Neural Process Lett (2011) 34:101–116
DOI 10.1007/s11063-011-9186-9

Weighting Efficient Accuracy and Minimum Sensitivity for Evolving Multi-Class Classifiers

Javier Sánchez-Monedero · Pedro A. Gutiérrez · F. Fernández-Navarro · C. Hervás-Martínez

Published online: 28 May 2011

Abstract Recently, a multi-objective Sensitivity–Accuracy based methodology has been proposed for building classifiers for multi-class problems. This technique is especially suitable for imbalanced and multi-class datasets. Moreover, the high computational cost of multi-objective approaches is well known so more efficient alternatives must be explored. This paper presents an efficient alternative to the Pareto based solution when considering both Minimum Sensitivity and Accuracy in multi-class classifiers. Alternatives are implemented by extending the Evolutionary Extreme Learning Machine algorithm for training artificial neural networks. Experiments were performed to select the best option after considering alternative proposals and related methods. Based on the experiments, this methodology is competitive in Accuracy, Minimum Sensitivity and efficiency.

Keywords Artificial neural networks · Extreme learning machine · Evolutionary ELM · Multi-class · Imbalanced datasets · Accuracy · Sensitivity · Differential evolution

1 Introduction

Global performance measures such as the Correct Classification Rate (CCR) or Accuracy are not enough to evaluate classifiers [1]. Even in two class problems, Accuracy reflects a one-dimensional ordering where two different types of errors can be found. Alternative

J. Sánchez-Monedero (✉) · P. A. Gutiérrez · F. Fernández-Navarro · C. Hervás-Martínez
Department of Computer Science and Numerical Analysis, University of Córdoba, Rabanales Campus, Albert Einstein building 3rd floor, 14071 Córdoba, Spain
e-mail: jsanchezm@uco.es

P. A. Gutiérrez
e-mail: pagutierrez@uco.es

F. Fernández-Navarro
e-mail: i22fenaf@uco.es

C. Hervás-Martínez
e-mail: chervas@uco.es

measures are especially needed in imbalanced datasets—problems where the number of patterns in each class is significantly different.

Martinez et al. [2] address the problem of the one dimensional consideration in multi-class problems. In this work, two measures are considered to evaluate a classifier: traditionally used accuracy (C) and the minimum sensitivities of all classes (MS); that is, the premise assumed is that a good classifier should combine a high correct classification rate level in the generalization set with an acceptable accuracy level for each class. For further details about these measures please check [2,3].

We consider a classification problem with Q classes and N training patterns with g as a classifier obtaining a $Q \times Q$ contingency or confusion matrix $M(g) = \left\{n_{ij};\ \sum_{i,j=1}^{Q} n_{ij} = N\right\}$ where n_{ij} represents the number of times the patterns are predicted by classifier g to be in class j when they really belong to class i.

Two scalar measures are defined considering different points of view of the elements in the confusion matrix. Let us denote the number of patterns associated with class i by $n_i = \sum_{j=1}^{Q} n_{ij},\ i = 1, \ldots, Q$. Let $S_i = n_{ii}/n_i$ be the number of patterns correctly predicted to be in class i with respect to the total number of patterns in i (sensitivity S_i for class i). Therefore, the sensitivity for class i estimates the probability of correctly predicting a class i example. From the above quantities, Sensitivity MS of the classifier is defined as the minimum value of the sensitivities for each class, $MS = \min\{S_i;\ i = 1, \ldots, Q\}$. The Correct Classification Rate or Accuracy is defined as $C = (1/N)\sum_{j=1}^{Q} n_{jj}$, which is the rate of all correct predictions.

These measures have recently been proposed to address evolutionary training for multilayer perceptron (MLP) neural networks [3]. In [3] C and MS are presented as objectives which can be competitive. This fact justifies the use of an evolutionary multi-objective algorithm to train artificial neural networks (ANNs). The results are presented by showing the performance, in terms of C and MS measures of the individuals belonging to the two extremes of the Pareto front for C and MS.

In general, a weighted combination of objective functions does not necessary perform better (in accuracy) than a Pareto approach, performing worse in some cases [4]. However, it is well known that they are computationally costly [5] and weighted combination can be a more efficient alternative, specially when only two objectives are considered. In addition, these algorithms' output is a set of possible solutions to the problem; thus, an expert or a decision-making system is needed to select the proper solution belonging to the Pareto front. To overcome these issues, this paper attempts to improve the minimum sensitivity of binary and multi-class classifiers by using a weighted linear combination as an alternative to the Pareto based solution. The Evolutionary Extreme Learning Machine (E-ELM) algorithm [6] is extended with different fitness functions which are automatically adjusted to each dataset by the proposed algorithm.

This article is a significant extension and improvement of a previous work [7]. The fitness function design has been extended from a discrete function to a continuous one with improved results. In addition the fitness function λ parameter is automatically selected by the revised algorithm so that no extra parameters are needed.

The rest of the paper is organized as follows. Section 2 briefly presents ELM, Evolutionary ELM, and proposes different fitness functions and the Evolutionary ELM considering C and MS (E-ELM-CS) algorithm. Section 3 shows experiments comparing several proposed fitness functions and different related state of the art methods. Finally, some conclusions are drawn.

2 Proposed Method

2.1 Differential Evolution and Extreme Learning Machine

The Extreme Learning Machine (ELM) algorithm has been proposed by Huang et al. [8,9]. It has been successfully applied to problems such as QoS Violation detection in multimedia transmission [10] or microarray gene expression cancer diagnosis [11]. This section briefly presents ELM algorithm and the Evolutionary ELM.

Let us consider the training set given by N samples $D = \{(\mathbf{x}_j, \mathbf{y}_j) : \mathbf{x}_j \in R^K, \mathbf{y}_j \in R^Q, j = 1, 2, \ldots, N\}$, where $\mathbf{x}_j$ is a $K \times 1$ input vector and $\mathbf{y}_j$ is a $Q \times 1$ target vector.

