Applied Soft Computing 11 (2011) 534–550

Contents lists available at ScienceDirect

Applied Soft Computing

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

Memetic Pareto Evolutionary Artificial Neural Networks to determine growth/no-growth in predictive microbiology

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

[a] Department of Computer Science and Numerical Analysis, University of Cordoba, Rabanales Campus, Albert Einstein Building, 3° Floor, 14071 Córdoba, Spain
[b] Department of Management and Quantitative Methods, ETEA, Escritor Castilla Aguayo 4, 14005 Córdoba, Spain

ARTICLE INFO

Article history:
Received 18 November 2008
Received in revised form
27 November 2009
Accepted 6 December 2009
Available online 16 December 2009

Keywords:
Accuracy
Classification
Memetic algorithms
Predictive microbiology
Multiobjective
Neural networks
Sensitivity

ABSTRACT

The main objective of this work is to automatically design neural network models with sigmoid basis units for binary classification tasks. The classifiers that are obtained achieve a double objective: a high classification level in the dataset and a high classification level for each class. We present MPENSGA2, a Memetic Pareto Evolutionary approach based on the NSGA2 multiobjective evolutionary algorithm which has been adapted to design Artificial Neural Network models, where the NSGA2 algorithm is augmented with a local search that uses the improved Resilient Backpropagation with backtracking—*IRprop+* algorithm. To analyze the robustness of this methodology, it was applied to four complex classification problems in predictive microbiology to describe the growth/no-growth interface of food-borne microorganisms such as *Listeria monocytogenes*, *Escherichia coli R31*, *Staphylococcus aureus* and *Shigella flexneri*. The results obtained in Correct Classification Rate (*CCR*), Sensitivity (*S*) as the minimum of sensitivities for each class, Area Under the receiver operating characteristic Curve (*AUC*), and Root Mean Squared Error (*RMSE*), show that the generalization ability and the classification rate in each class can be more efficiently improved within a multiobjective framework than within a single-objective framework.

1. Introduction

There are many fields of study, such as medicine and predictive microbiology, where it is very important to predict a binary response variable or, equivalently, the probability of occurrence of an event, in terms of the values of a set of explicative variables related to it [1,2].

A classification problem occurs when an object needs to be assigned to a predefined group or class based on a number of observed attributes related to that object. Many techniques have been proposed to improve the overall generalization capability for the classifier design [3], but very few maintain their sensitivity capacity in all classes (which is studied here), an objective that is essential in some datasets (such as predictive microbiology, medicine, remote sensing, economy, etc.) to ensure the benefits of one classifier with respect to another.

Artificial Neural Networks (ANNs) [4,5] have become 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 and pattern recognition problems [6,7]. In this work, we discuss learning and the generalization improvement of classifiers designed using a Multiobjective Evolutionary learning Algorithm (MOEA) [8] for the determination of growth limits in predictive microbiology. Specifically, the study involves the generation of neural network classifiers that achieve a high classification level for each class. The methodology is based on two measures: the Correct Classification Rate, *CCR*, and Sensitivity, *S*, as the minimum of the sensitivities of all classes.

The basic structure of our MOEA has been modified by introducing an additional step, where some individuals in the population have been enhanced by a local search method. For this purpose, a Memetic Pareto Evolutionary NSGA2 (MPENSGA2) algorithm has been developed.

Recently, several more flexible classification models have been developed in the field of predictive microbiology to evaluate the behaviour of microorganisms under a given set of conditions [9], due to the demand for healthier and more suitable food products, because scientists recognize that there is an increasing need to model microbial growth limits and microbial growth as an alternative to the time-consuming traditional microbiological enumeration technique [10,11]. Growth/no-growth models or boundary models quantify the probability of microbial growth and define combinations of factors that prevent growth. This is because microbial growth is confined to a limited range of factors, and it sometimes even drops sharply when the level of each factor is increased. Growth predictive models have been widely accepted as informative tools that provide quick and cost-effective assessments of microbial growth for product development, risk assessment, and

* Corresponding author. Tel.: +34 696550558.
E-mail address: jcfernandez@uco.es (J.C. Fernández).

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.12.013

educational purposes. The importance of growth boundary models for empowering the hurdle concept has been discussed by various authors [12,13].

Given an adequate database, the response of many microbes in food could be predicted with sufficient knowledge about the formulation of the food, its processing and storage conditions applied in later food product development and food-safety risk assessment.

Models of the probability of pathogen outgrowth in foods appeared in the early 1970s literature when Genigeorgis et al. [14], motivated by a need to predict combinations of conditions required to prevent pathogen growth and toxin formation, modelled the decimal reduction of *Staphylococcus aureus*. A further development was the ability to predict whether an organism would grow under a given set of environmental conditions or not. Ratkowsky and Ross [15] modified a model for bacterial growth rate, incorporating the effects of temperature, pH, water activity and nitrite concentration to such an extent that it could predict the probability of growth versus the probability of no-growth.

Our application lies in the demand for healthier and more suitable food products, as scientists recognize that there is an increasing need to model microbial growth limits. In order to compare the results of our algorithm with some baseline methods, we used four performance metrics for binary classification problems [16,17], Accuracy or Correct Classification Rate, *CCR*, Area Under the Curve of receiver operating characteristics, *AUC*, the Sensitivity, *S*, as the minimum of sensitivities of each class and the Root Mean Squared Error, *RMSE*. The *CCR*, the *AUC*, and the *RMSE* represent the three most often used metrics, which represent the threshold metric, the probability metric, and the rank metric, respectively. The experiments show that our methodology obtains the best results in almost all datasets with almost all metrics.

The rest of the paper is organized as follows. Section 2 covers background materials. Section 3 shows an explanation of Accuracy and Sensitivity. In Section 4, the base classifier framework and the fitness functions used in this work are explained. The MPENSGA2 algorithm is described in Section 5, followed by the experimental design in Section 6. Section 7 shows the results obtained and finally, the conclusions are drawn in Section 8.

2. Related works

2.1. Evolutionary Artificial Neural Networks

ANNs have been a key research area in computer science for the last two decades [5]. On the one hand, methods and techniques have been developed to find better approaches for evolving ANNs, and more specifically, multilayer feedforward ANNs. On the other hand, finding a good ANN architecture has also been a debatable issue in the field of Artificial Intelligence. Methods for network-growing denominated constructive algorithms [18,19], start with a small network (usually a single neuron). This network is trained until it is unable to continue learning. Then, new components are added to the network. This process is repeated until a satisfactory solution is found. Destructive methods, also known as pruning algorithms [20], start with a big network that is able to learn but usually ends in over-fitting, and then some processes are applied in order to remove the connections and nodes that are not useful. These methods are based on the classic Backpropagation algorithm, BP, and all these usually suffer from slow convergence and a long training time. In addition, they are gradient-based techniques and, therefore, can easily get stuck at a local minimum.

Evolutionary computation has been widely used in the last few years to evolve neural-network architectures and weights. This is known as Evolutionary Artificial Neural Networks (EANNs), and it has been used in many applications [21,22]. EANNs provide a more successful platform for optimizing network performance and architecture simultaneously. There have been many applications for parametric learning [23] and for both parametric and structural learning [24]. This may indicate that there is an extensive need for finding better ways to evolve ANNs. A major advantage of the evolutionary approach over traditional learning algorithms such as BP is the ability to escape a local optimum. More advantages include robustness and an ability to adapt to changing environments. In the literature, research into EANNs has usually taken one of three approaches: evolving the weights of the network, evolving the architecture, or evolving both simultaneously [25]. The major disadvantage of the EANN approach is that it is computationally expensive, as the evolutionary approach is usually slow. To overcome this slow convergence of the evolutionary approach, hybrid techniques were used to speed up convergence by augmenting evolutionary algorithms with a local search technique (i.e. memetic approach), such as BP [26].

2.2. Artificial Neural Networks in predictive microbiology

In predictive microbiology, some jobs with ANNs have been used for modelling complex time-dependent bacterial growth [27–29], or for predicting growth parameters such as lag time and exponential growth rate [30–33] as affected by extrinsic biochemical and environmental conditions. Basheer and Hajmeer [27] proposed feedforward neural networks based on Backpropagation minimization criterion applied to the area of predictive microbiology, along with applications for the estimation of bacterial growth parameters and growth curve modelling. They found that feedforward neural networks outperform the most traditional statistical classification approaches. In [28] ANNs are used as efficient approximators for highly dimensional complex functions because of their high non-linearity and tolerance to noise data, so that the models obtained were used to include the effect of time as well as a multitude of parameters pertaining to experimental conditions for pathogenic *Escherichia coli 0157:H7* and *Shigella flexneri*. In [29], simple neural networks were compared with statistical and approximate methods to find the best descriptive model for a set of 20 *Lactobacillus helveticus* growth curves, obtaining good results. In [30] General Regression Neural Networks, GRNN, are compared to other statistical models using six statistical indices, obtaining good performance results in unseen data for the growth curves for three pathogens. In [31] ANN models are used and compared to other methodologies to predict thermal inactivation for *E. coli* bacteria obtaining the best results in accuracy due to the ANNs ability to compute the combined effects of environmental factors. In [32,33] García-Gimeno et al. use ANNS models with sigmoidal units for the estimation of several kinetic parameters of *Leuconostoc mesenteroides* and *E. coli* under aerobic and anaerobic conditions, comparing the results with the Response Surface Model and obtaining single predictive models. In [34], Probabilistic Neural Networks, PNNs, are combined with Bayes theorem of conditional probability and Parzen's method for estimating the probability density functions of random variables in the classification of the growth/no-growth state of a pathogenic *E. coli R31* in response to temperature and water activity. In [35,36], a support vector machine classifier based on the Gaussian RBF Kernel and a neuro-fuzzy system classifier with Gaussian bell shaped membership functions and feedforward neural networks were used for the classification of the growth limits of *E. coli R31*. This same pathogen was studied in [37], where feedforward error backpropagation ANNs and PNN based classifiers were developed and compared with respect to their accuracy in the classification of bacterial growth/no-growth data from temperature and water activity values. Recently, a new approach has been proposed to determine the growth probability of *Listeria monocytogenes* applying logistic regression over a combination of linear functions and non-linear transformations of them, where the linear

functions are made up of the input variables, and the transformations are trained by a Evolutionary Product Unit Neural Network algorithm, EPUNN [38].

As can be seen, some work has been done with ANNs to solve classification tasks in predictive microbiology, but in no case is sensitivity used to improve a classifier in a multiobjective framework as second objective. This is because the usual objectives to be optimized through the use of ANNs in binary classification tasks are the maximization of accuracy and the minimization of network complexity. There are classifiers to optimize sensitivity and specificity in a multiobjective framework, but they can only be used for binary classification, while the definition of sensitivity that we propose in this work can be used for multiclass problems.

2.3. Multiobjective Evolutionary Algorithms

A general Multiobjective Optimization Problem (MOP) solution method ranges from linear objective function aggregation to Pareto-based techniques. Aggregation methods usually present disadvantages, for example they only generate one Pareto-solution at a time [39] and assume convexity of the Pareto-frontier. In an attempt to stochastically solve problems of this generic class in an acceptable timeframe, specific Multiobjective Evolutionary Algorithms (MOEAs) were initially developed in the mid-eighties for application to the MOP domain and were efficient in the evaluation of the Pareto-optimal set in difficult multiobjective optimization problems. Several MOEAs, capable of dealing with a population of points, have been suggested to define an approximation to the Pareto set with a single run. There are already a number of favourable reviews on MOEA methods [8].

The use of ANNs together with Evolutionary Pareto-based Algorithms [40] is known as Multiobjective Evolutionary Artificial Neural Networks (MOEANNs), and this technique is being used to solve classification tasks with several competitive objectives, and is able to find multiple solutions in a single execution [41–43].

During the last few years, new methods called Memetic Algorithms (MAs) have been developed in order to improve the EAs using local optimization algorithms [44]. Some of the most important works in the literature about MOEAs, local optimizers and ANNs used to speed up the convergence are [41,45–47].

3. Accuracy and Sensitivity

In this section we present two measures to evaluate a classifier: the Correct Classification Rate or Accuracy, *CCR*, and Sensitivity, *S*. We will show that these quantities in general do not cooperate on certain levels. This fact justifies the use of a MOEA.

