Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 52–64
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

Logistic regression using covariates obtained by product-unit neural network models

César Hervás-Martínez[a], Francisco Martínez-Estudillo[b,*]

[a]*Department of Computing and Numerical Analysis, University of Córdoba, Spain*
[b]*Department of Management and Quantitative Methods, ETEA, Spain*

Received 10 November 2005; accepted 1 June 2006

Abstract

We propose a logistic regression method based on the hybridation of a linear model and product-unit neural network models for binary classification. In a first step we use an evolutionary algorithm to determine the basic structure of the product-unit model and afterwards we apply logistic regression in the new space of the derived features. This hybrid model has been applied to seven benchmark data sets and a new microbiological problem. The hybrid model outperforms the linear part and the nonlinear part obtaining a good compromise between them and they perform well compared to several other learning classification techniques. We obtain a binary classifier with very promising results in terms of classification accuracy and the complexity of the classifier.

Keywords: Logistic regression; Product-unit neural network; Classification

1. Introduction

There are many fields of study such as medicine, microbiology and others, where it is very important to predict a binary response variable, or equivalently the probability of an event's occurrence in terms of the values of a set of explicative variables related to it. Therefore, in binary supervised learning problems, the goal is to learn how to distinguish between examples from two classes (herein labeled as $y = 0$ and $y = 1$) on the basis of k observed predictor variables (also known as features or covariates) $x_1, x_2, \ldots, x_k$. The logistic regression (LR) model has been widely used in statistics for many years and has recently been the object of extensive study in the machine learning community. This traditional statistical tool arises from the desire to model the posterior probabilities of the class level given its observation via linear functions in the predictor variables. In this way, the LR model admirably serves the purpose of predicting a binary response variable and it is the most used in these cases as we can see, for example, in [1].

The LR is a simple and useful procedure, although it poses problems when is applied to a real-problem of classification, where frequently we cannot make the stringent assumption of additive and purely linear effects of the covariates.

A traditional technique to overcome these difficulties is to augment/replace the vector of inputs with additional variables, basis functions, which are transformations of the input variables and then to use linear models in this new space of derived input features. The beauty of this method is that once the basis functions have been determined, the models are linear in these new variables and the fitting is a standard procedure. Methods like sigmoidal feed-forward neural networks [2], projection pursuit learning [3], generalized additive models [4] and multivariate adaptive splines (MARS) [5] can be seen as different basis function. The major drawback of these approaches is to state the number and the typology of the corresponding basis functions.

The simplest method to build basis functions is to augment the inputs with polynomial terms to achieve higher-order Taylor expansions, for example, with quadratic terms

* Corresponding author. Department of Management and Quantitative Methods, ETEA, Spain. Tel.: +34 957 222120; fax: +34 957 222107.
E-mail address: fjmestud@etea.com (F. Martínez-Estudillo).

0031-3203/$30.00
doi:10.1016/j.patcog.2006.06.003

and multiplicative interactions. Note, however, that the number of variables grows exponentially in the degree of the polynomial.

Our approach overcomes the nonlinear effects of the covariates proposing a LR model based on the hybridation of linear and product-units models (LRLPU), introducing into the model nonlinear basis functions constructed with the product of the inputs raised to arbitrary powers. These basis functions express the possible strong interactions between the covariates, where the exponents are not fixed and may even take real values. Moreover, we avoid the huge number of coefficients involved in the polynomial model.

The nonlinear basis functions of the proposed model corresponds to a special class of feed-forward neural network, namely product-unit neural networks (PUNN), introduced by Durbin and Rumelhart [6]. They are an alternative to standard sigmoidal neural networks and are based on multiplicative nodes instead of additive ones. The error surface associated with PUNN is extremely convoluted with numerous local optimums and plateaus. This is because small changes in the exponents can cause large changes in the total error surface. The estimation of the coefficients is carried out in several steps. In a first step, an evolutionary algorithm (EA) is applied to the design of the structure and training of the weights in a PUNN. The evolutionary process determines the number of basis functions in the model and the corresponding exponents. The complexity of the error surface of the proposed model justifies the use of an EA as part of the process of estimation of the model coefficients. That step can be seen as a global search in the coefficients' model space. On the other hand, it is well known that EA are efficient at exploring an entire search space; however, they are relatively poor at finding the precise optimum solution in the region where the algorithm converges to. In order to improve the lack of precision of the EA, we use, in a second step, a local optimization algorithm. More precisely, once the basis functions have been determined by the EA, the model is linear in these new variables together with the initial covariates and the fitting proceeds with standard maximum likelihood optimization method for LR.

Finally, we apply a backward method to select the best covariates to explain the response. By controlling the number of coefficients in the final model we can decrease the risk of building overly complex models that overfit the training data, and therefore obtain simpler models. It should be pointed that most of the classification techniques are principally used to improve the precision of the classifier, while their comprehensibility and interest are of secondary importance [7]. That comprehensibility is becoming more and more important for researchers who need to be able to make a sensitive analysis of each and every covariate of the model, which is why the last few years have seen some articles dealing specifically with comprehensibility [8] and others that are about it as well as precision [9]. Thus, throughout this paper we will do our best to obtain the maximum precision in classification while maintaining the simplest models possible as far as the number of model coefficients is concerned.

We evaluate the performance of our methodology on seven data sets of two classes taken from the UCI repository [10] and a classification microbiology problem. The empirical results show that the proposed hybrid method is very promising in terms of classification accuracy, simplicity as well as very efficient in terms of the total number of coefficients and basis functions needed for constructing the final binary classifiers, and yielding a state-of-the-art performance. It is interesting to point out that the proposed hybrid model outperforms the linear model constructed by means of LR with initial covariates and also the nonlinear part of the model obtained with a logistic regression with all covariate product units (LRPU). In this way, the hybrid model (LRLPU) determines a good balance between the linear and nonlinear part.

The paper is arranged as follows: Section 2 briefly reviews and discusses some related papers. Section 3 introduces LR and our model in depth. Section 4 describes the process of coefficient estimation. Section 5 introduces the datasets and explains the experiments carried out and finally Section 6 summarizes the conclusions of our work.

2. Related works

Regression models play an important role in many data analysis, providing prediction and classification rules, where the linear models are the most frequency used because they are very simple and comprehensible, although in general that traditional linear model often fails in real situations. In this section, we give a brief overview of the different methods that use basis functions for moving beyond linearity. Moreover, we point out some recent works that show a close relationship between LR and machine learning methods.

The generalized additive models [4] comprise automatic and flexible statistical methods that may be used to identify and characterize nonlinear regression effects. For two class classification, the additive LR model is an example of generalized additive model and it replaces each linear term by a more general functional form approximating multidimensional functions as a sum of univariate curves. The univariate functions are estimated in a flexible manner, using an algorithm whose basic building block is a scatter plot smoother, for example, the cubic smoothing spline. The additive model manages to retain interpretability by restricting nonlinear effects in the predictors to enter into the model independently of one another. Generalized additive models provide a natural first approach to relaxing strong linear assumptions. A way to capture the interaction terms is to generalize the additive model including spline terms that possibly depend on more than one variable [11]. In a further paper, [12], the same authors extend this work to other basis functions, such as thin-plate spline, multiquadric and cubic basis functions. They are all examples of radial basis functions

that rely on distances between vectors in predictor space. Unfortunately, because of the curse of dimensionality, these basis functions can perform poorly in a high dimensional input space.

Another approach to overcome the difficulties associated with the curse of dimensionality is to consider basis functions given by a sum of lower-dimensional functions. In [13], Gustafson introduced the Bayesian regression model with interactions and smooth effects (BWISE). The model is particularly well suited to modelling data when the deviations from linearity are not great.

Finally, one of the most popular multiple nonlinear regression methods is the MARS [5] model, where the basis functions are given by a product of some number of one-dimensional spline functions. Basis functions are added gradually during learning, using a technique of sequential forward selection.