Let us consider the MLP with M nodes in the hidden layer given by $f(\mathbf{x}, \theta) = (f_1(\mathbf{x}, \boldsymbol{\theta}_1), f_2(\mathbf{x}, \boldsymbol{\theta}_2), \ldots, f_Q(\mathbf{x}, \boldsymbol{\theta}_Q))$:

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

where $\theta = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_Q)^T$ is the transpose matrix containing all the neural net weights, $\boldsymbol{\theta}_l = (\beta^l, \mathbf{w}_1, \ldots, \mathbf{w}_M)$ is the vector of weights of the l output node, $\beta^l = \beta_0^l, \beta_1^l, \ldots, \beta_M^l$ is the vector of weights of the connections between the hidden layer and the lth output node, $\mathbf{w}_j = (w_{1j}, \ldots, w_{Kj})$ is the vector of weights of the connections between the input layer and the jth hidden node, Q is the number of classes in the problem, M is the number of sigmoidal units in the hidden layer and $\sigma_j\left(\mathbf{x}, \mathbf{w}_j\right)$ the sigmoidal function:

$$\sigma_j\left(\mathbf{x}, \mathbf{w}_j\right) = \frac{1}{1 + \exp\left(-\left(w_{0j} + \sum_{i=1}^{K} w_{ij} x_i\right)\right)}, \tag{2}$$

where w_{0j} is the bias of the jth hidden node. Suppose that a MLP is being trained with M-nodes in the hidden layer to learn the N samples of set D. The linear system $f(\mathbf{x}_j) = \mathbf{y}_j$, $j = 1, 2, \ldots, N$, can be written in a more compact format as $\mathbf{H}\boldsymbol{\beta} = \mathbf{Y}$, where $\mathbf{H}$ is the hidden layer output matrix of the network:

$$H\left(\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{w}_1, \ldots, \mathbf{w}_M\right) = \begin{bmatrix} \sigma\left(\mathbf{w}_1 \cdot \mathbf{x}_1\right) & \cdots & \sigma\left(\mathbf{w}_M \cdot \mathbf{x}_1\right) \\ \vdots & \ddots & \vdots \\ \sigma\left(\mathbf{w}_1 \cdot \mathbf{x}_N\right) & \cdots & \sigma\left(\mathbf{w}_M \cdot \mathbf{x}_N\right) \end{bmatrix}_{N \times M},$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_M \end{bmatrix}_{M \times Q} \quad \text{and} \quad \mathbf{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}_{N \times Q}$$

The ELM algorithm randomly selects the $\mathbf{w}_j = (w_{1j}, \ldots, w_{Kj})$, $j = 1, \ldots, M$, weights and biases for hidden nodes, and analytically determines the output weights $\beta_0^l, \beta_1^l, \ldots, \beta_M^l$ for $l = 1 \ldots Q$ by finding the least square solution to the given linear system. The minimum norm least-square solution (LS) to the linear system is $\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{Y}$, where $\mathbf{H}^{\dagger}$ is the Moore–Penrose (MP) generalized inverse of matrix $\mathbf{H}$. The minimum norm LS solution is unique and has the smallest norm among all the LS solutions.

The Evolutionary Extreme Learning Machine (E-ELM) [6] improves the original ELM by using a Differential Evolution (DE) algorithm. Differential Evolution was proposed by Storn and Price [12] and it is known as one of the most efficient evolutionary algorithms which many applications such as artificial neural networks training [13]. The E-ELM uses DE to select the input weights between input and hidden layers and the Moore–Penrose generalized inverse to analytically determine the output weights between hidden and output layers.

2.2 Fitness Function Design

As mentioned at the introduction, our approach tries to build classifiers with simultaneously optimized C and MS. Since these objectives are not always cooperative [2,3], an evolutionary multi-objective approach could be used.

In this work, a linear combination of these objectives is used to obtain the maximization of objectives C and MS. This option is a good alternative to proper multi-objective evolutionary algorithms when there are two objectives and when the first Pareto front has a very small number of models. In addition, its computational cost is noticeably lower.

Weighted linear combination proves to be very efficient in practice for certain types of problems, for example in combinatorial multi-objective optimization. Some of the applications of this technique are to schedule evaluation of a resource scheduler or to design multiplierless infinite impulse response (IIR) filters [5].

We assume we do not have a priori information about the proper weighting of C and MS for each dataset. Thus, both measures are considered equally important. Thereby, we deal with this problem by adapting the algorithm to each dataset through a nested cross-validation procedure. Our purpose is to design a fitness function able to weight up both C and MS objectives in the algorithm.

There is no rule for establishing priorities between C and MS. Thus, we include a parameter for weighing the two objectives. The underlying idea is to have an automatically adjustable fitness function which could be optimized for each dataset via this parameter, called λ, ranging between [0, 1]. In this work, three fitness functions are proposed to try to balance the two objectives.

The first fitness function is based on C and MS. This function evaluates the performance of a classifier depending on a weighted Accuracy level and a weighted Sensitivity. It is defined by:

$$F_{\lambda CS} = (1-\lambda)C + \lambda MS, \tag{3}$$

According to [14,15], in general terms, the use of continuous function for training neural networks for classification problems makes the convergence of the algorithm more robust. By using the root mean square error (RMSE) or the cross-entropy error [14] the fitness function is turned into a continuous function.

In order to properly calculate the RMSE and the cross-entropy error, the neural network outputs need to be interpreted as probabilities p_l, thereby, they must satisfy the following constraints [14]:

$$\sum_{l=1}^{Q} p_l(\mathbf{x}, \boldsymbol{\theta}_l) = 1, \tag{4}$$

$$0 \leq p_l(\mathbf{x}, \boldsymbol{\theta}_l) \leq 1. \tag{5}$$

The first constraint also ensures that the distribution is correctly normalized, so that $\int p(\mathbf{y}|\mathbf{x})d\mathbf{y} = 1$. These constraints can be satisfied by choosing a p_l output to be related to corresponding network outputs f_l by a softmax function [16]. Then, the softmax activation function is added to standard ELM model outputs:

$$p^l = p_l(\mathbf{x}, \boldsymbol{\theta}_l) = \frac{\exp(f_l(\mathbf{x}, \boldsymbol{\theta}_l))}{\sum_{i=1}^{Q} \exp(f_i(\mathbf{x}, \boldsymbol{\theta}_i))}, \quad 1 \leq l \leq Q, \tag{6}$$

where f_l are the ELM outputs defined in Eq. 1 and p_l is the posterior probability that a pattern $\mathbf{x}$ has of belonging to class l. A pattern will belong to the class with the greatest membership probability, this is:

$$C(\boldsymbol{\theta}_l, \mathbf{x}) = \arg\max_l p_l(\mathbf{x}, \boldsymbol{\theta}_l), \quad 1 \leq l \leq Q. \tag{7}$$

Then, once a probabilistic function from the ELM output is determined, a fitness function based on RMSE is proposed:

$$F_{\lambda R}(\theta) = (1-\lambda)\frac{1}{1+\frac{1}{N}\sum_{l=1}^{Q}(n_l R_l(\theta))} + \lambda\frac{1}{1+\max\{R_l(\theta), l=1,\ldots,Q\}}, \tag{8}$$

where $n_l = \sum_{l=1}^{Q} n_{il}, i = 1, \ldots, Q$ is the number of patterns associated with class l. $R_l(\theta)$ is the RMSE per class in the problem, defined by:

$$R_l(\theta) = \sqrt{\frac{\sum_{i=1}^{N}\sum_{l=1}^{Q}\left(y_i^l - p_i^l(\mathbf{x}_i, \theta)\right)^2}{n_l Q}}, \tag{9}$$

where N is the number of patterns, y_i^l is the target value for class l of pattern $\mathbf{x}_i$ (y_i^l will be equal to 1, in terms of probability, if the pattern $\mathbf{x}_i$ belongs to class l and 0 otherwise), p_i^l is the probability pattern $\mathbf{x}_i$ has of belonging to class l, and n_l is the number of patterns associated with class l.