To evaluate a classifier, the machine learning community has traditionally used *CCR* to measure its default performance. However, the pitfalls of using accuracy have been pointed out by several authors [48]. Actually, we simply have to realize that accuracy cannot capture all the different behavioural aspects found in two different classifiers. Even in the simplest case, where there are only two classes, accuracy states a one-dimensional ordering where two different types of errors are found. We consider traditionally used accuracy, *CCR*, and the minimum of the sensitivities of all classes, *S*, that is, the lowest percentage of examples correctly predicted as belonging to each class, S_j, with respect to the total number of examples in the corresponding class, $S = \min\{S_j\}$. The sensitivity versus accuracy pair (*S*,*CCR*) expresses two features associated with a classifier: global performance (*CCR*) and the rate of the worst classified class (*S*). The selection of *S* as a complementary measure of *CCR* can be justified upon considering that

$$CCR = \frac{f_1}{N} S_1 + \frac{f_2}{N} S_2$$

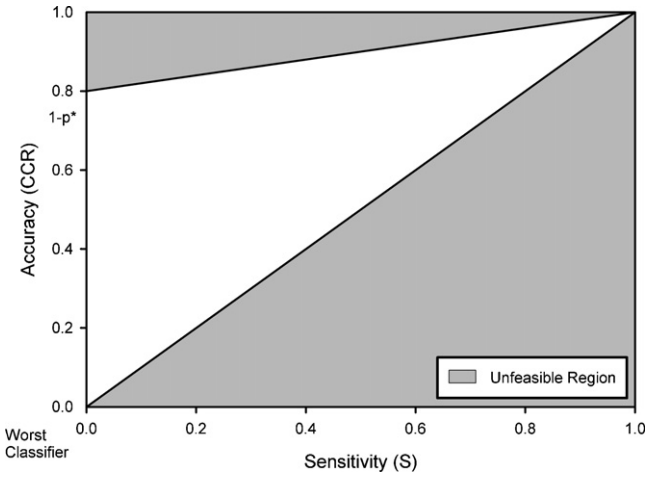

Fig. 1. Unfeasible region in the two-dimensional (*S*,*CCR*) space of a concrete classification problem.

is the weighted average of the sensitivities of each of the two classes, and f_i is the size of the C_i class. From a statistical point of view, since *CCR* is a weighted average, it will be a good and representative measurement of the set of sensitivities if they are homogeneous enough.

One point in (*S*,*CCR*) space dominates [8] another if it is above it and to the right, i.e. it has more accuracy and greater sensitivity. Let *CCR* and *S* be respectively the accuracy and the sensitivity associated with a classifier *g*, then $S \leq CCR \leq 1-(1-S)p^*$, where p^* is the minimum of the estimated prior probabilities. Therefore, each classifier will be represented as a point in the white region in Fig. 1, hence the area outside of the triangle is marked as unfeasible.

The area inside the triangle may be feasible (attainable), or may not be, depending upon the classifier and the difficulty of the problem. Observe that the optimum classifier is not feasible for all problems/classifiers, especially for problems with stochastic elements. For this reason it is better to say that a classifier cannot be located in the unfeasible region. Furthermore, the points on the vertical axis correspond to classifiers that are not able to correctly predict any pattern of a given class. Note that it is possible to find among them classifiers with a high level of accuracy, particularly in problems with low p^* (unbalanced problems).

A priori, we can think that *S* and *CCR* objectives can be positively correlated, but while this may be true for small values of *S* and *CCR*, it is not for values close to 1 on both *S* and *CCR*. In this way competitive objectives are at the top right corner of the white region.

4. MLP classifiers and fitness functions

4.1. Multilayer Perceptron

We consider standard feedforward Multilayer Perceptron (MLP) neural networks for binary growth/no-growth classification problems, with one input layer with *K* independent variables or features, one hidden layer with *M* sigmoidal hidden nodes and one output node. A scheme of the MLP models considered in this paper is given in Fig. 2.

Let us take a binary outcome variable *y* and a vector $\mathbf{x} = (1, x_1, x_2, \ldots, x_K)$ of input variables (we assume that the vector of inputs includes the constant term 1 to accommodate the intercept or bias). We coded the two class via a 1/0 response variable *y*, where $y = 1$ for the first class (growth) and $y = 0$ for the second class (no-growth); and then the output layer is interpreted from a probability point of view which considers the softmax activation function given by the

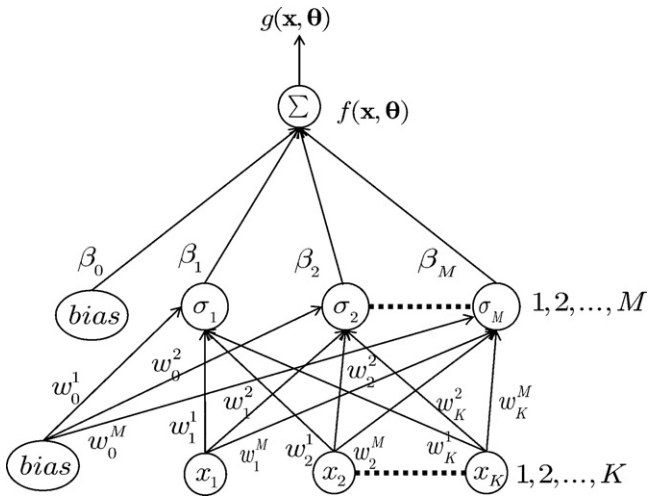

Fig. 2. Feedforward neural network with sigmoidal basis units for binary classification.

following expression:

$$g(\mathbf{x}, \boldsymbol{\theta}) = \frac{\exp f(\mathbf{x}, \boldsymbol{\theta})}{1 + \exp f(\mathbf{x}, \boldsymbol{\theta})}, \tag{1}$$

where $g(\mathbf{x},\boldsymbol{\theta})$ is the probability a pattern $\mathbf{x}$ has of belonging to the growth class, and $1 - g(\mathbf{x},\boldsymbol{\theta})$ is the probability a pattern $\mathbf{x}$ has of belonging to the no-growth class, $\boldsymbol{\theta} = (\beta_0, \ldots, \beta_M, \mathbf{w}_1, \ldots, \mathbf{w}_M)$ is the vector of weights of the output node, $\mathbf{w_j} = \{w_0^j, \ldots, w_K^j\}$ is the vector of inputs weights of the hidden node j, and $f(\mathbf{x},\boldsymbol{\theta})$ is the output of the output node for pattern $\mathbf{x}$ given by

$$f(\mathbf{x}, \boldsymbol{\theta}) = \beta_0 + \sum_{j=1}^{M} \beta_j \sigma \left(w_0^j + \sum_{i=1}^{K} w_i^j x_i \right)$$

where $\sigma(\cdot)$ is the sigmoidal activation function.

The classification rule $C(\mathbf{x})$ of the MLP model is $C(\mathbf{x}) = \arg\max\{g(\mathbf{x},\boldsymbol{\theta}), 1 - g(\mathbf{x},\boldsymbol{\theta})\}$, this classification rule coinciding with the optimal Bayes' rule.

4.2. Fitness functions

When there is an available training dataset $D = \{(\mathbf{x}_n, y_n); n = 1, 2, \ldots, N\}$, where $\mathbf{x}_n = (x_{1n}, \ldots, x_{Kn})$ is the random vector of measurements taking values in $\Omega \subset R^K$, and y_n is the class level of the nth individual, we define the Correctly Classified Rate (*CCR*) or accuracy by:

$$CCR = \left(\frac{1}{N}\right) \sum_{n=1}^{N} (I(C(\mathbf{x}_n) = y_n))$$

where $I(\cdot)$ is the zero-one loss function, y_n is the desired output for pattern n and N is the total number of patterns in the dataset. A good classifier tries to achieve the highest possible *CCR* in a given problem. However, the *CCR* measure is a discontinuous function, which makes convergence very difficult in neural network optimization.

Thus, instead of accuracy, we consider the continuous function given by cross-entropy, E:

$$E(g, \boldsymbol{\theta}) = -\frac{1}{N} \sum_{n=1}^{N} [y_n \log g(\mathbf{x}_n, \boldsymbol{\theta}) + (1 - y_n) \log(1 - g(\mathbf{x}_n, \boldsymbol{\theta}))] \tag{2}$$

Then, we propose a strictly decreasing transformation of the entropy error $E(g,\boldsymbol{\theta})$ as the fitness measure to maximize:

$$A(g) = \frac{1}{1 + E(g, \boldsymbol{\theta})}$$

The second objective to maximize is the sensitivity of the classifier, S, defined as the minimum value of the sensitivities for each class $S = \min\{S_i;\ i = 1, 2\}$. That is, maximizing the lowest percentage of examples correctly predicted as belonging to each class with respect to the total number of examples in the corresponding class.

5. Memetic Pareto Algorithm

This section introduces a MOEA with a local search algorithm, called MPENSGA2 (Memetic Pareto Evolutionary NSGA2), that tries to move the classifier population towards the optimum classifier located at the (1,1) point in the (S,C) space. The MOEA proposed is based on the NSGA2 algorithm [49] and the local search algorithm is the *Improved Resilient Backpropagation—IRprop+* [50], which will be discussed at the next subsection.

The Memetic Multiobjective Evolutionary Neural Network algorithm used in this work evolves architectures and connection weights simultaneously, each individual being a fully specified ANN. The ANNs are represented using an object-oriented approach and the algorithm deals directly with the ANN phenotype. Each connection is specified by a binary value, which indicates whether the connection exists and a real value representing its weight. The crossover operator is not considered due to its potential disadvantages in evolving ANNs [51]. This object-oriented representation does not assume a fixed order among the different hidden nodes. With these features, the algorithms fall into the class of evolutionary programming.

Mutators used in this work are divided into structural mutators (add/delete neurons, add/delete connections) and a parametric mutator, in this case a new parametric mutation that involves the alteration of all weights of the network by adding a Gaussian noise, where the variance of the Gauss distribution follows a geometric decline which is configurable (for specific structural mutation details see [52–54]).

Structural mutation introduces diversity in the population that leads to different locations in the search space. The number of neurons that can be added or deleted is configurable, and this value has been established at a minimum of one neuron and a maximum of two (random value every time a mutation is used). With regard to adding or deleting link mutations, the number of links to add or delete are calculated between the input layer and the hidden layer and between the hidden layer and output layer. Specifically, 30% of the total number of links in the hidden layer have been added or deleted and the 5% of the total in the output layer.

Parametric mutation is done on each weight $w \in \theta$ of the neural network with Gaussian noise $w(t+1) = w(t) + \xi(t)$, where $\xi(t) \in N(0,T(t))$, represents a one-dimensional normally distributed random variable with mean 0 and variance $T(t)$, and $T(t)$ represents a temperature function decreasing throughout evolution, making abrupt changes at the beginning (exploration) and soft changes at the end (exploitation), whose expression in the tth generation is:

$$T(t) = \begin{cases} \alpha T(t-1), & \text{if } t \text{ is multiple of } G \\ T(t), & \text{in any other case} \end{cases} \tag{3}$$

According to this rule, the new temperature is equal to the current temperature multiplied by a temperature factor α. The initial temperature $T(0)$ and α must be defined in the algorithm, as well as the number of generations, G, that pass between two consecutive temperature updates.

