Learning Artificial Neural Networks Multiclassifiers by Evolutionary Multiobjective Differential Evolution Guided by Statistical Distributions

M. Cruz-Ramírez, C. Hervás-Martínez, *Member, IEEE,* J. C. Fernández, *Member, IEEE* and J. Sánchez-Monedero

***Abstract*— This work presents an Evolutionary Artificial Neural Network (EANN) approach based on the Pareto Differential Evolution (PDE) algorithm where the crossover operator is determined using a Gaussian distribution associated with the best models in the evolutionary population. The crossover operator used in a real-coded genetic algorithm is based on confidence intervals. The PDE is used to localize the most promising search regions for locating the best individuals. Confidence intervals use mean localization and standard deviation estimators that are highly recommendable when the distribution of the random variables is Gaussian. It has always been an issue to find good ANN architecture in both multiclassification problems and in the field of ANNs. EANNs provide a better method to optimize simultaneously both network performance (based on the Correct Classification Rate, C) and the network performance of each class (Minimum Sensitivity, MS). The proposal with respect to methodology performance is evaluated using a well characterized set of multiclassification benchmark problems. The results show that crossover performance based on confidence intervals is less dependent on the problem than crossover performance based on a random selection of three parents in the PDE.**

I. Introduction

As multiclass classification tasks are ubiquitous in the real world, there is a growing interest in multiclassification problems in the machine learning community. A classifier design method is usually an algorithm that develops a classifier to approximate an unknown input-output mapping function from finitely available data, i.e., training samples. Once this classifier has been designed, it can be used to predict class labels that correspond to unseen samples. Hence, the objective in developing a good classifier is to ensure high prediction accuracy for unseen future data, i.e., testing capability. Many techniques have been proposed to improve the overall testing capability of classifier designed methods (assuming, for example, the maximization of the correct classification rate), but very few methods maintain this capacity in all classes (assuming, for example, maximization of the correct classification of each class). This second objective is very important in some research areas (such as medicine, remote sensing, economy, etc.) to ensure the benefits of one classifier over another. Thus, in this paper we have proposed the simultaneous optimization of the two conflicting objectives in multiclass problems. The solution for these "multi-objective" problems is different from that of a single-objective optimization. The main difference is that multi-objective optimization problems normally have not one but a set of solutions which are all equally good. The different classification methods found in the literature [1], include a very important tool used in the last few years, Artificial Neural Networks (ANNs) [2].

M. Cruz-Ramírez is with the Department of Computer Science and Numerical Analysis, University of Córdoba, Campus de Rabanales, Albert Einstein building, 3rd floor, 14071 - Córdoba, Spain, Tel.: +34 957 218 349; fax: +34 957 218 630; e-mail: i42crram@uco.es.

The training of Artificial Neural Networks by Evolutionary Pareto-based algorithms [3] is known as Multiobjective Evolutionary Artificial Neural Networks (MOEANNs), and has been used to solve classification tasks [4], some of its main exponents being H. Abbass [5] and Y. Jin [3].

Differential Evolution (DE) [6], an evolutionary optimization method for continuous search spaces, has been used by Ilonen [7] to train the weights of feed-forward neural networks. Other authors, like Bhuiyan [8], use DE to optimize the architecture for Artificial Neural Networks.

This paper presents a new perspective on the Memetic Pareto Differential Evolution (MPDE) algorithm because the selection of three parents in the crossover operator is guided by statistical distributions using Gaussian distribution. This Multiobjective Evolutionary Algorithm (MOEA) [9] is based on Differential Evolution and on the Pareto dominance concept for solving multiclass classification problems. MPDE is improved with a local search algorithm, concretely improved Resilient Backpropagation (iRprop$^+$) [10]. The methodology proposed is tested throughout six datasets taken from the UCI repository [11], specifically Breast-Cancer, Breast-Wisconsin, Heart-Disease, Heart-Statlog, Horse and Newthyroid.

The rest of the paper is organized as follows. Section 2 shows an explanation of Accuracy and Sensitivity. Section 3 describes the algorithms presented, followed by the experimental design in Section 4. Section 5 shows the results obtained while conclusions and future research are outlined in Section 6.

II. Related Work

A. Accuracy and Minimum Sensitivity

In this section we present two measures to evaluate a classifier: the Correct Classification Rate or Accuracy, C, and Minimum Sensitivity, MS. To evaluate a classifier, the machine learning community has traditionally used C to measure its default performance. Actually, we simply have to realize that C cannot capture all the different behavioral aspects found in two different classifiers in multiclassification problems. For these problems, two performance measures are considered; traditionally-used C and MS in all classes, that is, the lowest percentage of examples correctly predicted as belonging to each class, S_i, with respect to the total number of examples in the corresponding class, $MS = \min\{S_i\}$. The pair made up of Minimum Sensitivity versus Accuracy (MS, C) expresses two features associated with a classifier: global performance (C) and the rate of the worst classified class (MS). The selection of MS as a complementary measure of C can be justified by considering that C is the weighted average of the Sensitivities of each class. For a more detailed description of these measures, see [12].

