A Hybrid Evolutionary Approach to Obtain Better Quality Classifiers

David Becerra-Alonso, Mariano Carbonero-Ruz,
Francisco José Martínez-Estudillo, and Alfonso Carlos Martínez-Estudillo

Department of Management and Quantitative Methods
ETEA - University of Córdoba
{dbecerra, mariano, fjmestud, acme}@etea.com

Abstract. We present an extra measurement for classifiers, responding to the need to evaluate them with more than accuracy alone. This measure should be able to express, at least to some degree, the extent to which all classes are taken into account in a classification problem. In this communication we propose sensitivity dispersion (being as it is, the associated statistical dispersion measurement of accuracy), as the appropriate measure to have a more complete evaluation of the quality of classifiers. We use the Evolutionary Extreme Learning Machine algorithm, with a specific fitness function to optimize both measures simultaneously, and we compare it with other classifiers.[1]

1 Introduction

Accuracy has generally been used to study the performance of evolutionary classifiers. Many works extend on the idea that accuracy, although a critical piece of information, does not capture all the different aspects of how classification takes place for a given method and a given set. This becomes even more relevant when some of the classes of the dataset are shown to be predominant over others, and the system is unbalanced. The search for combined measurements as a way to evaluate a classifier is already found in a number of machine learning publications [2, 7, 9]. Here, we propose a new measure that we call sensitivity dispersion given by the dispersion between the accuracy results in each class.

In 2005, Huang et al. [5, 6] proposed the original algorithm called Extreme Learning Machine (ELM) which randomly chooses hidden nodes and analytically determines the output weights of the network. A hybrid algorithm called Evolutionary ELM (E-ELM) [11] was proposed by using the Differential Evolution algorithm [8]. In this paper, the simultaneous optimization of accuracy and sensitivity dispersion is carried out by means of the E-ELM algorithm combination. The key point of the algorithm is the fitness function considered, which tries to achieve a good balance between the classification rate level in the global

[1] This work was supported in part by the Spanish Inter-Ministerial Commission of Science and Technology under Project TIN 2008-06681-C06-03, the European Regional Development fund, and the Junta de Andaluca, Spain, under Project P08-TIC-3745.

J. Cabestany, I. Rojas, and G. Joya (Eds.): IWANN 2011, Part II, LNCS 6692, pp. 161–168, 2011.

dataset, and an acceptable level for each class. The base classifier considered is the Multilayer Perceptron (MLP) neural network.

The paper is structured as follows: Section 2 presents the two measurements we propose and the relationship between them. In Section 3 we propose the hybrid algorithm based on the Evolutionary Extreme Learning Machine with the specific fitness function built for the optimization of accuracy and sensitivity dispersion. Section 4 presents the experimental results, where 9 methods are tested for 6 datasets. Section 5 ends the paper with conclusions.

2 C vs. S^2

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 $M(g) = \{n_{ij}; \sum_{i,j=1}^{Q} n_{ij} = N\}$ 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. The accuracy C can be obtained as a weighted average of the classification rate of each class:

$$C = \sum_{Q} f_i p_i \tag{1}$$

where p_i represents the relative frequency of patterns that belong to a given class, while f_i is the rate of those correctly classified in that i class. The sensitivity dispersion (i.e. the associated statistical dispersion measurement of accuracy) is given by:

$$S^2 = \sum f_i^2 p_i - C^2 \tag{2}$$

There is also a way to relate C with S^2 in terms of how large must the latter be for every value of the former. Given our definition of S^2 as in (2), and since $\sum_i f_i^2 p_i \leq \sum_i f_i p_i = C$, we have the boundary:

$$S^2(C) \leq C - C^2 \tag{3}$$

This of course is just a tentative upper boundary for S^2. Each classifier will be represented as a point in the (C,S^2) region (see Figure 1). The worst classifier returns a confusion matrix that has $C = 0$ and $S^2 = 0$, while the best classifier returns $C = 1$, $S^2 = 0$.