1. t=0 and generate a random population P(t) of size N_p, where each individual presents a sigmoidal basis function structure and where "t" is the number of the actual generation.
2. Evaluate the individuals P(t) on basis of Entropy and Sensitivity.
3. Use the fast-non-dominated-sort for obtaining a list F with the fronts of the population P(t).
4. Assign to each individual a rank value equal to his dominance level and a crowding distance value for the case of tie in the following selection process.
5. Use binary tournament for selecting N_p individuals of P, according to their rank and crowding distance value
6. Do mutation (one of the five mutations randomly) on each of the individuals selected to generate a new offspring population Q(t) of size N_p.
7. Evaluate the individuals Q(t) on basis of Entropy and Sensitivity.
8. While stopping criterion is not met do
   a) f=1
   b) R(t) = P(t) U Q(t).
   c) F= fast-non-dominated-sort(R(t)).
   d) If (number of generation t is equal to 2/7 or 4/7 or 6/7 of the total number of generations)
      e) Apply iRprop+ to the individuals of the first Pareto front F^1 of F and evaluate the individuals F^1 on basis of Entropy and Sensitivity, R'(t) being the R(t) population with the first front modified.
      f) F= fast-non-dominated-sort(R'(t))
   g) end if
   h) While size of population P(t+1) is < N_p to do
      I. Calculates crowding-distance for front F^f
      II. P(t+1) = P(t+1) U F^f
      III. f=f+1 //The number (rank from the fast-non-dominated-sort) of Pareto front is incremented
   i) end while
   j) Sort population P(t+1) according to their rank and crowding value and select the fisrt N_p individuals. The new population P(t+1) of size N_p is now completed.
   k) Use binary tournament and according to their rank and crowding value obtain N_p individuals from P(t+1).
   l) Do mutation (one of the five mutations randomly) on each of the individuals selected by binary tournament to generate a new population Q(t+1) of size N_p.
   m) Evaluate the individuals Q(t+1) on basis of Entropy and Sensitivity.
   n) t=t+1
9. end while

Fig. 3. MPENSGA2 algorithm pseudocode.

In Fig. 3 we show the pseudocode of the MPENSGA2 algorithm, where the local search steps are in italics.

The algorithm starts generating a random population $P(0)$ of size N. The population is sorted according to non-domination, assigning to each solution a rank equal to its non-domination level (1 is the best level, 2 is the next-best level, and so on). Then, the usual binary tournament selection and mutation operators are used to create an offspring population $Q(0)$ of size N. Since elitism is introduced by comparing current population with previously found best non-dominated solutions, the procedure is different after the initial generation. Next, we describe any generation of the proposed algorithm:

First, a combined population $R(t) = P(t) \cup Q(t)$ is formed, the size of $R(t)$ population being equal to $2N$. Second, the population $R(t)$ is sorted according to non-domination criteria. Third, the *IRprop+* local procedure is applied to the first Pareto front F^1 of the $R(t)$ population. Fourth, $R(t)$ is sorted again with the fast non-dominated sort procedure. Fifth, solutions belonging to the best non-dominated set F^1 are the best solutions in the population. If the size of F^1 is smaller than N then all members of the set F^1 are definitely chosen for the new population $P(t+1)$. The remaining members of the population $P(t+1)$ are chosen from subsequent non-dominated fronts in their order of ranking. Thus, solutions from the set F^2 are chosen next, followed by solutions from the set F^3, and so on. This procedure is continued until no more sets can be accommodated. Sixth, the new population $P(t+1)$ is sorted according to rank and crowding values and the first N individuals are selected. Seventh, we use binary tournament on $P(t+1)$ to obtain N individuals. Then these individuals are mutated using one of the five mutations selected randomly, and the new offspring population $Q(t+1)$ is generated. The reader can see [49] to compare the proposed algorithm to the original NSGA2 proposed by Deb et al.

5.1. Local search algorithm

An improvement in the EAs is the incorporation of local search procedures throughout evolution. Some studies carried out on the convergence process of a genetic algorithm in a concrete optimization problem show that, although the genetic algorithm quickly finds good solutions to the problem, it needs many generations to reach the optimum solution, and it poorly finds the best solution when it is in a region near a global optimum [55]. Thus is well-known that certain local procedures are able to find the local optimum when the search is carried out in a small region of the

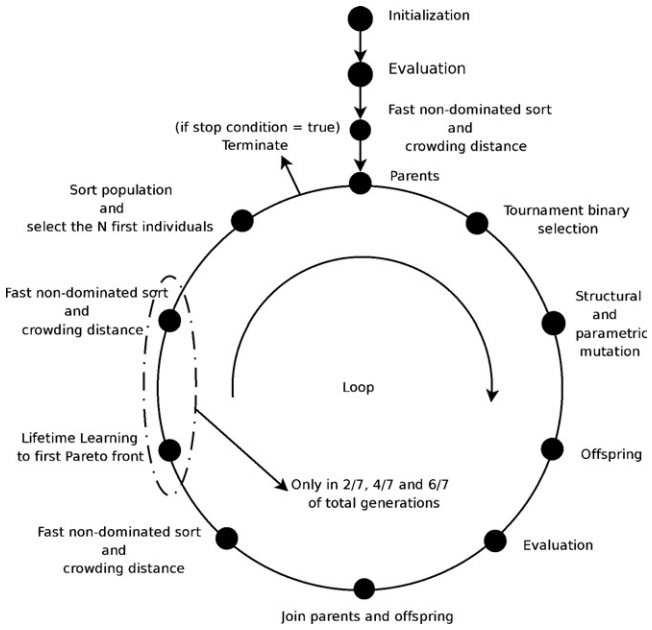

Fig. 4. Memetic framework proposed.

space. Therefore, in the combination of EA and local procedures, EAs would carry out a global search inside the space of solutions, locating ANNs near the global optimum, and the local procedure would quickly and efficiently find the best solution. This type of algorithm receives the name of Memetic or hybrid Algorithms (MAs) [44,56,57].

MAs can be considered a combination of population-based global search and heuristic-based local search. Gradient descent techniques are the most widely used class of algorithms for supervised learning in ANNs.

There are several studies [58–60] that make use of MOEAs along with local optimizers to fine-tune the weights. This is called *lifetime learning* and it consists of the updating of each individual regarding the approximation error. In addition, the weights modified during *lifetime learning* are encoded back to the chromosome, which is known as the Lamarckian type of inheritance. The main problem with this type of algorithms is the computational cost. All these authors use the local search on all individuals in the population size M after having made the crossover and mutation operations in each generation. This implies a high computational cost, something that we wanted to avoid. For these reasons we propose the following:

The local search algorithm is applied when combining parent and offspring population in NSGA2. Then, only the individuals of the first Pareto front of this combined population are optimized by the local gradient-based optimizer, reducing the computational cost considerably. In addition to that, *lifetime learning* occurs only three times in the evolutionary process, in generations 2/7, 4/7 and 6/7 of the total number of generations. In Fig. 4 the reader can see the framework proposed.

This local search will improve the Pareto front obtained in only one objective, specifically that which tries to minimize the classification error (Cross-Entropy).

We find that one of the best of these techniques in terms of convergence speed, accuracy and robustness with respect to its parameters is the *Rprop* (resilient Backpropagation) algorithm [61,62], although classic algorithms like Backpropagation are also frequently used. *Rprop* is a learning heuristic for supervised learning in artificial neural networks. Similarly to the Manhattan update rule, *Rprop* takes into account only the sign of the partial derivative throughout all patterns (not the magnitude), and acts independently on each weight. For each weight, if there was a sign of change in the partial derivative of the total error function compared to the previous iteration, the update value for that weight would be multiplied by a factor η^-. If the last iteration produced the same sign, the update value is multiplied by a factor η^+. The update values are calculated for each weight in the above manner, and finally each weight is changed by its own update value, in the opposite direction of that weight's partial derivative, so as to minimize the total error function.

A recent proposal has been the *improved Rprop—IRprop+* algorithm, which applies a backtracking strategy (i.e. it decides whether to take a step back in a weight direction or not, by means of a heuristic, and "+" is the incorporation of backtracking). The improvement is based on the consideration that a change of sign in the partial derivative implies that the algorithm has jumped over a local minimum, but does not indicate whether the weight update has caused an increase or a decrease. The idea of the modification of *Rprop+* is to make the step reversal dependent on the evolution of the error. These considerations lead to the rule that those weight updates that have caused changes to the signs of the corresponding partial derivatives are reverted, but only in case of an error increase. It has been shown on several benchmark problems [50] that the *improved Rprop* with backtracking exhibits consistently better performance than the original *Rprop* algorithm, and that is why we use it. We have carried out the adaptation of the *IRprop+* local optimizer to (1) the softmax activation function, and (2) the cross-entropy error function, modifying the gradient function for the weights in the hidden and output layers.

6. Experiments

To analyze the robustness of the proposed methodology in the experimental design we consider four complex problems in predictive microbiology to describe the behaviour of pathogen and spoilage microorganism under a given set of environmental conditions. The objective is to determine the conditions under which these microorganisms do or do not grow and to create a neural classifier for this purpose. Specifically, the problems considered have been the pathogen growth limits of *L. monocytogenes*, *E. coli R31*, *S. aureus* and *S. flexneri*.

In all experiments, the population size for MPENSGA2 is established to $N_p = 100$. The mutation probability for each operator is equal to 1/5. For *IRprop+*, the adopted parameters are $\eta^+ = 1.2$, $\eta^- = 0.5$, $\Delta_0 = 0.0125$ (the initial value of the Δ_{ij}), $\Delta_{\min} = 0$, $\Delta_{\max} = 50$ and $Epochs = 25$, see [61,62]. To start processing data, each of the input variables were scaled in the ranks $[-1.0, 1.0]$ to avoid the saturation of the signal. The α and $T(0)$ values in (3) were set to 0.95 and 1, respectively and the $\#G$ value, although dependent on the dataset, is usually assigned values of 50 or 100 generations.

In Table 1 we can see the features for each dataset. We show the total number of instances in each dataset, the number of instances in training and testing sets, the number of input variables, the total number of instances per class and the p^* value (the minimum of the estimated prior probabilities). For each database we had used the fractional factorial design present in different papers ([63] for *L. monocytogenes*, [64] for *E. coli R31*, [9] for *S. aureus* and [65] for *S. flexneri*) in order to find out the growth limits of each microorganism.

Once the Pareto front is built, two methodologies are considered in order to construct a neural network model with the information of the models on it. These are called MPENSGA2E and MPENSGA2S. These methodologies provide us with single models that can be compared with other classification methods existing in the literature. The process followed in these methodologies is the next one: once the first Pareto front is calculated using the

Table 1
Characteristics of *Listeria monocytogenes*, *Escherichia coli R31*, *Staphylococcus aureus* and *Shigella flexneri* pathogens. First class (growth) and second class (no-growth).

	Listeria monocytogenes	*Escherichia coli R31*	*Staphylococcus aureus*	*Shigella flexneri*
#Patterns	539	179	287	123
#Training Patterns	404	134	146	76
#Test Patterns	135	45	141	47
#Input Variables	4	2	3	4
#Patterns per class	299–240	99–80	162–125	79–44
p^*	0.44	0.44	0.43	0.32

patterns of the training set, the best individual belonging to the Pareto front on Entropy (*EI*) is chosen for MPENSGA2E, and the best individual in terms of sensitivity (*SI*) is selected for MPENSGA2S. Once this is done, the values of *CCR* and *S* are obtained by testing the *EI* and *SI* individuals. Therefore we will have an individual $EI_{testing} = (CCR_{testing}, S_{testing})$ and an individual $SI_{testing} = (CCR_{testing}, S_{testing})$. This is repeated 30 times and then the average and standard deviation obtained from the individuals is estimated, $\overline{EI}_{testing} = (\overline{CCR}_{testing}, \bar{S}_{testing})$, $\overline{SI}_{testing} = (\overline{CCR}_{testing}, \bar{S}_{testing})$. The first expression is the average obtained taking entropy into account as the primary objective, and the second taking sensitivity into account as the primary objective. So, the opposite extremes of the Pareto front are taken in each of the executions. Hence, the first procedure is called MPENSGA2E (Entropy) and the second MPENSGA2S (Sensitivity). In Fig. 5, the process is shown graphically.

Four metrics are used to test the performance of our methodology: *CCR*, *S*, *RMSE* and *AUC*. *CCR* and *S* represent threshold metrics, *AUC* is a probability metric, and *RMSE* a rank metric. *CCR* and *S* have been previously defined in Section 3.

RMSE or Root Mean Square Error [16] is a metric corresponding to the expected value of the squared error loss or quadratic loss. *RMSE* is a frequently used measurement of the differences between values predicted by a model or an estimator, and the values actually observed in what is being modelled or estimated.