The fitness function defined in Eq. 8 introduces an alternative for considering C and MS. The first term represents the global accuracy error for all the classes while the second term represents the isolated error of the worst classified class as defined in Eq. 9. The maximum error is selected because it is the equivalent to consider sensitivity, which is the minimum accuracy for each class.

The third fitness function proposed is based on cross-entropy error in a similar way to $F_{\lambda R}$ defined in Eq. 8:

$$F_{\lambda E} = (1-\lambda)\frac{1}{1+\frac{1}{N}\sum_{l=1}^{Q}(n_l E_l)} + \lambda\frac{1}{1+\max\{E_l, l=1,\ldots,Q\}}, \tag{10}$$

where E_l is the cross-entropy error per class in the problem, defined by:

$$E_l = \frac{-\sum_{i=1}^{N}\sum_{l=1}^{Q}(y_i^l \ln(p_l(x_i, \boldsymbol{\theta}_l))}{n_l}, \tag{11}$$

This error function is also known as the negative log likelihood and, when it is minimized, maximum likelihood estimates ($p_l(\mathbf{x}_i, \boldsymbol{\theta}_l)$) are obtained for the event observed.

2.3 The E-ELM-CS Algorithm

Our proposed method is implemented by using the Evolutionary ELM (E-ELM) [6]. E-ELM for classification problems only considers the misclassification rate of the classifier. Further details about the algorithm can be consulted in [6]. The E-ELM has been extended in two ways. First, the three fitness functions designed in Sect. 2.2 are added, including the addition of the softmax layer to the model. Secondly, a 10-fold cross-validation is applied, using exclusively the training data, which aims to optimally configure the λ parameter of the fitness function. Note that after several experiments with different λ values no generic optimal value for λ is found to maximize both C and MS, that is, λ depends on the data set. Therefore, a cross-validation process is mandatory for each different dataset. Then, the algorithm

```
Require: E-ELM-CS (P (Training Patterns), T (Training Tags), F (Fitness Function), $\lambda$)
1: $\hat{\lambda} \leftarrow$ Calculate optimal $\lambda$ for F and (P,T)
2: Create a random initial population $\boldsymbol{\theta} = [\mathbf{w}_1, \ldots, \mathbf{w}_k, b_1, \ldots, b_k]$ of size $N$
3: for each individual do
4:     $\hat{\boldsymbol{\beta}} \leftarrow$ ELM_output($\mathbf{w}, P, T$) {Calculate output weights}
5:     fitness $\leftarrow$ getFitness($\mathbf{w}, \hat{\boldsymbol{\beta}}, F, \hat{\lambda}, P, T$) {Evaluate individual}
6: end for
7: Select best individual of initial population
8: while Stop condition is not met do
9:     Mutate random individuals and apply crossover as described in [27]
10:    for each individual in the new population do
11:        $\hat{\boldsymbol{\beta}} \leftarrow$ ELM_output($\mathbf{w}, P, T$) {Calculate output weights}
12:        fitness $\leftarrow$ getFitness($\mathbf{w}, \hat{\boldsymbol{\beta}}, F, \hat{\lambda}, P, T$) {Evaluate model}
13:        Select new individuals for replacing individuals in old population
14:    end for
15:    Select the best model in the generation
16: end while
17: return Best ELM model

18: function $\hat{\boldsymbol{\beta}} = ELM\_output(\mathbf{w}, P, T)$
19: Calculate the hidden layer output matrix H
20: Calculate the output weight $\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{Y}$
21: return $\hat{\boldsymbol{\beta}}$

22: function $F_{\lambda} = getFitness(\mathbf{w}, \boldsymbol{\beta}, F, \lambda, P, T)$
23: if $F_{\lambda CS}$ then
24:     Build training confusion matrix $\mathbf{M}$
25:     Calculate $C$ and $S$ from $\mathbf{M}$
26:     Get classifier fitness with Eq. (3)
27: else if $F_{\lambda R}$ or $F_{\lambda E}$ then
28:     Add softmax layer to the ELM model ($\mathbf{w}, \hat{\boldsymbol{\beta}}$)
29:     if $F_{\lambda R}$ then
30:         Get classifier fitness with Eq. (8)
31:     else
32:         Get classifier fitness with Eq. (10)
33:     end if
34: end if
35: return Individual fitness
```

Fig. 1 E-ELM-CS algorithm pseudocode

extension is called E-ELM-CS (Evolutionary ELM considering C and MS). The E-ELM-CS algorithm pseudocode is shown in Fig. 1. Mutation, crossover and selection operators work as described in [6].

The cross-validation is performed by testing a range of λ values for the chosen fitness function in E-ELM-CS and a given configuration for the remaining parameters. The training set is stratified into 10 sets so 10 validation configurations can be formed. Each one of the 10 validation tests consists of different combinations of 9 sets for training and a different one for validation. Note that generalization data is never used during this cross-validation procedure so that the final algorithm performance will be measured only with unseen data. For each λ value to validate, the E-ELM-CS algorithm is run three times with the same data partition. Therefore the total number of executions over the training data is 30. The λ considered as optimal is the one shown in the following equation:

$$\hat{\lambda} = \arg\max_{\lambda_i} \frac{\overline{C_{\lambda_i}} + \overline{MS_{\lambda_i}}}{2}, \tag{12}$$

where $\overline{C_{\lambda_i}}$ is the mean C and $\overline{MS_{\lambda_i}}$ is the mean MS obtained by the algorithm in the different validation folds using λ_i for the fitness function.

The parameters related to the evolutionary algorithm are the same for the whole cross-validation process with the exception of the parameter related to the number of generations,

Table 1 Datasets used for the experiments

Dataset	Size	#Input	#Classes	Distribution	p^*
Two classes					
BreastC-W	699	9	2	(458,241)	0.3448
Card	690	51	2	(307,383)	0.4449
Hepatitis	155	19	2	(32,123)	0.2069
Multi-class					
Balance	625	4	3	(288,49,288)	0.0784
Gene	3175	120	3	(762,765,1648)	0.2400
Iris	150	4	3	(50,50,50)	0.3333
Lymph	148	38	4	(2,81,61,4)	0.0135
Anneal	898	59	5	(8,99,684,67,40)	0.0089
Glass	214	9	6	(70,76,17,13,9,29)	0.0421
Zoo	101	16	7	(41,20,5,13,4,8,10)	0.0396

which is reduced to $1/5$ of the final number of generations because, experimentally, it is not necessary to go further in the number of generations in order to find the best λ. The λ values to be tested are selected from the range [0, 1] in intervals of 0.25. Previous experiments confirmed that there were no significant differences if the cross-validation was performed with more values. So, considering a few λ values and reducing the number of generations in the algorithm, the cross-validation process time is drastically reduced.

