Available online at www.sciencedirect.com

ScienceDirect

Neurocomputing 72 (2008) 548–561

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Evolutionary product-unit neural networks classifiers

F.J. Martínez-Estudillo[a,*], C. Hervás-Martínez[b], P.A. Gutiérrez[b], A.C. Martínez-Estudillo[a]

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

Received 13 April 2007; received in revised form 26 October 2007; accepted 25 November 2007
Communicated by G.-B. Huang
Available online 15 January 2008

Abstract

This paper proposes a classification method based on a special class of feed-forward neural network, namely product-unit neural networks. Product-units are based on multiplicative nodes instead of additive ones, where the nonlinear basis functions express the possible strong interactions between variables. We apply an evolutionary algorithm to determine the basic structure of the product-unit model and to estimate the coefficients of the model. We use softmax transformation as the decision rule and the cross-entropy error function because of its probabilistic interpretation. The approach can be seen as nonlinear multinomial logistic regression where the parameters are estimated using evolutionary computation. The empirical and specific multiple comparison statistical test results, carried out over several benchmark data sets and a complex real microbial *Listeria* growth/no growth problem, show that the proposed model is promising in terms of its classification accuracy and the number of the model coefficients, yielding a state-of-the-art performance.
© 2008 Elsevier B.V. All rights reserved.

Keywords: Classification; Product-unit neural networks; Evolutionary neural networks

1. Introduction

The simplest method for the classification of patterns provides the class level given their observations via linear functions in the predictor variables. This process of model fitting is quite stable, resulting in low variance but a potentially high bias. Frequently, in a real-problem of classification, we cannot make the stringent assumption of additive and purely linear effects of the variables. A traditional technique to overcome these difficulties is augmenting/replacing the input vector with new variables, the basis functions, which are transformations of the input variables, and then using linear models in this new space of derived input features. One first approach is to augment the inputs with polynomial terms to achieve higher-order Taylor expansions, for example, with quadratic terms and multiplicative interactions. Once the number and the structure of the basis functions have been determined, the models are linear in these new variables and their fitting is a standard procedure. Methods like sigmoidal feed-forward neural networks [6], projection pursuit learning [23], generalized additive models [31], and PolyMARS [43], a hybrid of multivariate adaptive splines (multiadaptive regression splines, MARS) [22], specifically designed to handle classification problems, can be seen as different basis function models. The major drawback of these approaches is stating the optimal number and typology of corresponding basis functions.

We tackle this problem proposing a nonlinear model along with an evolutionary algorithm (EA) that finds the optimal structure of the model and estimates its corresponding coefficients. Concretely, our approach tries to overcome the nonlinear effects of the input variables by means of a model based on nonlinear basis functions constructed with the product of the inputs raised to arbitrary powers. These basis functions express possible strong interactions between the variables, where the exponents may even take on real values and are suitable for automatic adjustment. The model proposed

*Corresponding author.
E-mail addresses: fjmestud@etea.com (F.J. Martínez-Estudillo), chervas@uco.es (C. Hervás-Martínez), i02gupep@uco.es (P.A. Gutiérrez), acme@etea.com (A.C. Martínez-Estudillo).

corresponds to a special class of feed-forward neural network, namely product-unit based neural networks (PUNN) introduced by Durbin and Rumelhart [16]. They are an alternative to sigmoidal neural networks and are based on multiplicative nodes instead of additive ones. Unfortunately, the error surface associated with PUNNs is extremely convoluted with numerous local optima and plateaus. The main reason for this difficulty is that small changes in the exponents can cause large changes in the total error surface. Because of this, their training is more difficult than the training of standard sigmoidal-based networks. For example, it is well known [7] that back-propagation is not efficient in the training of product-units. Section 3 will briefly show the most relevant techniques that have been used so far to apply learning methods to product-unit networks.

On the other hand, the evolutionary approach is used to optimize both the weights and the architecture of the network simultaneously. In general, classical neural networks training algorithms assume a fixed architecture; nevertheless, it is very difficult to know beforehand what the most suitable structure of the network for a given problem will be. There have been many attempts to design the architecture automatically, such as constructive and pruning algorithms [51,56]. We used EAs to design a nearly optimal neural network architecture because better results have been obtained using this heuristic [4,62]. This fact, together with the complexity of the error surface associated with a PUNN, justifies the use of an EA to design the topology of the network and to train its corresponding weights. The evolutionary process determines the number of basis functions, associated coefficients and corresponding exponents in the model.

We use the softmax activation function and the cross-entropy error function [6]. Therefore, from a statistical point of view, the approach can be seen as a nonlinear multinomial logistic regression [30], where we optimize log-likelihood using evolutionary computation. Actually, we attempt to estimate conditional class probabilities using a multilogistic model with the nonlinear model given by PUNNs.

We evaluate the performance of our methodology on seven data sets taken from the UCI repository [7], and on a real microbial growth/no growth problem in order to determine the growth limits of *Listeria monocytogenes* [2,18] to assure microbial safety and quality in foods. Empirical and statistical test results show that the proposed method performs well when compared to several other learning classification techniques. We obtain a classifier with interesting results in terms of classification accuracy and number of hidden nodes. Moreover, we show graphically the classification task carried out by the product-unit model together with its capability to both capture the interactions between the variables and to reduce the dimension of the input space. This reduction of dimensionality facilitates the study of the behavior of corresponding basis functions and the relevance of each input variable in the final model. This paper is organized as follows: Section 2 shows the main related works; Section 3 is devoted to a description of PUNNs; Section 4 describes the evolution of PUNNs; Section 5 explains the experiments and the comparison test carried out; and finally, Section 6 summarizes the conclusions of our work.

2. Related works

We start by giving a brief overview of the different methods that use basis functions to move beyond linearity. The first method cited to solve classification problems is conventional statistical discriminant analysis [30], which assumes that the measurement vectors in each class follow a normal multivariate distribution. If the covariance matrices of the measurements in each class are the same, the method shows that the regions created by Bayes' decision rule are separated by boundaries, which are linear in the input variables. When the conventional assumption of the equality of covariate matrices is dropped, Bayes' decision rule gives quadratic boundaries. In many examples, the inadequacy of linear or quadratic discriminant analysis for the purpose of classification made it necessary to look for approaches that could approximate highly nonlinear class boundaries. Instead of assuming specific distributions for the inputs and using them to calculate conditional class probabilities, one can estimate these classes directly from training sample cases.

A number of methods based on nonparametric regression [30], which are capable of approximating highly nonlinear class boundaries in classification problems, have been developed in the last few years.

Generalized additive models [31] comprise automatic and flexible statistical methods that may be used to identify and characterize nonlinear effects. The generalized additive model approximates multidimensional functions as a sum of univariate curves. 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 in order to enter them into the model independently of each other. Generalized additive models provide a natural first approach to relaxing strong linear assumptions.

Bose [8] presented a method, classification using splines (CUS), somewhat similar to the neural network method, which uses additive cubic splines to estimate conditional class probabilities. Afterwards, the same author presented a modification of CUS, named "the method of successive projections", to solve more complex classification problems [9]. Although this method was presented using CUS, it is possible to replace CUS by any nonparametric regression-based classifier.

Kooperberg et al. [43] propose an automatic procedure that uses linear splines and their tensor products. This method is a hybrid of the MARS [22] called PolyMars, specifically designed to handle classification problems. It grows the model in a forward stage-wise fashion like MARS, but at each stage it uses quadratic approximation to multinomial log-likelihood to search for the next basis function pair. Once found, the enlarged model is fit according to the maximum likelihood method, and the process is repeated.

From a different point of view, neural networks have been an object of renewed interest among researchers, both in statistics and computer science, owing to the significant results obtained in a wide range of classification problems. Many different types of neural network architectures have been used, but the most popular one has been that of the single-hidden-layer feed-forward network. Amongst the numerous approaches using neural networks in classification problems, we focus our attention on that of evolutionary artificial neural networks (EANNs). EANNs have been a key research area in the past decade providing a better platform for simultaneously optimizing both network performance and architecture. Miller et al. [48] proposed evolutionary computation as a very good candidate to search the space of architectures because the fitness function associated with that space is complex, noisy, nondifferentiable, multimodal and deceptive. Since then, many evolutionary methods have been developed to evolve artificial neural networks, see, for example, [12,19,24,52,60,62,63]. In these works we find several methods that combine architectural evolution with weight learning and use different mutation operators, including, in some cases, partial training after each architectural mutation or approaches that hybridize EANNs with a local search technique to improve the slowness of the convergence. The problem of finding suitable architecture and the corresponding weights of the network is a very complex task (for a very interesting review of the matter the reader can consult [61]). Among the EANN paradigm it is worthwhile to point out two different approaches. Cooperative co-evolution [50] is a recent paradigm in the area of evolutionary computation focused on the evolution of co-adapted subcomponents without external interaction [24]. COVNET is a new cooperative co-evolutionary model for evolving artificial neural networks [25]. The method is based on the idea of co-evolving subnetworks that must cooperate to form a solution for a specific problem, instead of evolving complete networks.