One point in (MS, C) space *dominates* another if it is above and to the right, i.e. it has greater C and the best MS. Let C and MS be associated with a classifier g, then $MS \leq C \leq 1 - (1 - MS)p^*$, where p^* is the minimum for estimated prior probabilities. Therefore, each classifier will be represented as a point in the white region in Figure 1, hence the area outside of the triangle is marked as unfeasible.

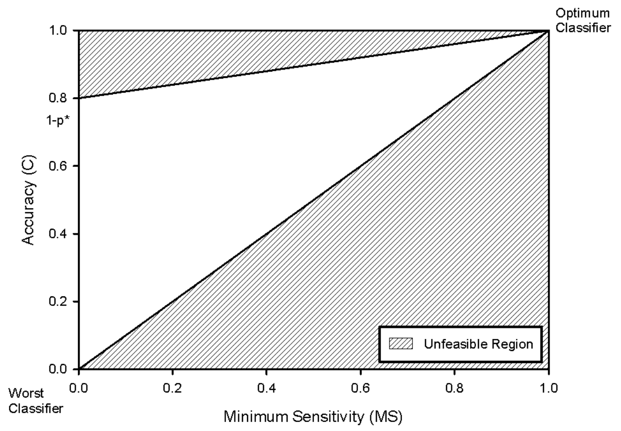

Fig. 1: Unfeasible region in two dimensional (MS, C) space.

The area inside the triangle in Figure 1 may be feasible or not (attainable), depending upon the classifier and the difficulty of the problem. *A priori*, we could think that MS and C objectives could be positively correlated, but while this may be true for small values of MS and C, it is not for values close to 1 in either MS or C. Thus competitive objectives are at the top right corner of the white region. This fact justifies the use of a MOEA.

III. Learning Methodology

The beginning of this section describes the neural networks and the Fitness Functions employed. The proposed algorithms are shown and the section concludes with a description of the local search algorithm used.

A. Base Classifier

For multi-classification problems, we consider standard feed forward Multilayer Perceptron (MLP) neural networks with one input layer with independent variables or features, one hidden layer with sigmoidal hidden nodes and one output node.

Let a coded "1-of-J" outcome variable $\mathbf{y}$, (that is the outcomes have the form $\mathbf{y} = (y^{(1)}, y^{(2)}, ..., y^{(J)})$, where $y^{(j)} = 1$ if the pattern belongs to class j, and $y^{(j)} = 0$, in other cases); and a vector $\mathbf{x} = (1, x_1, x_2, ..., x_K)$ of input variables, where K is the number of input (we assume that the vector of inputs includes the constant term to accommodate the intercept or bias).

Then, the output layer is interpreted from the point of view of probability which considers the softmax activation function. The activation function of the *j-th* node in the hidden layer is given by:

$$g_l(\mathbf{x}, \boldsymbol{\theta}_l) = \frac{\exp f_l(\mathbf{x}, \boldsymbol{\theta}_l)}{1 + \exp f_l(\mathbf{x}, \boldsymbol{\theta}_l)}, \text{for } l = 1, ..., J$$

where $g_l(\mathbf{x}, \boldsymbol{\theta}_l)$ is the probability a pattern $\mathbf{x}$ has of belonging to class l, $\boldsymbol{\theta}_l = (\beta_0^l, ..., \beta_M^l, \mathbf{w}_1, ..., \mathbf{w}_M)$ is the vector of weights of the output node, M is the number of hidden nodes, $\mathbf{w}_j = \{w_0^j, ..., w_K^j\}, \text{for } j = 1, ..., M,$ is the vector of input weights of the hidden node j, and $f_l(\mathbf{x}, \boldsymbol{\theta}_l)$ is the output of the output node for pattern x given by:

$$f_l(\mathbf{x}, \boldsymbol{\theta}_l) = \beta_0^l + \sum_{j=1}^{M} \beta_j^l \sigma \left(w_0^j + \sum_{i=1}^{K} w_i^j x_i \right), \text{for } l = 1, ..., J$$

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

In order to tackle this classification problem, the outputs of the model have been interpreted from the point of view of probability through the use of the softmax activation function [13], which is given by:

$$p_l(\mathbf{x}, \boldsymbol{\theta}_l) = \frac{\exp f_l(\mathbf{x}, \boldsymbol{\theta}_l)}{\sum_{j=1}^{J} \exp f_j(\mathbf{x}, \boldsymbol{\theta}_j)}, \text{for } l = 1, ..., J \qquad (1)$$

where $f_j(\mathbf{x}, \boldsymbol{\theta}_l)$ is the output of the j output neuron for pattern $\mathbf{x}$ and $p_l(\mathbf{x}, \boldsymbol{\theta}_l)$ is the probability that pattern $\mathbf{x}$ has of belonging to class j.