3 Experiments

The purpose of the experiments is to evaluate which fitness function is more suitable for E-ELM-CS with the purpose of simultaneous optimization of C and MS. Computational cost, in terms of training time T, is also considered. Results are compared with related state of the art methods.

There were ten UCI repository datasets with different features under study [17] (see Table 1). The experimental 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 generalization set. All the data have been standardized and the experiments have been conducted using Matlab R2009a running on a Ubuntu Server (x86_64 architecture) on a Intel Xeon at 2.00 GHz with 8 Gb RAM.

3.1 Machine Learning Methods Used for Comparison Purposes

The experimental section compares two basically different methodologies with different extensions for training MLP neural networks. The first group of classifiers are variations of the Evolutionary ELM (results are also compared to the original ELM and OPELM [18]):

- EELM. This method is set up with two fitness functions: CCR (EELM(C)) and MS (EELM(S)).
- EELMCS. This algorithm is set up with the three fitness functions proposed in Sect. 2.2: $F_{\lambda CS}$ (EELMCS(CS)), $F_{\lambda R}$ (EELMCS(R)) and $F_{\lambda E}$ (EELMCS(E)).

The second type of neural network training algorithm is the Memetic Pareto Differential Evolution Neural Network (MPDENN) presented in [19]. MPDENN is a Multi-Objective Evolutionary Algorithm (MOEA) based on the Pareto Differential Evolution algorithm (PDE) presented by [20,21]. The MPDENN trains ANNs considering C and MS as conflicting objectives which should be simultaneously optimized. In addition, the MPDENN applies a local search procedure to some individuals in the population. The local search algorithm used is the improved Resilient Backpropagation (iRprop$^+$) algorithm [22]. Moreover, local search can improve classification performance although it is computationally costly. MPDENN can be used without local search; when doing so, we will refer to it to as PDENN.

In addition, the experiments include two additional popular learning algorithms: a standard MLP trained with Resilient backpropagation (Rprop) algorithm [23] and the Support Vector Machine (SVM) [24,25]. The Rprop Matlab's implementation is used for the former algorithm, while Cost Support Vector Classificacion (SVC) available in libSVM 3.0 [26] is used as the SVM classifier implementation.

All the ELM-based methods were trained with different algorithms (ELM, OPELM, EELM and EELMCS) using the Sigmoid function as the basis function. For ELM and OPELM, the number of hidden nodes gradually increases in intervals of 5 within the interval $[5, 200]$ and the nearly optimal number of nodes for ELM and OPELM are then selected based on the 10-fold cross-validation method using the training set. For the E-ELM-CS(R), the range of the interval has been reduced for cross-validation of the number of hidden nodes to $[5, 20]$. PDENN and MPDENN automatically prune nodes, so a range of maximum and minimum numbers of hidden nodes must be provided. We have used the range $[1, 6]$ according to the author's recommendations. Regarding Rprop, a nested cross-validation was also performed using a range of hidden nodes of $[1, 30]$ with an interval of one. The radial basis kernel was used with the SVC method. For the selection of SVC hyperparameters (regularization parameter, C, and width of the Gaussian functions, γ), a grid search was performed with a 10-fold cross-validation, using the following ranges: $C \in \{10^2, 10^1, \ldots, 10^{-1}\}$ and $\gamma \in \{10^2, 10^0, \ldots, 10^{-8}\}$. For all the methods, except PDENN and MPDENN, the optimal hyperparameters $\hat{\boldsymbol{\theta}}$ cross-validation criteria was the following:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}_i} \frac{\overline{C_{\boldsymbol{\theta}_i}} + \overline{MS_{\boldsymbol{\theta}_i}}}{2}. \tag{13}$$

3.2 Statistical Results

Tables 4 and 5 in Appendix A present results in values of the mean and the standard deviation (SD) for $\%C_{\mathrm{G}}$, $\%MS_{\mathrm{G}}$ and T (in seconds) for 30 runs. Subindex G in $\%C_{\mathrm{G}}$ and $\%MS_{\mathrm{G}}$ indicates that results belongs to the generalization dataset. For these tables and Table 2 the best result is in bold face and the second best result is in italics.

The mean rankings of C_{G}, MS_{G} and T are obtained to compare the different methods (see Table 2). A Friedman's non-parametric test for a significance level of $\alpha = 0.1$ has been carried out to determine the statistical significance of the differences in rank in each method. The test rejected the null-hypothesis stating that all algorithms performed equally in the mean ranking of C_{G}, MS_{G} and T so a Nemenyi post-hoc test [27] ($\alpha = 0.1$) was used to compare all the methods and their variations. Figure 2 shows Critical Difference (CD) diagrams proposed in [27]. Figure 2a shows that there are two groups of classifiers regarding C_{G}. SVC has the best C_{G} ranking mean but it does not have significant differences compared with EELM variants but with EELM(MS). Regarding, MS_{G}, all the methods but Rprop, OPELM, PDE(C) and ELM have no significant differences when they are compared to one another.

Table 2 Mean statistical results and average rankings

Method	$\overline{C}_{\mathrm{G}}$ (%)	$\overline{R}_{C_{\mathrm{G}}}$	$\overline{MS}_{\mathrm{G}}$ (%)	$\overline{R}_{MS_{\mathrm{G}}}$	$\overline{T}$ (s)	$\overline{R}_T$
EELMCS(R)	86.19	5.50	**58.86**	**2.45**	1.42E+002	7.00
EELMCS(E)	*86.87*	5.50	56.52	4.95	1.46E+002	8.20
EELMCS(CS)	86.47	6.50	*58.50*	*4.75*	1.39E+002	7.00
EELM(C)	86.78	*5.40*	48.81	6.75	1.41E+002	7.00
EELM(MS)	84.86	8.60	57.54	5.20	1.40E+002	6.80
OPELM	85.71	7.10	42.70	10.35	3.32E+000	3.70
ELM	86.09	7.20	44.33	7.90	**9.12E−002**	*1.70*
PDE(C)	82.92	10.25	41.81	9.45	2.08E+003	10.05
PDE(MS)	82.06	10.55	51.62	7.60	2.08E+003	10.35
HPDE(C)	86.07	6.90	46.91	7.25	1.37E+006	12.25
HPDE(MS)	84.89	8.30	55.60	6.35	1.37E+006	12.35
Rprop	85.06	7.20	38.05	11.20	8.39E−001	3.30
SVC	**88.60**	**2.00**	53.97	6.80	*1.44E−001*	**1.30**

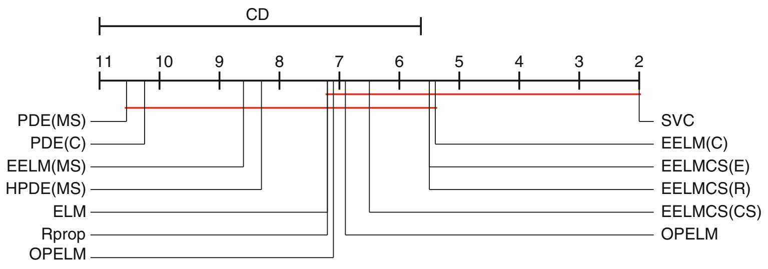
Regarding T, all the non-evolutionary approaches have similar performance, in addition, EELM methods [except EELMCS(E)] show similar computational time. As expected, the Pareto based solutions have the highest computational cost (especially the hybrid approaches).