On the other hand, multiobjective evolutionary optimization has recently appeared as an enhanced approach to optimize both the structure and weights of the neural network [5,27]. The idea of designing neural networks within a multiobjective setup was first considered by Abbass in [1]. Here, the multiobjective problem formulation essentially involves setting up two objectives: complexity of the network (number of weights, number of connections or a combination of both) and the training error. Finally, an interesting idea has been developed in subsequent works related to the topic of network ensembles [11,13,14]. In these papers, the authors tackle the ensemble-learning problem within a multiobjective setup where diversity and accuracy objectives are in conflict and the evolutionary process searches for a good trade-off between them.

Finally, a new learning algorithm called extreme learning machine (ELM) for single hidden-layer feed-forward neural networks has been recently proposed [35,36]. This novel procedure, unlike the conventional implementations of gradient-based learning algorithms, chooses randomly hidden nodes and analytically determines the output weights of the network. This algorithm provides good generalization performances at extremely fast learning speeds and it has been proved in theory that the universal approximator property holds [34]. However, ELM may need a higher number of hidden nodes due to random determination of input weights and hidden biases. Several algorithms based on the ELM method (hybrid proposals which use the differential evolution algorithm [64] and a convex optimization method [33]) have been developed to achieve good generalization performances with more compact networks.

3. Product-unit neural networks

In this section we present the family of product-unit basis functions used in the classification process and its representation by means of a neural network structure. This class of multiplicative neural networks comprises such types as sigma–pi networks and product-unit networks. A multiplicative node is given by

$$y_j = \prod_{i=1}^{k} x_i^{w_{ji}}, \qquad (1)$$

where k is the number of the inputs. If the exponents w_{ji} in (1) are $\{0,1\}$ we obtain a higher-order unit, also known as sigma–pi unit. In contrast to the sigma–pi unit, in the product-unit the exponents are not fixed and may even take real values.

Some advantages of PUNNs are their increased information capacity and ability to form higher-order input combinations. Durbin and Rumelhart [16] determined empirically that the information capacity of product-units (measured by their capacity for learning random Boolean patterns) is approximately $3N$, as compared to $2N$ in a network with additive units for a single threshold logic function, where N denotes the number of inputs to the network. Besides, it is possible to obtain the upper bounds of the VC dimension [58] in PUNNs similar to those obtained in sigmoidal neural networks [55]. Schmitt [55] derives the upper bounds of the VC dimension and the pseudo-dimension for various types of networks with multiplicative units. As the most general case, Schmitt [55] considers feed-forward networks consisting of product and sigmoidal units, and showing that their pseudo-dimension is bounded from above by a polynomial with the same order of magnitude as the currently best known bound for purely sigmoidal networks. Concretely, he shows that a product-unit network has a pseudo-dimension at least $O(W^2k^2)$, where the number of parameters is W and k is the number of hidden nodes (see Theorem 1 and Corollary 2). As a consequence, the complexity of a PUNN (from the point of view of the VC dimension or the pseudo-dimension) depends on the number of parameters and the number of hidden nodes instead of on the values of the weights.

Finally, it is a straightforward consequence of the Stone–Weierstrass theorem to prove that PUNNs are universal approximators [46] (observe that polynomial functions in several variables are a subset of product-unit models).

Despite these advantages, PUNNs have a major drawback. They have more local minima and a higher probability of becoming trapped in them [38]. Several efforts have been made to carry out learning methods for product-units. Janson and Frenzel [41] developed a genetic algorithm for evolving the weights of a network based on product-units with a predefined architecture. The major problem with this kind of algorithm is how to obtain the optimal architecture beforehand. Ismail and Engelbrecht [38,39] applied four different optimization methods to train PUNNs: random search, particle swarm optimization, genetic algorithms and leapfrog optimization. They concluded that random search is not efficient for training this type of network, and that the other three methods show an acceptable performance in three problems of function approximation with low dimensionality. In a later paper [40] they used a pruning algorithm to develop both the structure and the training of the weights in a PUNN. Leerink et al. [44] tested different local and global optimization methods for PUNNs. Their results show that local methods, such as back-propagation, are prone to be trapped in local minima, and that global optimization methods, such as simulated annealing and random search, are impractical for larger networks. They suggested some heuristics to improve back-propagation, and the combination of local and global search methods. In short, the studies carried out on PUNNs have not tackled the problem of the simultaneous design of the structure and weights in this kind of neural network, using either classical or evolutionary based methods. Moreover, PUNNs have been applied mainly to solve regression problems so far [17,46,47,53,54].

On the other hand, it is interesting to note that a problem arises with networks containing product-units that receive negative inputs and have weights that are not integers. A negative number raised to some noninteger power yields a complex number. Since neural networks with complex outputs are rarely used in applications, Durbin and Rumelhart [16] suggest discarding the imaginary part and using only the real component for further processing. This manipulation would have disastrous consequences for the VC dimension when we consider real-valued inputs. No finite dimension bounds can, in general, be derived for networks containing such units [55].

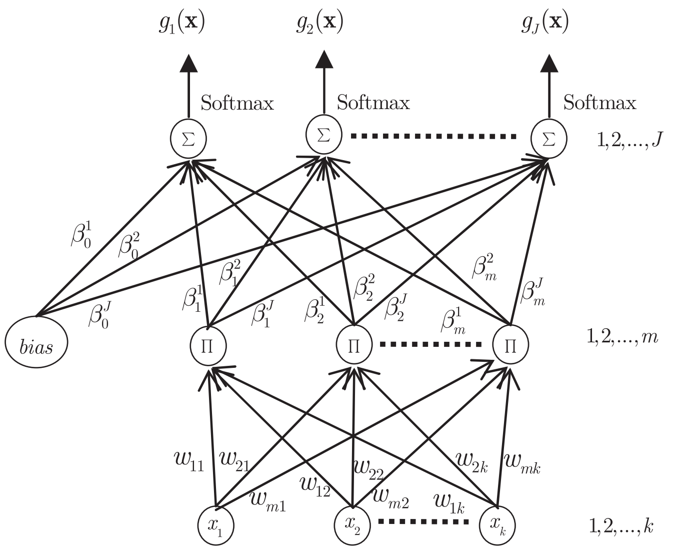

Fig. 1. Model of a product-unit neural network.

To avoid this problem, the input domain is restricted, and we define the set given by $\{\mathbf{x} = (x_1, x_2, ..., x_k) \in \mathbb{R}^k : x_i > 0, i = 1, 2, \ldots, k\}$.

We consider a PUNN with the following structure (Fig. 1): an input layer with k nodes, a node for every input variable, a hidden layer with m nodes and an output layer with J nodes, one for each class level. There are no connections between the nodes of a layer, and none between the input and output layers either. The activation function of the 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 and $\mathbf{w}_j = (w_{j1}, \ldots, w_{jk})$ the weights vector. The activation function of the output node l is given by $\beta_0^l + \sum_{j=1}^{m} \beta_j^l B(\mathbf{x}, \mathbf{w}_j)$, where β_j^l is the weight of the connection between the hidden node j and the output node l and β_0^l the corresponding bias. The transfer function of all hidden and output nodes is the identity function. In this way, the estimated function $f_l(\mathbf{x}; \boldsymbol{\theta}_l)$ from each output is given by

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

where $\boldsymbol{\theta}_l = (\boldsymbol{\beta}^l, \mathbf{w}_1, \ldots, \mathbf{w}_m)$ and $\boldsymbol{\beta}^l = (\beta_0^l, \beta_1^l, \ldots, \beta_m^l)$.

4. Classification problem

In a 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 the J classes based on these measurements. It is assumed that J is finite, and the measurements x_{i} are random observations from these classes.