From another point of view, LR has gained popularity recently in the machine learning community due to its close relations to well-known techniques such as SVM [14], AdaBoost [15] and artificial neural networks (ANN) [16,17]. Vapnik [14] compared LR and SVM in terms of minimizing the loss functional, and showed that the loss function of LR can be very well approximated by SVM loss with multiple knots (SVMn). In [18], Friedman et al. discussed SVM, LR and boosting on top of their different loss functions. Lebanon and Lafferty showed in [19] that the only difference between AdaBoost and LR is that the latter requires the model to be normalized to a probability distribution. Finally, the LR model can be seen as equivalent to a perceptron with a logistic activation function representing the simplest neural network. Finally, a comparative investigation of LR and neural networks can be found in [16,17].

3. Logistic regression

We consider the situation where we observe a binary outcome variable y and a vector $\mathbf{x} = (1, x_1, x_2, \ldots, x_k)$ of covariates for each of N individuals (we assume that the vector of inputs includes the constant term 1 to accommodate the intercept). We code the two-class via a 0/1 response y_i, where $y_i = 1$ for the first class and $y_i = 0$ for the second one. Let p be the conditional probability associated with the first class. LR [20,21] is a widely used statistical modelling technique in which the probability p of the dichotomous outcome event is related to a set of explanatory variables $\mathbf{x}$ in the form:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = f(\mathbf{x}, \boldsymbol{\beta}) = \boldsymbol{\beta}^{\mathbf{T}}\mathbf{x}, \tag{1}$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \ldots, \beta_k)$ is the vector of the coefficients of the model and $\boldsymbol{\beta}^{\mathbf{T}}$ the transpose vector. We refer to $p/(1-p)$ as odds-ratio and to the expression (1) as the log-odds or logit transformation. A simple calculation in Eq. (1) shows that the probability of occurrence of an event as a function of the covariates is nonlinear and is given by

$$p(\mathbf{x}; \boldsymbol{\beta}) = \frac{e^{\boldsymbol{\beta}^{\mathbf{T}}\mathbf{x}}}{1 + e^{\boldsymbol{\beta}^{\mathbf{T}}\mathbf{x}}}. \tag{2}$$

When the conditional probability function (2) is known, one can construct the Bayesian (optimal) decision rule

$$r(\mathbf{x}) = \text{sign}\left\{\ln\left(\frac{p}{1-p}\right)\right\}. \tag{3}$$

The decision boundary is the set of points for which the log-odd is zero, that is, in this linear model, the hyperplane defined by $\boldsymbol{\beta}^{\mathbf{T}}\mathbf{x} = 0$. Observe that LR not only constructs a decision rule but also finds a function that for any input vector defines the probability p that the vector $\mathbf{x}$ belongs to the first class.

Let $D = \{(\mathbf{x}_l, y_l) : l = 1, 2, \ldots, n_T\}$ be the training data set, where the number of samples is n_T. Here, we assume that the training sample is a realization of a set of independent and identically distributed random variables. The unknown regression coefficients β_i, which have to be estimated from the data, are directly interpretable as log-odds ratios or, in term of $\exp(\beta_i)$, as odds ratios. That log-likelihood for n_T observations is

$$\begin{aligned} l(\boldsymbol{\beta}) &= \sum_{l=1}^{n_T} \{y_l \log p(\mathbf{x}_l; \boldsymbol{\beta}) + (1 - y_l) \log(1 - p(\mathbf{x}_l; \boldsymbol{\beta})\} \\ &= \sum_{l=1}^{n_T} \{y_l \boldsymbol{\beta}^{\mathbf{T}}\mathbf{x}_l - \log(1 + e^{\boldsymbol{\beta}^{\mathbf{T}}\mathbf{x}_l})\}. \end{aligned} \tag{4}$$

The associated Hessian matrix is negative semidefinite [22], which implies that the log-likelihood is a concave function of the coefficient $\boldsymbol{\beta}$. Concavity, together with the fact that the coefficient vector $\boldsymbol{\beta}$ varies freely in a convex set, guarantee that there are no local maxima on the log-likelihood surface of a LR model. The estimation of coefficient $\boldsymbol{\beta}$ is usually carried out by means of an iterative procedure like the Newton–Raphson algorithm or the iteratively reweighted least squares (IRLS) [23]. Typically the algorithm converges, since the log-likelihood is concave, but overshooting can occur. In the rare cases where log-likelihood decreases, step size halving will guarantee convergence. The conditions under which a global maximum exists and the maximum likelihood estimators do not diverge are discussed in [22] and references therein.

We propose a LR model based on the hybridation of the standard linear model and product-unit models, introducing a nonlinear term in the model constructed with basis functions given by products of the inputs raised to real powers, which express the possible strong interactions between the covariates. The general expression of the model is given by

$$f(\mathbf{x}, \boldsymbol{\theta}) = \alpha_0 + \sum_{i=1}^{k} \alpha_i x_i + \sum_{j=1}^{m} \beta_j \prod_{i=1}^{k} x_i^{w_{ji}}, \tag{5}$$

with the corresponding matricial expression:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \boldsymbol{\alpha}^{\mathbf{T}}\mathbf{x} + \boldsymbol{\beta}^{\mathbf{T}}\mathbf{B}(\mathbf{x}, \mathbf{W}),$$

where the basis functions are $\mathbf{B}(\mathbf{x}, \mathbf{W}) = \{B_1(\mathbf{x}, \mathbf{w}_1), B_2(\mathbf{x}, \mathbf{w}_2), \ldots, B_m(\mathbf{x}, \mathbf{w}_m)\}$, with $B_j(\mathbf{x}, \mathbf{w_j}) = \prod_{i=1}^{k} x_i^{w_{ji}}$ and $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{W})$, $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_k)$, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_m)$ and $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w_m})$, with $\mathbf{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jk})$, $w_{ji} \in \mathbb{R}$.

In this way, the new conditional distribution is

$$p(\mathbf{x}; \boldsymbol{\theta}) = \frac{e^{f(\mathbf{x},\boldsymbol{\theta})}}{1 + e^{f(\mathbf{x},\boldsymbol{\theta})}} \tag{6}$$

and the logit transformation

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = f(\mathbf{x}, \boldsymbol{\theta}). \tag{7}$$

In this case the decision boundaries are nonlinear and defined by the hypersurface $f(\mathbf{x}, \boldsymbol{\theta}) = 0$ in the $\mathbb{R}^k$ space. Depending on the coefficient $\boldsymbol{\theta}$ the hypersurface could even be nonconnected.

The nonlinear part of $f(\mathbf{x}, \boldsymbol{\theta})$ corresponds to a special class of feed-forward neural networks, namely PUNN, introduced by Durbin and Rumelhart [6]. They are an alternative to the standard sigmoidal neural networks and are based on multiplicative nodes instead of additive ones. This class of multiplicative neural networks comprise such types as sigma–pi networks and product-unit networks. In contrast to the sigma–pi unit, the exponents of the product-unit are not fixed and may even take real values. Advantages of product-unit based neural networks include increased information capacity and the ability to form higher-order combinations of inputs. They are universal approximators and it is possible to obtain upper bounds of the VC dimension of PUNN similar to those obtained for sigmoidal neural networks [24]. Despite these obvious advantages, product-unit based neural networks have a major drawback. Their training is more difficult than the training of standard sigmoidal based networks [6]. The main reason for this difficulty is that small changes in the exponents can cause large changes in the total error surface. Hence, networks based on product units have more local minima and a greater probability of becoming trapped in them [25,26]. It is a well known issue [27] that back-propagation is not efficient in training product units. Several efforts have been made to develop learning methods for product units [28–31].