A receiver operating characteristics (ROC) graph [17] is a technique for visualizing, organizing and selecting classifiers based on their performance. Recent years have seen an increase in the use of ROC graphs in the machine learning community, due in part to the realization that simple classification accuracy is often a poor way to measure performance [48]. A ROC curve is a two-dimensional depiction of classifier performance. To compare classifiers we may want to reduce ROC performance to a single scalar value representing the performance expected. A common method is to calculate the area under the ROC curve, abbreviated *AUC* [66]. Since the *AUC* is a portion of the area of the unit square, its value will always be between 0 and 1.0. The *AUC* has an important statistical property: the *AUC* of a classifier is equivalent to the probability that the classifier has of ranking a randomly chosen positive instance higher than a randomly chosen negative instance.

The metrics mentioned are used to compare the performance of MPENSGA2 along with 11 machine learning. The description of these algorithms can be found in [67] and they are available as part of the WEKA machine learning workbench [68] (http://www.cs.waikato.ac.nz/ml/weka/) and LibSVM is available as a continuously updated software library for Support Vector Machines [69] (http://www.csie.ntu.edu.tw/~cjlin/libsvm). The parameter values for each algorithm and for each dataset were chosen based on a battery of tests, selecting in each dataset the best results obtained for each algorithm. The test battery uses a grid of parameters taking into account extreme and intermediate values. The optimized values for these parameters can be found in Table 2. In continuation, a brief description of the parameters for each algorithm is shown. For a better comprehension of the values for each parameter see the description of the algorithm parameters in Weka workbench [68]:

- AdaBoost M1 (AB): Base classifier (BC), number of iterations to be performed (IT), use of resampling (RS) and weight threshold for weight pruning (WT).
- BayesNet (BN): Estimator algorithm (EA) and algorithm for searching network structures (SA).

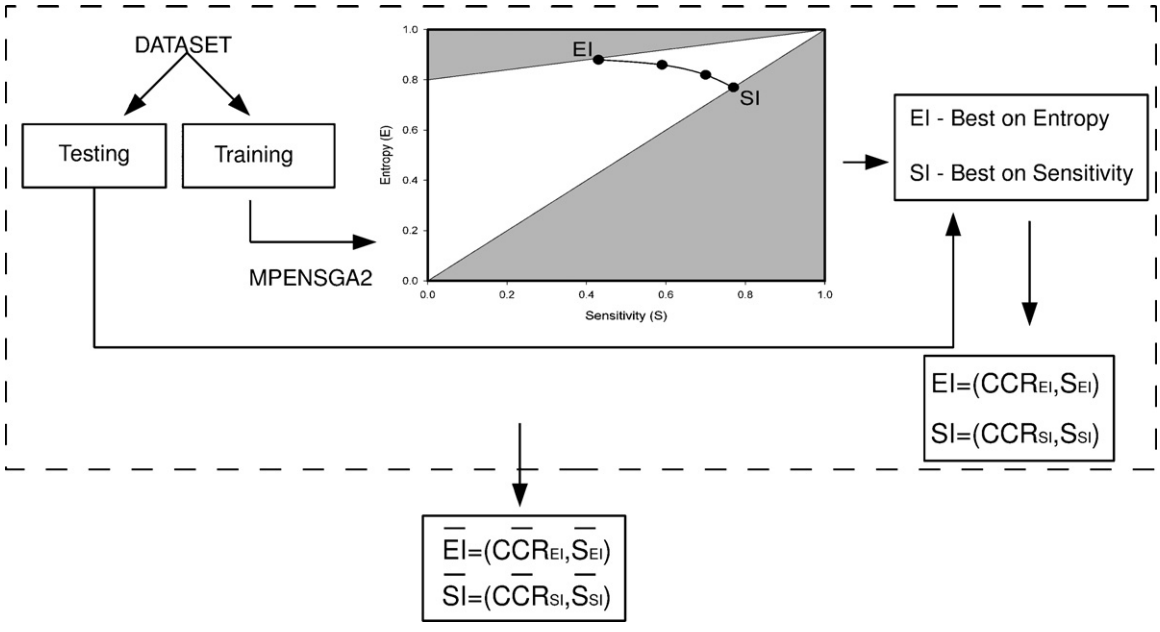

Fig. 5. Achieving statistical results from MPENSGA2.

Table 2
Optimized parameters for AdaBoost M1, BayesNet, C4.5 or J48, LMT, MultiLogistic and SimpleLogistic algorithms. See a complete description of each parameter for a better understanding of this values in Weka machine learning workbench [67].

AdaBoost M1

Dataset	BC	IT	RS	WT
Listeria	C4.5	100	False	100
E. coli	C4.5	100	False	150
S. aureus	C4.5	100	False	50
S. flexneri	C4.5	100	False	150

BayesNet

Dataset	EA	SA
Listeria	Simple Estimator	Simulated Annealing
E. coli	Simple Estimator	Simulated Annealing
S. aureus	BMA Estimator	Tabu Search
S. flexneri	Simple Estimator	Simulated Annealing

C4.5 or J48

Dataset	CF	NI	AD
Listeria	0.1	2	2
E. coli	0.5	5	5
S. aureus	0.25	3	3
S. flexneri	0.1	2	2

LMT

Dataset	MI	LB	BV
Listeria	15	10	0.1
E. coli	5	−1	0.0
S. aureus	15	10	0.1
S. flexneri	25	20	0.2

MultiLogistic

Dataset	IT	RL
Listeria	5	10^{-8}
E. coli	5	10^{-8}
S. aureus	5	10^{-8}
S. flexneri	5	10^{-8}

SimpleLogistic

Dataset	ER	HS	MI	NI	AIC	CV	BV
Listeria	False	100	900	10	False	True	0.0
E. coli	False	100	900	10	False	True	0.0
S. aureus	False	100	900	10	False	True	0.0
S. flexneri	False	−1	900	−10	False	True	0.0

- C4.5 or J48: Confidence factor for pruning (CF), minimum number of instances per leaf (NI) and amount of data for reduced-error pruning (AD).
- KStar (KS): Global blending (GB) is the only parameter for this algorithm in Weka. All experiments used the value 20, except *S. flexneri* which used the value 50.
- LibSVM (LSVM): This is a software package for the optimization of Support Vector Machines (SVM). This library contains a script for automatically adjusting the hyper-parameters associated to this kind of models, including the cost parameter and the width of the Gaussian kernels. The library searches the best hyper-parameter values using a grid search and choosing the best configuration by a 10-fold cross-validation process [69].
- Logistic Model Tree (LMT): Minimum number of instances at which a node is considered for splitting (MI), number of iterations for LogitBoost (LB) and ratio of weight trimming in LogitBoost (BV).
- MultiLogistic or Logistic (ML): Maximum number of iterations (IT) and ridge value in the log-likelihood (RL).
- NaivesBayesUpdateable (NB): Kernel estimator for numeric attributes rather than a normal distribution (KS). For this algorithm, the kernel estimator was selected in all the experiments.
- NBTree (NBT): Weka only provides a configuration for this algorithm. All experiments used that configuration.
- SimpleLogistic (SL): Error on the probabilities (ER), heuristic stop (HS), maximum number of iterations for LogitBoost (MI), number of iterations for LogitBoost (NI), AIC to determine when to stop LogitBoost iterations (AIC), cross-validation in LogitBoost (CV) and ratio of weight trimming in LogitBoost (BV).
- RandomForest (RF): Maximum depth of the trees, number of attributes to be used in random selection and number of trees to be generated. For this algorithm, all experiments used the values 0, 0 and 10, respectively (see algorithm's description in WEKA for understanding this values).

The next subsections deal with a description of the four problems selected for predictive microbiology.

6.1. *L. monocytogenes*

L. monocytogenes have been a serious problem concerning food industries due to their ubiquity in the natural environment [70] 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 has been the problem of listeriosis, and different strategies have been proposed to limit levels of contamination at the time of consumption to less than 100 CFU/g (European Commission [71]).

L. monocytogenes data were collected at CA and AA concentrations of 0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35 and 0.4% (w/v), at 4, 7, 10, 15 and 30 °C and pH levels of 4.5, 5, 5.5 and 6.5, as can be seen in [38]. Thus, 39 different conditions were tested with eight replicates per condition. This data set was divided so that 404 conditions were chosen for training, and 135 conditions were selected for testing the generalization capacity. Among the different conditions, there were 299 cases of growth and 240 cases with no-growth.

6.2. *E. coli*

This dataset is given by Salter et al. [64] and it pertaining to growth/no-growth of an *E. coli* strain *R31* affected by temperature and water activity.

The data consists of experiments performed with different combinations of temperature in the 7.7–37.0 range and water activity in the 0.943–0.987 range. All samples of *E. coli R31* were cultured in plates and L-tube observed daily. If growth in a sample occurred, it was scored positive. The growth in the sample was noticed by a visible increase in turbidity or deposit in the base of the tube. If after 50 days there was neither turbidity nor deposit, a loopful of culture was streaked onto plate count agar to determine if any growth was present. A total of 179 samples were observed using different values of temperature and water activity, with 99 as growth cases and 80 as cases without growth.

6.3. *S. aureus*

S. aureus has been recognized as an indicator of deficient hygiene of food and processing and a major cause of food gastroenteritis worldwide [72]. A fractional factorial design was followed in order to know the growth limits of *S. aureus* [9]. It was made by carefully choosing a subset (fraction) of the experimental runs of a full factorial design in order to reduce experimental time and resources. The selection was based on delimiting the levels of the environmental factors studied to the growth/no-growth domain of *S. aureus*. Since no growth was detected at 7.5 °C or below, data were collected at 8, 10, 13, 16 and 19 °C, at pH levels from 4.5 to 7.5 (0.5 intervals)

Table 3
Comparative performance in the testing set of the different methods applied to the predictive microbiology datasets: Mean and Standard Deviation (SD) of the accuracy (*CCR*(%)), sensitivity (*S*(%)), error (*RMSE*) and area under the ROC curve (*AUC*) results, mean values of these measures throughout all the datasets ($\overline{CCR}$(%), $\bar{S}$(%), $\overline{RMSE}$ and $\overline{AUC}$) and mean ranking of their performance ($\bar{R}$). The best results are in bold face and the second best results in italics, reading by rows. At the end of the table we can see the best model for each pathogen in the MPENSGA2 methodologies in testing.

