MultiLogistic Regression using Initial and Radial Basis Function covariates

Pedro Antonio Gutiérrez, César Hervás-Martínez, Francisco J. Martínez-Estudillo and Juan Carlos Fernández

Abstract—**This paper proposes a hybrid multilogistic model, named MultiLogistic Regression using Initial and Radial Basis Function covariates (MLRIRBF). The process for obtaining the coefficients is carried out in several steps. First, an Evolutionary Programming (EP) algorithm is applied, aimed to produce a RBF Neural Network (RBFNN) with a reduced number of RBF transformations and the simplest structure possible. Then, the input space is transformed by adding the nonlinear transformations of the input variables given by the RBFs of the best individual in the last generation. Finally, a maximum likelihood optimization method determines the coefficients associated with a multilogistic regression model built on this transformed input space. In this final step, two different multilogistic regression algorithms are applied, one that considers all initial and RBF covariates (MLRIRBF) and another one that incrementally constructs the model and applies cross-validation, resulting in an automatic covariate selection (MLRIRBF*). The methodology proposed is tested using six benchmark classification problems from well-known machine learning problems. The results are compared with the corresponding multilogistic regression methodologies applied over the initial input space, to the RBFNNs obtained by the EP algorithm (RBFEP) and to other competitive machine learning techniques. The MLRIRBF* models are found to be better than the corresponding multilogistic regression methodologies and the RBFEP method for almost all datasets, and obtain the highest mean accuracy rank when compared to the rest of methods in all datasets.**

I. Introduction

Multi-class pattern recognition is a problem of building a system that accurately maps an input feature space to an output space of more than two pattern classes. Whereas a two-class classification problem is well understood, multi-class classification is relatively less-investigated. In general, the extension from the two-class to the multi-class pattern classification problem is not trivial, and often leads to unexpected complexity or weaker performances. This paper presents a competitive study in multi-class neural learning which combines different statistical and soft computing elements such as multilogistic regression, Radial Basis Function Neural Networks (RBFNNs) and Evolutionary Algorithms (EAs).

In spite of the great number of techniques developed to solve classification problems, there is no optimum methodology or technique to solve specific problems. This point has encouraged the comparison and combination of different types of classification techniques [1]. Many of the works related to the hybridization of classifiers combine linear (or nonlinear) classifiers using the same functional typology, the combination of different functional structures being more unusual. A recently proposed combination of neural networks and logistic regression [2], [3] is based on the hybridization of a linear multilogistic regression model and a nonlinear Product-Unit Neural Network model for binary and multi-class classification problems. This methodology allows the generation of nonlinear classification surfaces and the identification of possible strong interactions that may exist between the covariates which define the classification problem.

In this paper, a new aspect of this methodology is presented, since we combine a linear model with a RBFNN nonlinear model and then we estimate the coefficients using logistic regression. RBFNNs, as an alternative to multilayer perceptrons, have been found to be very helpful to many engineering problems since: (1) they are universal approximators [4]; (2) they have more compact topology than other neural networks [5]; and (3) their learning speed is fast because of their locally tuned neurons [6]. The approach, named MultiLogistic Regression using Initial and Radial Basis Function covariates (MLRIRBF), consists of a multilogistic regression model built on the combination of the linear covariates and the RBFs of a RBFNN.

Although logistic regression is a simple and useful procedure, it poses problems when applied to real classification problems, where we cannot frequently make the stringent assumption of additive and purely linear effects of the covariates [7]. In this way, our technique overcomes these difficulties by augmenting the input vector with new RBF variables. From the opposite point of view, adding linear terms to a RBFNN yields simpler and easier to interpret models. Specifically, if a covariate only appears linearly in the final model, then the model is a traditional parametric model with respect to this covariate. Moreover, the linear terms reduce the variance associated with the overall modeling procedure and the likelihood of ending up with unnecessary RBFs in the final model.

Logistic regression models are usually fit by maximum likelihood, where the Newton-Raphson algorithm is the tradi-

P.A. Gutiérrez, C. Hervás-Martínez and J.C. Fernández are with the Department of Computer Science and Numerical Analysis, University of Córdoba, Campus de Rabanales, Albert Einstein building, 3rd floor, 14071 - Córdoba, Spain, e-mail: i02gupep@uco.es, chervas@uco.es, fernandezcaballero@gmail.com.

F.J. Martínez-Estudillo is with the Department of Management and Quantitative Methods, ETEA, Escritor Castilla Aguayo 4, 14004 - Córdoba, Spain, e-mail: fjmestud@etea.com.

