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

aDepartment of Computer Science and Numerical Analysis, University of Cordoba, Rabanales Campus, Albert Einstein building 3º floor, 14071, Córdoba, Spain

bDepartment of Management and Quantitative Methods, ETEA, Escritor Castilla Aguayo 4, 14005, Córdoba, Spain

Received date here; revised date here; accepted date here

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.

© 2009 Elsevier Science. All rights reserved

Keywords: Accuracy; Classification, Memetic Algorithms; Predictive Microbiology; Multiobjective; Neural Networks; Sensitivity

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 Multi-Objective 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 behavior 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 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 1970’s literature when Genigeorgis [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 is shown. 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, multi-layer feed-forward 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 modeling complex time-dependent bacterial growth [27,28,29], or for predicting growth parameters such as lag time and exponential growth rate [30,31,32,33] 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 nonlinearity 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 O157: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 Escherichia 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 Escherichia 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 Escherichia Coli R31 in response to temperature and water activity. In [35] and [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 Escherichia 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 [41,42,43].

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 [41,45,46,47].

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, 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

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

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 feed forward 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 \( x = (1, x_1, x_2, \ldots, x_N) \) 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 following expression:

\[
g(x, \theta) = \frac{\exp f(x, \theta)}{1 + \exp f(x, \theta)},
\]

where \( g(x, \theta) \) is the probability a pattern \( x \) has of belonging to the growth class, and \( 1 - g(x, \theta) \) is the probability a pattern \( x \) has of belonging to the no-growth
class, $\theta = (\beta_0, \ldots, \beta_M, w_1, \ldots, w_n)$ is the vector of weights of the output node, $w_j = (w_j', \ldots, w_j')$ is the vector of inputs weights of the hidden node $j$, and $f(x, \theta)$ is the output of the output node for pattern $x$ given by

$$f(x, \theta) = \beta_0 + \sum_{j=1}^{M} \beta_j \sigma \left( \sum_{i=1}^{n} w'_j x_i \right)$$

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

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

4.2. Fitness Functions

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

$$CCR = \frac{1}{N} \sum_{n=1}^{N} I(C(x_n) = y_n)$$

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, $E$:

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

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

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

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 [52,53,54]).

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 $t-\text{th}$ generation is:

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

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

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.

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, it 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 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 [58,59,60] 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 \( \eta \). If the last iteration produced the same sign, the update value is multiplied by a factor \( \eta^2 \). 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 minimise 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 micro-organism
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
Listeria Monocytogenes, Escherichia Coli R31, Staphylococcus Aureus and Shigella 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 \( \eta' = 1.2, \eta = 0.5, \Delta_i = 0.0125 \) (the initial
value of the \( \Delta_j \)), \( \Delta_{\text{init}} = 0, \Delta_{\text{max}} = 50 \) and \( \text{Epochs} = 25 \), 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 \( \alpha \) 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 Listeria Monocytogenes, [64] for Escherichia Coli
R31, [9] for Staphylococcus Aureus and [65] for
Shigella 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:
onece the first Pareto front is calculated using the 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,

\[
\overline{E}I_{testing} = (\overline{CCR}_{testing}, \overline{S}_{testing}), \quad \overline{S}_{testing} = (\overline{CCR}_{testing}, \overline{S}_{testing})
\]

The first expression is the average obtained taking entropy into
account as the primary objective, and the second taking
to account as the primary objective. So, the
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 eleven 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].
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).
- 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 *Shigella 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).
- NaiveBayesUpdateable (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. Listeria Monocytogenes

*Listeria 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]).

*Listeria 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. Escherichia Coli

This dataset is given by Salter et al. [64] and it pertaining to growth/no-growth of an *Escherichia Coli* strain R31I affected by temperature and water activity. The data consists of experiments performed with different combinations of temperature in the 7.7 to 37.0 range and water activity in the 0.943 to 0.987 range. All samples of *Escherichia 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. Staphylococcus Aureus

*Staphylococcus 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 *Staphylococcus 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 *Staphylococcus 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) and at 19Aw levels (from 0.856 to 0.999 at regular intervals). The initial dataset (287 conditions) was divided into 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 in comparison to other studies to obtain the growth/no-growth transition.

### 6.4. *Shigella Flexneri*

*Shigella 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 summarized in Table 2 referenced. Growth of *Shigella 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 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 performing 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, 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 rankings 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 \(C_{95}=(0, F_{(1,6)crit}=2.03)\) and the \(F\)-distribution statistical values are \(F_{(1,6)crit}=3.76 \not\in C_{95}\), \(F_{(1,6)crit}=5.71 \not\in C_{95}\), \(F_{(1,6)crit}=6.07 \not\in C_{95}\) and \(F_{(1,6)crit}=6.01 \not\in C_{95}\). 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, 5, 6 and 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).
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 methodology 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 MPENSGA2 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 MPENSGA2 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.
Acknowledgment

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


List of Figures

Fig. 1. Unfeasible region in the two dimensional (S, CCR) space of a concrete classification problem.
Fig. 2. Feed Forward Neural Network with sigmoidal basis units for binary classification.
Fig. 3. MPENSGA2 algorithm pseudocode.
Fig. 4. Memetic Framework proposed
Fig. 5. Achieving statistical results from MPENSGA2.
Fig. 6. CCR , S , RMSE and AUC measurements on Box Plots for Listeria Monocytogenes and Escherichia Coli R31.
Fig. 7. CCR , S , RMSE and AUC measurements on Box Plots for Staphylococcus Aureus and Shigella Flexneri.
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.

List of Tables

Table 1. Characteristics of Listeria Monocytogenes, Escherichia Coli R31, Staphylococcus Aureus and Shigella Flexneri pathogens. First class growth, second class no-growth.
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].
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 (CCR(%) , S(%) , RMSE and AUC) and mean ranking of their performance (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.
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.
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.
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.
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.