Although S^2 takes on its full meaning on multi-class scenarios, it is illustrative to present its properties and boundaries in the $Q = 2$ case. For a two classes dataset we have:

$$C = f_1 p_1 + f_2 p_2 \tag{4}$$

$$S^2 = f_1^2 p_1 + f_2^2 p_2 - C^2 \tag{5}$$

Since $p_1 + p_2 = 1$, both measures can be expressed in terms of p_1, that we will call p from this point:

$$C = p f_1 + (1 - p) f_2 \tag{6}$$

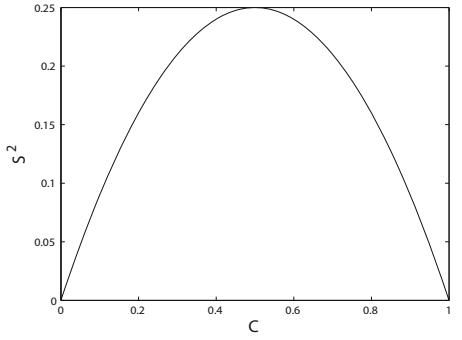

Fig. 1. Top boundary for the C vs. S^2 representation

$$S^2 = p(1-p)(f_1 - f_2)^2 \tag{7}$$

All classifiers with optimal S^2 meet on $f_1 = f_2$. Optimal S^2 is found when the relative correctly classified patterns are equal in both classes.

3 Differential Evolution and Extreme Learning Machine Algorithm

3.1 Extreme Learning Machine (ELM)

Given a dataset with Q classes, we have N patterns $D = \{(\mathbf{x}_j, \mathbf{y}_j) : \mathbf{x}_j \in R^K, \mathbf{y}_j \in R^Q, j = 1, 2, ..., N\}$ where $\mathbf{x}_j$ is a $k \times 1$ input vector and $\mathbf{y}_j$ is $Q \times 1$ target vector. Our Multilayer Perceptron system has M nodes in the hidden layer, and is given by $f = (f_1, f_2, ..., f_Q)$, where every f transforms the input $\mathbf{x}$ according to:

$$f_j(\mathbf{x}, \boldsymbol{\theta}_l) = \beta_0^l + \sum_{j=1}^{M} \beta_j^l \sigma_j(\mathbf{x}, \mathbf{w}_j), l = 1, 2, ..., Q \tag{8}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_J)^T$ is the transposed matrix that includes all the neural network weights. Thus, $\boldsymbol{\theta}_l = (\beta_0^l, \beta_1^l, ..., \beta_M^l, \mathbf{w}_1, ..., \mathbf{w}_M)$ includes the M weights from the hidden layer to the output node, and the vectors $\mathbf{w}_j = (w_{1j}, ..., w_{Kj})$ for the weights and biases between the input and the hidden layer. Each element of $\mathbf{x}$ represents a pattern of the dataset, and the activation function (sigmoidal for our case) is given by $\sigma_j(\mathbf{x}, \mathbf{w}_j)$.

Suppose we want to train a Multilayer Perceptron to learn from N patterns of a set D as proposed above. The relation $f(\mathbf{x}_j) = \mathbf{y}_j$ where $j = 1, ..., N$ can also be expressed as $\mathbf{H}\boldsymbol{\beta} = \mathbf{Y}$, where $\mathbf{H}$ represents the hidden layer output matrix of the network.

The ELM algorithm, as it is proposed in [5, 6] assigns random values to $\mathbf{w}_j = (w_{ij}, ..., w_{Kj})$. By doing so, they convert the nonlinear system into a linear one. Thus they can use these weights to analytically obtain $\{\beta_1^l, ..., \beta_M^l\}$

by finding the least square solution to the given linear system. This can be done by solving $\hat{\boldsymbol{\theta}} = \mathbf{H}^{\dagger}\mathbf{Y}$, where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of $\mathbf{H}$.

3.2 Differential Evolution Algorithm

The Evolutionary Extreme Learning Machine (E-ELM) takes on what has been proposed in Section 3.1 and tries to improve the classifier by applying Differential Evolution (DE) to $\mathbf{w}_j$ (as proposed by Storn and Price [8, 10], and later by Zhu [11]) . This provides an opportunity to use a fitness function that involves C and S^2. DE requires a population of classifiers to compete and combine their weight chromosomes. The following is the basic strategy used for DE:

