Evolutionary Learning Using a Sensitivity-Accuracy Approach for Classification

Javier Sánchez-Monedero[1,⋆], C. Hervás-Martínez[1], F.J. Martínez-Estudillo[2], Mariano Carbonero Ruz[2], M.C. Ramírez Moreno[2], and M. Cruz-Ramírez[1]

[1] Department of Computer Science and Numerical Analysis, University of Córdoba, Spain
[2] Department of Management and Quantitative Methods, ETEA, Spain

Abstract. Accuracy alone is insufficient to evaluate the performance of a classifier especially when the number of classes increases. This paper proposes an approach to deal with multi-class problems based on Accuracy (C) and Sensitivity (S). We use the differential evolution algorithm and the ELM-algorithm (Extreme Learning Machine) to obtain multi-classifiers with a high classification rate level in the global dataset with an acceptable level of accuracy for each class. This methodology is applied to solve four benchmark classification problems and obtains promising results.

1 Introduction

To evaluate a classifier, the machine learning community has traditionally used the correct classification rate or accuracy to measure its default performance. In the same way, accuracy has been frequently used as the fitness function in evolutionary algorithms when solving classification problems. However, the pitfalls of using accuracy have been pointed out by several authors [1]. Actually, it is enough to simply realize that accuracy cannot capture all the different behavioural aspects found in two different classifiers.

Assuming that all misclassifications are equally costly and that there is no penalty for a correct classification, we start from the premise that a good classifier should combine a high classification rate level in the testing set with an acceptable level for each class. Concretely, we consider traditionally used accuracy (C) and the minimum of the sensitivities of all classes (S), that is, the lowest percentage of examples correctly predicted as belonging to each class with respect to the total number of examples in the corresponding class.

Recently, in [2], Huang et al. proposed an original algorithm called extreme learning machine (ELM) which randomly chooses hidden nodes and analytically determines (by using Moore-Penrose generalized inverse) the output weights of the network. The algorithm tends to provide good testing performance at

⋆ Corresponding author at: E-mail address: i02samoj@uco.es Phone: +34-957218349.
This work has been partially subsidized by TIN 2008-06681-C06-03 (MICYT), FEDER funds and the P08-TIC-3745 project of the "Junta de Andalucía" (Spain).

E.S. Corchado Rodriguez et al. (Eds.): HAIS 2010, Part II, LNAI 6077, pp. 288–295, 2010.

an extremely fast learning speed. However, ELM may need a higher number of hidden nodes due to the random determination of the input weights and hidden biases. In [3], a hybrid algorithm called Evolutionary ELM (E-ELM) was proposed by using the differential evolution algorithm [4]. The experimental results obtained show that this approach reduces the number of hidden nodes and obtains more compact networks.

In this paper, the simultaneous optimization of accuracy and sensitivity is carried out by means of the E-ELM algorithm combination. The key point of the algorithm is the fitness function considered, as the convex linear combination of accuracy and sensitivity, which tries to achieve a good balance between the classification rate level in the global dataset and an acceptable level for each class. The base classifier considered is the standard multilayer perceptron (MLP) neural network. The paper is structured as follows. First, we present our approach based on the sensitivity versus accuracy pair (S, C). The third section contains the evolutionary approach. Finally, the paper concludes with an analysis of the results obtained in four benchmark classification problems.

2 Accuracy and Sensitivity

We consider a classification problem with Q classes and N training or testing 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.

Let us denote the number of patterns associated with class i by $f_i = \sum_{j=1}^{Q} n_{ij}$, $i = 1, \ldots, Q$. We start by defining two scalar measures that take the elements of the confusion matrix into consideration from different points of view. Let $S_i = n_{ii}/f_i$ be the number of patterns correctly predicted to be in class i with respect to the total number of patterns in i (sensitivity for class i). Therefore, the sensitivity for class i estimates the probability of correctly predicting a class i example. From the above quantities we define the sensitivity S of the classifier as the minimum value of the sensitivities for each class, $S = \min\{S_i;\ i = 1, \ldots, Q\}$. We define the Correct Classification Rate or Accuracy, $C = (1/N) \sum_{j=1}^{Q} n_{jj}$, which is the rate of all the correct predictions.