Using the softmax activation function presented in expression 1, the class predicted by the MLP corresponds to the node in the output layer whose output value is the greatest. In this way, the optimum classification rule $C(\mathbf{x})$ is the following:

$$C(\mathbf{x}) = \widehat{l}, \text{where } \widehat{l} = \text{argmax}_l \; p_l(\mathbf{x}, \boldsymbol{\theta}_l), \text{for } l = 1, ..., J$$

The best MLP is determined by means of a MOEA (detailed in Section III-C) that optimizes the error function given by the negative log-likelihood for N observations associated with the MLP model:

$$L^*(\theta) = \frac{1}{N}\sum_{n=1}^{N}\left[-\sum_{l=1}^{J-1} y_n^{(l)} f_l(\mathbf{x}_n, \theta_l) + \right. \qquad (2)$$
$$\left. + \log \sum_{l=1}^{J-1} \exp f_l(\mathbf{x}_n, \theta_l)\right]$$

where $y_n^{(l)}$ is equal to 1 if pattern $\mathbf{x}_n$ belongs to the *l-th* class and equal to 0 otherwise. From a statistical point of view, this approach can be seen as nonlinear multinominal logistic regression, where we optimize log-likelihood using a MOEA.

B. Fitness Functions

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

$$C = (1/N)\sum_{n=1}^{N}(I(C(\mathbf{x}_n) = \mathbf{y}_n))$$

where $I(\cdot)$ is the zero-one loss function, $\mathbf{y}_n$ 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 C in a given problem. However, the C measure is a discontinuous function, which makes convergence more difficult in neural network optimization.

Thus, instead of C, we consider the continuous function given in expression 2, also called Entropy (E). The advantage of using the error function $E(g, \theta)$ instead of C is that this is a continuous function, which makes the convergence more robust.

As a first objective, we propose a strictly decreasing transformation of the $E(g, \theta)$ as the fitness measure to maximize:

$$A(g) = \frac{1}{1 + E(g, \theta_l)}, 0 < A(g) \leq 1$$

where g is the multivaluated function:

$$g(\mathbf{x}, \theta) = (g_1(\mathbf{x}, \theta_1), ..., g_J(\mathbf{x}, \theta_J))$$

The second objective to maximize is the MS of the classifier. 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.

C. Memetic Pareto Algorithm

We construct two MOEAs with a local search algorithm. They are called the Memetic Pareto Differential Evolutionary Neural Network (MPDENN) and the Memetic Pareto Differential Evolutionary Neural Network using Confidence Intervals with L2 Norm (MPDENN-L2), which tries to move the classifier population towards the optimum classifier located at the $(1, 1)$ point in (MS, C) space. The MOEAs proposed are based on the PDE [14] and the local search algorithm is the Improved Resilient Backpropagation–*iRprop*$^+$ [15].

The MOEAs used in this study consider a fully specified ANN to be an individual which evolves architectures and connection weights simultaneously. 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 MPDENN is based on the algorithm described in [16]. In MPDENN, local search does not apply to all the children to be added to the population. Instead, the most representative children in the population are optimized throughout several generations. The pseudocode of MPDENN is shown in Figure 2.

```
1:  Create a random initial population $P_0$
2:  while Stop condition is not met do
3:      Evaluate population
4:      Adjust the size of the population
5:      while The population is not complete do
6:          Select parents
7:          Cross parents
8:          Mutate the child
9:          Evaluate the child
10:         Add the child to the population according to dominance relationships with the main parent
11:     end while
12:     if Local search in this generation then
13:         if Number of individuals of the first Pareto front of $P_k < num$ then
14:             Apply iRprop$^+$ to the individuals in the first Pareto front
15:         else
16:             Generate $num$ cluster in the first Pareto front using K-means
17:             Apply iRprop$^+$ to the $num$ centers
18:         end if
19:     end if
20: end while
```

Fig. 2: MPDENN algorithm pseudocode.

The algorithm starts generating a random population P_0 of size M. The population is sorted according to the non-domination concept explained in Section II-A. Dominated individuals are removed from the population. Then the population is adjusted until its size is between 3 and half the maximum size by adding dominated individuals or deleting individuals according to their respective distance from their nearest neighbor. After that, the population is completed with new children generated from three randomly selected individuals in the population. The child is generated by crossing the three parents. The resultant child is a perturbation of the main parent. This perturbation occurs with a probability of P_c for each neuron. This perturbation may be structural, according to expression (3), so that neurons are removed or added to the hidden layer; or parametric, according to expression (4) (for the hidden layer) or (5) (for the output layer), where the weights of the primary parent are modified with the difference of the weights of the secondary parents.

$$\rho_h^{child} \leftarrow \begin{cases} 1 & if\left(\rho_h^{\alpha_1} + N(0,1)\left(\rho_h^{\alpha_2} - \rho_h^{\alpha_3}\right)\right) \geq 0.5 \\ 0 & otherwise \end{cases} \tag{3}$$

$$w_{ih}^{child} \leftarrow w_{ih}^{\alpha_1} + N(0,1)\left(w_{ih}^{\alpha_2} - w_{ih}^{\alpha_3}\right) \tag{4}$$

$$w_{ho}^{child} \leftarrow w_{ho}^{\alpha_1} + N(0,1)\left(w_{ho}^{\alpha_2} - w_{ho}^{\alpha_3}\right) \tag{5}$$