In our framework, the product units have the following structure (Fig. 1): an input layer with a node for every input variable, a hidden layer with several nodes, and an output layer with just one node. There are no connections between the nodes in a layer and none between the input and output layers either. The network has k inputs that represent the independent variables of the model, m nodes in the hidden layer and one node in the output layer. The activation of jth node in the hidden layer is given by $B_j(\mathbf{x}, \mathbf{w}_j) = \prod_{i=1}^{k} x_i^{w_{ji}}$ where w_{ji} is the weight of the connection between input node i and hidden node j. The activation of the output node is given by $\beta_0 + \sum_{j=1}^{m} \beta_j B(\mathbf{x}, \mathbf{w_j})$ where β_j is the weight of the connection between the hidden node j and the output node. The transfer function of all nodes is the identity function.

In this case the log-likelihood for n_T observations is

$$l(\boldsymbol{\theta}) = \sum_{l=1}^{n_T} \{y_l f(\mathbf{x}_l, \boldsymbol{\theta}) - \log(1 + e^{f(\mathbf{x}_l,\boldsymbol{\theta})})\}. \tag{8}$$

The nature and the properties of the function $f(\mathbf{x}, \boldsymbol{\theta})$ imply that the associated Hessian matrix in Eq. (8) is generally indefinite and has more local maximum and more probability of becoming trapped in them. To overcome this problem we use an EA as part of the process of coefficient estimation. In the following section we describe in detail the process of obtaining the estimated coefficients $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\mathbf{W}})$ of log-likelihood.

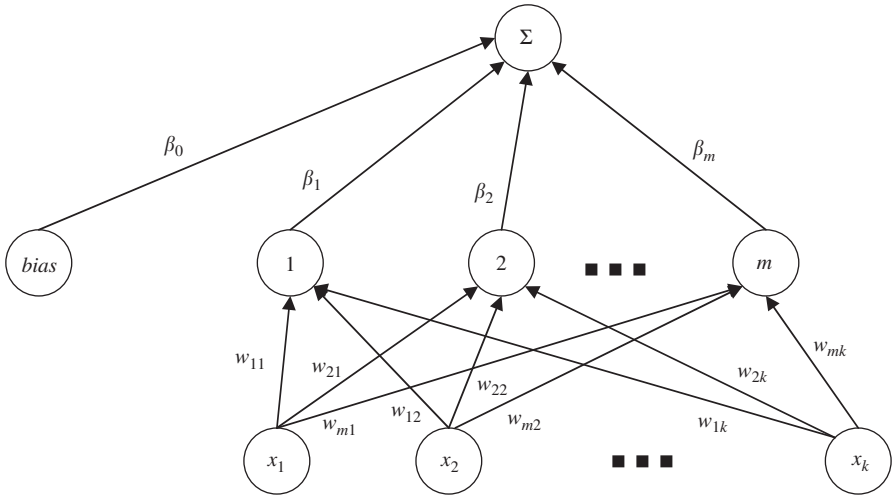

Fig. 1. Model of a product-unit based neural network.

4. Estimation of the coefficients

The methodology proposed is based on the combination of an EA (global explorer) and a local optimization procedure (local exploiters) carried out by standard maximum likelihood optimization method. In a first step, an EA is applied to design the structure and training of the weights of a PUNN. The evolutionary process determines the number m of potential basis functions of the model and the corresponding vector $\mathbf{w}_j$ of exponents. Once the basis functions $B(\mathbf{x}, \mathbf{W}) = \{B_1(\mathbf{x}, \mathbf{w}_1), B_2(\mathbf{x}, \mathbf{w}_2), \ldots, B_m(\mathbf{x}, \mathbf{w}_m)\}$ have been determined by the EA, we consider a transformation of the input space adding the nonlinear transformations of the input variables given by the basis functions obtained by the EA. The model is linear in these new variables together with the initial covariates. The remaining coefficients $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are calculated by the maximum likelihood optimization method. Finally, we use a backward stepwise procedure, pruning variables sequentially to the model obtained previously, until further prunes do not improve the fit.

In the following paragraphs we describe each one of the different aspects of the algorithm in detail.

Algorithm.

Step 1: We apply an EA to find the basis functions $B(\mathbf{x}, \hat{\mathbf{W}}) = \{B_1(\mathbf{x}, \hat{\mathbf{w}}_1), B_2(\mathbf{x}, \hat{\mathbf{w}}_2), \ldots, B_m(\mathbf{x}, \hat{\mathbf{w}}_m)\}$ corresponding to the nonlinear part of $f(\mathbf{x}, \boldsymbol{\theta})$. We have to determine the number of basis functions m and the weights $\hat{\mathbf{W}} = (\hat{\mathbf{w}}_1, \hat{\mathbf{w}}_2, \ldots, \hat{\mathbf{w}}_m)$. 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 EA for architectural design and the estimation of real-coefficients have points in common with other EAs in [32–34]. The search begins with an initial population, and 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 [32].

We consider the mean squared error (MSE) of an individual g of the population

$$MSE(g) = \frac{1}{n_T} \sum_{l=1}^{n_T} (y_l - g(\mathbf{x}_l))^2, \tag{9}$$

where y_l are the predicted values. We define the fitness function $A(g)$ by means of a strictly decreasing transformation of the mean squared error (9):

$$A(g) = \frac{1}{1 + MSE(g)}. \tag{10}$$

The general structure of the EA is the following:

(1) The initial population of size N_R is generated.
(2) Repeat until the stopping criterion is fulfilled
   (a) calculate the fitness of every individual in the population;
   (b) rank the individuals with respect to their fitness;
   (c) the best individual is copied into the new population;
   (d) the best 10% of individuals of the population are replicated and substitute the worst 10% of individuals;
   (e) apply parametric mutation to the best 10% of individuals;
   (f) apply structural mutation to the remaining 90% of individuals.

Parametric mutation consists of a simulated annealing algorithm [35]. The severity of a mutation to an individual g is dictated by the temperature $T(g)$, given by

$$T(g) = 1 - A(g), \quad 0 \leqslant T(g) < 1. \tag{11}$$

Parametric mutation is accomplished for each coefficient w_{ji}, β_j of the model with Gaussian noise and where the variance of the normal distribution depends on the temperature (11):

$$w_{ji}(t+1) = w_{ji}(t) + \xi_1(t) \quad \text{and} \quad \beta_j(t+1) = \beta_j(t) + \xi_2(t),$$

where $\xi_k(t) \in N(0, \alpha_k T(g))$, $k = 1, 2$ represents a one-dimensional normally distributed random variable with mean 0 and variance $\alpha_k T(g)$. It should be pointed that the modification of the exponents w_{ji} is different from the modification of the coefficients β_j, therefore $\alpha_1 \ll \alpha_2$.

Structural mutation implies a modification of the structure of the function and allows the explorations of different regions of the search space and helps to keep the diversity of the population. There are four different structural mutations similar to the mutations of the GNARL model [32]: Node addition (AN), node deletion (DN), connection addition (AC) and connection deletion (DC). All the above mutations are made sequentially in the given order, with probability $T(g)$, in the same generation on the same network. If the probability does not select any mutation, one of the mutations is chosen at random and applied to the network.

The stop criterion is reached whenever one of the following two conditions is fulfilled: (i) for a determined number of generations there is no improvement either in the average performance of a percentage of the best individuals in the population or in the fitness of the best individual; (ii) the algorithm achieves a determined number of generations. For more details about the EA see [36,37].