Specifically, we consider the two-dimensional measure (S, C) associated with classifier g. The measure tries to evaluate two features of a classifier: global performance and the performance in each class. We represent S on the horizontal axis and C on the vertical axis. One point in (S, C) space dominates another if it is above and to the right, i.e. it has more accuracy and greater sensitivity. straightforward It is straightforward to prove the following relationship between C and S (see [5]). Let us consider a Q-class classification problem. Let C and S be respectively the accuracy and sensitivity associated with a classifier g, then $S \leq C \leq 1 - (1 - S)\, p^*$, where $p^* = f_Q/N$ is the minimum of the estimated prior probabilities.

Therefore, each classifier will be represented as a point outside the shaded region in Fig. 2 (Fig. 2 is built from experimental data, see Section 4.2). Several points in (S, C) space are important to note. The lower left point $(0, 0)$ represents the worst classifier and the optimum classifier is located at the $(1, 1)$ point. Furthermore, the points on the vertical axis correspond to classifiers that are not able to predict any point in a concrete class correctly. Note that it is possible to find among them classifiers with a high level of C, particularly in problems with small p^* [6].

Our objective is to build an evolutionary algorithm that tries to move the classifier population towards the optimum classifier located in the $(1, 1)$ point in the (S, C) space. We think an evolutionary algorithm could be an adequate scheme allowing us to improve the quality of the classifiers, measured in terms of C and S, directing the solutions towards the $(1, 1)$ point.

3 The Proposed Method

3.1 Differential Evolution and Extreme Learning Machine

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 $f = (f_1, f_2, \ldots, f_Q)$:

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

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J)^T$ is the transpose matrix containing all the neural net weights, $\boldsymbol{\theta}_l = (\beta_0^l, \beta_1^l, \ldots, \beta_M^l, \mathbf{w}_1, \ldots, \mathbf{w}_M)$ is the vector of weights of the l 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, $\mathbf{x}$ is the input pattern and $\sigma_j(\mathbf{x}, \mathbf{w}_j)$ the sigmoidal function.

Suppose we are training a MLP 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.

The ELM algorithm randomly selects the $\mathbf{w}_j = (w_{1j}, \ldots, w_{Kj})$ weights and biases for hidden nodes, and analytically determines the output weights $\beta_0^l, \beta_1^l, \ldots, \beta_M^l$ 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) [3] improves the original ELM by using a Differential Evolution (DE) algorithm. Differential Evolution was proposed by Storn and Price [4] and it is known as one of the most efficient evolutionary algorithms.

Require: P (Training Patterns), T (Training Tags)
1: Create a random initial population $\boldsymbol{\theta} = [\mathbf{w}_1, \ldots, \mathbf{w}_k, b_1, \ldots, b_k]$ of size N
2: **for** each individual **do**
3: $\hat{\boldsymbol{\beta}} = ELM\_output(\mathbf{w}, P, T)$ {Calculate output weights}
4: $\phi_\lambda = Fitness(\mathbf{w}, \hat{\boldsymbol{\beta}}$ {Evaluate individual}
5: **end for**
6: Select best individual of initial population
7: **while** Stop condition is not met **do**
8: Mutate random individuals and apply crossover
9: **for** each individual in the new population **do**
10: $\hat{\boldsymbol{\beta}} = ELM\_output(\mathbf{w}, P, T)$ {Calculate output weights}
11: $\phi_\lambda = Fitness(\mathbf{w}, \hat{\boldsymbol{\beta}})$ {Evaluate model}
12: Select new individuals for replacing individuals in old population
13: **end for**
14: Select the best model in the generation
15: **end while**
16: **function** $\hat{\boldsymbol{\beta}} = ELM\_output(\mathbf{w}, P, T)$
Calculate the hidden layer output matrix H
Calculate the output weight $\hat{\boldsymbol{\beta}} = \mathbf{H}^\dagger \mathbf{Y}$
17: **function** $\phi_\lambda = Fitness(\mathbf{w}, \hat{\boldsymbol{\beta}}, \lambda, P, T)$
Build training confusion matrix $\mathbf{M}$
Calculate C and S from $\mathbf{M}$
Get classifier fitness with (1)

Fig. 1. E-ELM-CS algorithm pseudocode

3.2 The E-ELM-CS Algorithm

As mentioned in Section 2, our approach tries to build classifiers with C and S simultaneously optimized. These objectives are not always cooperative. This fact justifies the use a multi-objective approach for the evolutionary algorithm [6].