Afterwards, the mutation operator is applied to the child. The mutation operator consists of adding or deleting neurons in the hidden layer depending on a P_m probability for each of them. Taking into account the maximum number of hidden neurons that may exist in an individual in a specific problem, the probability will be used the same number of times as the number of neurons that are found in the classifier. If the neuron exists, it is deleted, but if it does not exist, then it is created and the weights are established randomly, according to expression (6).

$$\rho_h^{child} \leftarrow \begin{cases} 1 & if \rho_h^{child} = 0 \\ 0 & otherwise \end{cases} \tag{6}$$

Finally, the child is added to the population according to dominance relationships with the main parent. In some generations, depending on the size of the first Pareto front, local search is applied to all the individuals in the first Pareto front or the most representative individuals in this front (obtained by the K-means algorithm [17]).

On the other hand, the MPDENN-L2 is based on the MPDENN algorithm and in MPDENN, parents are randomly selected from among individuals in the population while MPDENN-L2 uses virtual parents generated from confidence intervals. These confidence intervals are generated using the L2 norm. The rest of the algorithm works like MPDENN. The pseudocode of MPDENN-L2 is shown in Figure 3.

To generate virtual parents, it is necessary to calculate the mean and standard deviation of each of the weights of individuals in the population. Then, for each of the weights, the limits of the interval must be calculated according to expression (7).

$$limit_i \leftarrow t_{n-1,\alpha} \frac{S_i}{\sqrt{n}} \tag{7}$$

where n is the number of individuals in the population and $t_{n-1,\alpha}$ is the value of Student's *t* distribution with $n-1$ degrees of freedom and $1-\alpha$ as the confidence coefficient.

With the average value of each weight and its limits, an interval is formed according to expression (8).

$$CI_i = \left[\hat{\mu}_i - t_{n-1,\alpha}\frac{S_i}{\sqrt{n}}; \hat{\mu}_i + t_{n-1,\alpha}\frac{S_i}{\sqrt{n}}\right] \tag{8}$$

```
1:  Create a random initial population $P_0$
2:  while Stop condition is not met do
3:      Evaluate population
4:      Adjust the size of the population
5:      while The population is not complete do
6:          Generate virtual parents:
7:              Calculate the average of each of the weights of
                individuals belonging to $P_k$
8:              $CIM \leftarrow$ individual generated from the averages
9:              Calculate the standard deviation of each of the
                weights of individuals belonging to $P_k$
10:             Calculate the width of the confidence interval of
                each of the weights
                    $limit_i \leftarrow t_{n-1,\alpha}\frac{S_i}{\sqrt{n}}$
11:             $CILL_i \leftarrow CIM_i - limit_i$
12:             $CIUL_i \leftarrow CIM_i + limit_i$
13:             Set CIM as main parent and CILL and CIUL as
                secondary parents
14:         Cross parents
15:         Mutate the child
16:         Evaluate the child
17:         Add the child in the population according to domi-
            nance relationships with the main parent
18:     end while
19:     if Local search in this generation then
20:         Apply local search
21:     end if
22: end while
```

Fig. 3: MPDENN-L2 algorithm pseudocode.

From the confidence interval of expression (8), the three individuals built are considered to be the parents in the crossover. These three parents are formed by: all the lower limit values of the confidence intervals of the chromosome gene individual, CILL$_i = \hat{\mu}_i - t_{n-1,\alpha}\frac{S_i}{\sqrt{n}}$; all the upper limit values of the confidence intervals of the chromosome gene individual, CIUL$_i = \hat{\mu}_i + t_{n-1,\alpha}\frac{S_i}{\sqrt{n}}$; and all the means of the confidence intervals of the chromosome gene individual, CIM$_i = \hat{\mu}_i$. These parents have statistical information on the localization features and dispersion of the best individuals in the population, that is, the genetic information that the fittest individuals share.

D. Local Search Algorithm

The Evolutionary Algorithms, EAs, are improved by the incorporation of local search procedures throughout their evolution. Some studies that were 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 has great difficulties in finding the best solution when it is in a region near a global optimum. It is well-known that certain local

TABLE I: CHARACTERISTICS FOR DATASETS

Dataset	#Patterns	#Training patterns	#Test patterns	#Input variables	#Classes	#Patterns per class	p^*
Breast-Cancer	286	215	71	15	2	(201,85)	0.2957
Breast-Wisconsin	699	524	175	9	2	(458,241)	0.3428
Heart-Disease	302	226	76	26	2	(164,138)	0.4605
Heart-Statlog	270	202	68	13	2	(150,120)	0.4411
Horse	364	273	91	58	3	(224,88,52)	0.1428
Newthyroid	215	161	54	5	3	(150,35,30)	0.1296

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, EA was going to carry out a global search inside the solution space, 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 Algorithm [18].