This work has been partially subsidized by the TIN 2008-06681-C06-03 project of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P08-TIC-3745 project of the "Junta de Andalucía" (Spain). The research done by P.A. Gutiérrez and J.C. Fernández has been financed respectively by the FPU (grant reference AP-2006-01746) and FPI (grant reference BES-2006-12543) Predoctoral Programs (Spanish Ministry of Science and Innovation).

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

tional way to estimate the maximum a posteriori parameters. Usually, the algorithm converges since the log-likelihood is concave. However, in our approach, the nonlinearity of the RBFs with respect to the centres and radii implies that the corresponding Hessian matrix is generally indefinite and the likelihood could have local maxima. These reasons justify, in our opinion, the use of an alternative heuristic procedure as an EA to estimate the parameters of the model.

The estimation of the coefficients is carried out in several steps. In a first step, an Evolutionary Programming (EP) algorithm determines the number of RBFs in the model and their corresponding centres and radii. 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 covariates for the construction of these transformations. This step can be seen as a global search in the coefficients' model space. 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 last generation. The final model is linear in the set of variables formed by the RBFs and the initial covariates. Now, the Hessian matrix is definite and fitting proceeds with a maximum likelihood optimization method. In this final step, two different multilogistic regression algorithms are applied, one that considers all initial and RBF covariates (MLRIRBF) and another one that incrementally constructs the model and applies cross-validation, resulting in an automatic covariate selection (MLRIRBF*).

We evaluate the performance of our methodology on six datasets taken from the UCI repository [8]. The results are compared with the corresponding multilogistic regression methodologies applied over the initial input space, to the RBFNNs obtained by the EP algorithm (RBFEP) and to other competitive machine learning techniques. The MLRIRBF* models are found to be better than the corresponding multilogistic regression methodologies and the RBFEP method for almost all datasets, and obtain the highest mean accuracy rank when compared to the rest of methods in all datasets.

This paper is organized as follows: Section II is devoted to a description of the MultiLogistic Regression using Initial and Radial Basis Function covariates (MLRIRBF) model; Section III describes the MLRIRBF learning algorithm; Section IV explains the experiments carried out; and finally, Section V summarizes the conclusions of our work.

II. MLRIRBF MODEL

In the classification problem, measurements x_i, $i = 1, 2, ..., k$, are taken on a single individual (or object), and the individuals are to be classified into one of 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. A training sample $D = \{(\mathbf{x}_n, \mathbf{y}_n); n = 1, 2, ..., N\}$ is available, where $\mathbf{x}_n = (x_{1n}, ..., x_{kn})$ is the vector of measurements taking values in $\mathbf{\Omega} \subset \mathbb{R}^k$, and $\mathbf{y}_n$ is the class level of the n-th individual. The common technique of representing the class levels using a "1-of-J" encoding vector is adopted, $\mathbf{y} = \left(y^{(1)}, y^{(2)}, ..., y^{(J)}\right)$, such as $y^{(l)} = 1$ if $\mathbf{x}$ corresponds to an example belonging to class l and $y^{(l)} = 0$ otherwise. Based on the training sample, we wish to find a decision function $F : \mathbf{\Omega} \to \{1, 2, ..., J\}$ for classifying the individuals. In other words, F provides a partition, say $D_1, D_2, ..., D_J$, of $\mathbf{\Omega}$, where D_l corresponds to the l-th class, $l = 1, 2, ..., J$, and measurements belonging to D_l will be classified as coming from the l-th 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 corrected classified 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 in 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: $p\left(y^{(l)} = 1 \middle| \mathbf{x}\right) > 0$, $l = 1, 2, ..., J$, $\mathbf{x} \in \mathbf{\Omega}$, and sets the function:

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

where θ_l is the weight vector corresponding to class l, and $f_J(\mathbf{x}, \theta_J) = 0$. Under a multinomial logistic regression, the probability that $\mathbf{x}$ belongs to class l is then given by:

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

where $\theta = (\theta_1, \theta_2, ..., \theta_{J-1})$. For binary problems ($J = 2$), this is known as logistic regression (or soft-max in neural network literature).

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 $\mathbf{x}$:

$$F(\mathbf{x}) = \hat{l}, \text{where } \hat{l} = \arg\max_{l} f_l(\mathbf{x}, \hat{\theta}_l), \text{for } l = 1, ..., J.$$

On the other hand, due to the normalization condition we have:

$$\sum_{l=1}^{J} p\left(y^{(l)} = 1 \middle| \mathbf{x}, \theta\right) = 1,$$

and the probability for one of the classes (the last one, in our case) does not need be estimated. Observe that we have considered $f_J(\mathbf{x}, \theta_J) = 0$.