Based on the robustness regarding C_{G} and MS_{G}, and considering computational cost, EELMCS(R) promises to be the best alternative. However, when comparing them to each other for MS_{G}, EELMCS(R) shows no significant differences with EELM(C), so our proposal cannot be justified. From a statistical point of view, this can be justified considering the effects of the results of competitive methods, that is, the presence of EELM related methods. Therefore, the more powerful Holm post-hoc test is used to compare EELMCS(R) to all the other classifiers in order to justify our proposal. The Holm test is a multiple comparison procedure that can work with a control algorithm and compares it to the remaining methods [27]. The test statistics for comparing the ith and the jth method using this procedure is

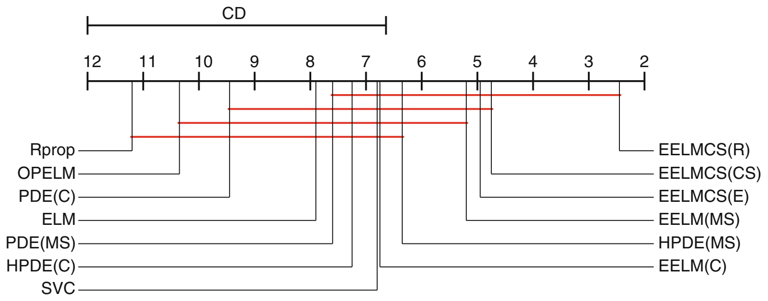
$$z = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6N}}} \tag{14}$$

where k is the number of algorithms and N the number of datasets. The z value is used to find the corresponding probability from the table of normal distribution, which is then compared to an appropriate level of confidence α. Holm's test adjusts the value for α in order to compensate for multiple comparisons.

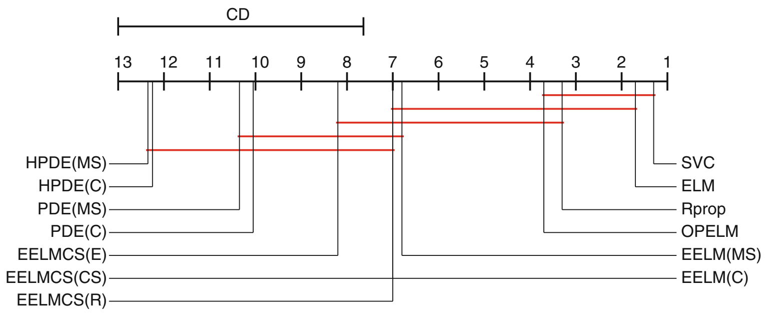
The results of the Holm tests ($\alpha = 0.1$) for MS_{G} can be seen in Table 3, using the corresponding p and adjusted α (α'_{Holm}) values. The EELMCS(R) is used as the Control Method. The horizontal line shows the division between methods significantly different from EELMCS(R) (in terms of MS_{G} when $p < \alpha'_{Holm}$) and methods which are not significantly different. Considering the results of these tests, it can be concluded that the EELMCS(R) algorithm obtains a significantly higher ranking of MS_{G} when compared to most of the remaining methods, especially ELM, OPELM and EELM(C). Standard methods such as Rprop or SVC are not competitive when considering MS.



(a)



(b)



(c)

Fig. 2 Ranking tests for CCR, MS and training time. Nemenyi CD diagrams comparing the generalization (**a**) CCR and (**b**) MS mean results of different methods. **c** Nemenyi CD diagrams comparing the training time mean results of different methods.

4 Conclusions

This paper presents an efficient alternative to the Pareto based approach to train multi-class classifiers with a simultaneous improvement in C and MS.

Three different fitness functions were evaluated by extending the Evolutionary Extreme Learning Machine algorithm for training ANNs and were compared with different machine learning methodologies. Continuous fitness functions have proved to be more robust and suitable for evolutionary algorithms (see results details in Tables 4, 5).

Table 3 Table with the different algorithms compared with the EELMCS(R) (i.e. the control algorithm) using the Holm procedure in terms of MS_{G}

i	Algorithm	z	p	α'_{Holm}
1	Rprop	5.02398	0.00000	0.00833
2	OPELM	4.53594	0.00001	0.00909
3	PDE(C)	4.01918	0.00006	0.01000
4	ELM	3.12922	0.00175	0.01111
5	PDE(MS)	2.95697	0.00311	0.01250
6	HPDE(C)	2.75601	0.00585	0.01429
7	SVC	2.49764	0.01250	0.01667
8	EELM(C)	2.46893	0.01355	0.02000
9	HPDE(MS)	2.23926	0.02514	0.02500
10	EELM(MS)	1.57897	0.11434	0.03333
11	EELMCS(E)	1.43542	0.15117	0.05000
12	EELMCS(CS)	1.32059	0.18664	0.10000

Statistical tests demonstrate that experimentally the E-ELM-CS(R) methodology gets similar results in CCR with respect to the other methods. However, statistical tests for MS demonstrate experimentally that it is significantly different than most algorithms. Considering the methods with the best significant results in CCR and MS, and considering the statistical test for training time T, we conclude that the weighted combination of global RMSE and the worst classified class RMSE give competitive results.

MS results in anneal, balance, glass or hepatitis datasets (see Tables 4, 5) show that our proposal is specially suitable for problems with high number of classes and/or with small size classes, i.e. imbalanced datasets. Experimental results show that some methods focus only on the bigger classes.

Acknowledgments This work has been partially subsidized by the TIN 2008-06681-C06-03 project of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P08-TIC-3745 project of the "Junta de Andalucía" (Spain). The research of Javier Sánchez-Monedero has been funded by the "Junta de Andalucía" Ph. D. Student Program.

Appendix A: Statistical results tables for C, MS and T

See Tables 4 and 5.