Many MOEAs use local optimizers to fine tune ANN weights. This is called "lifetime learning" and it consists of updating each individual with respect to 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. This procedure has a high computational cost, something that we wanted to avoid. For this reason we propose the following:

The local search algorithm is only applied in three generations of evolution (the first to start, the second in the middle and the third at the end) once the population is completed. Thus, local search is not applied to those children who are rejected. Local search does not apply to all individuals, only to the most representative. The process for selecting these individuals is as follows: if the number of individuals in the first Pareto front is lower than or equal to the desired number of clusters (num), a local search is carried out without needing to apply K-means [17]. But, if the number of individuals in the first front is greater than num, the K-means is applied to the first front to get the most representative num individuals, who will then be the object of a local search.

This local search will improve the Pareto front obtained with respect to only one objective, specifically that which seeks to minimize classification error.

As far as we are concerned, *Rprop* (resilient Backpropagation) algorithm [10] is used because it is one of the best techniques in terms of convergence speed, accuracy and robustness.

IV. EXPERIMENTS

Six datasets taken from the UCI repository are considered for the experimental design. This design was conducted using a stratified holdout procedure with 30 runs, where approximately 75% of the patterns were randomly selected for the training set and the remaining 25% for the test set.

In all the experiments, the population size is established at $M = 25$. The crossover probability is 0.8 and the mutation probability is 0.1. For iRprop$^+$, the parameters adopted 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 $Epochs = 5$, see [10] for iRprop$^+$ parameter description. The optimization process is applied 3 times during execution (every 33.33% generations) and uses $num = 5$ cluster in the clustering algorithm. For confidence intervals, alpha takes values in the range $[0.9, 1]$. To start processing data, each one of the input variables was scaled in the ranks $[-1.0, 1.0]$ to avoid the saturation of the signal.

In Table I we can see the features for each dataset. The total number of instances or patterns in each dataset appear, as well as 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 prior estimated probabilities).

During the experiment, models are trained using the fitness function $A(g)$ (based on E, see Section III-B) and MS as objective functions, but when validated, we use C and MS. $A(g)$ is used instead of C in training because C is a discontinuous function, which makes convergence more difficult in optimization.

Once the Pareto front is built, two methodologies are considered in order to build a neural network model which then includes the information about the models within it. These are called MethodName-E and MethodName-MS. These methodologies provide single models that can be compared to other classification methods found in the literature. The process followed in these methodologies is the following: once the first Pareto front is calculated using training set patterns, the best individual belonging to the Pareto front on E (EI) is chosen for MethodName-E, and the best individual in terms of MS (MSI) is selected for MethodName-MS. Once this is done, the values of C and MS are obtained by testing the EI and MSI individual models. Therefore we obtain an individual $EI_G = (C_G, MS_G)$ and an individual $MSI_G = (C_G, MS_G)$. This is repeated 30 times and then estimations are carried out of the average and standard deviation obtained from the individuals $\overline{EI}_G = (\overline{C}_G, \overline{MS}_G)$ and $\overline{MSI}_G = (\overline{C}_G, \overline{MS}_G)$. The first expression is the average obtained taking E into account as the primary objective, and the second one is obtained by taking MS into account

TABLE II: Statistical Results for PDENN, MPDENN and MPDENN-L2 in testing

Methodology	Dataset	$C_G(\%)$ Mean±SD	$S_G(\%)$ Mean±SD	Dataset	$C_G(\%)$ Mean±SD	$S_G(\%)$ Mean±SD
PDENN-E	Breast Cancer	66.66±3.16	38.09±12.62	Breast-Wisconsin	95.15±1.11	89.61±3.08
PDENN-MS		63.66±3.59	56.32±7.49		95.12±1.04	90.00±2.55
MPDENN-E		67.98±3.05	41.27±13.07		95.22±1.06	89.89±3.24
MPDENN-MS		63.99±3.80	**58.77±6.23**		95.20±0.99	90.17±2.71
MPDENN-L2-E		**69.34±3.36**	30.16±12.90		**95.68±1.00**	**90.17±2.6**0
MPDENN-L2-MS		63.43±4.61	52.35±9.18		95.30±0.92	90.06±2.34
PDENN-E	Heart-Disease	83.64±2.01	78.80±3.41	Heart-Statlog	76.07±1.38	63.77±2.99
PDENN-MS		82.41±2.67	78.60±3.94		76.47±1.89	**65.88±3.68**
MPDENN-E		83.55±2.46	77.32±4.54		77.25±1.00	61.56±1.59
MPDENN-MS		83.33±2.38	79.21±4.45		76.37±1.59	62.89±2.43
MPDENN-L2-E		**85.31±2.83**	**80.78±3.70**		**77.89±1.47**	62.12±4.24
MPDENN-L2-MS		**85.31±2.83**	**80.78±3.70**		77.65±1.52	62.44±4.10
PDENN-E	Horse	57.95±10.00	0.00±0.00	Newthyroid	97.28±1.99	83.91±11.72
PDENN-MS		57.95±10.00	0.00±0.00		96.91±2.44	84.23±11.20
MPDENN-E		60.66±5.25	0.00±0.00		96.73±2.21	82.91±10.51
MPDENN-MS		60.66±5.25	0.00±0.00		96.60±2.12	82.45±10.35
MPDENN-L2-E		**65.31±3.05**	0.00±0.00		**97.90±1.99**	**85.56±11.70**
MPDENN-L2-MS		**65.31±3.05**	0.00±0.00		96.73±2.74	83.60±12.46

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 MethodName-E and the second MethodName-MS.