A training sample $D = \{(\mathbf{x}_n, \mathbf{y}_n); n = 1, 2, \ldots, N\}$ is available, where $\mathbf{x}_n = (x_{1n}, \ldots, x_{kn})$ is the random vector of measurements taking values in $\Omega \subset \mathbb{R}^k$, and $\mathbf{y}_n$ is the class level of the n-th individual. We adopt the common technique of representing the class levels using a "1-of-J" encoding vector $\mathbf{y} = (y^{(1)}, y^{(2)}, \ldots, y^{(J)})$, such as $y^{(l)} = 1$ if $\mathbf{x}$ corresponds to an example belonging to class l, and otherwise $y^{(l)} = 0$. Based on the training sample we wish to find a decision function $C : \Omega \to \{1, 2, \ldots, J\}$ for classifying individuals. In other words, Ω provides a partition, say $D_1, D_2, \ldots, D_J$, of Ω, where D_l corresponds to the l class, $l = 1, 2, \ldots, J$, and measurements belonging to D_l will be classified as coming from the l-th class. A misclassification occurs when a decision rule Ω assigns an individual (based on measurements vector) to a class j when it actually comes from a class $l \neq j$. We define the corrected classification rate (CCR) by $\mathrm{CCR} = (1/N)\sum_{n=1}^{N} I(C(\mathbf{x}_n) = \mathbf{y}_n)$, where $I(\bullet)$ is the zero-one loss function. A good classifier tries to achieve the highest possible CCR in a given problem.

We consider the softmax activation function [6] given by

$$g_l(\mathbf{x}, \boldsymbol{\theta}_l) = \frac{\exp f_l(\mathbf{x}, \boldsymbol{\theta}_l)}{\sum_{l=1}^{J} \exp f_l(\mathbf{x}, \boldsymbol{\theta}_l)}, \quad l = 1, 2, \ldots, J. \tag{3}$$

If we use the training data set $D = \{(\mathbf{x}_n, \mathbf{y}_n)\}$, where $x_{in} > 0\ \forall i, n$, then the cross-entropy error function (K-class multinomial deviance) for those observations is

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

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J)$. The error surface associated with the model is very convoluted with numerous local optimums and the Hessian matrix of the error function $l(\boldsymbol{\theta})$ is, in general, indefinite. Moreover, the optimal number of basis functions in the model (i.e. the number of hidden nodes in the neural network) is unknown. Thus, we determine the estimation of the vector parameters $\hat{\boldsymbol{\theta}}$ by means of an EA (see Section 5).

The optimum rule $C(\mathbf{x})$ is the following:

$$C(\mathbf{x}) = \hat{l}, \quad \text{where } \hat{l} = \arg\max_{l} g_l(\mathbf{x}, \hat{\boldsymbol{\theta}}), \\ \text{for } l = 1, 2, \ldots, J. \tag{5}$$

From a statistical point of view, with the softmax activation function (3) and the cross-entropy error (4), the neural network model can be seen as a multilogistic regression model. Nevertheless, the nonlinearity of the model with respect to the parameters $\boldsymbol{\theta}_j$ and the indefinite character of the associated Hessian matrix do not recommend the use of gradient-based methods (for example, iteratively reweighted least squares (IRLS) commonly used in the optimization of log-likelihood in linear multinomial logistic regression) to minimize the negative log-likelihood function.

Observe that softmax transformation produces positive estimates that sum to one and, therefore, the outputs can be interpreted as conditional probability of class membership. Specifically, the probability that $\mathbf{x}$ belongs to class l is written as

$$p(y^{(l)} = 1 | \mathbf{x}, \boldsymbol{\theta}_j) = g_l(\mathbf{x}, \boldsymbol{\theta}_l), \quad l = 1, 2, \ldots, J \tag{6}$$

and the classification rule (5) coincides with the optimal Bayes rule. In other words, an individual should be assigned to the class which has the maximum probability, given vector measurement $\mathbf{x}$. On the other hand, because of the normalization condition

$$\sum_{l=1}^{J} p(y^{(l)} = 1 | \mathbf{x}, \boldsymbol{\theta}_l) = 1, \tag{7}$$

the probability for one of the classes does not need to be estimated. There is a redundancy in the functions $f_l(\mathbf{x}, \boldsymbol{\theta}_l)$, since adding an arbitrary $h(\mathbf{x})$ to each output leaves the model (3) unchanged. Traditionally one of them is set to zero and we reduce the number of parameters to estimate. With loss generality, we set $f_J(\mathbf{x}, \boldsymbol{\theta}_J) = 0$.

5. Evolutionary algorithm

We apply an evolutionary neural networks algorithm to estimate the parameter that minimizes the cross-entropy error function. EA designs the structure and learns the weights of PUNNs. The search begins with an initial population of PUNNs, and in each iteration the population is updated using a population-update algorithm. The population is subjected to the operations of replication and mutation. Crossover is not used due to its potential disadvantages in evolving artificial networks [4,62]. With these features the algorithm falls into the class of evolutionary programming [20,21]. The general structure of EA is similar to the one presented in [32]:

```
(1) Generate a random population of size $N_P$.
(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 population individuals are replicated and substitute the worst 10% of individuals.
        Over that intermediate population we
    (e) Apply parametric mutation to the best 10% of individuals.
    (f) Apply structural mutation to the remaining 90% of individuals.
```

In the current approach, $l(\boldsymbol{\theta})$ is the error function of an individual g of the population, where g is a PUNN given by the multivaluated function $g(\mathbf{x}, \boldsymbol{\theta}) = (g_1(\mathbf{x}, \boldsymbol{\theta}_1), \ldots, g_l(\mathbf{x}, \boldsymbol{\theta}_l))$ and the fitness measure is a strictly decreasing transformation of the error function, $l(\boldsymbol{\theta})$ given by $A(g) = 1/(1 + l(\boldsymbol{\theta}))$.

Parametric mutation is accomplished for each coefficient w_{ji}, β_j^l of the model with Gaussian noise, where the variances of the normal distribution are updated throughout the evolution of the algorithm. Once the mutation is performed, the fitness of the individual is recalculated and the usual simulated annealing process [42,49] is applied. On the other hand, structural mutation implies a modification in the neural network structure and allows explorations of different regions in the search space while helping to keep up the diversity of the population. There are five different structural mutations: node deletion, connection deletion, node addition, connection addition and node fusion. These five mutations are applied sequentially to each network. For further details about the parametric and structural mutations see [32,46].

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

The parameters used in EA are common for all problems. The exponents w_{ji} are initialized in the $[-5,5]$ interval, the coefficients β_j^l are initialized in $[-5,5]$. The maximum number of hidden nodes is $m=6$. The size of the population is $N_P=1000$. 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 $[1,c]$ interval, where c is a third of the number of connections of the model. We consider 400 as the maximum number of generations.

We have done a simple linear rescaling of the input variables in the interval [1,2], X_i^* being the transformed variables. The lower bound is chosen to avoid input values near 0, which can produce very large values of the outputs for negative exponents. The upper bound is chosen to avoid dramatic changes in the outputs of the network when there are weights with large values (especially in the exponents).

6. Experiments

In this section we compare the Evolutionary Product Unit Neural Network method (EPUNN) with different evolutionary neural network learning algorithms. First of all, in order to show the abilities of the product-unit models, we run the same evolutionary setup with standard neural networks with sigmoidal units, and we compare the results of the models based on multiplicative units against additive unit models. Next, we compare our model EPUNN to COVNET, a new cooperative co-evolutionary neural network method. Afterwards, the comparison is made with respect to two recent approaches based on multiobjective evolutionary neural networks: MPANN and SPANN. Furthermore, we analyze in detail two classification models obtained by EA in two data sets, to show graphically the task of classification carried out by the product-unit model and how well it can capture the interactions between the variables as well as reduce the dimension of the input space. Finally, the performance of our model is tested in a complex real microbial *Listeria* growth/no growth problem.

The different experiments were conducted using a software package developed in JAVA by the authors as an extension of JCLEC framework (http://jclec.sourceforge.net/) [59]. The software package is available in the noncommercial JAVA tool named KEEL (http://www.keel.es) [3].

6.1. Neural network models comparisons

6.1.1. Product-units versus additive sigmoidal units

In order to justify the benefits of applying the product-unit model, we evolve the traditional feed-forward neural networks with sigmoidal additive units with the same EA. This approach will be known as evolutionary sigmoidal unit neural networks (ESUNN). The same evolutionary setup (EA and parameters) will be considered for the EPUNN and ESUNN models in seven classification problems with different features (see Table 1). The seven data sets are available by anonymous ftp from [7].

