A Cost-Sensitive AdaBoost algorithm for Ordinal Regression based on Extreme Learning Machine

Annalisa Riccardi, Francisco Fernández-Navarro, *Member IEEE* and Sante Carloni

***Abstract*—In this paper, the well-known Stagewise Additive Modeling using a Multi-class Exponential (SAMME) boosting algorithm is extended to address problems where there exists a natural order in the targets using a cost-sensitive approach. The proposed ensemble model uses as a base classifier an Extreme Learning Machine (ELM) model, (with the Gaussian kernel and the additional regularization parameter). The closed form of the derived Weighted Least Squares Problem (WLSP) is provided and it is employed to estimate analytically the parameters connecting the hidden layer to the output layer at each iteration of the boosting algorithm. Compared to the state-of-the-art boosting algorithms, in particular those using ELM as base classifier, the suggested technique doesn't require the generation of a new training dataset at each iteration. The adoption of the weighted least squares formulation of the problem has been presented as an unbiased and alternative approach to the already existing ELM boosting techniques. Moreover, the addition of a cost model for weighting the patterns, according to the order of the targets, extends further the classifier to tackle ordinal regression problems. The proposed method has been validated by an experimental study with comparison to already existing ensemble methods and ELM techniques for ordinal regression, showing competitive results.**

***Index Terms*—Ordinal Regression, Boosting, SAMME algorithm, Extreme Learning Machine, Neural Networks**

I. INTRODUCTION

Ordinal regression resides between multi-classification and standard regression in the area of supervised learning. In an ordinal regression problem, the patterns are labeled with a set of discrete ranks [1], [2], [3], [4]. It is commonly formulated as a multi-class problem with ordinal constraints [5], [6]. The goal of learning in ordinal regression is to find a model based on training set which can predict the rank of the patterns in the test set. Several approaches for ordinal regression were proposed in recent years from a machine learning perspective. Vast majority of the algorithms are based on the idea of transforming the ordinal scales into numeric values, and then solving the problem as a standard regression problem [5], [7], [8], [9], [10]. This kind of algorithms are called threshold models. Two examples of threshold algorithms are the support vector based formulations [11], [12] and the Gaussian Process for Ordinal Regression (GPOR) [13] method.

In the field of Extreme Learning Machines (ELMs), Deng et al. [14] proposed a modification in the encoding scheme to adapt the standard ELM algorithm to the ordinal scenario. They considered three methodologies with its corresponding encoding schemes: the single multi-output classifier approach, the multiple binary-classifications with one-against-all decomposition method and the one-against-one method. After that, the models parameters are trained using the corresponding encoding framework. From another perspective, Becerra et al. [15] proposed an evolutionary approach based on the Evolutionary ELM (E-ELM) [16] to address the ordinal regression problem. The authors relied on the assumption that the ordinal structure of the set of class labels is also reflected in the topology of the instance space. Under this idea, Becerra et al. [15] proposed an evolutionary algorithm in two stages. The first stage makes a projection of the ordinal structure of the feature space. Next, an evolutionary algorithm tunes the first projection working with the misclassified patterns near the border of their right class.

On the other hand, ensembles are a promising machine learning research field, where several models are combined to generate a final output [17], [18], [19]. Two factors must be considered in order to enhance the generalization performance of a neural network ensemble. One is diversity and the other one is the performance of the models that comprise the ensemble. A trade-off study between the optimal measures of diversity and performance is available in [18]. The approaches for designing neural network ensembles can be divided in two groups: the first one iterates between different architectures and parameters settings while the second one gets diverse models by training them on different training sets. Some approaches on this idea are bagging, boosting or cross-validation [20], [21], [22]. Both groups of methodologies directly generate a group of neural networks which are error uncorrelated.

For ordinal regression problems, there are some ensemble-related approaches. The main idea of these approaches is to transform the classification problem into a nested binary classification one, and then combine the resulting classifier predictions to obtain the final ensemble model. For example, Frank and Hall [23] proposed a general algorithm that enables binary classifiers to make use of order information in the targets, using as base binary classifier a tree model. Waegeman and Boullart [24] proposed an enhanced method based on an ensemble of Support Vector Machines (SVMs). In their proposal, each binary classifier is trained with specific weights for each pattern of the training set.