V. Results

Table II presents the values of the mean and standard deviation for C and MS in 30 runs for all the experiments performed. It can be seen that the MPDENN-L2 algorithm produces good results with respect to C and MS. In fact, from a descriptive point of view, the MPDENN-L2-E algorithm obtains the best result in C_G in all the datasets and the best result in MS_G in three datasets.

In the Heart-Disease and Horse datasets, some algorithms get the same results with both methodologies. This is because the first Pareto front, in the 30 executions, is formed by a single individual.

In Figure 4, we can see the graphic results obtained by the MPDENN-L2 algorithm for the Breast-Cancer and Heart-Disease datasets in the training $(MS, A(g))$ and test (MS, C) spaces. For the $(MS, A(g))$ space, the Pareto front is selected for one specific run output of the 30 done for each dataset, concretely the execution that presents the best E individual in training, where $A(g)$ and MS are the objectives that guide MPDENN-L2. The (MS, C) testing graphs show MS and C values throughout the testing set for the individuals who are reflected in the $(MS, A(g))$ training graphs. Observe that the (MS, C) values do not form Pareto fronts in testing, and the individuals that had been in the first Pareto front in the training graphics may now find themselves located in a worse region in the space. 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.

We can see that an increase in $A(g)$ in Breast-Cancer causes a decrease in MS in training, and that an increase in C causes a decrease in MS for testing. The result is a Pareto front that is quite dispersed and has a fair number of models. However, for the Heart-Disease dataset, we observe a Pareto front with a single individual. This is because during the evolutionary process, there was no individual who was not dominated by the individual of the Pareto front.

In the graph of Breast-Cancer in training, we have identified the two ends of the first Pareto front. The testing graph shows the position of these models in space (MS, C).

The ANalysis Of the VAriance of one factor (ANOVA I) statistical method or the non parametric Kruskal-Wallis, (K-W) test were used to determine the best methodology for training MLP neural networks (with respect to their influence on C and MS in the test dataset), depending on the satisfaction of the normality hypothesis of C and MS values. The results of the ANOVA analysis for test C values show that for the six datasets, the effect of the six training methodologies is statistically significant at a 5% level of significance. The results of the ANOVA or KW analysis for C and MS show that for the six datasets, the effect of the methodologies is statistically significant at a 5% level of significance.

Because there exists a significant difference in mean for C and MS using the Snedecor's F or the K-W test; we perform, in the first case, under the normality hypothesis, a post hoc multiple comparison test of the mean C and MS is perfor-