Table 4 Statistical results for C, MS and T

Dataset	Algorithm	C	MS	T
Anneal	EELMCS(R)	96.46 ± 01.75	80.43 ± 11.13	1.71E + 002 ± 1.84E + 000
	EELMCS(E)	95.48 ± 01.93	76.76 ± 11.14	1.79E + 002 ± 3.99E + 000
	EELMCS(CS)	*98.07 ± 01.71*	**89.01 ± 11.49**	1.62E + 002 ± 2.59E + 000
	EELM(C)	**99.11 ± 00.90**	59.95 ± 47.29	1.67E + 002 ± 7.36E + 000
	EELM(MS)	96.07 ± 02.65	*83.61 ± 13.90*	1.61E + 002 ± 2.47E + 000

Table 4 continued

Dataset	Algorithm	C	MS	T
	OPELM	95.87 ± 01.21	51.67 ± 06.48	5.97E + 000 ± 8.86E−001
	ELM	96.74 ± 01.01	55.60 ± 12.83	*2.26E−001 ± 9.96E−002*
	PDE(C)	90.24 ± 03.23	34.40 ± 27.22	3.02E + 003 ± 3.92E + 002
	PDE(MS)	86.22 ± 06.42	53.76 ± 15.37	3.02E + 003 ± 3.92E + 002
	HPDE(C)	92.22 ± 02.56	41.40 ± 27.91	3.01E + 003 ± 4.77E + 002
	HPDE(MS)	87.08 ± 06.81	59.14 ± 14.01	3.01E + 003 ± 4.77E + 002
	Rprop	95.82 ± 00.03	19.47 ± 00.37	2.24E + 000 ± 8.14E−001
	SVC	97.78 ± 00.00	50.00 ± 00.00	**4.84E−002 ± 0.00E + 000**
Balance	EELMCS(R)	91.65 ± 00.94	**87.67 ± 06.83**	6.14E + 001 ± 2.41E + 000
	EELMCS(E)	91.86 ± 00.79	*87.42 ± 06.78*	6.38E + 001 ± 2.08E + 000
	EELMCS(CS)	90.92 ± 01.47	83.32 ± 10.85	6.07E + 001 ± 1.29E + 000
	EELM(C)	91.32 ± 01.70	36.33 ± 26.46	6.10E + 001 ± 1.67E + 000
	EELM(MS)	90.49 ± 02.00	81.86 ± 19.57	5.91E + 001 ± 3.10E + 000
	OPELM	91.97 ± 01.61	16.33 ± 18.29	6.89E + 000 ± 1.17E + 000
	ELM	88.55 ± 01.39	06.67 ± 06.06	*3.54E−001 ± 1.07E−001*
	PDE(C)	90.36 ± 01.30	23.33 ± 16.68	1.03E + 002 ± 1.03E + 001
	PDE(MS)	91.05 ± 01.15	85.15 ± 07.68	1.03E + 002 ± 1.03E + 001
	HPDE(C)	91.24 ± 01.23	29.00 ± 11.85	1.20E + 002 ± 1.34E + 001
	HPDE(MS)	91.22 ± 01.49	84.62 ± 06.82	1.20E + 002 ± 1.34E + 001
	Rprop	*92.56 ± 00.04*	17.33 ± 00.27	8.41E−001 ± 2.89E−001
	SVC	**93.59 ± 00.00**	80.00 ± 00.00	**1.19E−002 ± 0.00E + 000**
Breastw	EELMCS(R)	96.30 ± 00.93	**94.46 ± 02.07**	1.77E + 001 ± 3.43E−001
	EELMCS(E)	96.02 ± 00.79	93.24 ± 02.11	1.88E + 001 ± 4.19E−001
	EELMCS(CS)	96.23 ± 00.86	93.54 ± 02.23	1.79E + 001 ± 3.90E−001
	EELM(C)	*96.38 ± 00.77*	*94.42 ± 01.79*	1.72E + 001 ± 4.41E−001
	EELM(MS)	95.70 ± 00.92	91.69 ± 02.52	1.73E + 001 ± 3.60E−001
	OPELM	95.92 ± 00.88	93.76 ± 01.98	1.40E + 000 ± 2.97E−001
	ELM	95.81 ± 00.66	92.06 ± 01.89	*1.96E−002 ± 2.53E−002*
	PDE(C)	95.14 ± 00.99	90.22 ± 02.69	2.47E + 001 ± 3.56E + 000
	PDE(MS)	95.18 ± 00.93	90.33 ± 02.85	2.47E + 001 ± 3.56E + 000
	HPDE(C)	95.03 ± 00.86	89.56 ± 02.19	2.92E + 001 ± 3.77E + 000
	HPDE(MS)	94.93 ± 00.85	89.44 ± 02.16	2.92E + 001 ± 3.77E + 000
	Rprop	96.13 ± 00.01	93.43 ± 00.02	4.78E−001 ± 1.46E−001
	SVC	**96.57 ± 00.00**	91.67 ± 00.00	**7.26E−003 ± 0.00E + 000**
Card	EELMCS(R)	88.07 ± 01.64	**86.09 ± 01.59**	4.95E + 001 ± 3.95E−001
	EELMCS(E)	*88.13 ± 01.55*	*85.95 ± 03.12*	5.22E + 001 ± 3.43E−001
	EELMCS(CS)	87.15 ± 01.94	85.59 ± 02.25	4.85E + 001 ± 1.59E + 000
	EELM(C)	86.92 ± 01.50	84.82 ± 02.50	4.74E + 001 ± 8.91E−001
	EELM(MS)	87.23 ± 01.62	85.02 ± 02.28	4.86E + 001 ± 1.39E + 000
	OPELM	84.84 ± 01.84	82.93 ± 02.72	6.21E + 000 ± 7.78E−001
	ELM	85.53 ± 01.93	84.33 ± 02.03	*5.02E−002 ± 5.34E−002*
	PDE(C)	85.39 ± 02.32	83.01 ± 03.08	2.98E + 001 ± 7.36E + 000