Dataset	AB Mean	BN Mean	C4.5 Mean	KS Mean	LSVM Mean	LMT Mean	ML Mean	NB Mean	NBT Mean	SL Mean	RF Mean ± SD	M-E Mean ± SD	M-S Mean ± SD
Method (*CCR*(%))													
Listeria	89.62	77.03	85.92	86.66	**91.85**	89.62	83.70	80.74	87.40	82.22	88.83 ± 1.31	*90.37 ± 1.30*	89.03 ± 2.23
E. coli R31	*93.33*	80.00	*93.33*	88.88	91.11	*93.33*	80.00	*93.33*	88.88	80.00	92.44 ± 1.08	**94.51 ± 2.90**	91.85 ± 3.05
S. aureus	92.30	63.82	86.52	90.07	92.19	90.07	86.52	79.43	80.85	85.10	90.54 ± 1.28	**92.52 ± 1.43**	*92.43 ± 1.43*
S. flexneri	**80.85**	68.08	*78.72*	75.90	65.95	74.46	72.34	74.46	**80.85**	76.00	*78.72 ± 3.47*	76.26 ± 2.66	73.54 ± 3.19
$\overline{CCR}$	**89.02**	72.23	86.12	85.38	85.27	86.87	80.64	81.99	84.49	80.83	87.63	*88.41*	86.71
$\bar{R}$	*2.88*	12.50	6.13	7.75	6.50	5.50	10.37	9.00	7.25	9.75	5.12	**2.25**	6.00
Method (*S*(%))													
Listeria	88.00	71.00	80.00	83.00	88.00	88.00	80.00	76.00	83.00	76.00	86.71 ± 2.07	**89.33 ± 1.16**	*88.26 ± 1.79*
E. coli R31	85.00	76.00	85.00	80.00	85.00	85.00	65.00	85.00	83.00	65.00	83.16 ± 2.40	**89.66 ± 4.72**	*85.66 ± 5.83*
S. aureus	**91.00**	16.00	83.00	85.00	90.00	90.00	86.00	70.00	69.00	78.00	88.35 ± 1.86	90.11 ± 1.51	*90.2 ± 1.43*
S. flexneri	*73.00*	0.00	*73.00*	38.00	40.00	*73.00*	66.00	33.00	**78.00**	43.00	69.25 ± 12.35	72.63 ± 3.15	70.54 ± 3.98
$\bar{S}$	*84.25*	40.75	80.25	71.5	75.75	84.00	74.25	66.00	76.25	65.50	81.87	**85.43**	83.66
$\bar{R}$	3.25	12.25	6.62	8.87	5.87	4.12	9.25	9.87	7.87	10.75	6.75	**2.50**	*3.00*
Method (*RMSE*)													
Listeria	0.30	0.39	0.34	0.29	**0.25**	0.29	0.35	0.38	0.33	0.36	0.29 ± 0.02	*0.26 ± 0.01*	*0.26 ± 0.02*
E. coli R31	0.25	0.38	0.26	0.27	0.23	0.27	0.42	0.35	0.32	0.42	0.27 ± 0.01	**0.20 ± 0.03**	*0.23 ± 0.04*
S. aureus	0.26	0.47	0.33	0.31	**0.24**	0.28	0.32	0.41	0.39	0.35	0.27 ± 0.01	**0.24 ± 0.01**	*0.24 ± 0.02*
S. flexneri	**0.38**	0.47	0.41	0.43	0.41	*0.40*	*0.40*	0.42	**0.38**	0.47	**0.38 ± 0.02**	*0.40 ± 0.01*	0.43 ± 0.02
$\overline{RMSE}$	0.30	0.43	0.33	0.32	*0.28*	0.31	0.37	0.39	0.35	0.40	0.30	**0.27**	0.29
$\bar{R}$	3.25	11.37	6.62	6.37	*2.25*	4.75	7.87	9.75	6.50	10.50	3.75	**1.62**	3.37
Method (*AUC*)													
Listeria	0.96	0.83	0.87	0.95	0.97	0.96	0.91	0.87	0.90	0.90	0.93 ± 0.01	**0.98 ± 0.01**	*0.97 ± 0.01*
E. coli R31	0.94	0.86	0.92	*0.98*	**0.99**	0.91	0.84	**0.99**	0.89	0.84	0.92 ± 0.01	**0.99 ± 0.01**	**0.99 ± 0.01**
S. aureus	*0.96*	0.58	0.91	*0.96*	**0.98**	0.94	0.92	0.84	0.81	0.92	0.95 ± 0.01	**0.98 ± 0.01**	**0.98 ± 0.01**
S. flexneri	*0.83*	0.50	0.77	*0.83*	0.80	0.82	0.81	0.77	*0.83*	0.68	**0.85 ± 0.03**	*0.83 ± 0.02*	0.82 ± 0.03
$\overline{AUC}$	0.92	0.69	0.87	*0.93*	*0.93*	0.91	0.87	0.87	0.86	0.83	0.91	**0.94**	**0.94**
$\bar{R}$	4.62	12.50	9.87	4.75	4.00	6.75	9.25	8.87	8.75	10.62	5.37	**2.25**	*3.37*

Best Classifiers (testing)

Dataset	M-E				M-S			
	CCR	*S*	*RMSE*	*AUC*	*CCR*	*S*	*RMSE*	*AUC*
Listeria	91.851	90.000	0.238	0.982	93.333	91.666	0.223	0.986
E. coli R31	97.777	95.000	0.159	0.998	97.777	95.000	0.169	0.998
S. aureus	95.035	91.803	0.215	0.984	95.035	91.803	0.215	0.984
S. flexneri	78.958	75.732	0.396	0.851	78.825	75.682	0.395	0.981

Table 4
Critical differences in values and differences in rankings in the Nemenyi and Bonferroni–Dunn tests when comparing *CCR*(%), using the two proposals of this paper as the control methods for the last test mentioned.

Method(i)	Nemenyi test												
	Method(j)												
	AB	BN	C4.5	KS	LSVM	LMT	ML	NB	NBT	SL	RF	M-E	M-S
AB	–	9.62●	3.25	4.87	3.62	2.62	7.50	6.12	4.37	6.87	2.25	0.625	3.12
BN	–	–	6.37	4.75	6.00	7.00	2.12	3.50	5.25	2.75	7.37	10.25+,●	6.50
C4.5	–	–	–	1.62	0.37	0.62	4.25	2.87	1.12	3.62	1.00	3.87	0.12
KS	–	–	–	–	1.25	2.25	2.62	1.25	0.50	2.00	2.62	5.50	1.75
LSVM	–	–	–	–	–	1.00	3.87	2.50	0.75	3.25	1.37	4.25	0.50
LMT	–	–	–	–	–	–	4.87	3.50	1.75	4.25	0.37	3.25	0.50
ML	–	–	–	–	–	–	–	1.37	3.12	0.62	5.25	8.12	4.37
NB	–	–	–	–	–	–	–	–	1.75	0.75	3.87	6.75	3.00
NBT	–	–	–	–	–	–	–	–	–	2.50	2.12	5.00	1.25
SL	–	–	–	–	–	–	–	–	–	–	4.62	7.50	3.75
RF	–	–	–	–	–	–	–	–	–	–	–	2.87	0.87
M-E	–	–	–	–	–	–	–	–	–	–	–	–	3.75
M-S	–	–	–	–	–	–	–	–	–	–	–	–	–

$CD_{(\alpha=0.05)} = 9.11$; $CD_{(\alpha=0.1)} = 8.47$

Control method	Bonferroni–Dunn test												
	Compared method												
	AB	BN	C4.5	KS	LSVM	LMT	ML	NB	NBT	SL	RF	M-E	M-S
M-E	0.62	10.25+,●	3.87	5.50	4.25	3.25	8.12+,●	6.75	5.00	7.50+,○	2.87	–	3.75
M-S	3.12	6.50	0.12	1.75	0.50	0.50	4.37	3.00	1.25	3.75	0.87	3.75	–

$CD_{(\alpha=0.05)} = 7.89$; $CD_{(\alpha=0.1)} = 7.26$

Statistically significant differences with $\alpha = 0.05$ (●) and $\alpha = 0.1$ (○). +, the difference is in favour of Method(j) (Nemenyi test) or control method (Bonferroni–Dunn test).

and at 19Aw levels (from 0.856 to 0.999 at regular intervals). The initial dataset (287 conditions) was divided in two parts: model data (training set, 146 conditions covering the extreme domain of the model) and validation data (generalization set, 141 conditions within the interpolation region of the model). Among the different conditions, there were 162 growth cases and 125 no-growth cases. The purpose of this selection was to define a dataset for model data focused on the extreme regions of the growth/no-growth domain that actually represent the boundary zones. In this study, the number of replicates per condition ($n = 30$) was increased

Table 5
Critical differences in values and differences in rankings between the Nemenyi and Bonferroni–Dunn tests when comparing *S*(%), using the two proposals of this paper as the control methods for the last test mentioned.

Method(i)	Nemenyi test												
	Method(j)												
	AB	BN	C4.5	KS	LSVM	LMT	ML	NB	NBT	SL	RF	M-E	M-S
AB	–	9.00○	3.37	5.62	2.62	0.87	6.00	6.62	4.62	7.50	3.50	0.75	0.25
BN	–	–	5.62	3.37	6.37	8.12	3.00	2.37	4.37	1.50	5.50	9.75+,●	9.25+,●
C4.5	–	–	–	2.25	0.75	2.50	2.62	3.25	1.25	4.12	0.12	4.12	3.62
KS	–	–	–	–	3.00	4.75	0.37	1.00	1.00	1.87	2.12	6.37	5.87
LSVM	–	–	–	–	–	1.75	3.37	4.00	2.00	4.87	0.87	3.37	2.87
LMT	–	–	–	–	–	–	5.12	5.75	3.75	6.62	2.62	1.62	1.12
ML	–	–	–	–	–	–	–	0.62	1.37	1.50	2.50	6.75	6.25
NB	–	–	–	–	–	–	–	–	2.00	0.87	3.12	7.37	6.87
NBT	–	–	–	–	–	–	–	–	–	2.87	1.12	5.37	4.87
SL	–	–	–	–	–	–	–	–	–	–	4.00	8.25	7.75
RF	–	–	–	–	–	–	–	–	–	–	–	4.25	3.75
M-E	–	–	–	–	–	–	–	–	–	–	–	–	0.50
M-S	–	–	–	–	–	–	–	–	–	–	–	–	–

$CD_{(\alpha=0.05)} = 9.11$; $CD_{(\alpha=0.1)} = 8.47$

Control method	Bonferroni–Dunn test												
	Compared method												
	AB	BN	C4.5	KS	LSVM	LMT	ML	NB	NBT	SL	RF	M-E	M-S
M-E	0.75	9.75+,●	4.12	6.37	3.37	1.62	6.75	7.37+,○	5.37	8.25+,○	4.25	–	0.50
M-S	0.25	9.25+,●	3.62	5.87	2.87	1.12	6.25	6.87	4.87	7.75+,○	3.75	0.50	–

$CD_{(\alpha=0.05)} = 8.47$; $CD_{(\alpha=0.1)} = 7.26$

Statistically significant differences with $\alpha = 0.05$ (●) and $\alpha = 0.1$ (○). +, the difference is in favour of Method(j) (Nemenyi test) or control method (Bonferroni–Dunn test).

Table 6
Critical differences in values and differences in rankings between the Nemenyi and Bonferroni–Dunn tests when comparing *RMSE*, using the two proposals of this paper as the control methods for the last test mentioned.

Method(i)	Nemenyi test												
	Method(j)												
	AB	BN	C4.5	KS	LSVM	LMT	ML	NB	NBT	SL	RF	M-E	M-S
AB	–	8.12○	3.37	3.12	1.00	1.50	4.62	6.50	3.25	7.25	0.50	1.62	0.12
BN	–	–	4.75	5.00	9.12+,●	6.62	3.50	1.62	4.87	0.87	7.62	9.75+,●	8.00
C4.5	–	–	–	0.25	4.37	1.87	1.25	3.12	0.12	3.87	2.87	5.00	3.25
KS	–	–	–	–	4.12	1.62	1.50	3.37	0.12	4.12	2.62	4.75	3.00
LSVM	–	–	–	–	–	2.50	5.62	7.50	4.25	8.25	1.50	0.62	1.12
LMT	–	–	–	–	–	–	3.12	5.00	1.75	5.75	1.00	3.12	1.37
ML	–	–	–	–	–	–	–	1.87	1.37	2.62	4.12	6.25	4.50
NB	–	–	–	–	–	–	–	–	3.25	0.75	6.00	8.12	6.37
NBT	–	–	–	–	–	–	–	–	–	4.00	2.75	4.87	3.12
SL	–	–	–	–	–	–	–	–	–	–	6.75	8.87+,○	7.12
RF	–	–	–	–	–	–	–	–	–	–	–	2.12	0.37
M-E	–	–	–	–	–	–	–	–	–	–	–	–	1.75
M-S	–	–	–	–	–	–	–	–	–	–	–	–	–

$CD_{(\alpha=0.05)} = 9.11$; $CD_{(\alpha=0.1)} = 8.47$

Control method	Bonferroni–Dunn test												
	Compared method												
	AB	BN	C4.5	KS	LSVM	LMT	ML	NB	NBT	SL	RF	M-E	M-S
M-E	1.62	9.75+,●	5.00	4.75	0.62	3.12	6.25	8.12+,●	4.87	8.87+,●	2.12	–	1.75
M-S	0.12	8.00+,●	3.25	3.00	1.12	1.37	4.50	6.37	3.12	7.12	0.37	1.75	–

$CD_{(\alpha=0.05)} = 7.89$; $CD_{(\alpha=0.1)} = 7.26$

Statistically significant differences with $\alpha = 0.05$ (●) and $\alpha = 0.1$ (○). +, the difference is in favour of Method(j) (Nemenyi test) or control method (Bonferroni–Dunn test).

in comparison to other studies to obtain the growth/no-growth transition.