To obtain the maximization of objectives C and S we use a linear combination of these objectives. This option is a good method when there are two objectives and when the first Pareto front has a very small number of models, in some cases only one (see results from MPANN methodology in Balance and Newthyroid datasets in Table 2). 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 schedule evaluation of a resource scheduler or design multiplierless IIR filters.

We consider the fitness function defined by $\phi_\lambda = (1-\lambda)C + \lambda S$, where λ is a user parameter in $[0, 1]$. This function evaluates the performance of a classifier depending on a weighted Accuracy level and a weighted Sensitivity.

Our proposed method is implemented by using the Evolutionary ELM (E-ELM)[3]. E-ELM for classification problems only considers the misclassification rate of the classifier. We have extended the E-ELM to consider both C and S (E-ELM-CS, Evolutionary ELM considering C and S). Since E-ELM considers

Table 1. Datasets used for the experiments

Dataset	Size	#Input	#Classes	Distribution	p^*
BreastC	286	15	2	(201,85)	0.2957
BreastCW	699	9	2	(458,241)	0.3428
Balance	625	4	3	(288,49,288)	0.0641
Newthyroid	215	5	3	(150,35,30)	0.1296

an error measure as the fitness which should be minimized, we reformulate our fitness function as:

$$\phi_\lambda = \frac{1}{(1-\lambda)C + \lambda S} \tag{1}$$

The E-ELM-CS algorithm pseudocode is shown in Fig.1. Mutation, crossover and selection operations work as described in [3].

4 Experiments

We consider four datasets with different features taken from the UCI repository (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 test set.

4.1 Comparison Procedure

E-ELM-CS is compared to two popular classification algorithms using ANNs:

1. MPANN (Memetic Pareto Artificial Neural Networks) [7]. MPANN is a Multi-objective Evolutionary Algorithm based on Differential Evolution [8] with two objectives; one is to minimize the mean squared error (MSE) and the other is to minimize ANN complexity (the number of hidden units). We have implemented a Java version using the pseudocode shown in [7] and the framework for evolutionary computation JCLEC [1]. Thus, the methodology is named MPANN-MSE when the extreme the Pareto front chosen, that is provided by the algorithm, has better MSE; or is called MPANN-HN if the extreme that is chosen provided by the algorithm has better complexity value.
2. TRAINDIFFEVOL (Differential evolution training algorithm for Neural-Networks) [8]. TRAINDIFFEVOL is an algorithm to train feed forward multilayer perceptron neural networks based on Differential Evolution [4]. This algorithm uses the MSE and mean squared weights and biases for training the networks. To obtain the sensitivity for each class, a modification of the source code provided by the author[2] has been implemented.

[1] `http://jclec.sourceforge.net/`
[2] `http://www.it.lut.fi/project/nngenetic/`

Table 2. Statistical results for E-ELM-CS, E-ELM, TRAINDIFFEVOL , MPANN-MSE and MPANN-HN