Our logistic regression model proposal is based on the combination of the standard linear model and a nonlinear term constructed with RBFs, which captures possible locations in the covariate space. The general expression of the

model is given by:

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

where $l = 1, 2, ..., J-1$, $\theta_l = (\boldsymbol{\alpha}^l, \boldsymbol{\beta}^l, \mathbf{W})$ is the vector of parameters for each discriminant function, $\boldsymbol{\alpha}^l = (\alpha_0^l, \alpha_1^l, ..., \alpha_k^l)$ and $\boldsymbol{\beta}^l = (\beta_1^l, ..., \beta_m^l)$ are the coefficients of the multilogistic regression model and $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_m)$ are the parameters of the RBF nonlinear transformations, where $\mathbf{w}_j = (\mathbf{c}_j, r_j)$, $\mathbf{c}_j = (c_{j1}, c_{j2}, ..., c_{jk})$ is the centre or average of the j-th Gaussian RBF transformation, $c_{ji} \in \mathbb{R}$ and r_j is the corresponding radius or standard deviation.

III. MLRIRBF LEARNING ALGORITHM

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

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

where $f_l(\mathbf{x}, \theta_l)$ corresponds to the MLRIRBF model defined in (2).

The error surface associated with the model is very convoluted with numerous local optima. The nonlinearity of the model with respect to the parameters $\mathbf{c}_j$ and r_j, 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 hidden nodes in the RBFNN) is unknown. Thus, the estimation of the vector parameter $\hat{\theta}$ is carried out by means of a hybrid procedure described below.

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

The model is now linear in these new variables and the initial covariates. The remaining coefficient vector $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are calculated by the *maximum likelihood* optimization method: choose the parameters that maximize the probability of the observed data points. For the multilogistic regression model, there are no closed-form solutions for these estimates. Instead, numeric optimization algorithms that approach the maximum likelihood solution iteratively and reach it in the limit have to be used. In the next subsection, the algorithms for obtaining this maximum likelihood solution are described. Then, the different steps of the MLRIRBF 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 in the WEKA machine learning workbench [9]:

1) MultiLogistic: MultiLogistic is an algorithm for building a multinomial logistic regression model with a ridge estimator to guard against overfitting by penalizing large coefficients, based on work of Le Cessie and Van Houwelingen [10].

In order to find the coefficient matrix θ for which $L(\theta)$ is minimized, a Quasi-Newton Method is used. Specifically, the method used is the active-sets' method with Broyden-Fletcher-Goldfarb-Shanno (BFGS) update [11].

2) SimpleLogistic: This algorithm builds multinomial logistic regression models fitting them using the LogitBoost algorithm [12], 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 described above.

SimpleLogistic algorithm is based on applying LogitBoost with simple regression functions and determining the optimum number of iterations by a five fold cross-validation: the data is equally splitted five times into training and test, LogitBoost is run on every training set up to a maximum number of iterations (500) and the classification error on the respective test set is logged. Afterwards, LogitBoost is run again on all data using the number of iterations that gave the smallest error on the test set averaged over the five folds. Further details about the algorithm can be found in [13].

B. Estimation of the model coefficients

In this subsection we describe the steps of MLRIRBF learning algorithm in detail. The process is structured in three steps.

Step 1. We apply an EP algorithm to find the basis functions:

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

corresponding to the nonlinear part of $f(\mathbf{x}, \theta)$. We have to determine the number of basis functions m and the weight vector $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_m)$. To apply evolutionary neural network 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 nodes; and an output layer with one node for each class minus one.

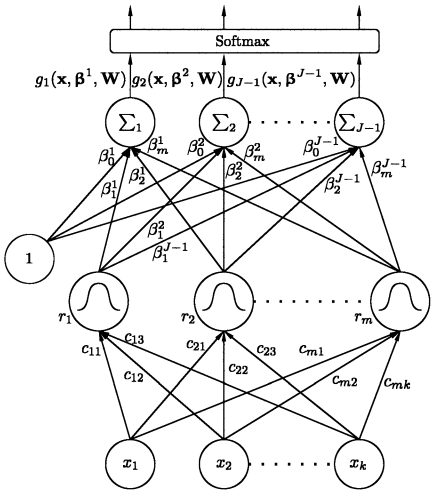

Fig. 1. Structure of Radial Basis Function Neural Networks: an input layer with k input variables, a hidden layer with m RBFs and an output layer with J nodes