Table 4 continued

Dataset	Algorithm	C	MS	T
	PDE(MS)	85.74 ± 02.06	84.08 ± 02.44	2.98E + 001 ± 7.36E + 000
	HPDE(C)	86.55 ± 01.14	84.85 ± 02.25	5.37E + 001 ± 1.10E + 001
	HPDE(MS)	86.51 ± 01.32	85.30 ± 01.83	5.37E + 001 ± 1.10E + 001
	Rprop	86.19 ± 00.08	82.85 ± 00.16	6.06E−001 ± 1.49E−001
	SVC	**88.44 ± 00.00**	85.42 ± 00.00	**3.31E−002 ± 0.00E + 000**
Gene	EELMCS(R)	*84.73 ± 01.15*	78.96 ± 02.14	9.79E + 002 ± 8.23E + 001
	EELMCS(E)	84.67 ± 01.16	78.20 ± 03.59	9.99E + 002 ± 8.63E + 001
	EELMCS(CS)	83.46 ± 01.30	78.84 ± 03.80	9.42E + 002 ± 8.57E + 001
	EELM(C)	84.50 ± 01.45	78.52 ± 03.34	9.61E + 002 ± 8.95E + 001
	EELM(MS)	83.69 ± 01.65	*79.57 ± 02.60*	9.61E + 002 ± 9.05E + 001
	OPELM	77.99 ± 01.41	62.86 ± 04.43	8.56E + 000 ± 1.06E−001
	ELM	80.50 ± 01.33	69.65 ± 03.23	**2.37E−001 ± 2.09E−002**
	PDE(C)	70.37 ± 04.18	56.74 ± 09.93	1.71E + 004 ± 4.38E + 003
	PDE(MS)	69.99 ± 04.34	65.25 ± 05.83	1.71E + 004 ± 4.38E + 003
	HPDE(C)	82.32 ± 02.83	75.15 ± 05.43	1.37E + 007 ± 2.70E + 006
	HPDE(MS)	82.06 ± 02.76	74.89 ± 05.47	1.37E + 007 ± 2.70E + 006
	Rprop	84.39 ± 00.07	73.52 ± 00.23	2.17E + 000 ± 5.43E−001
	SVC	**90.92 ± 00.00**	**89.32 ± 00.00**	*1.33E + 000 ± 0.00E + 000*

Table 5 Statistical results for C, MS and T

Dataset	Algorithm	C	MS	T
Glass	EELMCS(R)	62.83 ± 07.82	**19.17 ± 16.54**	7.31E + 001 ± 9.53E−001
	EELMCS(E)	69.62 ± 04.16	05.00 ± 12.11	7.52E + 001 ± 1.05E + 000
	EELMCS(CS)	68.55 ± 04.41	*18.61 ± 17.46*	7.87E + 001 ± 1.37E + 000
	EELM(C)	69.69 ± 05.22	07.78 ± 13.83	7.81E + 001 ± 1.64E + 000
	EELM(MS)	66.42 ± 06.08	15.47 ± 17.16	7.74E + 001 ± 1.76E + 000
	OPELM	**71.70 ± 03.43**	02.50 ± 07.63	2.88E + 000 ± 4.94E−001
	ELM	*70.44 ± 04.86*	00.00 ± 00.00	**4.38E−003 ± 1.42E−002**
	PDE(C)	61.89 ± 08.53	01.67 ± 06.34	3.26E + 002 ± 5.04E + 001
	PDE(MS)	57.99 ± 09.11	09.46 ± 15.06	3.26E + 002 ± 5.04E + 001
	HPDE(C)	69.18 ± 05.23	02.50 ± 07.63	3.61E + 002 ± 7.80E + 001
	HPDE(MS)	64.53 ± 07.41	18.43 ± 17.96	3.61E + 002 ± 7.80E + 001
	Rprop	59.69 ± 00.11	00.00 ± 00.00	5.52E−001 ± 1.80E−001
	SVC	64.15 ± 00.00	00.00 ± 00.00	*8.35E−003 ± 0.00E + 000*
Hepatitis	EELMCS(R)	74.27 ± 04.23	42.92 ± 13.80	1.71E + 000 ± 8.55E−002
	EELMCS(E)	76.24 ± 04.26	38.75 ± 14.44	1.70E + 000 ± 9.52E−003
	EELMCS(CS)	*77.18 ± 04.43*	*45.90 ± 12.95*	1.67E + 000 ± 1.06E−002
	EELM(C)	76.41 ± 04.64	35.00 ± 15.54	1.67E + 000 ± 8.34E−003
	EELM(MS)	74.87 ± 04.92	41.25 ± 13.99	1.67E + 000 ± 9.07E−003
	OPELM	76.32 ± 03.79	30.00 ± 14.53	2.27E−001 ± 6.25E−003
	ELM	76.75 ± 03.92	32.50 ± 11.65	**1.53E−003 ± 3.65E−005**
	PDE(C)	74.70 ± 04.24	33.33 ± 13.67	1.14E + 001 ± 1.78E + 000

Table 5 continued

Dataset	Algorithm	C	MS	T
	PDE(MS)	74.70 ± 04.35	35.42 ± 12.32	1.14E + 001 ± 1.78E + 000
	HPDE(C)	75.38 ± 03.34	31.67 ± 12.17	1.40E + 001 ± 3.38E + 000
	HPDE(MS)	75.21 ± 04.11	33.33 ± 12.43	1.40E + 001 ± 3.38E + 000
	Rprop	76.07 ± 00.04	22.08 ± 00.20	2.59E−001 ± 8.00E−002
	SVC	**79.49 ± 00.00**	**50.00 ± 00.00**	*1.58E−003 ± 0.00E + 000*
Iris	EELMCS(R)	96.00 ± 01.48	89.16 ± 03.29	1.76E + 000 ± 2.17E−002
	EELMCS(E)	95.70 ± 01.64	88.65 ± 03.08	1.82E + 000 ± 1.78E−002
	EELMCS(CS)	94.96 ± 01.93	87.56 ± 03.81	1.80E + 000 ± 8.17E−003
	EELM(C)	95.26 ± 02.16	89.09 ± 04.04	1.81E + 000 ± 1.10E−002
	EELM(MS)	94.74 ± 01.89	87.32 ± 03.63	1.80E + 000 ± 1.07E−002
	OPELM	92.89 ± 03.16	86.98 ± 07.09	8.08E−001 ± 2.09E−001
	ELM	95.70 ± 02.18	90.54 ± 05.07	*1.29E−002 ± 1.55E−002*
	PDE(C)	96.58 ± 01.82	90.26 ± 04.01	7.98E + 000 ± 9.75E−001
	PDE(MS)	95.81 ± 01.43	87.69 ± 04.33	7.98E + 000 ± 9.75E−001
	HPDE(C)	*97.09 ± 00.89*	*91.28 ± 02.66*	9.89E + 000 ± 2.43E + 000
	HPDE(MS)	95.73 ± 01.23	87.18 ± 03.69	9.89E + 000 ± 2.43E + 000
	Rprop	90.00 ± 00.13	71.76 ± 00.36	4.20E−001 ± 1.21E−001
	SVC	**97.78 ± 00.00**	**93.33 ± 00.00**	**6.18E−004 ± 0.00E + 000**
Lymph	EELMCS(R)	79.82 ± 03.74	04.83 ± 18.41	2.95E + 001 ± 2.00E−001
	EELMCS(E)	78.02 ± 05.35	02.33 ± 12.78	3.10E + 001 ± 2.06E−001
	EELMCS(CS)	77.75 ± 05.10	02.67 ± 14.61	3.11E + 001 ± 3.93E−001
	EELM(C)	79.01 ± 05.10	00.00 ± 00.00	3.13E + 001 ± 3.64E−001
	EELM(MS)	70.45 ± 06.46	06.28 ± 19.29	3.12E + 001 ± 2.55E−001
	OPELM	*82.43 ± 04.36*	00.00 ± 00.00	1.79E−001 ± 4.19E−003
	ELM	79.28 ± 04.83	07.00 ± 21.36	*4.85E−003 ± 1.22E−004*
	PDE(C)	79.46 ± 04.63	02.33 ± 12.78	9.75E + 001 ± 2.67E + 001
	PDE(MS)	79.28 ± 04.78	02.33 ± 12.78	9.75E + 001 ± 2.67E + 001
	HPDE(C)	80.99 ± 05.74	**14.83 ± 30.44**	1.21E + 002 ± 2.57E + 001
	HPDE(MS)	80.81 ± 05.74	**14.83 ± 30.44**	1.21E + 002 ± 2.57E + 001
	Rprop	80.99 ± 00.10	00.00 ± 00.00	3.21E−001 ± 1.29E−001
	SVC	**83.78 ± 00.00**	00.00 ± 00.00	**2.84E−003 ± 0.00E + 000**
Zoo	EELMCS(R)	91.73 ± 04.06	**08.89 ± 18.69**	3.61E + 001 ± 3.84E−001
	EELMCS(E)	*92.93 ± 04.54*	**08.89 ± 27.59**	3.76E + 001 ± 2.98E−001
	EELMCS(CS)	90.40 ± 04.15	00.00 ± 00.00	4.22E + 001 ± 3.24E−001
	EELM(C)	89.20 ± 03.95	02.22 ± 12.17	4.22E + 001 ± 3.83E−001
	EELM(MS)	88.93 ± 04.42	03.33 ± 18.26	4.23E + 001 ± 4.78E−001
	OPELM	87.20 ± 05.79	00.00 ± 00.00	4.60E−002 ± 7.87E−004
	ELM	91.60 ± 04.74	05.00 ± 20.13	*1.68E−003 ± 1.30E−004*
	PDE(C)	85.07 ± 06.12	02.78 ± 10.80	6.15E + 001 ± 1.68E + 001
	PDE(MS)	84.67 ± 06.31	02.78 ± 10.80	6.15E + 001 ± 1.68E + 001
	HPDE(C)	90.67 ± 05.49	**08.89 ± 27.59**	7.98E + 001 ± 3.89E + 001