Recently, two neural network threshold ensemble models for ordinal regression have been proposed in [10], [25]. For the first ensemble method, the thresholds are fixed a priori and are not modified during training. The second one considers the thresholds of each member of the ensemble as free parameters, allowing their modification during the training

All the authors are with the Advanced Concepts Team, European Space Research and Technology Centre (ESTEC), European Space Agency (ESA), 2201 AZ Noordwijk, Netherlands, e-mail: annalisa.riccardi@esa.int, i22fenaf@uco.es,francisco.fernandez.navarro@esa.int and sante.carloni@esa.int

process. This is achieved through a reformulation of the tunable thresholds to avoid the definition of constraints in the ordinal regression problem. During training diversity, existing in the different projections generated by each member, is taken into account for the parameter updating according to the Negative Correlation Learning (NCL) framework [26], [27]. In the NCL framework, an ensemble of M neural networks are trained in parallel using gradient descent techniques. The error function for each neural network, in addition to the usual squared error term, contains a penalty term proportional to the correlation of the network projections with those of all the other networks. The ordinal thresholds ensemble models of [10], [25] were validated using an economic dataset and real benchmark ordinal datasets

From another point of view, Perez-Ortiz et al. [28] proposed a projection-based ensemble model where every single model is trained in order to distinguish between one given class (j) and all the remaining ones, while grouping them in those classes with a rank lower than j, and those with a rank higher than j. Actually, the proposal could be considered as a reformulation of the well-known one-versus-all scheme. In the study, the base algorithm for the ensemble could be any threshold (or even probabilistic) model.

From a boosting perspective, two algorithms (ORBoost and AdaBoost.OR) [29], [30] were proposed for the ordinal scenario. ORBoost is a thresholded ensemble model for ordinal regression which consists of a weighted ensemble of confidence functions and an ordered vector of thresholds. In [29], the authors also derived novel large margin bounds of common error functions, such as the classification error and the absolute error. Apart from this boosting approach based on binary confidence functions, the same authors proposed an extension of the well-known AdaBoost using the reverse technique to directly improve the performance of existing cost-sensitive ordinal ranking algorithms, AdaBoost.OR [30].

In this paper, the Stagewise Additive Modeling using a Multi-class Exponential (SAMME) boosting algorithm [31] is extended to address ordinal problems. The SAMME model is an alternative approach to the multi class boosting algorithm called AdaBoost.MH [32]. The AdaBoost.MH algorithm addresses the multi class problem performing J one-against-all classifications, where J is the number of classes, while SAMME performs directly the J class classification problem. SAMME only needs weak classifiers better than random guess (e.g. correct probability larger than $1/J$), rather than better than $1/2$ as the two-class AdaBoost requires.

The proposed ensemble model uses as a base classifier an Extreme Learning Machine (ELM) [33] model. Concretely, in this work the Gaussian kernel version of the ELM with the regularization parameter has been considered. The approach integrates the advantages of variable weighting and the speed of ELM. In each iteration of the SAMME algorithm, non-negative weights are assigned to different time steps of the boosting process, reflecting the importance of each pattern in each interval. The parameters corresponding to the linear part of the model are analytically determined in each iteration according to the closed form of the Weighted Least Squares Error (WLSE). Traditionally, the state-of-the-art boosting algorithms using ELM as base classifier generate a new training subset at each iteration. This task is unnecessary if the closed form of the weighted least squares problems is adopted.

Summarizing, the main contributions of this paper are:

- The adaptation of the multi-class SAMME algorithm to the ordinal scenario considering a cost-sensitive approach.
- The use of a ELM model with Gaussian kernel and the regularization parameter as base classifier (for its competitive trade-off between efficiency and accuracy).
- The WLS closed-form solution of the error function was considered to estimate the linear parameters of the individuals in the final ensemble model. This avoids to generate M different sub-datasets, where M is the size of the ensemble, differently from what has been done traditionally in the ELM community [34], [35], [36].

The remainder of the paper is organised as follows: a brief analysis of the SAMME algorithm for multi-class classification is given in Section II. Section III describes the cost-sensitive ensemble model proposed and Section IV draws the way to estimate analytically the parameters of the ELM classifier based on the WLSE. Section V presents the experimental framework while the results are discussed in Section VI. Finally, Section VII summarises the achievements and outlines some future developments of the proposed methodology.