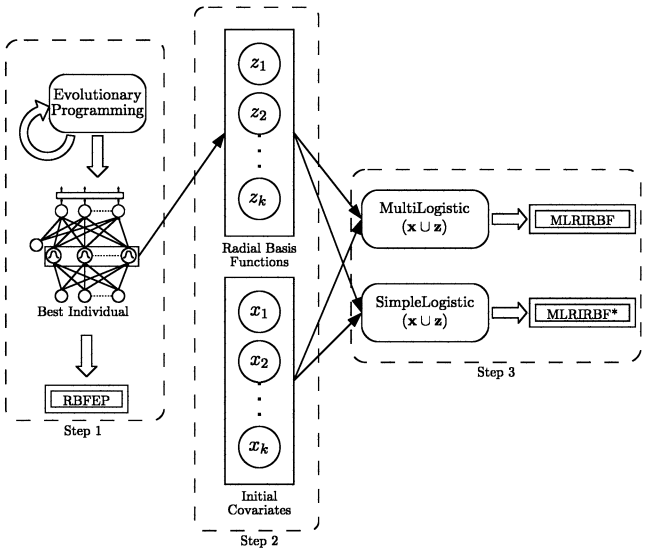

Fig. 2. Different steps of the proposed methodologies

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 j-th 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 $\mathbf{c}_j = (c_{j1}, ..., c_{jk})$ and c_{ji} is the weight of the connection between the i-th input node and the j-th hidden node. The activation function of the l-th output node is given by:

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

where β_j^l is the weight of the connection between the j-th hidden node and the l-th output node and β_0^l is the bias of l-th the output node. The transfer function of all output nodes is the identity function.

The weight vector $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_m)$ is estimated by means of an evolutionary neural network algorithm (detailed in Section III-C) that optimizes the error function given by the negative log-likelihood for N observations associated with the RBFNN model:

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

Although in this step the evolutionary process obtains a concrete value for the $\boldsymbol{\beta}$ vector, we only consider the estimated weight vector $\hat{\mathbf{W}} = (\hat{\mathbf{w}}_1, \hat{\mathbf{w}}_2, ..., \hat{\mathbf{w}}_m)$, which builds the basis functions. The values for the $\boldsymbol{\beta}$ vector will be determined in step 3 together with those of the $\boldsymbol{\alpha}$ coefficient vector.

Step 2. We consider the following transformation of the input space by including the nonlinear basis functions obtained by the EP algorithm in step 1:

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

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

Step 3. In the third step, we minimize the negative log-likelihood function for N observations:

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

where $\mathbf{x}_n = (1, x_{1n}, ..., x_{kn})$ and $\mathbf{z}_n = (z_{1n}, ..., z_{kn})$. Now, the Hessian matrix of the negative log-likelihood in the new variables $x_1, x_2, ..., x_k, z_1, ..., z_m$ is semi-definite positive. The estimated coefficient vector $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\mathbf{W}})$ determines the model:

$$f_l(\mathbf{x}, \hat{\boldsymbol{\theta}}_l) = \hat{\alpha}_0^l + \sum_{i=1}^{k} \hat{\alpha}_i^l x_i + \sum_{j=1}^{m} \hat{\beta}_j^l \exp\left(-\frac{\|\mathbf{x} - \hat{\mathbf{c}}_j\|^2}{\hat{r}_j^2}\right),$$

where $l = 1, 2, ..., J-1$. In this final step, both algorithms presented in subsection III-A have been used for obtaining the parameter matrix θ. This results in two different models, one with all $x_1, x_2, ..., x_k, z_1, ..., z_m$ covariates present in the model (*MultiLogistic* algorithm) and the other with only those variables selected by the *SimpleLogistic* algorithm (see subsection III-A). These two approaches will be called "MLRIRBF" and "MLRIRBF*", respectively.

Finally, in Fig. 2, a schematic view of the steps of the methodology and the different variants is presented. The variants are represented in a double squared box. The best individual obtained in the step 1 by the EP algorithm is also evaluated and called "RBFEP" variant.

C. Evolutionary acquisition of the RBF nonlinear transformations

In this section, the evolutionary algorithm used for obtaining the RBF nonlinear transformations is presented. 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 covariates for the construction of these transformations.

Among the different paradigms of Evolutionary Computation, we have chosen Evolutionary Programming (EP) due to the fact that we are evolving artificial neural networks. The population-based evolutionary algorithm for architectural design and the estimation of the real coefficients have points in common with other evolutionary algorithms in the bibliography [14], [15], [16]. The search begins with an initial population, and, in each iteration, the population is updated using a population-update algorithm. The population is subject to the operations of replication and mutation. Crossover is not used due to its potential disadvantages in evolving artificial networks [14]. 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 neural networks are represented using an object-oriented approach and the algorithm deals directly with the ANN phenotype. Each connection is specified by a binary value indicating if the connection exists, and a real value representing its weight. As the crossover is not considered, this object-oriented representation does not assumes a fixed order between the different hidden nodes.