We use a 10-fold stratified cross-validation and we carry out 10 runs of each fold for every data set. This gives a 100 data points for each data set, from which the average classification accuracy in the generalization set (CCR_G) and standard deviation are calculated.

The comparison between the predictive ability of the two models of artificial neural networks (EPUNN and ESUNN) was carried out using statistical tests in terms of accuracy (mean of the correct classification rate, CCR_G), homogeneity (standard deviation, SD, of the CCR_G), best

Table 1
Data sets used for the experiments, sorted by size

Data sets	Instances	Missing values (%)	Numeric attributes	Binary attributes	Nominal attributes	Classes
Heart-statlog	270	0.0	13	0	0	2
Ionosphere	351	0.0	33	1	0	2
Balance	625	0.0	4	0	0	3
Australian	690	0.6[a]	6	4	5	2
Diabetes	768	0.0	8	0	0	2
German	1000	0.0	6	3	11	2
Hypothyroid	3772	5.5[a]	7	20	2	4

[a]We have used original data sets without transforming the missing values.

and worst results and topology (mean and SD of the number of connections). See Table 2.

First, we study the normality of CCR_G and the number of connections by means of a Kolmogorov–Smyrnov test [10]. For this and the remaining comparisons, all tests were obtained using the SPSS statistical package [57].

From the results for CCR_G (see Table 3) a normal distribution can be assumed for Diabetes, German and Hypothyroid because the observed significance levels, p-Values, were higher than the significance level of the test, $\alpha = 0.05$. The normality hypothesis of the accuracy distribution cannot be assumed in Heart-statlog, Ionosphere, Balance, and Australian data sets. Regarding the number of connections, it is possible to assume the normal distribution hypothesis in all cases because p-Values were higher than 0.05. Therefore, under the hypothesis of normal distribution, the comparisons between EPUNN and ESUNN models were done by using the t-test for equal or different variance, according to the results previously obtained by the Levene test [45], in both cases with $\alpha = 0.05$. On the other hand, we use a nonparametric Mann–Whitney test [10] for differences between means in the non-normal distribution hypothesis. The statistical results obtained are shown in Table 4. We observe significant differences, in favor of EPUNN, at a significance level $\alpha = 0.05$ in Balance, Diabetes and German data sets in CCR_G results, while there are not significant differences in the remaining ones. As for the number of connections, there are a significantly lower number of connections for EPUNN in the Heart-statlog, Balance, Australian and Hypothyroid data sets, and there are a significantly lower

Table 4
Statistical comparison (p-Values of the Levene and Student's t-tests or Mann–Whitney test (M–W)) of the generalization ability (CCR_G) and number of connections (# Conn.) for EPUNN and ESUNN models

EPUNN versus ESUNN	CCR_G		♯ Conn.	
	Levene test	t-Test or M–W test	Levene test	t-Test
Heart-statlog	–	0.525	0.000	0.000[a]
Ionosphere	–	0.185	0.000	0.149
Balance	–	0.000[a]	0.300	0.000[a]
Australian	–	0.525	0.003	0.009[a]
Diabetes	0.086	0.047[a]	0.082	0.010[a]
German	0.129	0.000[a]	0.080	0.000[a]
Hypothyroid	0.080	0.618	0.212	0.001[a]

[a]There are significant differences on average using a significance level $\alpha = 0.05$.

Table 2
Statistical results for EPUNN and ESUNN in seven data sets

Data sets	Training				Generalization				♯ Conn.	
	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD
ESUNN										
Heart-statlog	86.21	1.15	88.89	83.54	83.22	6.61	92.59	66.67	16.99	2.52
Ionosphere	91.75	1.36	94.30	88.29	88.66	5.22	100.00	74.29	46.35	8.80
Balance	92.38	1.22	95.74	90.05	91.03	4.15	98.39	79.03	30.31	2.33
Australian	87.81	0.81	90.02	85.35	85.49	3.92	94.20	78.26	49.58	12.66
Diabetes	79.15	0.75	80.78	77.75	75.93	5.25	84.21	63.16	17.02	3.00
German	77.32	1.52	79.67	73.22	73.00	5.56	86.00	60.00	62.84	13.96
Hypothyroid	94.34	0.23	94.82	93.76	94.18	0.95	96.02	92.57	76.73	11.30
EPUNN										
Heart-statlog	84.65	1.63	88.48	80.25	81.89	6.90	96.30	62.96	14.78	3.83
Ionosphere	93.79	1.46	97.15	90.19	89.63	5.52	100.00	74.29	43.97	13.87
Balance	97.26	0.98	99.47	94.32	95.69	2.36	100.00	90.32	25.62	2.18
Australian	87.01	0.82	88.57	85.02	85.74	3.90	95.65	78.26	44.13	16.26
Diabetes	77.79	0.75	79.62	76.16	77.40	4.38	84.21	68.42	18.08	2.31
German	77.33	1.34	80.56	73.56	76.28	4.82	90.00	65.00	74.16	18.82
Hypothyroid	94.51	0.55	96.97	93.64	94.25	1.08	96.55	92.31	71.13	12.69

Table 3
Normality Kolmogorov–Smirnov test of the generalization ability (CCR_G) and number of connections (# Conn.) for EPUNN and ESUNN models

K–S test	Heart-statlog		Ionosphere		Balance		Australian		Diabetes		German		Hypothyroid	
	EPUNN	ESUNN	EPUNN	ESUNN	EPUNN	ESUNN	EPUNN	ESUNN	EPUNN	ESUNN	EPUNN	ESUNN	EPUNN	ESUNN
p-Value CCR_G	0.010[a]	0.001[a]	0.066	0.018[a]	0.002[a]	0.010[a]	0.034[a]	0.255	0.330	0.304	0.647	0.522	0.123	0.076
p-Value # Conn.	0.222	0.157	0.602	0.465	0.108	0.205	0.436	0.689	0.492	0.080	0.624	0.354	0.108	0.205

[a]Non-Gaussian distribution for a significance level $\alpha = 0.05$.

number of connections for ESUNN in the Diabetes and German ones, using a significance level $\alpha = 0.05$

We conclude that for the Heart-statlog, Australian and Hypothyroid data sets, the models using EPUNN have a significantly lower number of connections without significantly worsening or improving the mean value of CCR_G, as compared to those found with the ESUNN model.

For the Ionosphere data set there are no significant differences in the CCR_G and in the number of connections with respect to the type of base functions used.

The Balance data set has a significantly higher mean CCR_G value when using EPUNN models than when using ESUNN models; these models also manifesting a significantly lower mean number of connections. For Diabetes and German data sets the mean CCR_G values obtained with EPUNN models are significantly higher than when ESUNN models are used, although the mean number of connections is also significantly higher in the EPUNN models.

6.1.2. *Comparison with a cooperative co-evolutionary neural network method: COVNET*

In this section we compare our approach with COVNET, a new cooperative co-evolutionary model for evolving artificial neural networks [25]. The method is based on the idea of co-evolving subnetworks that must cooperate to form a solution for a specific problem, instead of evolving complete networks.

To be consistent with [25], the comparison is tested on three classification data sets: Diabetes, Australian and Heart disease. The features of Diabetes and Australian data sets can be seen in Table 1. Heart disease is a data set from the Cleveland Clinic Foundation and contains 13 attributes, 5 classes and 270 examples (the problem is described more deeply in [15]). We have carried out the same experimental design: 75% of patterns of each data set were used for training purposes and the remaining 25% were used as generalization assessment set. Three different random permutations of the patterns were made, and the evolutionary process was repeated 10 times for each permutation.

Table 5 shows, for each permutation of the data sets, the averaged CCR over 10 repetitions for training and generalization sets, the standard deviation, the best and worst individuals, and the mean and standard deviation of the number of connections of the best networks obtained for each experiment. Each permutation of the data set is labelled I, II and III in the table. The mean results over the three permutations are labelled All.