Dataset	Algorithm	C(%) Mean±SD	S(%) Mean±SD	Means Ranking of the C	Means Ranking of the S
BreastC	E-ELM-CS$_{\lambda=0.4}$	**68.97±3.19**	**33.97±6.82**	$\boldsymbol{\mu}_{\text{ELMCS}} \geq$ $\mu_{\text{TDIF}} \geq \mu_{\text{ELM}} >$ $\mu_{\text{MPANHN}} \geq$ μ_{MPAN}	$\boldsymbol{\mu}_{\text{ELMCS}} \geq$ $\mu_{\text{MPANHN}} \geq$ $\mu_{\text{MPAN}} \geq \mu_{\text{TDIF}} >$ $\mu_{\text{ELM}}; \boldsymbol{\mu}_{\text{ELMCS}} >$ μ_{MPANHN}, (○) (T-test)
	E-ELM	68.36±1.98	23.33±6.42		
	TDIF	*68.92±2.89*	26.35±11.71		
	MPANN-MSE	66.53±3.07	*28.73±14.23*		
	MPANN-HN	66.53±3.07	28.41±14.34		
BreastCW	E-ELM-CS$_{\lambda=0.4}$	**96.32±0.86**	**93.87±2.28**	$\boldsymbol{\mu}_{\text{ELMCS}} \geq$ $\mu_{\text{MPANHN}} \geq$ $\mu_{\text{MPAN}} \geq \mu_{\text{ELM}} >$ μ_{TDIF}	$\boldsymbol{\mu}_{\text{ELMCS}} \geq$ $\mu_{\text{MPANHN}} \geq$ $\mu_{\text{MPAN}} \geq \mu_{\text{ELM}} >$ μ_{TDIF}
	E-ELM	95.68±1.19	92.61±3.21		
	TDIF	93.98±1.75	86.22±4.69		
	MPANN-MSE	96.04±1.08	92.75±3.40		
	MPANN-HN	*96.27±1.00*	*93.30±3.36*		
Balance	E-ELM-CS$_{\lambda=0.7}$	91.48±1.50	**86.74±10.01**	$\boldsymbol{\mu}_{\text{MPANHN}} \geq$ μ_{ELMCS}, (*) MW test	$\boldsymbol{\mu}_{\text{ELMCS}} >$ μ_{MPANHN}, (*) MW test
	E-ELM	90.56±1.38	14.00±17.73		
	TDIF	87.12±2.56	2.00±6.10		
	MPANN-MSE	**92.94±1.81**	*60.00±14.14*		
	MPANN-HN	**92.94±1.81**	*60.00±14.14*		
Newthy	E-ELM-CS$_{\lambda=0.9}$	**96.23±2.31**	**80.85±11.88**	$\boldsymbol{\mu}_{\text{ELMCS}} \geq$ $\mu_{\text{MPANHN}} \geq$ $\mu_{MPAN} \geq \mu_{\text{ELM}} >$ $\mu_{\text{TDIF}}; \boldsymbol{\mu}_{\text{ELMCS}} >$ μ_{MPANHN}, (○)(T-test)	$\boldsymbol{\mu}_{\text{ELMCS}} \geq$ μ_{MPANHN}, MW test $\boldsymbol{\mu}_{\text{ELMCS}} \geq$ μ_{MPANHN}, MW test
	E-ELM	94.26±2.35	75.77±10.16		
	TDIF	91.11±4.77	59.47±22.74		
	MPANN-MSE	*94.87±3.82*	*72.11±22.29*		
	MPANN-HN	*94.87±3.82*	*72.11±22.29*		
(*) (○) The average difference is significant with p-values= 0.05 or 0.10					

From a statistical point of view, these comparisons are possible because we use the same partitions of the datasets. If not, it would be difficult to justify the equity of the comparison procedure. Regarding the settings of each algorithm that has been compared to E-ELM-CS, we have used the algorithm values advised by the authors in their respective studies.

4.2 Experimental Results

In Table 2 we present the values of the mean and the standard deviation (SD) for C and S for 30 runs obtained in generalization for the best models in each run over the generalization set in each dataset. In E-ELM-CS, the λ parameter is a user parameter. The λ parameter has been obtained for each dataset as the best result of a preliminary experimental design with $\lambda \in [0.0, 0.1, \ldots, 1.0]$.

If we analyze the results for C in the generalization set, we can observe that the E-ELM-CS methodology obtains results that are, in mean, better than or similar to the results of the second best methodology (MPANN-HN in three data sets, TRAINDIFFEVOL in two data sets or E-ELM in one data set). On the other hand, the results in mean of S show that the E-ELM-CS methodology obtains a performance that is better than the second best methodology (MPANN-HN in four data sets and TRAINDIFFEVOL or E-ELM in one data set respectively).

In order to determine the best methodology for training MLP neural networks (in the sense of its influence on C and S in the test dataset), an ANalysis Of the VAriance of one factor (ANOVA I) statistical method or the non parametric

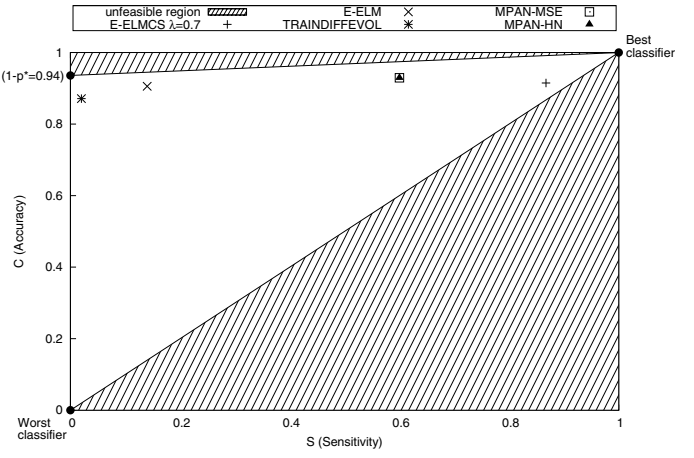

Fig. 2. Comparison of E-ELM-CS, E-ELM, TDIF, MPAN-MSE and MPAN-HN methods for Balance database