6.4. *S. flexneri*

S. flexneri is an important causative agent of gastrointestinal illness [65]. An incomplete factorial design was used to assess the effects of temperature (12, 15, 19, 28, 37 °C), initial pH (5.5, 6.0, 6.5, 7.0, 7.5), sodium chloride (0.5, 2.5, 4.0%) and sodium nitrite (0, 50, 100, 200, 1000 ppm). Data is obtained from 375 cultures, representing 123 variable combinations. The number of replicate cultures tested for each variable combination is given in Table 2 in the referenced paper [65]. This data was used to derive the models to predict the anaerobic growth of *S. flexneri* as a function of temperature, sodium chloride and sodium nitrite concentrations and initial pH. The growth kinetics data for each variable combination

Table 7
Critical differences in values and differences in rankings between the Nemenyi and Bonferroni–Dunn tests when comparing *AUC*, using the two proposals of this paper as the control methods for the last test mentioned.

Method(i)	Nemenyi test												
	Method(j)												
	AB	BN	C4.5	KS	LSVM	LMT	ML	NB	NBT	SL	RF	M-E	M-S
AB	–	7.87	5.25	0.12	0.62	2.12	4.62	4.25	4.12	6.00	0.75	2.37	1.25
BN	–	–	2.62	7.75	8.50+,○	5.75	3.25	3.62	3.75	1.87	7.12	10.25+,●	9.12+,●
C4.5	–	–	–	5.12	5.87	3.12	0.62	1.00	1.12	0.75	4.50	7.62	6.50
KS	–	–	–	–	0.75	2.00	4.50	4.12	4.00	5.87	0.62	2.50	1.37
LSVM	–	–	–	–	–	2.75	5.25	4.87	4.75	6.62	1.37	1.75	0.62
LMT	–	–	–	–	–	–	2.50	2.12	2.00	3.87	1.37	4.50	3.37
ML	–	–	–	–	–	–	–	0.37	0.50	1.37	3.87	7.00	5.87
NB	–	–	–	–	–	–	–	–	0.12	1.75	3.50	6.62	5.50
NBT	–	–	–	–	–	–	–	–	–	1.87	3.37	6.50	5.37
SL	–	–	–	–	–	–	–	–	–	–	5.25	8.37	7.25
RF	–	–	–	–	–	–	–	–	–	–	–	3.12	2.00
M-E	–	–	–	–	–	–	–	–	–	–	–	–	1.12
M-S	–	–	–	–	–	–	–	–	–	–	–	–	–

$CD_{(\alpha=0.05)} = 9.11$; $CD_{(\alpha=0.1)} = 8.47$

Control method	Bonferroni–Dunn test												
	Compared method												
	AB	BN	C4.5	KS	LSVM	LMT	ML	NB	NBT	SL	RF	M-E	M-S
M-E	2.37	10.25+,●	7.62+,○	2.50	1.75	4.50	7.00	6.62	6.50	8.37+,●	3.12	–	1.12
M-S	1.25	9.12+,●	6.50	1.37	0.62	3.37	5.87	5.50	5.37	7.25	2.00	1.12	–

$CD_{(\alpha=0.05)} = 7.89$; $CD_{(\alpha=0.1)} = 7.26$

Statistically significant differences with $\alpha = 0.05$ (●) and $\alpha = 0.1$ (○). +, the difference is in favour of Method(j) (Nemenyi test) or control method (Bonferroni–Dunn test).

are summarized in Table 2 referenced. Growth of *S. flexneri* was not observed under the conditions corresponding to 40 of the variable combinations studied. Additional 15 variable combinations resulted in environments under which some of the replicate cultures grew, while others did not; these are listed in Table 3 in the cited paper.

7. Results

In Table 3 we present the values of the average and the standard deviation for *CCR*, *S*, *RMSE* and *AUC* in 30 runs of all the experiments performed, where MPENSGA2E and MPENSGA2S methodologies are denoted by M-E and M-S, respectively. It can be seen that the

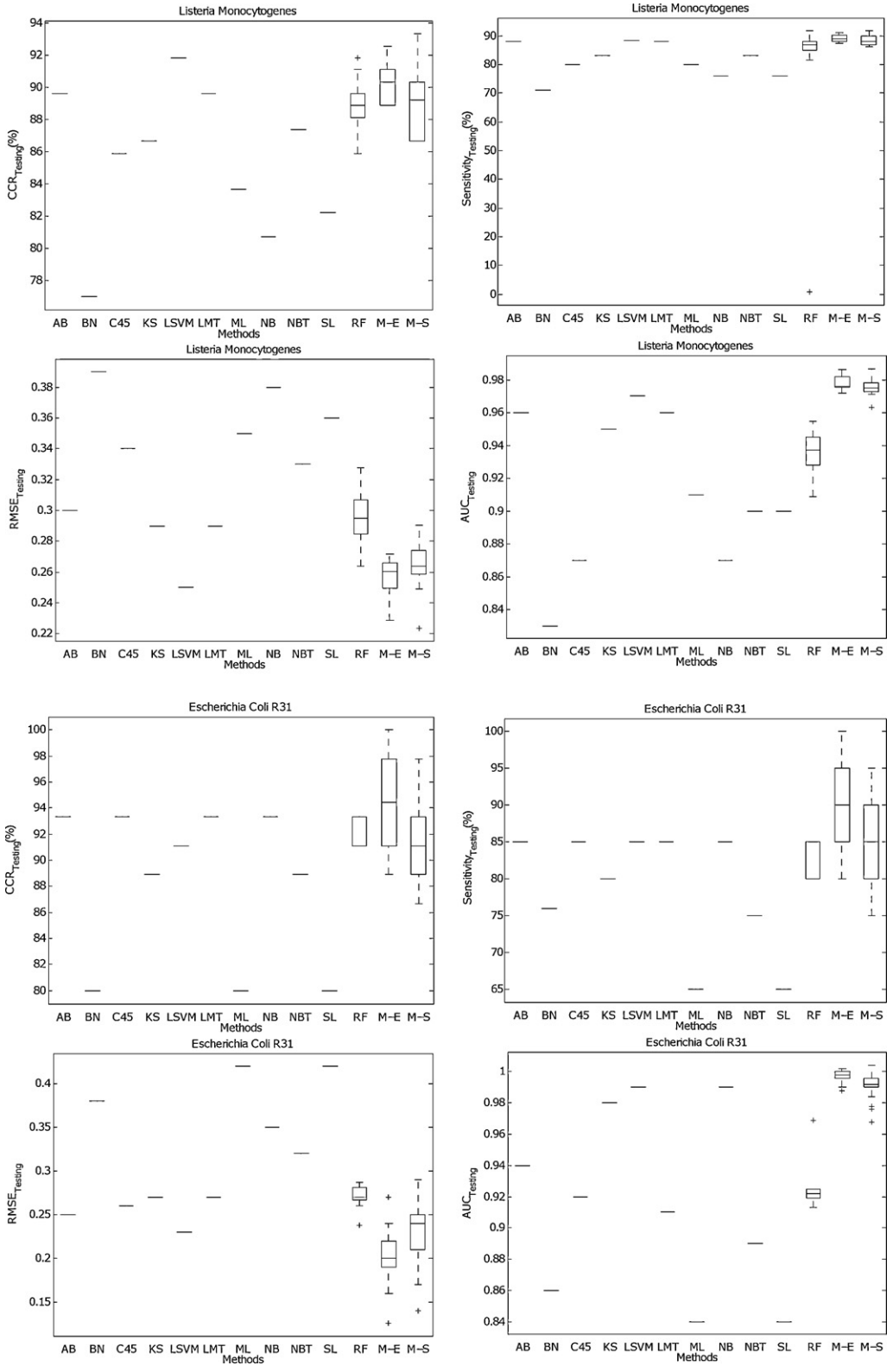

Fig. 6. *CCR*, *S*, *RMSE* and *AUC* measurements on Box Plots for *Listeria monocytogenes* and *Escherichia coli R31*.

M-E methodology produces good results with respect to *CCR*, *S*, *RMSE* and *AUC*. In fact, from a purely descriptive point of view, M-E methodology obtains the best result in *CCR* in two out of the four datasets analyzed and the second best result in another. Also it obtains the best result in *S* in two datasets. The best result in *RMSE* in two datasets, and the second best result in the other two remaining. And finally, it obtains the best *AUC* values in three out of the four datasets and the second best results in the remaining dataset.

On the other hand, the mean results obtained throughout the datasets show that the M-E methodology is the best one for *S*, *RMSE* and *AUC* measures, and the second best one for *CCR*. The M-S methodology is the best one for *AUC*.

Table 3 also include the mean ranking, ($\bar{R}$), of each method in each dataset and for each methodology ($R = 1$ for the best perform-

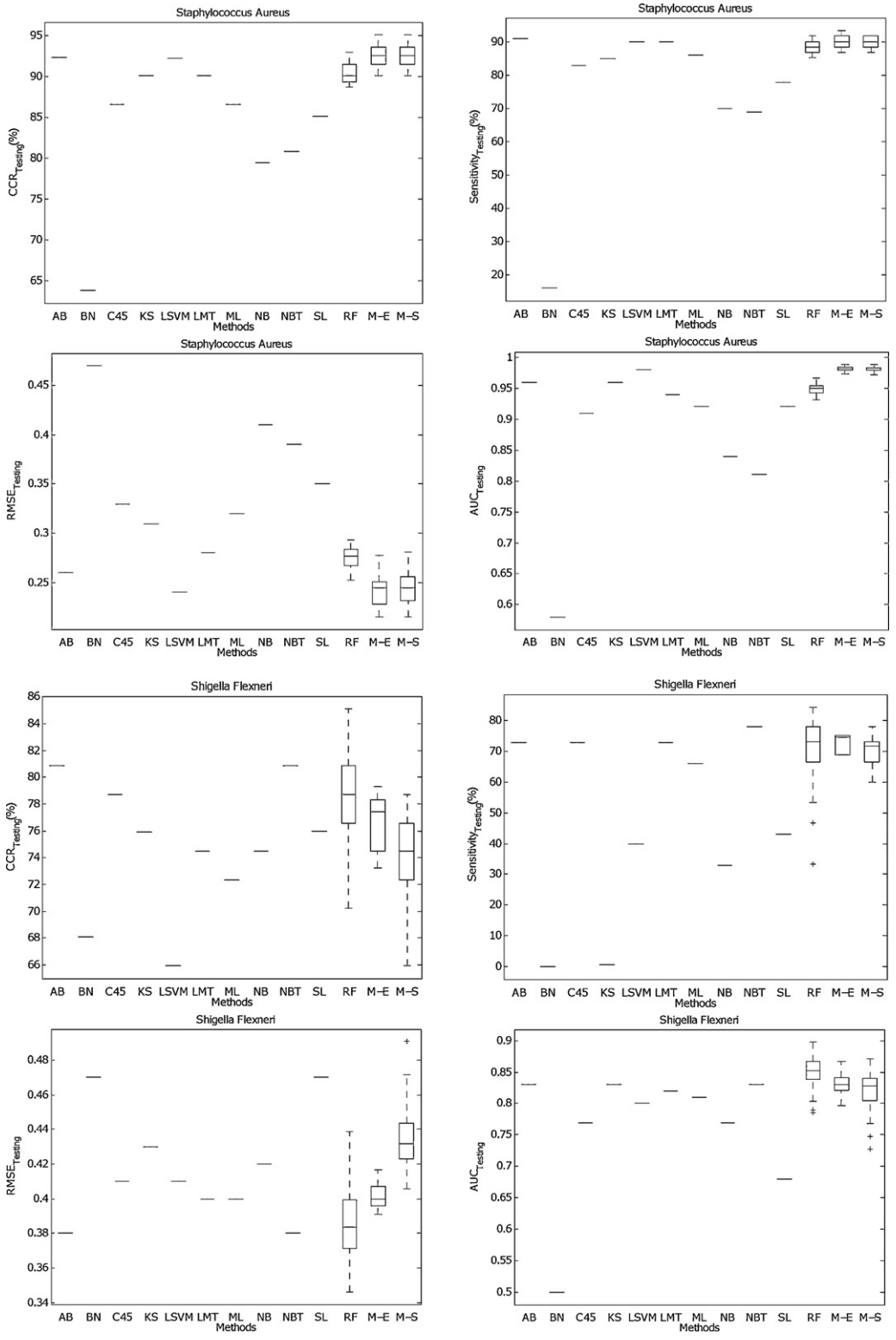

Fig. 7. *CCR*, *S*, *RMSE* and *AUC* measurements on Box Plots for *Staphylococcus aureus* and *Shigella flexneri*.

ing method, $R = 13$ for the worst one and summing 0.5 to the ranked position in case of tie). The M-E method obtains the best mean ranking for *CCR*, *S*, *RMSE* and *AUC* measures ($\bar{R} = 2.25, \bar{R} = 2.50, \bar{R} = 1.62, \bar{R} = 2.25$, respectively). Also, the results for the best individual or model obtained with the MPENSGA2 methodologies in 30 runs are shown for each pathogen.