In order to verify the true difference between the performance in EPUNN and the COVNET network, we conducted statistical tests (see Table 6). First, we corroborated by means of a Kolmogorov–Smyrnov test that the

Table 5
Statistical results for EPUNN and COVNET in three data sets

Data sets	Partition	Algorithm	Training				Generalization				♯ Conn.	
			Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD
Diabetes	I	COVNET	77.74	0.73	78.65	76.56	80.05	2.84	83.85	75.52	–	–
		EPUNN	77.48	0.91	78.99	76.22	79.84	0.95	81.25	78.65	20.43	2.80
	II	COVNET	77.31	0.70	78.30	76.04	79.37	2.36	82.29	76.04	–	–
		EPUNN	76.82	0.56	78.99	75.87	81.30	0.79	82.29	80.21	20.90	2.77
	III	COVNET	77.57	0.42	78.12	76.74	80.89	0.82	82.29	79.69	–	–
		EPUNN	76.91	0.61	78.47	76.39	82.03	1.21	84.38	80.73	21.42	2.91
	All	COVNET	77.54	0.63	–	–	80.10	2.20	–	–	24.60	–
		EPUNN	77.07	0.75	–	–	81.06	1.34	–	–	20.94	2.76
Heart disease	I	COVNET	87.43	0.63	88.61	86.63	85	1.35	86.76	82.35	–	–
		EPUNN	86.63	0.70	87.13	85.64	86.62	1.29	88.24	83.82	25.61	3.81
	II	COVNET	86.34	1.42	89.11	84.16	87.94	2.17	91.18	85.29	–	–
		EPUNN	83.76	1.04	87.129	82.18	90.88	1.52	94.12	88.24	28.20	4.10
	III	COVNET	86.88	0.75	87.62	85.64	84.26	3.18	88.24	79.41	–	–
		EPUNN	85.55	1.21	87.62	84.16	83.52	1.67	85.29	80.88	27.30	2.26
	All	COVNET	86.88	1.06	–	–	85.74	2.79	–	–	33.07	–
		EPUNN	85.31	1.55	–	–	87.01	3.39	–	–	27.03	3.54
Australian	I	COVNET	86.25	0.75	87.45	84.94	88.02	0.88	88.95	86.05	–	–
		EPUNN	84.17	0.32	84.36	83.59	88.66	0.63	88.95	87.21	11.31	11.84
	II	COVNET	85.69	0.92	87.64	84.36	88.95	1.19	90.71	86.63	–	–
		EPUNN	84.40	0.30	84.94	83.78	88.84	0.25	88.95	88.37	3.97	1.37
	III	COVNET	85.89	0.59	86.87	85.14	88.31	0.89	89.53	86.63	–	–
		EPUNN	84.48	0.24	84.94	84.36	88.72	0.30	88.95	88.37	6.40	4.95
	All	COVNET	85.95	0.78	–	–	88.43	1.04	–	–	34.23	–
		EPUNN	85.24	0.64	–	–	88.55	1.17	–	–	31.40	12.16

Table 6
Statistical test in three data sets

Data sets	Partition	K–S (p-Values)	t-Test (p-Values)
Diabetes	All	0.637	0.046[a]
Heart disease	All	0.702	0.125
Australian	All	0.061	0.685

[a]There are significant differences between the EPUNN and the COVNET means, for $\alpha = 0.05$.

distribution of the CCR_G values is normal; p-Values in the table are higher than $\alpha = 0.05$, and in this way, the normality hypothesis for the distribution of the CCR_G obtained using EPUNN is accepted. In [25], the normality hypothesis for the distribution of the CCR_G values for COVNET is accepted using the Kolmogorov–Smyrnov test, regarding p-Values presented for Diabetes data set ($\alpha = 0.44$), for Heart disease data set ($\alpha = 0.07$), and for Australian data set, ($\alpha = 0.56$). Then, we tested the hypothesis that the means of the CCR_G obtained from the experiments with EPUNN and COVNET networks do not result in significant differences, and therefore we performed a t-test that allowed us to ascertain if the mean of the CCR_G obtained with EPUNN was significantly higher than the mean of the CCR_G obtained with the COVNET network at a significance level $\alpha = 0.05$.

Table 6 shows that, when we consider the 30 results obtained in the three folds, the improvements on mean for the CCR_G of EPUNN are significant in Diabetes (for $\alpha = 0.05$), while there are not significant differences for the Australian and Heart disease data sets.

6.1.3. Multiobjective neural networks

Now we compare our proposal with some of the most recent algorithms that use the evolutionary neural network paradigm: MPANN (memetic pareto artificial neural network) and SPANN (a self-adaptive version of MPANN) [1]. We have tested EPUNN on two benchmark problems used in the references quoted: the Australian credit card assessment problem and the Diabetes problem. To be consistent with [1,13], we use a 10-fold stratified cross-validation for the Australian data set and a 12-fold stratified cross-validation for the Diabetes data set. For every data set we performed 10 runs of each fold. This gives 100 data points for the Australian, and 120 for the Diabetes data set, from which the average classification accuracy CCR_G and the standard deviation are calculated.

We carry out a standard comparison test to verify if there are significant differences in the CCR_G when comparing the EPUNN model to MPANN/SPANN methodologies. Following the hypothesis of the normality of the results, we carry out a t-test to compare the mean results of CCR_G in the EPUNN and MPANN/SPANN methodologies. We use the mean and standard deviation of the (100, 120) runs for MPANN and SPANN, 10 for each fold, and (100, 120) runs for EPUNN, 10 for each fold.

Table 7
Mean classification accuracy and standard deviation for generalization CCR_G for MPANN, SPANN and EPUNN

	MPANN	SPANN	EPUNN
Australian			
Test error rate	86.40 ± 4.50	86.90 ± 4.60	85.74 ± 3.90
Hidden units	5.00 ± 1.94	6.00 ± 1.83	2.90 ± 0.09
Diabetes			
Test error rate	74.90 ± 6.20	70.70 ± 5.00	77.89 ± 3.48
Hidden units	6.60 ± 1.51	7.17 ± 2.21	2.98 ± 0.14

Table 7 shows the statistical results of EPUNN, MPANN and SPANN for Australian and Diabetes data sets. The p-Values obtained for Australian (0.276 for MPANN and 0.058 for SPANN) and for Diabetes (0.000 for MPANN and 0.000 for SPANN) show that there are no significant differences between the mean results for MPANN and EPUNN algorithms and for SPANN and EPUNN in the Australian data set for a significant coefficient $\alpha = 0.05$; but there are significant differences between the mean results for MPANN and EPUNN algorithms and for SPANN and EPUNN in the Diabetes data set for the same significant coefficient. We conclude that in the latter data set, the EPUNN mean is higher than the MPANN and SPANN means. Moreover, there are differences between the EPUNN model and the MPANN and SPANN methodologies with respect to the number of hidden nodes in both databases, where this number is lower for EPUNN. Table 7 shows that the models constructed by EPUNN are smaller than the models built by the other algorithms. Therefore, the experiments show that the EPUNN algorithm produces models with an interesting trade-off between the accuracy and complexity of the classifier, determined by the low number of hidden nodes, possibly outperforming its interpretability.

Now we study in detail the best product-unit model obtained for the Australian and Diabetes data sets. For the Diabetes data set, we consider the best model of one of the 10-fold used in the experiments (specifically the seventh fold). These models can be easily implemented and the reader can reproduce and compare the results.[1]

The model for the Diabetes data set is determined by three basis functions:

$$B_1 = (X_2^*)^{2.749}(X_6^*)^{0.528},$$
$$B_2 = (X_1^*)^{-1.0785}(X_3^*)^{1.672}(X_5^*)^{0.904}(X_6^*)^{-3.319}(X_7^*)^{-1.349}(X_8^*)^{-1.662},$$
$$B_3 = (X_1^*)^{2.296}(X_2^*)^{-4.677}(X_4^*)^{0.634}(X_6^*)^{3.682}(X_7^*)^{2.832},$$

and the output of the softmax transformation is

$$\hat{g}_1(\mathbf{x}) = \frac{\exp\{-4.179 + 0.961B_1 - 4.569B_2 + 0.262B_3\}}{1 + \exp\{-4.179 + 0.961B_1 - 4.569B_2 + 0.262B_3\}}.$$

[1]The folds used in the cross-validation experiments and the EPUNN models obtained are available for the reader in http://www.uco.es/grupos/ayrna/.

By using the properties of the softmax, the decision rule can be expressed in a more simplified way:

$$C(x) = \begin{cases} 1 & \text{if } 0.961B_1 - 4.569B_2 + 0.262B_3 > 4.179, \\ 0 & \text{if } 0.961B_1 - 4.569B_2 + 0.262B_3 < 4.179. \end{cases}$$

If we observe the best model obtained for the Diabetes data set, and taking into account that the transformed variables X_i^* take values in the same interval $[1,2]$, we see that the most relevant variables are X_2, X_6 and X_7; likewise, X_4 is the least important variable in the model.