Given our $\mathbf{w}_j$ in generation G (we will define the whole as $\mathbf{w}_{j,G}$), we first apply mutation: for each target vector $\mathbf{w}_{j,G+1}$ we have a mutant vector according to:

$$\mathbf{m}_{j,G+1} = \mathbf{w}_{r_1,G} + F(\mathbf{w}_{r_2,G} - \mathbf{w}_{r_3,G}) \tag{9}$$

where r_1, r_2, r_3 are random mutually different indices, and $F \in [0, 2]$ defines the amplification of the differential variation $(\mathbf{w}_{r_2,G} - \mathbf{w}_{r_3,G})$. We then apply crossover: our population of chromosomes $\mathbf{w}_{j,G}$ has P members. For generation $G+1$, we can again define a trial vector as $\nu_{kj,G+1} = (\nu_{1j,G+1}, \nu_{2j,G+1}, ..., \nu_{Pj,G+1})$. Thus,

$$\nu_{kj,G+1} = \begin{cases} \mathbf{m}_{kj,G+1} \text{ if } rand_k \leq CR \text{ or } k = rnbr_j \\ \mathbf{w}_{kj,G+1} \text{ if } rand_k > CR \text{ and } k \neq rnbr_j \end{cases}$$

where $rand_k$ is the kth evaluation of a uniform random number generator in $[0, 1]$, $rnbr_j$ is a random chosen integer from 1 to P, and CR is the crossover constant. Finally, we apply selection: $\nu_{kj,G+1}$ will be used in generation $G + 1$, as long as our fitness function returns a better outcome than that obtained from $\mathbf{w}_{kj,G}$. Otherwise, we will have $\mathbf{w}_{kj,G} = \mathbf{w}_{kj,G+1}$. Thus, a fitness function for our particular problem needs to be defined. For each individual (a set of weights and biases), ELM returns the confusion matrix, and from it we can obtain its C and S^2. Let us define a classifier that returns both C_0 and S_0^2 in generation G, whereas in generation $G + 1$ (once the classifier has undergone crossover and

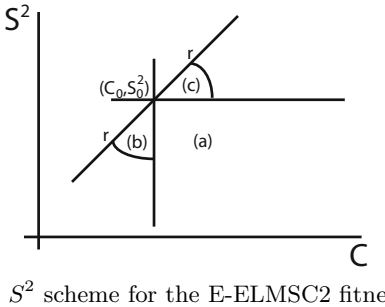

Fig. 2. C vs. S^2 scheme for the E-ELMSC2 fitness conditions

mutation) the condition for the offspring (that returns C_1 and S_1^2) to prevail over the parent is one of the following:

$$(a) \quad C_1 \geq C_0 \quad \texttt{and} \quad S_1^2 \leq S_0^2 \tag{10}$$

or

$$(b) \quad C_1 \leq C_0 \quad \texttt{and} \quad S_1^2 \leq S_0^2 \quad \texttt{and} \quad r \leq C_0/10 \tag{11}$$

when $(C_1 - C_0)^2 + (S_1^2 - S_0^2)^2 \leq r$ and $S_1^2 \leq C_1 - C_0 + S_0^2$, or

$$(c) \quad C_1 \geq C_0 \quad \texttt{and} \quad S_1^2 \geq S_0^2 \quad \texttt{and} \quad r \leq S_0^2/10 \tag{12}$$

when $(C_1 - C_0)^2 + (S_1^2 - S_0^2)^2 \leq r$ and $S_1^2 \leq C_1 - C_0 + S_0^2$.
These fitness conditions allow for small setbacks in either C or S^2 as long as these provide an opportunity for the $\mathbf{w}_j$ chromosome to thrive from a new position in the (C, S^2) space. The allowed new positions for a successful (C_1, S_1^2) can be seen in regions (a), (b) and (c) in Figure 2. We will refer to entire method with this particular fitness as E-ELMCS2.

4 Experiments

We consider six datasets with different features taken from the UCI repository [1] (see Table 1). The data was distributed in 10-fold subsets where all classes had an approximately equal representability. The method we present in this paper is compared with 7 well-known classification algorithms, as coded in Weka 3.6.3 (see [3, 4]). The methods listed include Support Vector Machines (SVM), C4.5, CART, Naive-Bayes/Decision-Tree (NBTree), Logistic, Simple Logistic and Multilayer Perceptron (MLP). Both Table 2 and the corresponding plots in Figure 3 show that we have found the best (C, S^2) results for Haberman, Pima, German and Vehicle. Glass returns an acceptable value of S^2, and our model does not compete well with others when we use Segmentation as dataset. The latter happens mostly when the dataset is very balanced, or when high accuracy C is generally easy to achieve by most methods.