Step 2: We consider the following transformation of the input space adding the nonlinear transformations of the input variables given by the basis functions obtained by the EA in step 1:

$$H : \mathbb{R}^k \to \mathbb{R}^{k+m}$$

$$(x_1, x_2, \ldots, x_k) \to (x_1, x_2, \ldots, x_k, z_1, \ldots, z_m),$$

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

Step 3: We apply a (standard) maximum likelihood optimization method in the new space of derived input features. We optimize the log-likelihood function for n_T observations

$$l(\boldsymbol{\theta}^1) = \sum_{i=1}^{n_T} \{y_i f(\mathbf{x}, \boldsymbol{\theta}^1) - \log(1 + e^{f(\mathbf{x}, \boldsymbol{\theta}^1)})\}, \quad (12)$$

where $\boldsymbol{\theta}^1 = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \hat{\mathbf{w}})$. The estimated coefficient $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\mathbf{w}})$ are obtained by means of the Newton–Raphson optimization algorithm. The estimation process determines the model:

$$f(\mathbf{x}, \hat{\boldsymbol{\theta}}) = \hat{\alpha}_0 + \sum_{i=1}^{k} \hat{\alpha}_i x_i + \sum_{j=1}^{m} \hat{\beta}_j \prod_{i=1}^{k} x_i^{\hat{w}_{ji}}. \quad (13)$$

Step 4: In order to select the final model, we use a backward stepwise procedure, starting with the full model with all the covariates and successively pruning variables sequentially to the model until further prunes do not improve the fit. At each step, we deleted the least significant covariate to predict the response variable, that is, the one which shows the greatest critical value (p-value) in the hypothesis test, where the associated coefficient equal to zero is the hypothesis to be contrasted. The procedure finishes when all tests provide p-values smaller than the fixed significance level and the model selected in the previous step fits well.

5. Experiments

We evaluate the performance of our methodology on seven, two classes, datasets taken from the UCI repository [10] and a predictive microbiology problem. Testing our model on this wide variety of problems can give us a clear idea of its performance.

The experimental design for the seven classification benchmarks and the microbian growth problem was conducted using a holdout cross-validation procedure where the size of the training set was approximately $3n/4$ and $n/4$ for the generalization set, where n is the size of the full data set.

The parameters used in the EA for learning the PUNN model are common for all problems: exponents w_{ji} are initialized in the $[-5, 5]$ interval, the coefficients β_j are initialized in $[-10, 10]$. The maximum number of nodes is $m = 6$. The size of the population is $N_R = 2000$. The number of nodes that can be added or removed in a structural mutation is within the $[1, 2]$ interval. The number of connections that can be added or removed in a structural mutation is within the $[1, 6]$ interval. The proposed conditions to the stop criterion are: (i) for five generations there is no improvement either in the average performance of 20% best of the population or in the fitness of the best individual. (ii) The algorithm achieves 100 generations.

5.1. Results of the benchmark data sets

In Table 1 we briefly describe the seven datasets used in the study. These seven, two classes, classification sets cover a wide variety of problems. There are problems with different numbers of available patterns, from 208 to 1000, different kind of inputs, nominal, binary and continuous and different areas of application, from medical diagnosis to sonar signals recognition. The input variables of the problems have different scales and present a large range of variability. For this reason it is advisable to carry out a pre-processing of the data. We have done a simple linear rescaling to the input variables to the interval $[0.1, 0.9]$.

To validate the model, we used the correct classification rate (CCR) for the generalization set defined as the percentage of subjects in the data set that are correctly classified. Table 2 shows the matrix results of classification for the seven benchmark problems. We consider the total CCR, and the partial correct classification rate PCR for training and generalization sets for the three models: a logistic regression with the initial covariates $x_1, x_2, \ldots, x_k$, (LR), logistic regression only with all covariate product units $z_1, z_2, \ldots, z_m$, (LRPU), in brackets; and our approach given by logistic regression with initial and product units covariates $x_1, x_2, \ldots, x_k, z_1, z_2, \ldots, z_m$ using attribute selection (LRLPU), in square brackets. In order to select the most significant variables of the logistic regression model, a backward method was selected by using SPSS software for Windows 11.0, [38].

Table 3 presents estimators of the coefficients from a LRLPU model, with their sample standard deviation, and the p-values associated with a Wald asymptotic t-test for each coefficient. We observed that in general the coefficients of the proposed models are significant for a coefficient $\alpha = 0.1$. In that table we give the percentage of initial attributes (after converting nominal attributes to binary ones) included in the final model where we used attribute selection. On average, the biggest reduction in initial covariates takes place for datasets with a high number of attributes because a high degree of interaction exists between them.

The experiments show that LRLPU models, in all the proposed data sets for the same partition (training set/generalization set), outperform the linear model constructed by means of logistic regression with initial covariates (LR) and also the nonlinear part of the model obtained with a logistic regression with the product-unit covariates (LRPU).

With the objective of presenting an empirical evaluation of the performance of LRLPU we compare to other learning schemes. We will use the most recent results [39] from six of the seven data sets considered. In [39] the authors compare their logistic model tree (LMT) algorithm to LR (with attribute selection, SLogistic, and for a full logistic model, MLogistic). Afterwards, the authors compare it with induction trees (C4.5 [40] and CART [41]). Also, they consider two logistic tree algorithms: LTree (using LR for splitting and at the leaves) [42] and Lotus [43] with two methodologies: one using simple logistic regression (LotusS) and another using multiple logistic regression (LotusM). Finally, they consider multiple-tree models M5' [44] for

Table 1
Summary of data sets

Data set	Cases train-test	Variables				Description
		C	B	N	k	
Cancer	525–174	9	0	0	9	This breast cancer data set was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg [50]. Each pattern has nine attributes, all of them ordinal in the range 1–10. There are two classes meaning if the cancer was benign (65.5% of the cases) or malignant (34.5%).
Card	518–172	6	4	5	51	The set contains data from applications to an Australian bank to get a credit card. There are two classes, meaning whether the application was granted (44.5% of the patterns) or denied (55.5%). Each record has 15 attributes, for confidentiality all attributes and values are not explained in the original data set.
German	750–250	6	3	11	61	This data set has been report for Hans Hofmann, where it classified a customer as good or bad. Several attributes that are ordered categorical have been coded as integer. This was the form used by StatLog. The number of attributes is 20 (6 numerical, 14 categorical).
Heart-statlog	202–68	13	0	0	13	This data set comes from the Cleveland Clinic Foundation and was supplied by Robert Detrano of the V.A. Medical Center, Long Beach, CA. The database contains 13 attributes, which have been extracted from a larger set of 75, that correspond to the results of various medical tests carried out on a patient. The goal is the prediction of the presence or absence of heart disease in those patients. The problem is described in detail in [51].
Ionosphere	264–87	33	1	0	34	This ionosphere classification comes from the Johns Hopkins University and was supplied by Sigillito et al. [52] The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not.
Pima	576–192	8	0	0	8	The data set contains data of 768 individuals, all of them females at least 21 years old of Pima Indian heritage. The patterns are divided into two classes. The class of each pattern shows whether the patient shows signs of diabetes according to the World Health Organization criteria. Former results can be found in [53].
Sonar	104–104	60	0	0	60	This is the data set used by Gorman and Sejnowski in their study of the classification of sonar signals using a neural network [54]. The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. The data set is interesting as an example of very redundant input variables.

The attributes of each data set can be C (continuous), B (binary) or N (nominal) where k is the total number of variables.

classification, and boosted C4.5 trees using AdaBoost.M1 with 10 and 100 boosting interactions. The authors affirm that LTM is comparable to AdaBoost (100) but that the relative performance of the two schemes really depends on the datasets.

For these six data sets, (see Table 4), the best results in three of them are obtained with AdaBoost (100), while in the other three the best results are found with SLogistic, MLogistic or LotusM. In four data sets, our results (marked with $*$ in the LRLPU column) are higher than the mean result of the best of the 10 compared algorithms.