On the other hand, we observe that the product-unit model transforms the eight-dimensional input space into a three-dimensional space given by the basis functions $B_1(\mathbf{x})$, $B_2(\mathbf{x})$, and $B_3(\mathbf{x})$. The model tries to capture the interactions among the variables and carries out a reduction in the dimensionality of the space. It is interesting to point out that this reduction allows us to depict the separation of the two classes into training and test points by means of linear functions in the transformed space. Fig. 2 shows the graphics for the training and test points for the Diabetes data set problem and the plane that separates the points in the two classes.

A similar analysis can be carried out for the Australian data set. We consider the best model of one of the 10-fold used in the experiments (the first fold) given by the two basis functions:

$$B_1 = (X_9^*)^{0.783}(X_{11}^*)^{3.142}(X_{12}^*)^{0.426}(X_{18}^*)^{0.661}(X_{26}^*)^{1.065} (X_{42}^*)^{2.745}(X_{43}^*)^{0.419}(X_{45}^*)^{0.112}(X_{49}^*)^{-2.320}(X_{51}^*)^{0.858},$$

$$B_2 = (X_1^*)^{-2.687}(X_3^*)^{-2.005}(X_4^*)^{-2.871}(X_6^*)^{2.676}(X_9^*)^{4.618} (X_{14}^*)^{1.935}(X_{15}^*)^{-1.099}(X_{16}^*)^{-2.957}(X_{17}^*)^{3.366}(X_{25}^*)^{3.688} (X_{27}^*)^{4.551}(X_{28}^*)^{1.164}(X_{30}^*)^{-1.679}(X_{38}^*)^{-0.029}(X_{39}^*)^{1.510} (X_{40}^*)^{3.462}(X_{41}^*)^{-2.613}(X_{42}^*)^{2.415}(X_{43}^*)^{-4.390}(X_{46}^*)^{-4.592} (X_{47}^*)^{-0.070}$$

and the output of the softmax transformation is

$$\hat{g}_1(\mathbf{x}) = \frac{\exp\{-3.230 + 0.416B_1 + 0.119B_2\}}{1 + \exp\{-3.230 + 0.416B_1 + 0.119B_2\}}.$$

The decision rule can be expressed as

$$C(\mathbf{x}) = \begin{cases} 1 & \text{if } 0.416B_1 + 0.119B_2 > 3.230, \\ 0 & \text{if } 0.416B_1 + 0.119B_2 < 3.230. \end{cases}$$

Fig. 3 shows the training and test points graphics for the Australian data set problem and the linear decision boundary that separates the points in the two classes. Observe that we have carried out a logarithmic transformation in the vertical axis to improve the graph. It is important to point out that, in this case, the product-unit model projects the 50-dimensional input space onto a two-dimensional space given by the basis functions $B_1(\mathbf{x})$ and $B_2(\mathbf{x})$. This reduction allows us a graphical analysis of the classification problem, facilitating the study of basis function behavior as well as the relevance of the input variables in the model. For example, the graphics for the Australian data set show that the basis function $B_1(\mathbf{x})$ is more important than the $B_2(\mathbf{x})$ ones in the model. Taking into account, again, that the transformed variables X_i^* take values in the same interval $[1,2]$, we can see that the most

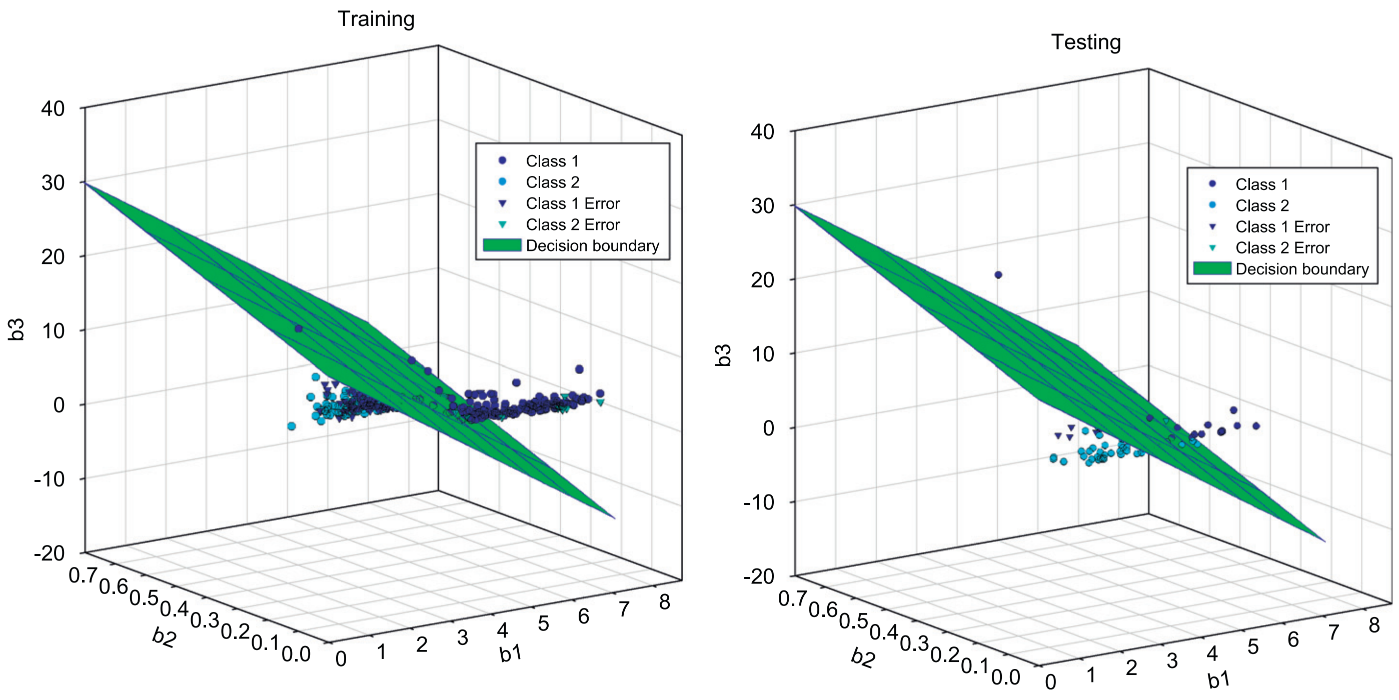

Fig. 2. Graphics of training and test points and decision boundary for the Diabetes data set problem.

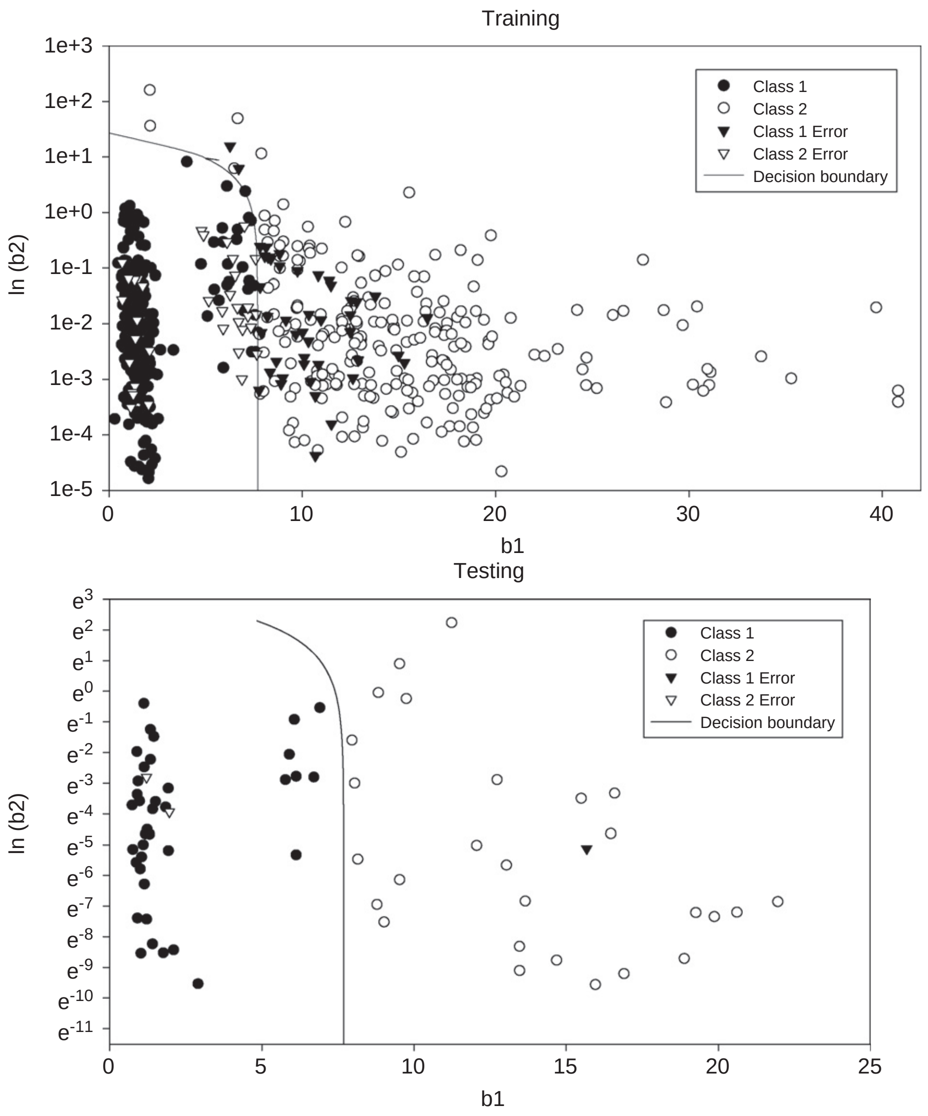

Fig. 3. Graphics of training and test points and decision boundary for the Australian data set problem.