Table 5 continued

Dataset	Algorithm	C	MS	T
	HPDE(MS)	90.80 ± 05.37	**08.89 ± 27.59**	7.98E + 001 ± 3.89E + 001
	Rprop	86.93 ± 00.12	00.00 ± 00.00	4.98E−001 ± 2.04E−001
	SVC	**96.00 ± 00.00**	00.00 ± 00.00	**1.50E−003 ± 0.00E + 000**

References

1. Provost F, Fawcett T (1997) Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In: Proceedings of the 3rd international conference on knowledge discovery and data mining, pp 43–48
2. Martínez-Estudillo FJ, Gutiérrez PA, Hervás-Martínez C, Fernández JC (2008) Evolutionary learning by a sensitivity-accuracy approach for multi-class problems. In: Proceedings of the 2008 IEEE congress on evolutionary computation (CEC'08), IEEE Press, Hong Kong, China, pp 1581–1588
3. Fernández-Caballero JC, Martínez-Estudillo FJ, Hervás-Martínez C, Gutiérrez PA (2010) Sensitivity versus accuracy in multiclass problems using memetic pareto evolutionary neural networks. IEEE Trans Neural Netw 21(5):750–770
4. Roth S, Gepperth A, Igel C (2006) Multi-objective neural network optimization for visual object detection. In: Jin Y (ed) Multi-objective machine learning, studies in computational intelligence, vol 16. Springer, Berlin, pp 629–655
5. Coello CA (2000) An updated survey of ga-based multiobjective optimization techniques. ACM Comput Surv 32(2):109–143
6. Zhu QY, Qin A, Suganthan P, Huang GB (2005) Evolutionary extreme learning machine. Pattern Recogn 38(10):1759–1763
7. Sánchez-Monedero J, Hervás-Martínez C, Martínez-Estudillo F, Ruz M, Moreno M, Cruz-Ramírez M (2010) Evolutionary learning using a sensitivity-accuracy approach for classification. In: Hybrid artificial intelligence systems, Lecture notes in computer science, vol 6077. Springer, Berlin, pp 288–295
8. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1-3):489–501
9. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
10. Chen L, Zhou L, Pung H (2008) Universal approximation and qos violation application of extreme learning machine. Neural Process Lett 28:81–95
11. Zhang R, Huang GB, Sundararajan N, Saratchandran P (2007) Multi-category classification using an extreme learning machine for microarray gene expression cancer diagnosis. IEEE/ACM Trans Computat Biol Bioinform 4(3):485–495
12. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
13. Ilonen J, Kamarainen JK, Lampinen J (2003) Differential evolution training algorithm for feed-forward neural networks. Neural Process Lett 17(1):93–105
14. Bishop CM (1996) Neural networks for pattern recognition, 1st edn. Oxford University Press, Oxford
15. Bishop CM (2007) Pattern recognition and machine learning, 1st edn. 2006. corr. 2nd printing edn. Springer, New York
16. Bridle JS (1990) Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In: Fogelman Soulie F, Herault J (eds) Neurocomputing: algorithms, architectures and applications. Springer-Verlag, New York, pp 227–236
17. Asuncion A, Newman D (2007) UCI machine learning repository. http://www.ics.uci.edu~mlearn/MLRepository.html
18. Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A (2010) OP-ELM: optimally pruned extreme learning machine. IEEE Trans Neural Netw 21(1):158–162
19. Cruz-Ramírez M, Sánchez-Monedero J, Fernández-Navarro F, Fernández J, Hervás-Martínez C (2010) Memetic pareto differential evolutionary artificial neural networks to determine growth multi-classes in predictive microbiology. Evolutionary intelligence, pp 1–13

20. Abbass HA, Sarker R, Newton C (2001) PDE: a pareto-frontier differential evolution approach formulti-objective optimization problems. In: Proceedings of the 2001 congress on evolutionary computation, vol 2. Seoul, South Korea
21. Abbass HA (2001) A memetic pareto evolutionary approach to artificial neural networks. In: Brooks M, Corbet D, Stumptner M (eds) AI2001, LNAI 2256. Springer, Berlin, pp 1–12
22. Igel C, Hsken M (2003) Empirical evaluation of the improved rprop learning algorithms. Neurocomputing 50(6):105–123
23. Riedmiller M., Braun H. (1993) A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: Proceedings of the 1993 IEEE International conference on neural networks, San Francisco, pp 586–591
24. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
25. Vapnik V (1999) An overview of statistical learning theory. IEEE Trans Neural Netw 10(5):988–999
26. Chang C.C., Lin C.J. (2001) LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
27. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res7:1–30