Table 1. Datasets used for the experiments

Dataset	Size	#Input	#Classes	Distribution
Haberman	306	3	2	225-81
Pima	768	8	2	500-268
German	1000	24	2	700-300
Vehicle	846	18	4	212-199-218-217
Glass	214	9	6	70-76-17-13-9-29
Segmentation	2310	19	7	330-330-330-330-330-330-330

Table 2. Comparative C vs. S^2

Dataset	Algorithm	C	S^2	Dataset	Algorithm	C	S^2
Haberman	SVM	0.7353	0.1818	Vehicle	SVM	0.7435	0.0467
	C4.5	0.7288	0.0969		C4.5	0.7246	0.0410
	CART	0.7451	0.1504		CART	0.6891	0.0438
	NBTree	0.7255	0.0957		NBTree	0.7293	0.0308
	Logistic	0.7418	0.1115		Logistic	0.7979	0.0224
	Simplelog.	0.7386	0.1203		Simplelog.	0.7719	0.0302
	MLP	0.6928	0.0498		MLP	0.8168	0.0191
	ELM	0.7763	0.0380		ELM	0.8246	0.0169
	E-ELM	0.7913	0.0311		E-ELM	0.8279	0.0128
Pima	SVM	0.7734	0.0289	Glass	SVM	0.5607	0.0756
	C4.5	0.7383	0.0107		C4.5	0.6682	0.0127
	CART	0.7513	0.0254		CART	0.7056	0.0324
	NBTree	0.7435	0.0115		NBTree	0.7089	0.0257
	Logistic	0.7721	0.0217		Logistic	0.6402	0.0275
	Simplelog.	0.7747	0.0248		Simplelog.	0.6402	0.0454
	MLP	0.7539	0.0114		MLP	0.6776	0.0353
	ELM	0.8021	0.0138		ELM	0.7736	0.0524
	E-ELM	0.8035	0.0105		E-ELM	0.7759	0.0419
German	SVM	0.7640	0.0362	Segmentation	SVM	0.9307	0.0079
	C4.5	0.7390	0.0273		C4.5	0.9693	0.0009
	CART	0.7500	0.0394		CART	0.9615	0.0013
	NBTree	0.7310	0.0423		NBTree	0.9494	0.0020
	Logistic	0.7690	0.0318		Logistic	0.9580	0.0018
	Simplelog.	0.7630	0.0429		Simplelog.	0.9511	0.0028
	MLP	0.7040	0.0190		MLP	0.9606	0.0018
	ELM	0.7917	0.0270		ELM	0.9411	0.0040
	E-ELM	0.8030	0.0153		E-ELM	0.9502	0.0024