relevant variables are X_{11} and X_{42}. Moreover, a value of $B_1(\mathbf{x})$ higher than 7 provides us with a simple and accurate rule of classification (see Fig. 3). Observe that the graphical study above can be carried out when the final model has two or three nodes in the hidden layer.

6.2. *A real classification problem: Listeria growth/no growth*

The main purpose of this section is to present and compare the efficacy, in terms of good classification, of different logistic regression approaches and the EPUNN on a data set of *L. monocytogenes* growth prediction, as a function of storage temperature *T*, pH, citric acid (CA) and ascorbic acid (AA).

L. monocytogenes has been a serious problem that has concerned food industries due to its ubiquity in the natural environment [2,37] and the specific growth conditions of the pathogen, which lead to its high prevalence in different kinds of food products. One impetus for this research was the problem of listeriosis [26]; so different strategies have been proposed to limit levels of contamination at the time of consumption to less than 100 CFU/g [18].

A fractional factorial design was followed in order to find the growth limits of *L. monocytogenes*. A number of 232 different conditions were chosen for the model with eight replicates per condition, from 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 data to form the generalization group. This experimental design was intended to explore the survival/death interface. Data were collected at concentrations of CA and AA between 0% and 0.4% (w/v), at 4, 7, 10, 15 and 30 °C with a pH range of 4.5–6.

We used two logistic regression methods, a full logistic model, MLogistic, and a backward selected variables

Table 8
CCR obtained for the growth limits of *L. monocytogenes* for MLogistic, SLogistic and EPUNN models

Models	CCR_T	CCR_G
SLogistic	82.30	76.10
Mlogistic	82.30	74.40
EPUNN	92.83	86.72

method, SLogistic, using SPSS software [57], in order to obtain the significant variables of the logistic regression model by stepwise selection. These methods are considered because there has been a renewed interest in the use of these statistical techniques in predictive microbiology (see, for example, [28]). Then, the classification efficiency of the training and generalization data sets of the EPUNN model was compared with MLogistic and SLogistic models. Table 8 shows a mean, in 30 runs, of the 86.72% of the CCR_G from the EPUNN model, a great improvement on the percentage obtained using the SLogistic (76.10%) or MLogistic (74.40%) models. These results are in line with those obtained by Hajmeer and Basheer [29].

7. Conclusions

We propose a classification method that combines a nonlinear model based on a special class of feed-forward neural network, namely PUNNs, and a learning EA that finds the optimal structure of the model and estimates the corresponding coefficients. To the best of our knowledge, this is the first study that applies evolutionary PUNNs to solve a wide range of multiclassification problems evolving both structure and weights. A review of the related literature shows that, up to now, research on product-units has mainly been applied to solve regression problems.

Our method uses softmax transformation and the cross-entropy error function. From a statistical point of view, the approach can be seen as nonlinear multinomial logistic regression where we use evolutionary computation to optimize log-likelihood. In fact, we attempt to estimate conditional class probabilities using a multilogistic model with nonlinear models given by product-units. The coefficients that minimize the cross-entropy error function are estimated by means of an EA. The algorithm proposed evolves both the weights and the structure of the network using evolutionary programming. The difficulty entailed in ascertaining the most suitable network structure at the outset of a given problem is well known; the evolution of the structure, however, partially alleviates this problem. The performance of our approach is estimated by means of comparison with other methodologies from different points of view. First, the results obtained in the comparison of EPUNN with the evolution of traditional feed-forward neural networks with sigmoidal additive units shows the capabilities of product-unit models versus the standard neural network model. Next, we compare our model EPUNN to a new cooperative co-evolutionary method (COVNET), and with respect to two recent approaches based on multiobjective evolutionary neural networks (MPANN and SPANN). In both cases, the empirical and statistical results show that EPUNN model performs well compared to other evolutionary neural network techniques for classification. Promising results are obtained in terms of classification accuracy, number of connections and number of nodes of the classifier. Moreover, we show the best model for each problem and graphically use two examples to illustrate the classification task carried out by the PUNN model as well as the capability of the product-unit to both capture the interactions between the variables and reduce the dimensionality of the problem.

Finally, for a nontrivial real problem of predictive microbiology, the result of the accuracy obtained using the EPUNN model outperforms the results obtained by consolidated statistical techniques for classification.

As future work, it would be of interest to state the relationship among the level of interaction of the data, the complexity level of each data set and the level of performance obtained in the product-unit models for the a corresponding problem.

Acknowledgments

This work has been financed in part by TIN 2005-08386-C05-02 projects of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT) and FEDER funds. The authors would like to thank the HIBRO Research Group of the University of Córdoba, Spain, which obtained the experimental data set used in Section 6.2 of this paper. The research of P.A. Gutiérrez has been subsidized by the FPU Predoctoral Program (Spanish Ministry of Education and Science), Grant reference AP2006-01746.

References

[1] H.A. Abbass, Speeding up backpropagation using multiobjective evolutionary algorithms, Neural Comput. 15 (11) (2003) 2705–2726.
[2] N. Ahamad, E.H. Marth, Behaviour of *Listeria monocytogenes* at 7, 13, 21 and 35 °C in tryptose broth acidified with acetic citric or lactic acid, J. Food Prot. 52 (1989) 688–695.
[3] J. Alcala-Fdez, L. Sánchez, S. García, M.J.D. Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput. 2007, accepted for publication.
[4] 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.
[5] M. Azevedo, A.D. Padua, B. Rodrigues, Improving generalization of MLPs with sliding mode control and the Levenberg–Marquardt algorithm, Neurocomputing 70 (7–9) (2007) 1342–1347.
[6] M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, 1995.
[7] C. Blake, C.J. Merz, UCI repository of machine learning data bases, 1998 ⟨http://www.ics.uci.edu/mlearn/MLRepository.thmlwww.ics.uci.edu/mlearn/MLRepository.thml⟩.

[8] S. Bose, Classification using splines, Comput. Stat. Data Anal. 22 (1996) 505–525.

[9] S. Bose, Multilayer statistical classifiers, Comput. Stat. Data Anal. 42 (2003) 685–701.

[10] W.J. Conover, Practical Nonparametric Statistics, Wiley, New York, 1971.

[11] A. Chandra, X. Yao. DIVACE: diverse and accurate ensemble learning algorithm, in: Proceedings of the Fifth International Conference on Intelligent Data Engineering and Automated Learning, Lecture Notes in Computer Science, vol. 3177, Springer, Berlin, 2004.

[12] A. Chandra, X. Yao, Evolutionary framework for the construction of diverse hybrid ensembles, in: Proceedings of the 13th European Symposium on Artificial Neural Networks, d-side, Brugge, Belgium, 2005.

[13] A. Chandra, X. Yao, Ensemble learning using multi-objective evolutionary algorithms, J. Math. Modelling Algorithms 5 (4) (2006) 417–445.

[14] A. Chandra, X. Yao, Evolving hybrid ensembles of learning machines for better generalization, Neurocomputing 69 (7–9) (2006) 686–700.

[15] R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu, K. Guppy, S. Lee, V. Froelicher, International application of a new probability algorithm for the diagnosis of coronary artery disease, Am. J. Cardiol. 64 (1989) 304–310.

[16] R. Durbin, D. Rumelhart, Products units: a computationally powerful and biologically plausible extension to backpropagation networks, Neural Comput. 1 (1989) 133–142.