Let us assume that the distributions of the results obtained with the best of the 10 algorithms proposed are normal for the six datasets (the values of these averages and typical deviations are estimated in boldface, in Table 4). Then for the data set Cancer, the probability that an execution of the algorithm Aboost (100) chosen at random could outdo our results is 0.231 (see the last column in Table 4). That is, for a random run of the ABoost (100) algorithm, the $P(\text{run}_{\text{AB}} > 98.3) = P(Z > 0.7349) = 0.231$, where Z is a Gaussian variable of mean 0 and variance 1. Similarly, for Card, the probability that a random execution of the best algorithm, ABoost (100), in this case, outdo our result is $P(\text{run}_{\text{AB}} > 90.7) = 0.142$. For German, this probability is 0.471, for a random execution of Slogistic. For Heart, this probability is 0.21 if compared to the result obtained randomly on executing MLogistic or LotusM. For Ionosphere the probability is 0.632 as compared to Aboost (100). For Pima this probability is 0.112 as compared to MLogistic and LotusM, and finally for Sonar the probability is 0.632 as compared to Aboost (100). Thus our results are competitive with those obtained in [39], providing a higher degree of interpretability because of a smaller structure compared to the other logistic tree models (observe that a model tree is a regression tree with regression functions at the leaves, see Table 5).

Table 2
Rate of the number of cases that were classified correctly in different two classes benchmark problems using logistic regression, LR, full nonlinear LRPU (in brackets) and LRLPU model (in squared brackets)

Predicted response	Training		PCR	Generalization		PCR
	$Y=1$	$Y=0$		$Y=1$	$Y=0$	
Cancer						
Target response						
$Y=1$	167(163) [169]	9(13) [7]	94.9(92.6) [96.0]	61(57) [62]	4(8) [3]	93.8(87.7) [95.4]
$Y=0$	11(13) [11]	338(336) [338]	96.8(96.3) [96.8]	0(0) [0]	109(109) [109]	100.0(100.0) [100]
CCR			96.2(95.0) [**96.6**]			97.7(95.4) [**98.3**]
Card						
Target						
$Y=1$	260(265) [245]	34(29) [49]	88.4(90.1) [83.3]	77(81) [76]	12(8) [13]	86.5(91) [85.4]
$Y=0$	27(67) [22]	197(157) [202]	87.9(70.1) [90.2]	7(25) [3]	76(58) [80]	91.6(69.9) [96.4]
CCR			**88.2**(81.5) [86.3]			89.0(80.8) [**90.7**]
German						
Target						
$Y=1$	123(0) [126]	103(226) [100]	54.4(0.0) [55.8]	35(0) [36]	39(74) [38]	47.3(0.0) [48.6]
$Y=0$	60(2) [55]	464(522) [469]	88.5(99.6) [89.5]	26(0) [23]	150(176) [153]	85.2(100) [86.9]
CCR			78.3(69.6) [**79.3**]			74.0(70.4) [**75.6**]
Heart						
Target						
$Y=1$	72(74) [75]	18(16) [15]	80.0(82.2) [83.3]	25(25) [25]	5(5) [5]	83.3(83.3) [83.3]
$Y=0$	11(10) [11]	101(102) [101]	90.2(91.1) [90.2]	2(4) [1]	36(34) [37]	94.7(89.5) [97.4]
CCR			85.6(**87.1**) [**87.1**]			89.7(86.8) [**91.2**]
Ionosphere						
Target						
$Y=1$	164(151) [164]	7(20) [7]	95.9(88.3) [95.9]	51(47) [53]	3(7) [1]	94.4(87.0) [98.1]
$Y=0$	22(18) [9]	71(75) [84]	76.3(80.6) [90.3]	9(7) [6]	24(26) [27]	72.7(78.8) [81.8]
CCR			89.0(85.6) [**93.9**]			86.2(83.9) [**92.0**]
Pima						
Target						
$Y=1$	318(325) [318]	47(40) [47]	87.1(89.0) [87.1]	122(121) [122]	13(14) [13]	90.4(89.6) [90.4]
$Y=0$	89(81) [80]	122(130) [131]	57.8(61.6) [62.1]	22(23) [20]	35(34) [37]	61.4(59.6) [64.9]
CCR			76.4(**79.0**) [78.0]			81.8(80.7) [**82.8**]
Sonar						
Target						
$Y=1$	49(34) [43]	0(15) [6]	100.0(75.5) [87.8]	47(43) [51]	15(19) [11]	75.8(69.4) [82.3]
$Y=0$	0(14) [4]	55(41) [51]	100.0(90.9) [92.7]	11(10) [7]	31(32) [35]	73.8(76.2) [83.3]
CCR			**100.0**(83.7) [90.4]			75.0(72.1) [**82.7**]

Table 3
Best models from LRLPU model to seven benchmark problems

Variables	Bias	X_1^*	X_3^*	X_6^*	X_7^*	X_8^*	B_1	B_2
Cancer								
Coeff.	10.236	−7.041	−3.779	−5.755	−5.570	−17.106	−262.844	20.187
Std error	1.399	1.706	1.806	1.324	1.790	8.500	156.785	11.023
p-value	0.000	0.000	0.036	0.000	0.002	0.044	0.094	0.067

$B_1 = (X_2^*)^{5.465}(X_5^*)^{5.955}(X_6^*)^{-0.089}(X_7^*)^{0.198}$
$B_2 = (X_1^*)^{0.080}(X_2^*)^{0.062}(X_6^*)^{0.100}(X_8^*)^{1.057}(X_9^*)^{0.022}$
rate of attributes used $= 8/9 = 0.89$; # coefficients $= 17$

Variables	X_8^*	X_{12}^*	X_{25}^*	X_{26}^*	X_{29}^*	X_{42}^*	X_{44}^*	X_{51}^*	B_1
Card									
Coeff.	−6.417	−5.406	1.110	3.173	−2.515	4.108	15.525	1.638	3.4e-008
Std error	0.796	0.705	0.596	1.029	1.049	0.511	4.350	0.781	2.1e-008
p-value	0.000	0.000	0.063	0.002	0.016	0.000	0.000	0.036	0.097

$B_1 = (X_4^*)^{1.560}(X_{10}^*)^{-4.115}(X_{11}^*)^{2.202}(X_{14}^*)^{-4.771}(X_{15}^*)^{0.317}(X_{23}^*)^{-1.573}(X_{27}^*)^{0.071}(X_{35}^*)^{1.397}$
$(X_{40}^*)^{-2.139}(X_{41}^*)^{0.296}(X_{42}^*)^{3.906}(X_{43}^*)^{0.189}(X_{46}^*)^{0.252}(X_{51}^*)^{0.462}$
rate of attributes used $= 20/60 = 0.33$; # coefficients $= 23$

Variables	Bias	X_1^*	X_2^*	X_5^*	X_7^*	X_{10}^*	X_{11}^*	X_{12}^*	X_{17}^*
German									
Coeff.	5.892	−1.408	−1.528	−2.685	−1.085	0.942	−0.870	1.390	−1.030
Std. error	1.045	0.456	0.306	0.874	0.536	0.299	0.290	0.522	0.542
p-value	0.000	0.002	0.000	0.002	0.043	0.002	0.003	0.008	0.057

Variables	X_{22}^*	X_{23}^*	X_{24}^*	X_{31}^*	X_{33}^*	X_{36}^*	X_{42}^*	X_{46}^*	X_{47}^*
Coeff.	−1.847	−1.326	−0.926	0.836	−1.212	0.509	1.472	−0.619	2.710
Std. error	1.082	0.316	0.458	0.347	0.419	0.258	0.572	0.349	1.231
p-value	0.088	0.000	0.043	0.016	0.004	0.048	0.010	0.076	0.028

Variables	X_{48}^*	X_{51}^*	X_{61}^*	B_1
Coeff.	0.860	−0.624	−1.707	−1.011
Std. error	0.324	0.313	0.890	0.530
p-value	0.008	0.046	0.055	0.057

$B_1 = (X_2^*)^{-0.181}(X_4^*)^{-0.245}(X_{26}^*)^{-0.054}(X_{33}^*)^{-0.098}$
$(X_{47}^*)^{0.479}(X_{48}^*)^{0.047}(X_{54}^*)^{0.041}$
rate of attributes used $= 23/61 = 0.38$; # coefficients $= 29$