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

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

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

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

3) Initialization of the population: The initial population is generated trying to obtain RBFNNs with the maximum possible fitness. First, 5.000 random RBFNNs are generated, where the number of hidden nodes 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 in the same random subspace of the input variables. 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 and the output layer.

The obtained individuals are evaluated using the fitness function. The initial population is finally obtained by selecting the best 500 RBFNNs and then improving them by applying the standard k-means clustering algorithm [17], using the randomly generated centres as the initial centroids and a maximum number of iterations of 100.

4) Parametric Mutation: Parametric mutation is accomplished for each coefficient $w \in \theta$ of the model with Gaussian noise, $w(t+1) = w(t) + \xi(t)$, where $\xi(t) \in N(0, \alpha(t))$ represents a one-dimensional 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 [18] is applied to accept or reject the modifications in each node. Thus, if ΔA is the difference in the fitness function before and after the random step, the criterion is: 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 [19]. The adaptation tries to avoid being trapped in local minima and to speed up the evolutionary process when the searching conditions are suitable.

5) Structural Mutation: Structural mutation 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. The node mutations are performed as follows:

- Node addition. One or more nodes are added to the hidden layer. The connections with the output nodes are chosen randomly and have a random value in the interval $[-I, I]$. The connections from the input layer are chosen randomly and its values are also random values in the interval $[-O, O]$.
- Node deletion. One or more nodes, together with their connections, are randomly selected and deleted.
- Node fusion. Two randomly selected nodes, a and b, are replaced by a new node c, which is a combination of the two. The connections common to both nodes are kept, with a weight given by:

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

  Those connections not shared by the nodes are inherited by c with probability 0.5 and their weight is unchanged.

The number of hidden 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 hidden nodes specified as parameters.

The connection structural mutations are performed as follows:

- 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 neuron from each layer is selected randomly, and then the connection is added with a random weight. A similar procedure is performed from the hidden to the output layer.
- Connection deletion. In the same way, connection deletion mutation is first performed in the hidden layer and then in the output layer, choosing randomly the origin neuron from the previous layer and the target neuron from the mutated layer.

We apply connection mutations sequentially for each mutated neural net, first, adding (or deleting) $1 + u[\Delta_{\mathrm{o}} n_{\mathrm{o}}]$ connections from the hidden layer to the output layer and then, adding (or deleting) $1 + u[\Delta_{\mathrm{h}} n_{\mathrm{h}}]$ connections from the input layer to the hidden layer, u being a random uniform variable in the interval $[0, 1]$, Δ_{o} and Δ_{h} previously defined ratios of number of connections in the hidden and the output layer, and n_{o} and n_{h} the current number of connections in the output and the hidden layer.

Parsimony is also encouraged in evolved networks by attempting the five structural mutations sequentially, where node or connection deletion and node fusion is 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 evolutionary algorithm except $M_{\min}$ and $M_{\max}$ have the same values in all the problems analyzed below. We have done a simple linear rescaling of the input variables in the interval $[-2, 2]$, X_i^* being the transformed variables. The centres 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 $\alpha(0) = 0.5$. The number of nodes that can be added or removed in a structural mutation is within the $[1, 2]$ interval. The ratio of the number of connections to add or delete in the hidden and the output layer during structural mutations is $\Delta_{\mathrm{o}} = 0.05$ and $\Delta_{\mathrm{h}} = 0.3$.

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

IV. Experiments

The proposed methodologies (MLRIRBF and MLRIRBF*) are applied to six datasets taken from the UCI repository [8], to test its overall performance when compared to other methods. The selected methods include:

- Multi-logistic regression methodologies, including the SimpleLogistic (SLogistic) and MultiLogistic (MLogistic) algorithms applied over the initial input space, which are explained in subsection III-A. As our models are logistic regression models, it is necessary to compare its performance to standard logistic regression algorithms.
- 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 its performance to the original RBFEP method.
- High performance machine learning methodologies: Logistic Model Trees (LMT), the C4.5 classification tree inducer, the Naive Bayes Tree learning algorithm (NBTree) and the Support Vector Machines (SVM) method. These are the best performing methods from those presented in [13], since a wide variety of algorithms with very competitive performance are gathered in this work. We have also included the $C-$SVM algorithm with RBF kernels. The description and some previous results of these methods can be found in [13] and [20].