[17] A.P. Engelbrecht, A. Ismail, Training product unit neural networks, Stability Control: Theory Appl. 2 (1–2) (1999) 59–74.

[18] European Commission, Opinion of the scientific committee on veterinary measures relating to public health on *Listeria monocytogenes*, 1999 ⟨http://www.europa.eu.int/comm/food/fs/sc/scv/out25⟩.

[19] D.B. Fogel, Using evolutionary programming to greater neural networks that are capable of playing Tic-Tac-Toe, in: International Conference on Neural Networks, IEEE Press, San Francisco, CA, 1993.

[20] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, IEEE Press, New York, 1995.

[21] D.B. Fogel, A.J. Owens, M.J. Wals, Artificial Intelligence Through Simulated Evolution, Wiley, New York, 1966.

[22] J. Friedman, Multivariate adaptive regression splines (with discussion), Ann. Stat. 19 (1991) 1–141.

[23] J. Friedman, W. Stuetzle, Proyection pursuit regression, J. Am. Stat. Assoc. 76 (376) (1981) 817–823.

[24] N. García-Pedrajas, C. Hervás-Martínez, J. Muñoz-Pérez, Multi-objective cooperative coevolution of artificial neural networks, Neural Networks 15 (10) (2002) 1255–1274.

[25] N. García-Pedrajas, C. Hervás-Martínez, J. Muñoz-Pérez, COVNET: a cooperative coevolutionary model for evolving artificial neural networks, IEEE Trans. Neural Networks 14 (3) (2003) 575–596.

[26] S.M. George, B.M. Lund, T.F. Brocklehurst, The effect of pH and temperature on the initiation of growth of *Listeria monocytogenes*, Lett. Appl. Microbiol. 6 (1988) 153–156.

[27] A. Gepperth, S. Roth, Applications of multi-objective structure optimization, Neurocomputing 69 (7–9) (2006) 701–713.

[28] K.P.M. Gysemans, K. Bernaerts, A. Vermeulen, A.H. Geeraerd, J. Debevere, F. Devlieghere, J.F.V. Impe, Exploring the performance of logistic regression model types on growth/no growth data of *Listeria monocytogenes*, Int. J. Food Microbiol. 114 (3) (2007) 316–331.

[29] M.N. Hajmeer, I.A. Basheer, Comparison of logistic regression and neural network-based classifier for bacterial growth, Food Microbiol. 20 (2003) 43–55.

[30] T. Hastie, R.J. Tibshirani, J. Friedman, The elements of statistical learning, in: Data Mining, Inference and Prediction, Springer, Berlin, 2001.

[31] T.J. Hastie, R.J. Tibshirani, Generalized Additive Models, Chapman & Hall, London, 1990.

[32] C. Hervás, F.J. Martínez-Estudillo, Logistic regression using covariates obtained by product-unit neural network models, Pattern Recognition 40 (2007) 52–64.

[33] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (16–18) (2007) 3056–3062.

[34] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Networks 17 (4) (2006) 879–892.

[35] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.

[36] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN2004), Budapest, Hungary, 2004.

[37] C.-A. Hwang, M.L. Tamplin, The influence of mayonnaise pH and storage temperature on the growth of *Listeria monocytogenes* in seafood salad, Int. J. Food Microbiol. 102 (2005) 277–285.

[38] A. Ismail, A.P. Engelbrecht, Training products units in feedforward neural networks using particle swarm optimization, in: V.B. Bajic, D. Sha (Eds.), Development and Practice of Artificial Intelligence Techniques, Proceeding of the International Conference on Artificial Intelligence, Durban, South Africa, 1999.

[39] A. Ismail, A.P. Engelbrecht, Global optimization algorithms for training product units neural networks, in: International Joint Conference on Neural Networks IJCNN'2000, Como, Italy, 2000.

[40] A. Ismail, A.P. Engelbrecht, Pruning product unit neural networks, in: Proceedings of the International Conference on Neural Networks, Honolulu, Hawaii, 2002.

[41] D.J. Janson, J.F. Frenzel, Training product unit neural networks with genetic algorithms, IEEE Expert 8 (5) (1993) 26–33.

[42] S. Kirkpatric, C.D.J. Gellat, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.

[43] C. Kooperberg, S. Bose, C.J. Stone, Polychotomous regression, J. Am. Stat. Assoc. 92 (1997) 117–127.

[44] L.R. Leerink, C.L. Giles, B.G. Horne, M.A. Jabri, Learning with products units, Adv. Neural Networks Process. Syst. 7 (1995) 537–544.

[45] H. Levene, Essays in honor of Harold Hotelling, in: Contributions to Probability and Statistics, 1960, pp. 278–292.

[46] A.C. Martinez-Estudillo, F.J. Martinez-Estudillo, C. Hervas-Martinez, N. Garcia-Pedrajas, Evolutionary product unit based neural networks for regression, Neural Networks 19 (4) (2006) 477–486.

[47] A.C. Martinez-Estudillo, C. Hervas-Martinez, F.J. Martinez-Estudillo, N. Garcia-Pedrajas, Hybridization of evolutionary algorithms and local search by means of a clustering method, IEEE Trans. Syst. Man Cybern. 36 (3) (2006) 534–545.

[48] G.F. Miller, P.M. Todd, S.U. Hedge, Designing neural networks using genetic algorithms, in: Proceedings of the Third International Conference Genetic Algorithms and their Applications, Morgan Kaufmann, San Mateo, CA, 1989.

[49] R.H.J.M. Otten, L.P.P.P. van Ginneken, The Annealing Algorithm, Kluwer, Boston, MA, 1989.

[50] M.A. Potter, K.A. de Jong, Cooperative coevolution: an architecture for evolving coadapted subcomponents, Evol. Comput. 8 (1) (2000) 1–29.

[51] R. Reed, Pruning algorithms—a survey, IEEE Trans. Neural Networks 4 (1993) 740–747.

[52] M. Rocha, P. Cortez, J. Neves, Evolution of neural networks for classification and regression, Neurocomputing 70 (2007) 2809–2816.

[53] K. Saito, R. Nakano, Numeric law discovery using neural networks, in: Proceedings of the Fourth International Conference on Neural Information Processing (ICONIP97), 1997.

[54] K. Saito, R. Nakano, Extracting regression rules from neural networks, Neural Networks 15 (2002) 1279–1288.

[55] M. Schmitt, On the complexity of computing and learning with multiplicative neural networks, Neural Comput. 14 (2001) 241–301.

[56] R. Setiono, L.C.K. Hui, Use of quasi-Newton method in a feedforward neural-network construction algorithm, IEEE Trans. Neural Networks 6 (1995) 273–277.

[57] I. SPSS, SPSS© para Windows 11.0.1, SPSS Inc., Chicago, IL, 1989–2001.

[58] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, Berlin, 1999.

[59] S. Ventura, C. Romero, A. Zafra, J.A. Delgado, C. Hervás-Martínez, JCLEC: a JAVA framework for evolutionary computation, Soft Computing—A Fusion of Foundations, Methodologies and Applications 12 (4) (2007) 381–392.

[60] W. Yan, Z. Zhu, R. Hu, Hybrid genetic /BP algorithm and its application for radar target classification, in: Proceedings of the IEEE National Aerospace Electronics Conference, IEEE Press, Piscataway, NJ, 1997.

[61] X. Yao, Evolving artificial neural network, Proc. IEEE 9 (87) (1999) 1423–1447.

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

[63] X. Yao, Y. Liu, Making use of population information in evolutionary artificial neural networks, IEEE Trans. Syst. Man Cybern. 28 (3) (1998) 417–425.

[64] Q.-Y. Zhu, A.K. Qin, P.N. Suganthan, G.-B. Huang, Evolutionary extreme learning machine, Pattern Recognition 38 (2005) 1759–1763.

Francisco J. Martínez-Estudillo was born in Villacarrillo, Jaén. He received the B.S. degree in mathematics in 1987 and the 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 research interests include structure optimization of neural networks, evolutionary algorithms and multiobjective optimization.

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.

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

Alfonso C. Martínez-Estudillo was born in Villacarrillo, Jaén. He received the B.S. degree in Computer Science in 1995 and the Ph. D. degree in 2005, both from the University of Granada, Granada, Spain. He is currently a lecturer in the Department of Management and Quantitative Methods in ETEA, University of Córdoba, Spain. His current research interests include neural networks, evolutionary algorithms and multiobjective optimization.