Kruskal-Wallis (KW) tests were chosen depending on the satisfaction of the normality hypothesis of C and S values. The factor methodologies analyze the effect on the C (or S) of each methodology applied with levels $i = 1 \dots 5$ to E-ELM-CS (ELMCS), E-ELM (ELM), TRAINDIFFEVOL (TDIF), MPANN-MSE (MPAN) and MPANN-HN (MPANHN). The results of the ANOVA or KW analysis for C and S show that, for the four data sets, the effect of the methodologies is statistically significant at a level of 5%.

Because there is a significant difference in mean for C and S using the Snedecor's F or the KW test; for the former, under the normality hypothesis, a post hoc multiple comparison test is performed of the mean C and S obtained with the different levels of the factor. We perform a Tukey test [20] under normality, and a pair-wise T-test, or a pair-wise Mann-Whitney test in other cases. Table 2 shows the results obtained. Columns 5 and 6 present the post hoc Tukey test; and T-test or Mann-Whitney (M-W) tests. The mean difference is significant with p-values= 0.05 (*) or 0.10 ($\circ$). The $\mu_A \geq \mu_B$: methodology A yields better results than methodology B, but the difference is not significant; $\mu_A > \mu_B$: methodology A yields better results than methodology B with significant differences. The binary relation $\geq$ is not transitive.

Observe that there is a relationship between the imbalanced degree of the dataset and the results obtained by the E-ELM-CS algorithm. It is worthwhile to point out that, for imbalanced datasets, E-ELM-CS gets the best performance results and the highest differences in S when comparing the algorithms (see Balance and Newthyroid results in Table 2). On the other hand, even for two class problems we can observe the same behaviour (compare the S results of BreastCW and BreastC datasets). Finally, our approach improves Sensitivity levels with respect to the original Evolutionary Extreme Learning Machine (E-ELM), while maintaining Accuracy at the same level.

Fig. 2 depicts the sensitivity-accuracy results of the four methodologies for the Balance dataset in the (S, C) space. A visual inspection of the figure allows us to easily observe the difference in the performance of E-ELM-CS with respect to E-ELM, TDIF and MPANN.

5 Conclusions

This work proposes a new approach to deal with multi-class classification problems. Assuming that a good classifier should combine a high classification rate level in the global dataset with an acceptable level for each class, we consider traditionally used Accuracy C and the minimum of the sensitivities of all classes, S. The differential evolution algorithm and the fast ELM algorithm are used to optimize both measures in a multi-objective optimization approach, by using a fitness function built as a convex linear combination of S and C. The procedure obtains multi-classifiers with a high classification rate level in the global dataset with a good level of accuracy for each class.

In our opinion, the (S, C) approach reveals an interesting point of view for dealing with multi-class classification problems since it improves sensitivity levels with respect to the Evolutionary Extreme Learning Machine, while maintaining accuracy at similar levels.

References

1. Provost, F., Fawcett, T.: Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In: Proc. of the 3rd International Conference on Knowledge Discovery and Data Mining, pp. 43–48 (1997)
2. Huang, G.B., Chen, L., Siew, C.K.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Transactions on Neural Networks 17(4), 879–892 (2006)
3. Zhu, Q.Y., Qin, A., Suganthan, P., Huang, G.B.: Evolutionary extreme learning machine. Pattern Recognition 38(10), 1759–1763 (2005)
4. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. of Global Opt. 11(4), 341–359 (1997)
5. Martínez-Estudillo, F., Gutiérrez, P., Hervás-Martínez, C., Fernández, J.: Evolutionary learning by a sensitivity-accuracy approach for multi-class problems. Accepted in the Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008), Hong Kong, China (2008)
6. Fernández, J., Martínez-Estudillo, F., Hervás, C., Gutiérrez, P.: Sensitivity versus accuracy in multi-class problems using memetic pareto evolutionary neural networks. IEEE Transacctions on Neural Networks (accepted, 2010)
7. Abbass, H.A.: Speeding up backpropagation using multiobjective evolutionary algorithms. Neural Computation 15(11), 2705–2726 (2003)
8. Ilonen, J., Kamarainen, J.K., Lampinen, J.: Differential evolution training algorithm for feed-forward neural networks. Neural Process. Lett. 17(1), 93–105 (2003)