The selected datasets present different numbers of instances, features and classes (see Table I). The minimum and maximum number of hidden nodes have been obtained as the best result of a preliminary experimental design, considering a small, medium and high value: $[M_{\min}, M_{\max}] \in \{[1, 3], [3, 6], [10, 12]\}$. This value is also included in Table I.

For the RBFEP, MLRIRBF and MLRIRBF* methods, the experimental design was conducted using a 10-fold cross-validation procedure, with 10 repetitions per each fold. For the other methods, the results have been obtained performing 10 times a 10-fold cross validation, because all are deterministic methods, i.e. they are not based in random values and return the same result for each execution. The results for all datasets have been taken from the paper of Landwehr et al. [13], except for the SVM method and the Post-Op. and Newthyroid datasets not included in Landwehr's work.

The RBFEP algorithm was implemented in JAVA using the Evolutionary Computation framework JCLEC [21] (http://jclec.sourceforge.net). For the MLRIRBF and MLRIRBF* methods, we slightly modified the RBFEP algorithm, applying the SLogistic and MLogistic algorithms from WEKA [9]. We also used "libsvm" [22] for obtaining the results of the SVM method and WEKA for obtaining the results of the Post-Op. and Newthyroid datasets.

The performance of each method has been evaluated using the correct classification rate or C in the generalization set [C_{G}, see (1)]. In Table II, the mean and the standard deviation

TABLE I

CHARACTERISTICS OF THE SIX DATASETS USED FOR THE EXPERIMENTS: NUMBER OF INSTANCES (Size), NUMBER OF REAL (R), BINARY (B) AND NOMINAL (N) INPUT VARIABLES, TOTAL NUMBER OF INPUTS (#In.), NUMBER OF CLASSES (#Out.), PER-CLASS DISTRIBUTION OF THE INSTANCES (Distribution) AND MINIMUM AND MAXIMUM NUMBER OF HIDDEN NODES USED FOR EACH DATASET ($[M_{\min}, M_{\max}]$)

Dataset	Size	R	B	N	#In.	#Out.	Distribution	$[M_{\min}, M_{\max}]$
Heart-c	302	6	3	4	26	2	(164, 138)	[1, 3]
Ionosphere	351	33	1	–	34	2	(126, 225)	[10, 12]
Post-Op.	90	0	1	6	20	3	(2, 24, 64)	[1, 3]
Newthyroid	215	5	–	–	5	3	(150, 35, 30)	[3, 6]
Balance	625	4	–	–	4	3	(288, 49, 288)	[3, 6]
Lymph.	148	3	9	6	38	4	(2, 81, 61, 4)	[3, 6]

All nominal variables are transformed to binary variables.

Lymph.: Lymphography; Post-Op.: Post-Operative.

TABLE II

MEAN AND STANDARD DEVIATION OF THE ACCURACY RESULTS (C_{G}) FROM 100 EXECUTIONS OF A 10-FOLD CROSS VALIDATION, MEAN ACCURACY ($\overline{C_{\mathrm{G}}}$) AND MEAN RANKING ($\overline{R}$) USING THE DIFFERENT METHODS PROPOSED

	$C_{\mathrm{G}}(\%)$							
	Heart-c	Ionoshere	Post-Op.	Newthyroid	Balance	Lymph.		
Method	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD	$\overline{C_{\mathrm{G}}}(\%)$	$\overline{R}$
SLogistic	83.30 ± 6.35	87.78 ± 4.99	70.44 ± 6.74	*96.74 ± 3.85*	88.74 ± 2.91	**84.37 ± 9.97**	85.23	4.50
MLogistic	83.70 ± 6.64	87.72 ± 5.57	59.67 ± 10.55	96.20 ± 4.86	89.44 ± 3.29	77.58 ± 10.59	82.38	6.33
LMT	83.51 ± 6.67	92.99 ± 4.13	70.11 ± 7.01	*96.74 ± 3.85*	89.71 ± 2.68	84.10 ± 10.00	*86.19*	3.92
C4.5	76.94 ± 6.59	89.74 ± 4.38	69.67 ± 7.04	92.62 ± 5.60	77.82 ± 3.42	75.84 ± 11.05	80.44	7.00
NBTree	80.60 ± 6.29	89.49 ± 5.12	66.67 ± 10.36	92.60 ± 5.57	75.83 ± 5.32	80.89 ± 8.77	81.01	7.00
SVM	56.75 ± 3.71	93.00 ± 4.28	70.44 ± 5.74	75.43 ± 4.57	89.98 ± 1.90	80.31 ± 8.44	77.65	5.75
RBFEP	*84.14 ± 4.79*	*93.83 ± 4.43*	**70.67 ± 5.81**	95.48 ± 3.13	91.15 ± 1.31	81.27 ± 10.52	86.09	*3.08*
MLRIRBF	82.81 ± 5.03	89.28 ± 4.92	62.44 ± 13.29	95.90 ± 2.96	**94.83 ± 2.69**	75.82 ± 12.07	83.52	6.00
MLRIRBF*	**84.77 ± 5.09**	**94.06 ± 4.18**	**70.67 ± 6.42**	**96.92 ± 2.57**	*94.34 ± 2.69*	*84.30 ± 10.23*	**87.51**	**1.42**