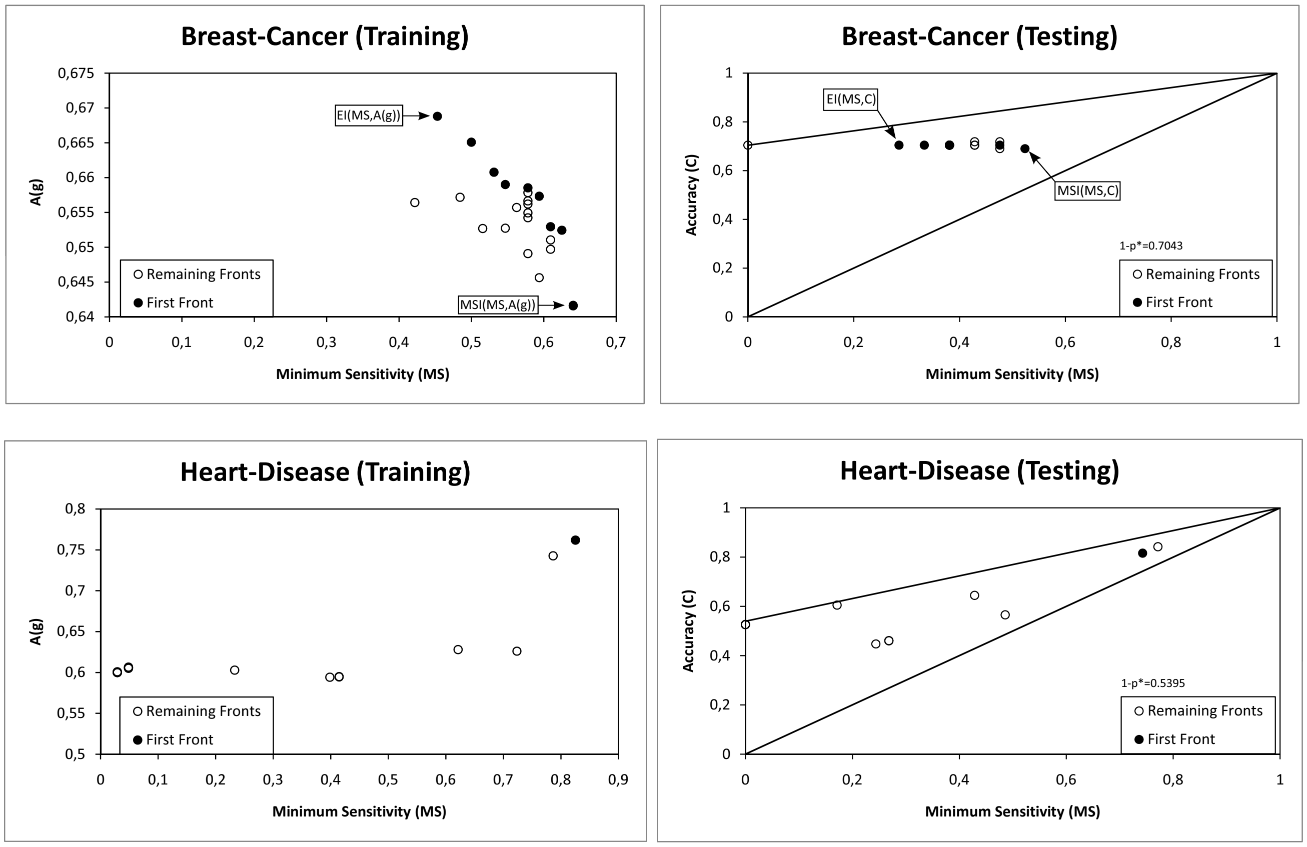

Fig. 4: Pareto front in training $(MS, A(g))$ and (MS, C) associated values in the testing of one specific run out of the 30 runs carried out.

med and obtained with different levels of the factor. A Tukey test under normality is carried out as well as a pair-wise T-test. Table III shows the results obtained (in first column C and then for MS). If we analyze the test results for C, we can observe that the MPDENN-L2-E methodology obtains results that are, in mean, better than or similar to the results of the second best methodology. In four databases there were significant differences when comparing the MPDENN-L2-E to the second best methodology in E (in 3 of these cases, the p-value is 0.10 and in the other, 0.05). On the other hand, the results in mean for MS show that MPDENN-L2-E is the best methodology in two databases, while in two others it is not, although there are no significant differences. MPDENN-L2-MS is the best methodology in one database (in this database, MPDENN-L2-E shows significant differences with respect to other methodologies in E).

VI. Conclusions

In this paper we study the use of two memetic algorithms based on differential evolution. One of them uses the random selection of parents (MPDENN) while the other uses statistical information distribution about individuals in the population to generate three virtual parents, who will then be used to generate a new individual (MPDENN-L2). In these algorithms, we have proposed applying local search to the most representative individuals in the population, selected through clustering techniques, to optimize the most promising individuals.

The best results for C_G and MS_G are obtained with MPDENN-L2, because the patterns of the datasets follow Gaussian distribution. Note that in the Horse dataset, all algorithms obtain 0 in MS_G because it is a not balanced dataset, and therefore resampling methods should be applied. Because MPDENN-L2 gets the best results, we recommend applying confidence intervals to generate virtual parents in differential evolution in those datasets that present Gaussian distribution.

This study suggests several future research directions. First, virtual parents can be generated from confidence intervals that are constructed differently. This can obtain good results in those databases that do not present Gaussian distribution or that have an unknown distribution of data. Second, resampling techniques can be used to solve imbalanced problems. Finally, we are studying the possibility of combining the ends of the first Pareto front to achieve a more robust classifier.

Acknowledgements

This work has been partially subsidized by TIN 2008-06681-C06-03 project of the Spanish Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P08-TIC-3745 project of the “Junta de Andalucía”