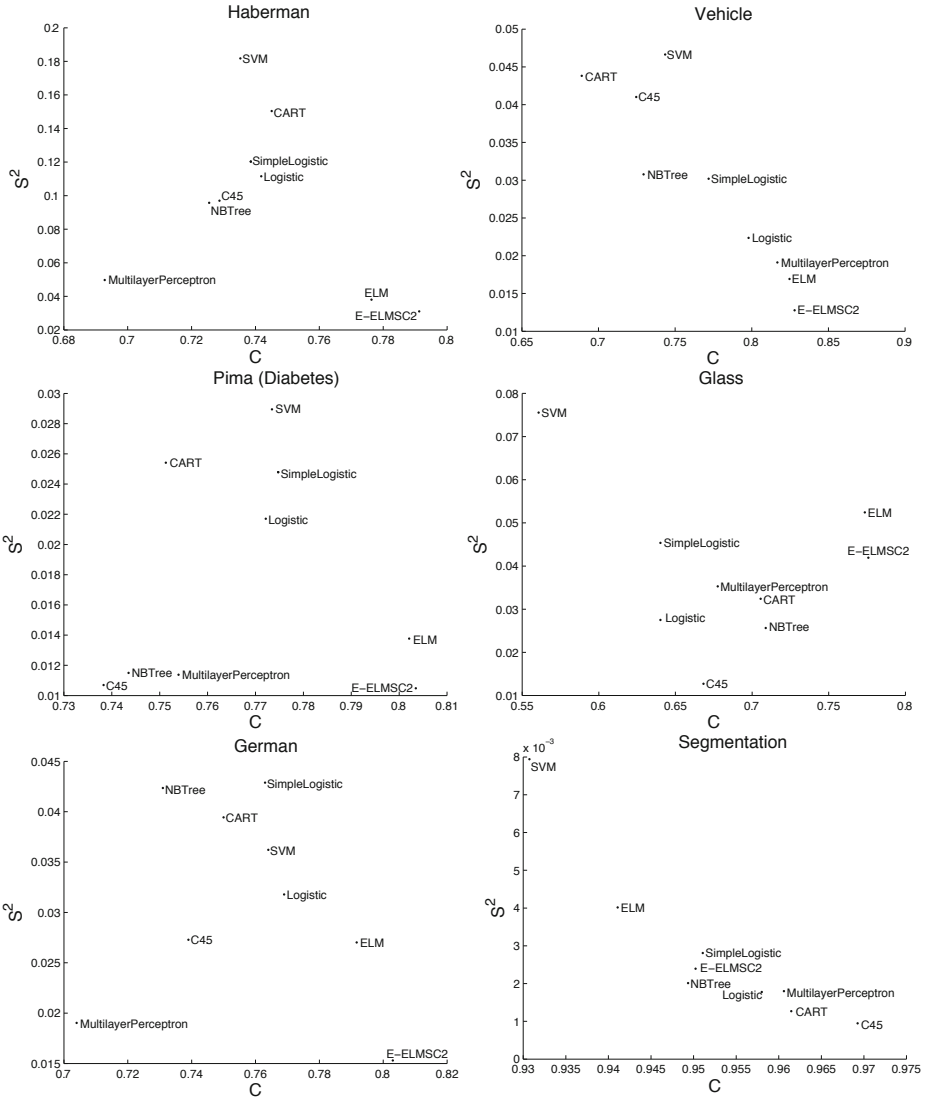

Fig. 3. C vs. S^2 for the databsets shown in Table 2

5 Conclusions

In this work we have presented a hybrid algorithm that returns results with a good balance between C and S^2. The results depicted in the (C, S^2) space show that the E-ELMCS2 algorithm is able to achieve high values of C while having a low S^2, i.e. the dispersion between the accuracy results in each class. Moreover, and looking at the methods used for comparison, the differences in S^2 between them (even when they share a similar C), indicate that this measure introduces the quality information that C did not have on its own.

For future work we will try to refine the fitness function in order to improve the robustness of the evolutionary side of the algorithm. A further analysis of a larger number of datasets will allow us to associate common features of these, with the outcomes of (C, S^2).

References

[1] Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998)
[2] Caballero, F., Martínez, F.J., Hervás, C., Gutiérrez, P.A.: Sensitivity versus accuracy in multiclass problems using memetic pareto evolutionary neural networks. IEEE Transactions on Neural Networks 21(5), 750–770 (2010)
[3] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)
[4] Holmes, G., Donkin, A., Witten, I.H.: Weka: A machine learning workbench. In: Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems, pp. 357–361 (2002)
[5] Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: A new learning scheme of feedforward neural networks. In: Proceedings 2004 IEEE International Joint Conference on Neural Networks, pp. 985–990 (2004)
[6] Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: theory and applications. Neurocomputing 70(1-3), 489–501 (2006)
[7] Martínez-Estudillo, F.J., Gutiérrez, P.A., Hervás, C., Fernández, J.C.: Evolutionary learning by a sensitivity-accuracy approach for multi-class problems. In: IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence) CEC 2008, pp. 1581–1588 (2008)
[8] Price, K.V., Storn, R.M., Lampinen, J.A.: Differential evolution: a practical approach to global optimization. Springer, Heidelberg (2005)
[9] Sánchez-Monedero, J., Hervás-Martínez, C., Martínez-Estudillo, F.J., Ruz, M.C., Moreno, M.C.R., Cruz-Ramírez, M.: Evolutionary learning using a sensitivity-accuracy approach for classification. In: Corchado, E., Graña Romay, M., Manhaes Savio, A. (eds.) HAIS 2010. LNCS, vol. 6077, pp. 288–295. Springer, Heidelberg (2010)
[10] Storn, R., Price, K.: Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11(4), 341–359 (1997)
[11] Zhu, Q.Y., Qin, A.K., Suganthan, P.N., Huang, G.B.: Evolutionary extreme learning machine. Pattern recognition 38(10), 1759–1763 (2005)