The best result for each column is in bold face and the second best result in italic.

of this C_{G} is shown for each dataset and a total of 100 executions. The ranking of each method and each dataset ($R = 1$ for the best performing method and $R = 9$ for the worst one) is obtained and the mean accuracy ($\overline{C_{\mathrm{G}}}$) 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 MLRIRBF* method obtains the best result in four out of the six datasets analyzed, and the second best result in the other two remaining datasets. Furthermore, the MLRIRBF* method obtains the best mean ranking ($\overline{R} = 1.42$) followed by the RBFEP method ($\overline{R} = 3.08$) and reports the highest mean accuracy ($\overline{C_{\mathrm{G}}} = 87.51\%$) followed by the LMT method ($\overline{C_{\mathrm{G}}} = 86.19\%$). The results obtained by the MLRIRBF* are higher than those obtained by MLRIRBF, which confirms the necessity of selecting the most important covariates (initial covariates or RBFs) in the final model in order to avoid overfitting (see Heart-c, Ionosphere, Post-Op. and Lymph. datasets).

To determine the statistical significance of the rank differences observed for each method in the different datasets, we have carried out a non-parametric Friedman test [23] with the ranking of C_{G} of the best models as the test variable (since a previous evaluation of the C_{G} values results in rejecting the normality and the equality of variances' hypothesis). The test shows that the effect of the methodology used for classification is statistically significant at a significance level of 5%, as the confidence interval is $C_0 = (0, F_{0.05} = 2.18)$ and the F-distribution statistical value is $F^* = 4.76 \notin C_0$. Consequently, we reject the null-hypothesis stating that all algorithms perform equally in mean ranking. On the basis of this rejection, a post-hoc non-parametric Bonferroni-Dunn test is applied [24], [25] with MLRIRBF* as the control algorithm. The results of the Bonferroni-Dunn test for $\alpha = 0.1$ and $\alpha = 0.05$ can be seen in Table III using the corresponding critical values for the two-tailed Bonferroni-Dunn test.

The methodology MLRIRBF* obtains a significant higher ranking of C_{G} when compared to MLogistic, C4.5, NBTree, SVM and MLRIRBF methodologies, although the differences in rank are not significant when compared to the other three methodologies.

V. CONCLUSIONS

This paper combines a linear model with a nonlinear model (the RBFNNs), the final coefficients being estimated using multilogistic regression. Specifically, this new approach consists of a multilogistic regression model built on the combination of linear covariates and Gaussian RBFs. The process for obtaining the coefficients is carried out in several

TABLE III
CRITICAL DIFFERENCE (CD) VALUES, MEAN RANKING AND DIFFERENCES OF RANKINGS OF THE BONFERRONI-DUNN TEST, USING MLRIRBF* AS THE CONTROL METHOD

Mean Ranking	Difference with $\overline{R}_{MLRIRBF*}$
$\overline{R}_{\text{MLRIRBF*}} = 1.42$	–
$\overline{R}_{\text{SLogistic}} = 4.50$	$\lvert\overline{R}_{\text{MLRIRBF*}} - \overline{R}_{\text{SLogistic}}\rvert = 3.08$
$\overline{R}_{\text{MLogistic}} = 6.33$	$\lvert\overline{R}_{\text{MLRIRBF*}} - \overline{R}_{\text{MLogistic}}\rvert = 4.92(*)$
$\overline{R}_{\text{LMT}} = 3.92$	$\lvert\overline{R}_{\text{MLRIRBF*}} - \overline{R}_{\text{LMT}}\rvert = 2.50$
$\overline{R}_{\text{C4.5}} = 7.00$	$\lvert\overline{R}_{\text{MLRIRBF*}} - \overline{R}_{\text{C4.5}}\rvert = 5.58(*)$
$\overline{R}_{\text{NBTree}} = 7.00$	$\lvert\overline{R}_{\text{MLRIRBF*}} - \overline{R}_{\text{NBTree}}\rvert = 5.58(*)$
$\overline{R}_{\text{SVM}} = 5.75$	$\lvert\overline{R}_{\text{MLRIRBF*}} - \overline{R}_{\text{SVM}}\rvert = 4.33(*)$
$\overline{R}_{\text{RBFEP}} = 3.08$	$\lvert\overline{R}_{\text{MLRIRBF*}} - \overline{R}_{\text{RBFEP}}\rvert = 1.67$
$\overline{R}_{\text{MLRIRBF}} = 6.00$	$\lvert\overline{R}_{\text{MLRIRBF*}} - \overline{R}_{\text{MLRIRBF}}\rvert = 4.58(*)$
	$CD = 3.95$ ($\alpha = 0.1$)
	$CD = 4.31$ ($\alpha = 0.05$)