Variables	Bias	X_1^*	X_2^*	X_3^*	X_4^*	X_5^*	X_6^*	X_{11}^*	X_{13}^*
Heart									
Coeff.	−4.157	2.923	−0.622	−2.070	−3.995	−2.421	0.746	−0.734	1.025
Std error	2.455	1.819	0.845	3.110	1.754	2.505	0.789	0.933	0.766
p-value	0.016	0.108	0.461	0.506	0.023	0.334	0.334	0.431	0.181

Variables	B_1	B_2
Coeff.	11.709	−12.689
Std error	2.143	5.222
p-value	0.000	0.015

$B_1 = (X_3^*)^{0.255}(X_8^*)^{0.253}(X_{12}^*)^{-0.116}(X_{13}^*)^{-0.083}$
$B_2 = (X_2^*)^{0.050}(X_3^*)^{0.803}(X_5^*)^{0.047}(X_7^*)^{0.043}$
$(X_8^*)^{0.205}(X_9^*)^{0.058}(X_{10}^*)^{0.083}(X_{12}^*)^{0.008}$
rate of attributes used $= 13/13 = 1$; # coefficients $= 23$

Variables	Bias	X_1^*	X_5^*	X_{11}^*	X_{14}^*	X_{19}^*	X_{22}^*	X_{23}^*	X_{24}^*
Ionosphere									
Coeff.	5.960	8.714	−9.001	5.848	−7.337	2.149	7.671	−9.144	−4.812
Std. error	3.490	2.141	4.023	2.731	2.918	2.356	3.588	3.429	2.679
p-value	0.088	0.000	0.025	0.032	0.012	0.362	0.033	0.008	0.072

Variables	X_{26}^*	X_{27}^*	X_{29}^*	X_{34}^*	B_1	B_2
Coeff.						
Std. error	−6.970	6.369	−5.542	8.920	−60.279	86.043
	2.115	3.072	2.461	2.358	11.713	16.053
p-value	0.001	0.038	0.024	0.000	0.000	0.000

Table 3 (*continued*)

Variables	X_{26}^*	X_{27}^*	X_{29}^*	X_{34}^*	B_1	B_2

$B_1 = (X_1^*)^{1.531}(X_3^*)^{0.509}(X_5^*)^{2.697}(X_8^*)^{0.141}(X_{10}^*)^{0.181}(X_{12}^*)^{-0.010}(X_{24}^*)^{0.167}(X_{25}^*)^{0.165}$

$B_2 = (X_1^*)^{3.854}(X_5^*)^{8.685}(X_6^*)^{-0.231}(X_7^*)^{-0.096}(X_{11}^*)^{-0.376}(X_{19}^*)^{-0.045}(X_{22}^*)^{0.428}(X_{23}^*)^{0.252}$
$(X_{26}^*)^{-0.0002}(X_{32}^*)^{0.117}(X_{33}^*)^{0.068}$

rate of attributes used = 20/33 = 0.61; # coefficients = 34

Variables	Bias	X_1^*	X_2^*	X_3^*	X_8^*	B_1	B_2
Pima							
Coeff.							
Std error	−10.166	2.172	13.212	−1.975	−1.698	15.977	−1688.149
	0.955	0.821	1.221	0.979	0.979	2.152	663.683
p-value	0.000	0.008	0.000	0.044	0.083	0.000	0.011

$B_1 = (X_2^*)^{-0.697}(X_4^*)^{-0.118}(X_6^*)^{1.560}(X_7^*)^{0.417}(X_8^*)^{0.453}$

$B_2 = (X_2^*)^{5.026}(X_3^*)^{-1.434}(X_4^*)^{-2.332}(X_5^*)^{1.409}(X_6^*)^{2.876}(X_7^*)^{4.435}(X_8^*)^{3.282}$

rate of attributes used = 1; # coefficients = 19

Variables	Bias	X_4^*	X_{17}^*	X_{18}^*	X_{22}^*	X_{28}^*	X_{37}^*	X_{39}^*	X_{40}^*
Sonar									
Coeff.									
Std. error	43.977	20.988	31.098	−30.534	−4.387	−5.009	27.547	−24.505	23.469
	20.217	11.226	10.560	10.852	2.785	2.771	9.554	8.428	7.447
p-value	0.030	0.062	0.003	0.005	0.115	0.071	0.004	0.004	0.002

Variables	X_{46}^*	X_{47}^*	X_{49}^*	X_{50}^*	X_{51}^*	X_{60}^*	B_1	B_2
Coeff.								
Std. error	−36.852	72.971	−29.178	30.721	−22.54	−22.790	20.088	−109.828
	14.952	26.131	8.763	9.349	9.517	11.483	7.909	39.551
p-value	0.014	0.005	0.001	0.001	0.018	0.047	0.011	0.005

$B_1 = (X_{11}^*)^{-0.117}(X_{19}^*)^{0.028}(X_{36}^*)^{0.135}$, $B_2 = (X_4^*)^{0.125}(X_{38}^*)^{0.010}(X_{47}^*)^{0.188}(X_{60}^*)^{-0.097}$

rate of attributes used = 18/60 = 0.3; # coefficients = 24

Table 4
Mean classification accuracy and standard deviation for LMT, SLogistic, MLogistic, C4.5, CART, Lotus using simple logistic regression (LotusS) and Lotus using multiple logistic regression (LotusM), M5′ for classification, ABoost and LRLPU model

Dataset	LMT	SLogistic	MLogistic	C4.5	CART	LotusS
Cancer	96.18 ± 2.20	96.21 ± 2.19	96.50 ± 2.18	95.01 ± 2.73	94.42 ± 2.70	94.61 ± 2.66
Card	85.04 ± 3.84	85.04 ± 3.97	85.33 ± 3.85	85.57 ± 3.96	84.55 ± 4.20	84.35 ± 4.18
German	75.37 ± 3.53	**75.34** ± 3.50	75.24 ± 3.54	71.25 ± 3.17	73.34 ± 3.66	72.69 ± 3.11
Heart	83.22 ± 6.50	83.30 ± 6.48	**83.67** ± 6.43	78.15 ± 7.42	78.00 ± 8.25	77.63 ± 7.16
Ionosphere	92.99 ± 4.13	87.78 ± 4.99	87.72 ± 5.57	89.74 ± 4.38	89.80 ± 4.78	89.04 ± 4.57
Pima	77.08 ± 4.65	77.10 ± 4.65	**77.47**(±4.39	74.49 ± 5.27	74.50 ± 4.70	75.08 ± 5.14
Sonar	76.32 ± 9.58	75.93 ± 8.51	72.47 ± 8.90	73.61 ± 9.34	74.77 ± 9.76	72.38 ± 9.13

Dataset	LotusM	M5′	ABoost (10)	ABoost (100)	LRLPU	*P*(Best > LRLPU)
Cancer	96.44 ± 2.13	95.85 ± 2.15	96.08 ± 2.16	**96.70** ± 2.18	98.3 (*)	0.231
Card	85.14 ± 4.03	85.39 ± 3.87	84.01 ± 4.36	86.43 ± 3.98	90.7 (*)	0.142
German	72.55 ± 3.36	74.99 ± 3.31	70.91 ± 3.60	74.53 ± 3.26	75.6 (*)	0.471
Heart	83.67 ± 6.43	82.15 ± 6.77	78.59 ± 7.15	80.44 ± 7.08	91.2 (*)	0.121
Ionosphere	87.72 ± 5.57	89.92 ± 4.18	93.05 ± 3.92	**94.02** ± 3.83	92.0	0.701
Pima	77.47 ± 4.39	76.56 ± 4.71	71.81 ± 4.85	73.89 ± 4.75	82.8 (*)	0.112
Sonar	72.27 ± 7.92	78.37 ± 8.82	79.22 ± 8.70	**85.14** ± 7.84	82.7	0.632

5.2. Application to microbial growth/no growth

Listeria monocytogenes has been a serious problem that has concerned food industries due to its ubiquity in the natural environment [45,46] and the specific growth conditions of the pathogen that lead to its high prevalence in different kinds of food products. One impetus for this research was the problem of listeriosis [47] and different strategies were proposed to limit levels of contamination at the time of consumption to less than 100 CFU/g (European Commission, [48]).