To quantify whether a statistical difference exists between any of these algorithms, a procedure for comparing multiple classifiers over multiple datasets is employed [73]. This procedure begins with the Friedman test [74], using the *CCR*, *S*, *RMSE* and *AUC* rankings of all the methods as the test variables. This test is a non-parametric equivalent to the repeated measures ANOVA test, and, in our case,

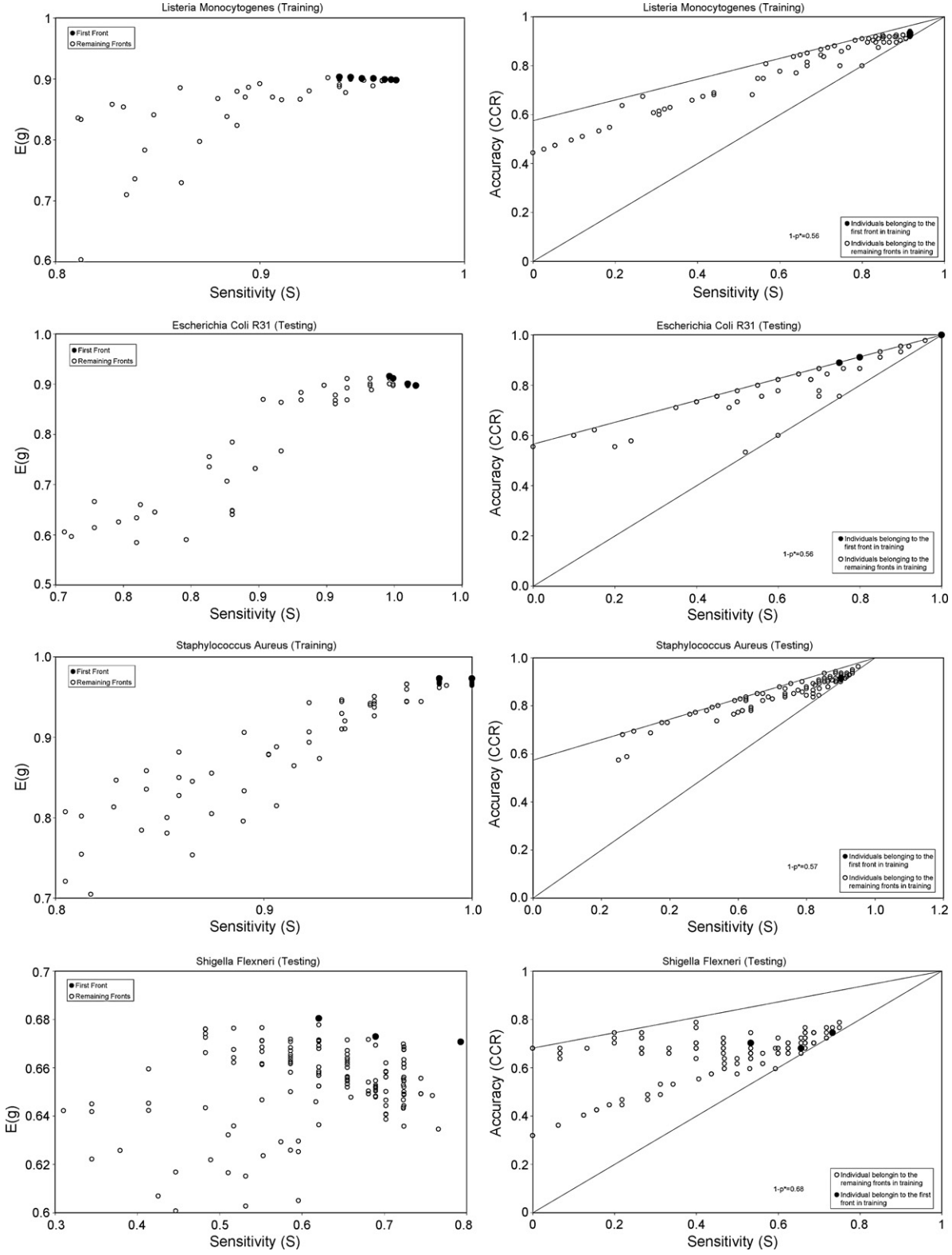

Fig. 8. Pareto front in training in (*S*,*E*) space and associated values in testing in (*S*,*CCR*) space for *Listeria monocytogenes*, *Escherichia coli R31*, *Staphylococcus aureus* and *Shigella flexneri* in one specific run.

it is applied since a previous evaluation of the *CCR*, *S*, *RMSE* and *AUC* values results in rejecting the normality and equality of the variances' hypothesis. Applying this test to the average ranks in Table 3, the test shows that the effect of the method used for classification is statistically significant at a significance level of 5%, as the confidence interval is $C_0 = (0, F_{0.05} = 2.03)$ and the *F*-distribution statistical values are $F_{CCR} = 3.76 \notin C_0$, $F_S = 5.71 \notin C_0$, $F_{RMSE} = 6.07 \notin C_0$ and $F_{AUCC} = 6.01 \notin C_0$. Consequently, we reject the null-hypothesis stating that all algorithms perform equally in mean ranking and a post hoc test is warranted for further investigation. On the basis of this rejection, the Nemenyi post hoc test is used to compare all the classifiers among themselves. The differences in rankings between the different algorithms for *CCR*, *S*, *RMSE* and *AUC* measures and the results of the Nemenyi test for $\alpha = 0.1$ and $\alpha = 0.05$ can be seen in Tables 4–7, respectively, using corresponding critical values. By using this test, it can be seen that the M-E method significantly outperforms the BayesNet method, BN, when using all the measures for $\alpha = 0.05$, and it outperforms SimpleLogistic, SL, when using *RMSE* measure for $\alpha = 0.1$. The M-S method significantly outperforms the BayesNet method when using *S* and *AUC* measures for $\alpha = 0.05$. However, it has been noted that the approach of comparing all the classifiers among themselves in a post hoc test is not as sensitive as the approach of comparing all the classifiers to a given classifier (a control method) [73]. One approach of this latter type of comparison is the Bonferroni–Dunn test. The results of the Bonferroni–Dunn test for $\alpha = 0.1$ and $\alpha = 0.05$ can be seen in the same tables, using corresponding critical values for the two-tailed Bonferroni–Dunn test. From the results of these tests, it can be seen that the M-E method significantly outperforms the BayesNet method for $\alpha = 0.05$ when using all the measurements, the MultiLogistic, ML, when using *CCR* measure, the Naïve Bayes, NB, when using *RMSE* measure; and the SimpleLogistic when using *S*, *RMSE* and *AUC* measures. It outperforms Naïve Bayes, for $\alpha = 0.1$ when using *S* measure; the Simple Logistic when using *CCR* measure; and C4.5 when using the *AUC* measure. The M-S method significantly outperforms BayesNet for $\alpha = 0.05$ when using all measures except *CCR*, and, for $\alpha = 0.1$, it outperforms SimpleLogistic when using the *S* measure.

From all these results, it can be concluded that the M-E method leads to a very competitive performance, obtaining the highest mean rank when considering all the datasets and all the measures. The statistical tests confirm that these differences are significant when the method is compared to the Bayes Net, MultiLogistic, Naive Bayes, C4.5 and Simple Logistic methods.

Figs. 6 and 7 show the Box Plots [75] for the *CCR*, *S*, *RMSE* and *AUC* measurements for each pathogen. The boxes show the lines at the lower quartile, median, and upper quartile values of the metrics. The bottom and top bars show the metric values at 1.5-IQR (Interquartile Range) below the lower quartile value and 1.5-IQR above the upper quartile value, respectively. Outliers are data with metric values beyond the ends of the two bars. For deterministic algorithms we only have one result for each dataset, therefore values as lower and upper quartile and the smallest and largest observation cannot be represented. As we can see in Figs. 6 and 7, in *CCR*, the degree of dispersion of M-E and R-F results is lower than the degree of dispersion of the results generated by M-S. In *S*, the degree of dispersion of M-E is, in general, lower than R-F and M-S. In *RMSE* the degree of dispersion is low and is very similar for the three methodologies. In *AUC*, the degree of dispersion is also low and similar in all methodologies.

In Fig. 8, we can see the graphical results obtained for the MPENSGA2 algorithm for the datasets *L. monocytogenes*, *E. coli R31*, *S. aureus* and *S. flexneri* in the training (*S*,*E*) and the test (*S*,*CCR*) spaces. For the (*S*,*E*) space we select the Pareto front for one specific run, out of the 30 realized for each dataset. Concretely the execution chosen is that which presents the best individual in Entropy in training, as Entropy and Sensitivity are the objectives that guide MPENSGA2. On the (*S*,*CCR*) testing graphics, we show the *S* (Sensitivity) and *CCR* (Accuracy) values throughout the testing set for the individuals who are reflected in the (*S*,*E*) training graphics. Observe that the (*S*,*CCR*) values do not form Pareto fronts in testing, and the individuals that were in the first Pareto front in the training graphics can now be found within the (*S*,*CCR*) space in a worse region. In general the structure of a Pareto front in training is not maintained in testing. Sometimes it is very difficult to obtain classifiers with a high percentage of classification and a high percentage of sensitivity, and for this reason some fronts have very few individuals.

8. Conclusions

In this paper we study the improvement of the generalization ability of neural classifiers with two classes aiming at maximizing the percentage of correctly classified patterns for each class. The main purpose is to optimize the accuracy, while keeping a balance of individual performance with respect to each class of the problem. The inclusion of the two-objective sensitivity and accuracy (*S*,*CCR*) approach reveals a new standpoint to deal with classification problems. An Evolutionary Algorithm based on Pareto dominance and hybridized with a modified local search algorithm has been applied in order to optimize these two objectives.

It should be emphasized that the evolutionary process obtains models of MLP networks with rising values of *E* and *S* at the beginning of the evolution. Then, when high values of *E* and/or *S* in the training set are reached, the MPENSGA2 methodologies obtains Pareto fronts with high accuracy models without reducing the level of sensitivity and vice versa.

Moreover, it can be noted that sensitivity values obtained by the MPENSGA2E methodology are similar to and even better than the values obtained with other well-known machine learning methodologies, and that greater classification accuracy in testing data are obtained for the two classes of each problem. On the other hand, the MPENSGA2S methodology also results in very high values of *CCR* and *S*.

These methodologies have been applied to different datasets from the field of predictive microbiology. The statistical results obtained and the multiple means comparison tests analyzed make MPENSGA2E a competitive method to be considered in this field, where there is a high necessity of obtaining good classification accuracy for both the growth and no-growth classes. This approach can help predictive modellers to better define the growth boundaries of microorganisms and to model the microbial variability associated to the conditions. In conclusion, the use of this method constitutes a valuable alternative for mathematical modelling to determine microbial growth probability under a certain set of conditions.

Acknowledgements

This work has been partially financed by the TIN 2008-06681-C06-03 project (Spanish Inter-Ministerial Commission of Science and Technology), FEDER funds and the P08-TIC-3745 project (Junta de Andalucia (Spain)).

