pp. 26–29.

[11] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis functions," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[12] K. J. Hunt, D. Sbarbaro, R. Zbikowski, and P. J. Gawthrop, "Neural networks for control systems—A survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, Nov. 1992.

[13] J. B. Gomm and D. L. Yu, "Selecting radial basis function network centers with recursive orthogonal least squares training," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 306–314, Mar. 2000.

[14] Z. Uykan and C. Guzelis, "Input-output clustering for determining centers of radial basis function network," in *Proc. Eur. Conf. Circuit Theory Design*, vol. 2. 1997, pp. 435–439.

[15] O. Buchtala, M. Klimek, and B. Sick, "Evolutionary optimization of radial basis function classifiers for data mining applications," *IEEE Trans. Syst., Man, Cybern. B: Cybern.*, vol. 35, no. 5, pp. 928–947, Oct. 2005.

[16] 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 *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 3. Jul. 2004, pp. 2083–2088.

[17] Z. Q. Zhao and D. S. Huang, "A mended hybrid learning algorithm for radial basis function neural networks to improve generalization capability," *Appl. Math. Model.*, vol. 31, no. 7, pp. 1271–1281, Jul. 2007.

[18] S. Chen, Y. Wu, and B. L. Luk, "Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1239–1243, Sep. 1999.

[19] J. González, I. Rojas, J. Ortega, H. Pomares, F. J. Fernández, and A. F. Días, "Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1478–1495, Nov. 2003.

[20] B. A. Whitehead and T. D. Choate, "Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction," *IEEE Trans. Neural Netw.*, vol. 7, no. 4, pp. 869–880, Jul. 1996.

[21] B. A. Whitehead, "Genetic evolution of radial basis function coverage using orthogonal niches," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1525–1528, Nov. 1996.

[22] B. A. Whitehead and T. D. Choate, "Evolving space-filling curves to distribute radial basis functions over an input space," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 15–23, Jan. 1994.

[23] 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. New York: Wiley, 2004, ch. 10, pp. 317–362.

[24] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, 1st ed. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[25] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, 1st ed. Cambridge, MA: MIT Press, 2001.

[26] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. ACM Workshop Comput. Learn. Theory*, Pittsburgh, PA, 1992, pp. 144–152.

[27] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[28] T.-F. Wu, C.-J. Lin, and R. C. Weng, "Probability estimates for multi-class classification by pairwise coupling," *J. Mach. Learn. Res.*, vol. 5, pp. 975–1005, 2004.

[29] S. S. Keerthi, K. B. Duan, S. K. Shevade, and A. N. Poo, "A fast dual algorithm for kernel logistic regression," *Mach. Learn.*, vol. 61, nos. 1–3, pp. 151–165, Nov. 2005.

[30] N. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse Gaussian process methods: The informative vector machine," in *Advances in Neural Information Processing Systems*, vol. 15. Cambridge, MA: MIT Press, 2001, pp. 625–632.

[31] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, Jun. 2001.

[32] H. Chen, P. Tino, and X. Yao, "Probabilistic classification vector machines," *IEEE Trans. Neural Netw.*, vol. 20, no. 6, pp. 901–914, Jun. 2009.

[33] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multi-class support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.

[34] L. Narlikar and A. J. Hartemink, "Sequence features of DNA binding sites reveal structural class of associated transcription factor," *Bioinformatics*, vol. 22, no. 2, pp. 157–163, Jan. 2006.

[35] J. Liu, J. Chen, and J. Ye, "Large-scale sparse logistic regression," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Paris, France, 2009, pp. 547–556.

[36] N. Subrahmanya and Y. Shin, "Sparse multiple kernel learning for signal processing applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 788–798, May 2010.

[37] Y.-J. Oyang, S.-C. Hwang, Y.-Y. Ou, C.-Y. Chen, and Z.-W. Chen, "Data classification with radial basis function network based on a novel kernel density estimation algorithm," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 225–236, Jan. 2005.

[38] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques* (Data Management Systems), 2nd ed. San Mateo, CA: Morgan Kaufmann, 2005.

[39] S. le Cessie and J. van Houwelingen, "Ridge estimators in logistic regression," *Appl. Stat.*, vol. 41, no. 1, pp. 191–201, 1992.