The use of ANN to describe the growth/no growth interface of several pathogens has already been implemented. Generally speaking, the use of ANNs were found to outperform the LR methods. Hajmeer and Basheer [49] reported the use of a Probabilistic Neural Network (PNN) approach combining the Bayes theorem of conditional probability and Parzen's method to estimate the probability density functions of random variables. This study was applied to classify growth/no growth of *E. coli R31* as a function of temperature and water activity.

In this section we present an application of the LRLPU model to determine the probability of *L. monocytogenes* growth as a function of storage temperature, pH, citric (CA) and ascorbic acid (AA) and to compare it to the result using LR models (linear and quadratic).

A fractional factorial design was followed in order to find the growth limits of *L. monocytogenes*. Two hundred and Thirty two different conditions were chosen for the model with eight replicates per condition, of which we have eliminated those that were far removed from the growth/no-growth range, so that we have considered 305 data to form the training group, 57% and 234 pieces of data to form the generalization group. This experimental design was intended to explore the survival/death interface. Data were collected at concentrations of citric and ascorbic acid between 0 and 0.4% (w/v), at 4, 7, 10, 15 and 30 °C with a pH range of 4.5–6 in order to homogenize the values of the input nodes that were scaled over the range $[0.1, 0.9]$.

A backward stepwise conditional method was used for determination modelling, where the coefficient estimates, and statistics with the significant ($P > 0.05$) effects are shown in Table 6. For the training set, the degree of agreement between predictions and observations was 81.3% for LR and 93.1% for the LRLPU model. For the test set 78.2%

Table 5
Mean tree size (number of leaves) and standard deviation for LMT, LotusS and LotusM

Dataset	LMT	LotusS	LotusM
Cancer	1.24 ± 0.99	4.51 ± 1.89	1.02 ± 0.14
Card	1.15 ± 0.59	2.58 ± 0.75	2.03 ± 0.30
German	1.00 ± 0.00	2.37 ± 0.65	2.11 ± 0.40
Heart	1.04 ± 0.32	3.57 ± 1.86	1.00 ± 0.00
Ionosphere	4.40 ± 1.86	3.59 ± 0.85	1.00 ± 0.00
Pima	1.03 ± 0.30	2.86 ± 0.97	1.00 ± 0.00
Sonar	2.12 ± 1.51	1.65 ± 0.95	1.00 ± 0.00

Table 6
Rate of the number of cases that were classified correctly in the recognition of growth/no growth of *Listeria monocytogenes* with LR and LRLPU models

	Training		PCR	Generalization		PCR
Predicted	$Y=1$	$Y=0$		$Y=1$	$Y=0$	
Target			LR			
$Y=1$	99	35	73.9	76	30	71.7
$Y=0$	22	149	87.1	21	107	83.6
CCR			81.3			78.2
			LRLPU			
$Y=1$	123	11	91.8	95	11	89.6
$Y=0$	10	161	94.2	9	119	93.0
CCR			**93.1**			**91.5**

Table 7
Best LRLPU model to growth/no growth of *Listeria monocytogenes*

Variable	AA^*	AC^*	B_1	B_2	B_3	B_4	B_5
Listeria							
Coeff.	−16.769	−11.741	34.707	120.219	−290.944	−14.709	−0.00032
Std error	4.526	2.859	5.827	20.225	48.996	3.513	5.657e-5
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000

$B_1 = (T^*)^{0.417}$; $B_2 = (pH^*)^{1.653}(AA^*)^{-0.915}$; $B_3 = (pH^*)^{1.777}(AA^*)^{-0.767}(AC^*)^{0.262}$
$B_4 = (pH^*)^{-0.111}(AA^*)^{0.270}(AC^*)^{0.089}$; $B_5 = (T^*)^{0.139}(pH^*)^{0.654}(AA^*)^{-3.532}(AC^*)^{-1.890}$
#coefficients = 19.

is for LR and 91.2% for LRLPU. The fact that there were 20 coefficients estimated should be noted: seven were associated with the regression coefficients and 13 with those of the product-unit basis functions, which is not excessive if we take into account that a complete quadratic polynomial model has 15 coefficients. The existence of 5 basis functions (see Table 7) shows a strong interaction among the four factors or covariates that determine the growth limits of *L. monocytogenes*.

Considering these results the model proposed using LRLPU is again the best whether using linear LR or not. Moreover, it is a sufficiently short model as far as the number of coefficients is concerned to allow a greater degree of interpretability.

6. Conclusions

In this paper, we focus on binary classification problems. We propose a LR method based on the hybridation of linear models and a special class of feed-forward neural network, namely product-unit neural networks. The nonlinear basis functions express the possible strong interactions between the covariates. The methodology to estimate the coefficients of the model is based on the combination of an EA to determine the basic structure of the product-unit model and a local optimization procedure carried out by standard logistic regression to find the final model. We include a backward stepwise method to select the best covariates to explain the response. In this way, we control the number of coefficients in the final model and we decrease the risk of building overly complex models that over fit the training data. The proposed model has been applied to seven benchmark data sets and a hard real microbiological world problem. The results obtained outperform the linear model constructed by means of logistic regression with initial covariates and also the nonlinear part of the model obtained with a logistic regression with all the product-unit covariates. In this way, the hybrid model determines a good balance between the linear and nonlinear part. Furthermore, the empirical results show that the proposed hybrid model is very promising in terms of classification accuracy and simplicity as well as very efficient in terms of the total number of coefficients and basis functions needed for constructing the final binary classifiers and yielding a state-of-the-art performance. Moreover, we can show the best model for each classification problem. It is important to note that the number of coefficients of the best models is considerably lower than in the alternative techniques considered.

Acknowledgments

This work has been financed in part by the TIC2002-04036-C05-02 and TIN 2005-08386-C05-02 projects of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT) and FEDER funds.