References

[1] A. Yardimci, Soft computing in medicine, Applied Soft Computing 9 (2009) 1029–1043.
[2] D. Garcia, A.J. Ramos, V. Sanchis, S. Marin, Predicting mycotoxins in foods: a review, Food Microbiology 26 (2009) 757–769.
[3] W. Wan, S. Mabu, K. Shimada, K. Hirasawa, J. Hu, Enhancing the generalization ability of neural networks through controlling the hidden layers, Applied Soft Computing 9 (1) (2009) 404–414.
[4] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[5] S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd edition, Prentice Hall, 1998.
[6] S. Ari, G. Saha, In search of an optimization technique for artificial neural network to classify abnormal heart sounds, Applied Soft Computing 9 (1) (2009) 330–340.
[7] C.N. Gupta, R. Palaniappan, S. Swaminathan, S.M. Krishnan, Neural network classification of homomorphic segmented heart sounds, Applied Soft Computing 7 (1) (2007) 286–297.
[8] C.A. Coello, G.B. Lamont, D.A.V. Veldhuizen, Evolutionary Algorithms for Solving Multi-objective Problems, Springer, 2007.
[9] A. Valero, F. Pérez-Rodríguez, E. Carrasco, J.M. Fuentes-Alventosa, R.M. García-Gimeno, G. Zurera, Modelling the growth boundaries of *Staphylococcus aureus*: effect of temperature, pH and water activity, International Journal of Food Microbiology 133 (2009) 186–194.
[10] T Ross, P. Dalgaard, S. Tienungoon, Predictive modelling of the growth and survival of *Listeria* in fishery products, International Journal of Food Microbiology 62 (2000) 231–245.
[11] T.A. McMeekin, T. Ross, Modeling applications, Journal of Food Protection Suppl. (1996) 1–88.
[12] D. Schaffner, T.P. Labuza, Summing it up: predictive microbiology, Food Technology 51 (1997) 95–99.
[13] T.A. McMeekin, K. Presser, D.A. Ratkowsky, T. Ross, M. Salter, S. Tienungoon, Quantifying the hurdle concept by modelling the bacterial growth/no growth interface, International Journal of Food Microbiology 55 (2000) 93–98.
[14] C. Genigeorgis, S. Martin, C.E. Franti, H. Reimann, Initiation of staphylococcal growth in laboratory media, Applied Microbiology 21 (1971) 934–939.
[15] D.A. Ratkowsky, T. Ross, Modelling the bacterial growth/no growth interface, Letters in Applied Microbiology 20 (1995) 29–33.
[16] R. Caruana, A. Niculescu-Mizil, Data mining in metric space: an empirical analysis of supervised learning performance criteria, in: Proceedings of the 10th Int. Conf. Knowl. Disc. Data Mining, 2004, pp. 69–78.
[17] T. Fawcett, An introduction to ROC analysis, Pattern Recognition Letters 27 (2006) 861–874.
[18] S. Gídlant, Neural-network Learning and Expert Systems, MIT Press, 1993.
[19] R. Parekh, J. Yang, V. Honavar, Constructive neural-network learning algorithms for pattern classification, IEEE Transactions on Neural Networks 11 (2000) 436–450.
[20] R. Reed, Pruning algorithms. A survey, IEEE Transactions on Neural Networks 4 (1993) 740–747.
[21] T. Kondo, Evolutionary design and behavior analysis of neuromodulatory neural networks for mobile robots control, Applied Soft Computing 7 (1) (2007) 189–202.
[22] A. Saxena, A. Saad, Evolving an artificial neural network classifier for condition monitoring of rotating mechanical systems, Applied Soft Computing 7 (1) (2007) 441–454.
[23] A.J. van Rooij, L.C. Jain, R.P. Johnson, Neural Networks Training Using Genetic Algorithms, World Scientific, Series in Machine Perception and Artificial Intelligence, 1996.
[24] X. Yao, Y. Liu, A new evolutionary system for evolving artificial neural networks, IEEE Transactions on Neural Networks 8 (3) (1997) 694–713.
[25] X. Yao, Evolving artificial neural networks, Proceedings of the IEEE (1999) 1423–1447.
[26] W. Yan, Z. Zhu, R. Hu, Hybrid genetic/BP algorithm and its application for radar target classification, in: Proceedings of the 1997 IEEE National Aerospace and Electronics Conference, NAECON, IEEE Press, 1997, pp. 981–984.
[27] I.A. Basheer, M.N. Hajmeer, Artificial neural networks: fundamentals, computation, design and application, Journal of Microbiological Methods 43 (2000) 3–31.
[28] M.N. Hajmeer, I.A. Basheer, J.L. Mardsden, D.Y.C. Fung, New approach for modeling generalized microbial growth curves using artificial neural networks, Journal of Rapid Methods & Automation in Microbiology 8 (4) (2000) 265–284.
[29] A.W. Schepers, J. Thibault, C. Lacroix, Comparison of simple neural network and nonlinear regression models for descriptive modeling of *Lactobacillus helveticus* growth in pH-controlled batch cultures, Enzyme and Microbial Technology 26 (2000) 431–445.
[30] S. Jeyamkondan, D.S. Jayas, R.A. Holley, Microbial growth modelling with artificial neural networks, International Journal of Food Microbiology 64 (2001) 343–354.
[31] W. Lou, S. Nakai, Application of artificial neural networks for predicting the thermal inactivation of bacteria: a combined effect of temperature, pH, and water activity, Food Research International 34 (7) (2001) 573–579.
[32] R.M. García-Gimeno, C. Hervas-Martinez, E. Barco-Alcala, G. Zurera-Cosano, E. Sanz-Tapia, An artificial neural network approach to *Escherichia coli* O157:H7 growth estimation, Journal of Food Science 68 (2003) 639–645.
[33] R.M. García-Gimeno, C. Hervas-Martinez, R. Rodriguez-Perez, G. Zurera-Cosano, Modelling the growth of *Leuconostoc mesenteroides* by artificial neural networks, International Journal of Food Microbiology (2005) 317–332.
[34] M. Hajmeer, I. Basheer, A probabilistic neural network approach for modeling and classification of bacterial growth/no-growth data, Journal of Microbiological Methods 51 (2002) 217–226.
[35] E.A. El-Sebakhy, K.A. Faisal, Bacterial growth classification with support vector machines: a comparative study, in: The 2007 International Conference on Machine Learning; Models, Technologies and Applications (MLMTA'07), CSREA Press, 2007, pp. 90–98.
[36] E.A. El-Sebakhy, I. Raharja, S. Adem, Y. Khaeruzzaman, Neuro-fuzzy systems modeling tools for bacterial growth, in: EEE/ACS International Conference on Computer Systems and Applications, 2007, pp. 374–380.
[37] M.N. Hajmeer, I.A. Basheer, Comparison of logistic regression and neural network-based classifiers for bacterial growth, Food Microbiology 20 (2003) 43–55.
[38] A. Valero, C. Hervás, R.M. García-Gimeno, G. Zurera, Product unit neural networks models for predicting the growth limits of *Listeria monocytogenes*, Food Microbiology 24 (2007) 452–464.
[39] K. Deb, Multi-objective Optimization using Evolutionary Algorithms, John Wiley & Sons, Ltd., 2004.
[40] Y. Jin, B. Sendhoff, Pareto-based multiobjective machine learning: an overview and case studies, IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews 38 (3) (2008) 397–415.
[41] H.A. Abbass, Speeding up backpropagation using multiobjective evolutionary algorithms, Neural Computation 15 (2003) 2705–2726.
[42] V.V.R. Silva, P.J. Fleming, J. Sugimoto, R. Yokoyama, Multiobjective optimization using variable complexity modelling for control system design, Applied Soft Computing 8 (1) (2008) 392–401.
[43] F. Pettersson, N. Chakraborti, H. Saxén, A genetic algorithms based multiobjective neural net applied to noisy blast furnace data, Applied Soft Computing 7 (1) (2007) 387–397.
[44] C. Cotta, P. Moscato, A gentle introduction to memetic algorithms, in: F. Glover, G.A. Kochenberger (Eds.), Handbook on Metaheuristics, Kluwer Academics, 2001.
[45] Y. Jin, B. Sendholf, E. Körner, Simultaneous generation of accurate and interpretable neural network classifiers, Studies in Computational Intelligence (SCI) 6 (2006) 291–312.
[46] S. Wiegand, C. Igel, Evolutionary multi-objective optimization of neural networks for face detection, International Journal of Computation Intelligence and Applications 4 (3) (2004) 237–253.
[47] S. Roth, A. Gepperth, C. Igel, Multi-objective neural network optimization for visual object detection, Multi-Objective Machine Learning (2006) 629–655.
[48] F. Provost, T. Fawcett, Robust classification system for imprecise environments, in: Proceedings of the Fifteenth National Conference on Artificial Intelligence, 1998, pp. 706–713.
[49] K. Deb, A. Pratab, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA2, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[50] C. Igel, M. Hüsken, Improving the Rprop learning algorithm, in: Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000), ICSC, Academic Press, 2000, pp. 115–121.
[51] P.J. Angeline, G.M. Saunders, J.B. Pollack, An evolutionary algorithm that constructs recurrent neural networks, IEEE Transactions on Neural Networks 5 (1) (1994) 54–65.
[52] C. Hervás, F.J. Martínez-Estudillo, Logistic regression using covariates obtained by product-unit neural network models, Pattern Recognition 40 (2007) 52–64.
[53] A. Martinez-Estudillo, F. Martinez-Estudillo, C. Hervas-Martinez, N. Garcia-Pedrajas, Evolutionary product unit based neural networks for regression, Neural Networks 19 (4) (2006) 477–486.
[54] F.J. Martínez-Estudillo, C. Hervás-Martínez, P.A. Gutiérrez, A.C. Martínez-Estudillo, Evolutionary product-unit neural networks classifiers, Neurocomputing 72 (1–3) (2008) 548–561.
[55] C.R. Houck, J.A. Joines, M.G. Kay, J.R. Wilson, Empirical investigation of the benefits of partial Lamarckianism, Evolutionary Computation 5 (1997) 31–60.
[56] Y.S. Ong, M.H. Lim, N. Zhu, K.W. Wong, Classification of adaptive memetic algorithms: a comparative study, IEEE Transactions on Systems, Man and Cybernetics Part B 36 (1) (2006) 141–152.
[57] J.E. Smith, Coevolving memetic algorithms: a review and progress report, IEEE Transactions on Systems, Man and Cybernetics Part B 37 (1) (2007) 6–17.
[58] Y Jin, T. Okabe, B. Sendhoff, Neural network regularization and ensembling using multi-objective evolutionary algorithms, in: Proceedings of 2004 Congress on Evolutionary and Ensembling Using Multi-objective Evolutionary Algorithms, 2004, pp. 1–8.
[59] H.A. Abbass, An evolutionary artificial neural networks approach for breast cancer diagnosis, Artificial Intelligence in Medicine 25 (2002) 265–281.
[60] A. Gepperth, S. Roth, Applications of multi-objective structure optimization, Neurocomputing 69 (2006) 701–703.
[61] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: the RPROP algorithm, neural networks, in: IEEE International Conference, 1993, pp. 586–591.
[62] C. Igel, M. Hüsken, Empirical evaluation of the improved Rprop learning algorithms, Neurocomputing 50 (6) (2003) 105–123.
[63] A. Valero, C. Hervás, R.M. García-Gimeno, G. Zurera, Searching for new mathematical growth model approaches for *Listeria monocytogenes*, Journal of Food Science 72 (1) (2007) 16–25.
[64] M.A. Salter, D.A. Ratkowsky, T. Ross, T.A. McMeekin, Modelling the combined temperature and salt (NaCl) limits for growth of a pathogenic *Escherichia coli* strain using nonlinear logistic regression, International Journal of Food Microbiology 61 (2000) 159–167.
[65] L.L. Zaika, E. Moulden, L. Weimer, J.G. Phillips, R.L. Buchanan, Model for the combined effects of temperature, initial pH, sodium chloride and sodium nitrite concentrations on anaerobic growth of *Shigella flexneri*, International Journal of Food Microbiology 23 (1994) 345–358.
[66] A.P Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognition 30 (7) (1997) 1145–1159.

[67] N. Landwehr, M. Hall, F. Eibe, Logistic model trees, Machine Learning 59 (2005) 161–205.
[68] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2005.
[69] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001.
[70] C.A. Hwang, M.L. Tamplin, The influence of mayonnaise pH and storage temperature on the growth of *Listeria monocytogenes* in seafood salad, International Journal of Food Microbiology 102 (2005) 277–285.
[71] E. 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.
[72] J.M. Soriano, G. Font, J.C. Moltó, J. Mañes, Enterotoxigenic staphylococci and their toxins in restaurant foods, Trends of Food Science and Technology 13 (2002) 60–67.
[73] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[74] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Annals of Mathematical Statistics 11 (1) (1940) 86–92.
[75] J.W. Tukey, Exploratory Data Analysis, Addison-Wesley, 1977.