[40] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. San Francisco, CA: Academic, 1982.

[41] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Ann. Stat.*, vol. 38, no. 2, pp. 337–374, 2000.

[42] N. Landwehr, M. Hall, and E. Frank, "Logistic model trees," *Mach. Learn.*, vol. 59, nos. 1–2, pp. 161–205, May 2005.

[43] P. J. Angeline, G. M. Sauders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 54–65, Jan. 1994.

[44] X. Yao, "Global optimization by evolutionary algorithms," in *Proc. 2nd Aizu Int. Symp. Parallel Algorithms/Architecutre Synthesis*, Aizu-Wakamatsu, Japan, 1997, pp. 282–291.

[45] A. C. Martínez-Estudillo, C. Hervás-Martínez, F. J. Martínez-Estudillo, and N. García-Pedrajas, "Hybridization of evolutionary algorithms and local search by means of a clustering method," *IEEE Trans. Syst., Man Cybern. B: Cybern.*, vol. 36, no. 3, pp. 534–545, Jun. 2006.

[46] F. J. Martínez-Estudillo, C. Hervás-Martínez, P. A. Gutiérrez, and A. C. Martínez-Estudillo, "Evolutionary product-unit neural networks classifiers," *Neurocomputing*, vol. 72, nos. 1–2, pp. 548–561, Dec. 2008.

[47] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. San Francisco, CA: Academic, 1999.

[48] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[49] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution*. Stuttgart, Germany: Frommann-Holzboog, 1973.

[50] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 148–156.

[51] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*. New York: Springer-Verlag, 2001.

[52] S. Ventura, C. Romero, A. Zafra, J. A. Delgado, and C. Hervas, "JCLEC: A Java framework for evolutionary computation," *Soft Comput.*, vol. 12, no. 4, pp. 381–392, Oct. 2008.

[53] C.-C. Chang and C.-J. Lin. (2001). *LIBSVM: A Library for Support Vector Machines* [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

[54] S. South, J. Qi, and D. P. Lusch, "Optimal classification methods for mapping agricultural tillage practices," *Remote Sens. Environ.*, vol. 91, no. 1, pp. 90–97, May 2004.

[55] J. Peña-Barragán, F. Léopez-Granados, M. Jurado-Expéosito, and L. García-Torres, "Spectral discrimination of ridolfia segetum and sunflower as affected by phenological stage," *Weed Res.*, vol. 46, no. 1, pp. 10–21, Feb. 2006.

[56] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.

[57] M. Friedman, "A comparison of alternative tests of significance for the problem of *m* rankings," *Ann. Math. Stat.*, vol. 11, no. 1, pp. 86–92, Mar. 1940.

Pedro Antonio Gutiérrez (M'08) was born in Córdoba, Spain, in 1982. He received the B.S. degree in computer science from the University of Sevillea, Sevillea, Spain, in 2006, and the Ph.D. degree in computer science and artificial intelligence from the University of Granada, Granada, Spain, in 2009.

He is currently an Assistant Professor in the Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba. His current research interests include neural networks and their applications, evolutionary computation, and hybrid algorithms.

César Hervás-Martínez (M'08) was born in Cuenca, Spain. He received the B.S. degree in statistics and operations research from the Universidad Complutense, Madrid, Spain, in 1978, and the Ph.D. degree in mathematics from the University of Seville, Seville, Spain, in 1986.

He is currently a Professor of Computer Science and Artificial Intelligence in the Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba, Spain, and an Associate Professor in the Department of Quantitative Methods, School of Economics, University of Córdoba. His current research interests include neural networks, evolutionary computation, and the modeling of natural systems.

Francisco J. Martínez-Estudillo (M'08) was born in Villacarrillo, Jaén, Spain. He received the B.S. degree in mathematics and the Ph.D. degree in specialization in differential geometry from the University of Granada, Granada, Spain, in 1987 and 1991, respectively.

He is currently a Professor in the Department of Management and Quantitative Methods, Faculty of Business Administration, University of Córdoba, Córdoba, Spain. From 1987 to 2002, he developed his research in non-Euclidean geometry, Lorentz spaces, and maximal surfaces. His current research interests include structure optimization of neural networks, evolutionary algorithms, and multiobjective optimization.