TABLE III: POST HOC TUKEY TEST AND T-TEST

Dataset	Means Ranking of the C	Means Ranking of the MS
Breast Cancer	$\boldsymbol{\mu_5} \geq \mu_3 \geq \mu_4 \geq \mu_6 \geq \mu_1 \geq \mu_2$; $(^\circ)\boldsymbol{\mu_5} > \mu_3$	$\boldsymbol{\mu_4} \geq \mu_2 \geq \mu_6 > \mu_3 \geq \mu_1 > \mu_5$
Breast-Wisconsin	$\boldsymbol{\mu_5} \geq \mu_2 \geq \mu_6 \geq \mu_1 \geq \mu_3 \geq \mu_4$; $(^\circ)\boldsymbol{\mu_5} > \mu_1$	$\boldsymbol{\mu_5} \geq \mu_4 \geq \mu_6 \geq \mu_2 \geq \mu_3 \geq \mu_1$
Heart-Disease	$\boldsymbol{\mu_5} \geq \mu_6 \geq \mu_1 \geq \mu_3 \geq \mu_4 \geq \mu_2$; $(^*)\boldsymbol{\mu_5} > \mu_1$	$\boldsymbol{\mu_6} \geq \mu_5 \geq \mu_4 \geq \mu_2 > \mu_3 \geq \mu_1$; $(^*)\boldsymbol{\mu_5} > \mu_1$
Heart-Statlog	$\boldsymbol{\mu_5} \geq \mu_6 \geq \mu_3 \geq \mu_2 \geq \mu_4 \geq \mu_1$; $(^\circ)\boldsymbol{\mu_5} > \mu_3$	$\boldsymbol{\mu_2} \geq \mu_1 \geq \mu_4 \geq \mu_6 \geq \mu_5 \geq \mu_3$
Horse	$\boldsymbol{\mu_5} \geq \mu_6 \geq \mu_3 \geq \mu_4 \geq \mu_1 \geq \mu_2$	$---$
Newthyroid	$\boldsymbol{\mu_5} \geq \mu_1 \geq \mu_2 \geq \mu_3 \geq \mu_6 \geq \mu_4$	$\boldsymbol{\mu_5} \geq \mu_2 \geq \mu_1 \geq \mu_6 \geq \mu_3 \geq \mu_4$

PDENN-E(1); PDENN-MS(2); MPDENN-E(3); MPDENN-MS(4); MPDENN-L2-E(5); MPDENN-L2-MS(6)
$(^*)(^\circ)$*The average difference is significant for MPDENN-L2-E with p-values= 0.05 or 0.10.* $\mu_A \geq \mu_B$*: methodology A yields better results than methodology B, but the differences are not significant;* $\mu_A > \mu_B$*: methodology A yields better results than methodology B with significant differences. The binary relation* $\geq$ *is not transitive.*

(Spain). The research of Manuel Cruz Ramírez has been funded by the "Junta de Andalucía", grant reference P08-TIC-3745.

REFERENCES

[1] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification, second edition*. Wiley Interscience, 2001.

[2] G. P. Zhang, "Neural networks for classification: A survey," *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, vol. 30, no. 4, pp. 451–462, 2000.

[3] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 38, no. 3, pp. 397–415, 2008.

[4] G. B. Ou and Y. L. Murphey, "Multi-class pattern classification using neural networks," *Pattern Recognition*, vol. 40, no. 1, pp. 4–18, 2007, ou, Guobin Murphey, Yi Lu.

[5] H. Abbass, "An evolutionary artificial neural networks approach for breast cancer diagnosis," *Artificial Intelligence in Medicine*, vol. 25, no. 3, pp. 265–281, 2002.

[6] R. Storn, *Differential evolution research - Trends and open questions*, 2008, vol. 143.

[7] J. Ilonen, J. Kamarainen, and J. Lampinen, "Differential evolution training algorithm for feed-forward neural networks," *Neural Processing Letters*, vol. 17, pp. 93–105, 2003.

[8] M. Bhuiyan, "An algorithm for determining neural network architecture using differential evolution," 2009, pp. 3–7.

[9] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*, 2nd ed. Springer, September 2007.

[10] C. Igel and M. Hüsken, "Empirical evaluation of the improved rprop learning algorithms," *Neurocomputing*, vol. 50, no. 6, pp. 105–123, 2003.

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

[12] J. C. Fernández, F. J. Martínez, C. Hervás, and P. A. Gutiérrez, "Sensitivity versus accuracy in multi-class problems using memetic pareto evolutionary neural networks," *IEEE Transactions Neural Networks, accepted*, 2009.

[13] D. Richard and E. R. David, "Product units: a computationally powerful and biologically plausible extension to backpropagation networks," *Neural Comput.*, vol. 1, no. 1, pp. 133–142, 1989, 78412.

[14] H. A. Abbass, "A memetic pareto evolutionary approach to artificial neural networks," in *AI2001*, M. Brooks, D. Corbet, and M. Stumptner, Eds. LNAI 2256, Springer-Verlag, 2001, pp. 1–12.

[15] C. Igel and M. Hüsken, "Improving the rprop learning algorithm," *Proc. Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000), ICSC Academic Press*, pp. 115–121, 2000.

[16] J. C. Fernández, C. Hervás, F. J. Martínez, P. A. Gutiérrez, and M. Cruz, "Memetic pareto differential evolution for designing artificial neural networks in multiclassification problems using cross-entropy versus sensitivity," in *Hybrid Artificial Intelligence Systems*, vol. 5572. Springer Berlin / Heidelberg, 2009, pp. 433–441.

[17] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. U. C. Berkeley Press, 1967, pp. 281–297.

[18] P. Moscato and C. Cotta, "A gentle introduction to memetic algorithms," in *Handbook of Metaheuristics*, ser. International Series in Operations Research and Management Science. Springer New York, 2003, vol. 57, pp. 105–144.