References

[1] R.L. Prentice, R. Pike, Logistic disease incidence models and case-control studies, Biometrika 66 (1979) 403–411.
[2] M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, New York, 1995.
[3] J. Friedman, W. Stuetzle, Projection pursuit regression, J. Am. Stat. Assoc. 76 (1981) 817–823.
[4] T.J. Hastie, R.J. Tibshirani, Generalized Additive Models, Chapman & Hall, London, 1990.
[5] J. Friedman, Multivariate adaptive regression splines (with discussion), Ann. Stat. 19 (1991) 1–141.
[6] R. Durbin, D. Rumelhart, Products units: a computationally powerful and biologically plausible extension to backpropagation networks, Neural Comput. 1 (1989) 133–142.
[7] A.A. Freitas, Data Mining and Knowledge Discovery with Evolutionary Algorithms, Springer, Berlin, 2002.
[8] E. Sommer, An approach to quantifying the quality of induced theories, Proceeding of the IJCAI 95—Workshop on Machine Learning and Comprehensibility, 1995.
[9] T. Endou, Q. Zhao, Generation of comprehensible decision trees through evolution of training data, in: Proceeding of the Congress on Evolutionary Computation, 2002.
[10] C. Blake, C.J. Merz, UCI repository of machine learning data bases, ⟨http://www.ics.uci.edu/mlearn/MLRepository.html⟩, 1998.
[11] M. Smith, R. Khon, A Bayesian approach to nonparametric bivariate regression, J. Amer. Stat. Assoc. 92 (1997) 1522–1535.
[12] R. Kohn, M. Smith, D. Chan, Nonparametric regression using linear combinations of basis functions, Stat. Comput. (2001) 313–322.
[13] P. Gustafson, Bayesian regression modelling with interactions and smooth effects, J. Amer. Stat. Assoc. 95 (2000) 795–806.
[14] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1999.
[15] Y. Freund, R. Shapire, Experiments with a new boosting algorithm, Machine Learning: Proceedings of the 13th International Conference, San Francisco, 1996.
[16] M. Schumacher, R. Robner, W. Vach, Neural networks and logistic regression: part I, Comput. Stat. Data Anal. 21 (1996) 661–682.
[17] W. Vach, R. Robner, M. Schumacher, Neural networks and logistic regression: part II, Comput. Stat. Data Anal. 21 (1996) 683–701.
[18] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting, Department of Statistics, Standford University, Technical Report, 1998.
[19] G. Lebanon, J. Lafferty, Boosting and maximum likelihood for exponential models, in: Proceedings of NIPS, 2002.
[20] D.W. Hosmer, S. Lemeshow, Applied Logistic Regression, Wiley, New York, 1989.
[21] T.P. Ryan, Modern Regression Methods, Wiley, New York, 1997.
[22] G. McLachlan, Discriminant Analysis and Statistical Pattern Recognition, Wiley, New York, 1992.
[23] T. Hastie, R.J. Tibshirani, J. Friedman, The elements of Statistical Learning. Data Mining, Inference and Prediction, Springer, Berlin, 2001.
[24] M. Schmitt, On the complexity of computing and learning with multiplicative neural networks, Neural Comput. 14 (2001) 241–301.
[25] A. Ismail, A.P. Engelbrecht, in: V.B. Bajic, D. Sha (Eds.), Training Products Units in Feedforward Neural Networks using Particle Swarm Optimisation, Durban, South Africa, 1999.
[26] A.P. Engelbrecht, A. Ismail, Global optimization algorithms for training product units neural networks, in: International Joint Conference on Neural Networks IJCNN'2000, 2000.
[27] D.J. Janson, J.F. Frenzel, Training product unit neural networks with genetic algorithms, IEEE Expert 8 (1993) 26–33.
[28] A.P. Engelbrecht, A. Ismail, Pruning product unit neural networks, in: Proceedings of the International Conference on Neural Networks, Honolulu, Hawai, 2002.

[29] L.R. Leerink, C.L. Giles, B.G. Horne, et al., Learning with products units, Adv. Neural Networks Process. Syst. 7 (1995) 537–544.
[30] K. Saito, R. Nakano, MDL regularizer: a new regularizer based on MDL principle, in: Proceedings of International Conference on Neural Networks ICNN97, 1997.
[31] K. Saito, R. Nakano, Partial BFGS update and calculating optimal step-length for three-layer neural networks, Neural Comput. 9 (1997) 123–141.
[32] P.J. Angeline, G.M. Saunders, J.B. Pollack, An evolutionary algorithm that constructs recurrent neural networks, IEEE Trans. Neural Networks 5 (1) (1994) 54–65.
[33] X. Yao, Y. Liu, A new evolutionary system for evolving artificial neural networks, IEEE Trans. Neural Networks 8 (3) (1997) 694–713.
[34] N. García-Pedrajas, C. Hervás-Martínez, J. Muñoz-Pérez, Multi objective cooperative coevolution of artificial neural networks, Neural Networks 15 (2002) 1255–1274.
[35] S. Kirkpatric, C.D.J. Gellat, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[36] A.C. Martínez-Estudillo, F.J. Martínez-Estudillo, C. Hervás-Martínez, et al., Hybridation of evolutionary algorithms and local search by means of a clustering method, IEEE Trans. Syst. Man Cybernet. Part. B: Cybernet. 36 (3) (2006) 534–546.
[37] A.C. Martínez-Estudillo, F.J. Martínez-Estudillo, C. Hervás-Martínez, et al., Evolutionary product unit based neural networks for regression, Neural Networks 19 (2006) 477–486.
[38] I. SPSS ©, "SPSS ©11.0 advanced models," Inc, S., Ed. Chicago, IL, 1999.
[39] N. Landwehr, M. Hall, F. Eibe, Logistic model trees, Machine Learning 59 (2005) 161–205.
[40] R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kauffman, Los Atlos, CA, 1993.
[41] L. Breiman, H. Friedman, J.A. Olshen, et al., Classification and Regression Trees, Wadsworth, Belment, CA, 1984.
[42] J. Gama, Functional trees, Machine Learning 55 (2004) 219–250.
[43] K.Y. Chan, W.Y. Loh, LOTUS: an algorithm for building accurate and comprehensible logistic regression trees, J. Comput. Graphical Stat. 13 (2004) 826–852.
[44] Y. Wang, I. Witten, Inducing model trees for continuous classes, in: Proceedings of Poster Papers, European Conference on Machine Learning, Prague, Czech Republic, 1997.
[45] L.R. Beuchat, Listeria monocytogenes: incidence on vegetables, Food Control 7 (1996) 223–228.
[46] D.R. Fenlon, J. Wilson, W. Donachie, The incidence and level of Listeria monocytogenes contamination of food sources at primary production and initial processing, J. Appl. Bacteriol. 81 (1996) 641–650.
[47] S. Tienungoon, D.A. Ratkowsky, T.A. McMeekin, et al., Growth limits of Listeria monocytogenes as a function of temperature, pH, NaCl, and lactic acid, Appl. Env. Microbiol. (2000) 4979–4987.
[48] European Commission, Opinion of the scientific committee on veterinary measures relating to public health on Listeria monocytogenes, ⟨http://www.europa.eu.int/comm/food/fs/sc/scv/out25⟩, 1999.
[49] M.N. Hajmeer, I.A. Basheer, A hybrid Bayesian-neural network approach for probabilistic modeling of bacterial growth/no-growth interface, Int. J. Food Microbiol. 82 (2003) 233–243.
[50] O.L. Mangasarian, W.H. Wolberg, Cancer diagnosis via linear programming, SIAM News 5 (1990) 1–18.
[51] R. Detrano, A. Janosi, W. Steinbrunn, et al., Amer. J. Cardiol. 64 (1989) 304–310.
[52] V.G. Sigillito, S.P. Wing, L.V. Hutton, et al., Classification of radar returns from the ionosphere using neural networks, Johns Hopkins APL Technical Digest 10 (1989).
[53] J.W. Smith, J.E. Everhart, W.C. Dickson, et al., Using the ADAP learning algorithm to forecast the onset of diabetes mellitus, in: Proceedings of the Symposium on Computer Applications and Medical Care, 1988.
[54] R.P. Gorman, T.J. Sejnowski, Analysis of hidden units in a layered network trained to classify sonar targets, Neural Networks 1 (1988) 75–89.

About the Author—CÉSAR HERVÁS MARTÍNEZ was born in Cuenca, Spain. He received the B.S. degree in statistics and operating 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 a Professor with the University of Córdoba in the Department of Computing and Numerical Analysis in the area of computer science and artificial intelligence and an Associate Professor in the Department of Quantitative Methods in the School of Economics. His current research interests include neural networks, evolutionary computation, and the modelling of natural systems.

About the Author—FRANCISCO J. MARTÍNEZ ESTUDILLO was born in Villacarrillo, Jaén, Spain. He received a M.Sc. Degree in Mathematics in 1987 and his Ph.D. degree in Mathematics in 1991, speciality Differential Geometry, both from the University of Granada, Granada, Spain. From 1987 to 2002, he developed his research in non-Euclidean geometry, Lorentz spaces and maximal surfaces. He is currently a professor in the Department of Management and Quantitative Methods in ETEA, University of Córdoba, Spain. His current areas of interest include neural networks and evolutionary computation.