($*$): statistically significant differences.

steps: (1) an EP algorithm aimed to produce a reduced number of RBF transformations with the simplest structure possible, (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 last generation and (3) a maximum likelihood optimization method. In this final step, two different multilogistic regression algorithms are applied, one that considers all initial and RBF covariates (MLRIRBF) and another one that incrementally constructs the model and applies cross-validation, resulting in an automatically covariate selection (MLRIRBF*).

From the analysis of the results obtained, several conclusions can be drawn. First of all, the covariate selection process incorporated in the MLRIRBF* is necessary in some datasets for avoiding overfitting. Then, the results reported by the MLRIRBF and MLRIRBF* methods (built on initial and RBF covariates) are better for almost all datasets than those reported by their equivalent multilogistic regression methods, MLogistic and SLogistic (built only on the initial covariates). Finally, the MLRIRBF* is a very competitive method, obtaining high accuracy values when compared to other recent machine learning methods as LMT or SVM. It is found to be better than pure SLogistic and RBFEP method in almost all datasets, and obtains the significantly highest mean rank of C_{G} when compared to MLogistic, C4.5, NBTree, SVM and MLRIRBF methodologies in all datasets.

Granted that the computational requirements for training MLRIRBF* models were nearly insignificant once the RBFEP models are built, the higher generalization accuracy obtained by the proposed models justify the use of the second and the third steps of this methodology. However, the computational cost of these models is in general higher than the requirements of the rest of methodologies evaluated.

As far as we are concerned, the methodology we are proposing in this work has not been previously used. The results obtained encourage us to perform a more extended experimental design, considering a higher number of datasets and problems with more classes. Moreover, the method could be adapted to on-line and incremental learning by using more complex models and algorithms similar to DENFIS [26].

REFERENCES

[1] R. L. Major and C. T. Ragsdale, "Aggregating expert predictions in a networked environment," *Computers & Operations Research*, vol. 28, no. 12, pp. 1231–1244, 2001.

[2] C. Hervas-Martinez and F. Martinez-Estudillo, "Logistic regression using covariates obtained by product-unit neural network models," *Pattern Recognition*, vol. 40, no. 1, pp. 52–64, 2007.

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

[4] J. Park and I. W. Sandberg, "Universal approximation using radial basis function networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, 1991.

[5] S. Lee and R. Kil, "A gaussian potential function network with hierarchical self-organising learning," *Neural Networks*, vol. 4, no. 2, pp. 207–224, 1991.

[6] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, no. 2, pp. 281–294, 1989.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, August 2006.

[8] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

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

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

[11] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. Academic Press, 1982.

[12] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *The Annals of Statistics*, vol. 38, no. 2, pp. 337–374, 2000.

[13] N. Landwehr, M. Hall, and E. Frank, "Logistic model trees," *Machine Learning*, vol. 59, no. 1-2, pp. 161–205, 2005.

[14] P. J. Angeline, G. M. Sauders, and J. B. Pollack, "An evolutionary algorithm that constructs recurren neural networks," *IEEE Transactions on Neural Networks*, vol. 5, pp. 54–65, 1994.

[15] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694–713, 1997.

[16] F. J. Martínez-Estudillo, C. Hervás-Martínez, P. A. Gutiérrez, and A. C. Martínez-Estudillo, "Evolutionary product-unit neural networks classifiers," *Neurocomputing*, vol. 72, no. 1-2, pp. 548–561, 2008.

[17] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. Academic Press, 1999.

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

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

[20] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*. Springer, August 2001.

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

[22] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[23] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[24] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, pp. 52–56, 1961.

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

[26] N. Kasabov and Q. Song, "Denfis: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 144–154, 2001.