II. Multi-class AdaBoost

In this paper, the so-called Stagewise Additive Modeling using a Multi-class Exponential loss function (SAMME) [31], multi-class version of the AdaBoost method, is adopted. SAMME directly handles the J-class problem by building a single J-class classifier, instead of J binary ones. Zhu et al. [31] proves that the solution of SAMME is consistent with the Bayes classification rule, so it is optimal in minimizing the misclassification error. Given a training set $\mathbf{D} = \{\mathcal{X}, \mathcal{C}\} = \{\mathbf{x}_n, c_n\}_{n=1}^N$, where $\mathbf{x}_n = (x_n^1, x_n^2, \ldots, x_n^K) \in \mathbb{R}^K$ and $c_n \in \{1 \ldots J\} \subset \mathbb{N}$ is the n-th input pattern and its corresponding target, the goal is to find a regression function $\mathbf{f} : \mathbb{R}^K \to \mathbb{R}^J$, i.e., $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_J(\mathbf{x}))$ such that minimizes the following error function:

$$\begin{aligned} \min_{\mathbf{f}(\mathbf{x})} \quad & \sum_{n=1}^{N} L(\mathbf{y}_n, \mathbf{f}(\mathbf{x}_n)) \\ \text{s.t} \quad & f_1(\mathbf{x}_n) + \ldots + f_J(\mathbf{x}_n) = 0, \ \forall n = 1, \ldots, N \end{aligned} \tag{1}$$

where

$$\begin{aligned} L(\mathbf{y}_n, \mathbf{f}(\mathbf{x}_n)) &= \exp\left(-1/J(y_n^1 f_1(\mathbf{x}_n) + \ldots + y_n^J f_J(\mathbf{x}_n))\right) \\ &= \exp\left(-1/J\, \mathbf{y}_n^T \mathbf{f}(\mathbf{x}_n)\right), \end{aligned}$$

is the exponential loss function for the n-th pattern and

$$\mathbf{y}_n = (y_n^1, \ldots, y_n^J), \tag{2}$$

is the J-dimensional vector, encoding of the target c_n, defined for all $j = 1, \ldots J$ as

$$y_n^j = \begin{cases} 1 & \text{if } c_n = j, \\ -\frac{1}{J-1} & \text{if } c_n \neq j. \end{cases} \tag{3}$$

SAMME Algorithm:
Require: Training dataset (D)
Require: Size of the ensemble (M)
Ensure: Ensemble model

1: $w_n^{(1)} \leftarrow 1/N, \forall n = 1, \ldots, N$ {Initialization of the patterns weights}
2: Initialization of the parameters of the ensemble model
3: **for** $m = 1, \ldots, M$ **do**
4: Fit a classifier to the training set using weights $w_n^{(m)}$
5: $e^{(m)} \leftarrow \sum_{n=1}^{N} w_n^{(m)} I(o^{(m)}(\mathbf{x}_n) \neq c_n) / \sum_{n=1}^{N} w_n^{(m)}$ {Computation of the error of the weighted ELM model}
6: $\alpha^{(m)} \leftarrow \log \frac{1-e^{(m)}}{e^{(m)}} + \log(J-1)$
7: $w_n^{(m+1)} \leftarrow w_n^{(m)} \exp(\alpha^{(m)} I(o^{(m)}(\mathbf{x}_n) \neq c_n)), \forall n = 1, \ldots, N$ {Updating the weights}
8: $w_n^{(m+1)} \leftarrow w_n^{(m+1)} / \sum_{n=1}^{N} w_n^{(m+1)}, \forall n = 1, \ldots, N$ {Normalization of the weights}
9: **end for**
10: Output: $C(\mathbf{x}) = \arg\max_{j} \sum_{m=1}^{M} \alpha^{(m)} I(o^{(m)}(\mathbf{x}) = j)$
11: **return** Ensemble model

Fig. 1: SAMME training algorithm framework

The symmetric constraint $f_1(\mathbf{x}_n) + \ldots + f_J(\mathbf{x}_n) = 0$ is included to guarantee the unicity of the solution $\mathbf{f}$, since adding a constant to all $f_j(\mathbf{x}_n)$ will give the same loss as $\sum_{j=1}^{J} y_n^j = 0$ for every $n \in \{1, ..., N\}$. As proved in [31] the formulation of Problem 1 is consistent with the Bayes classification rule.

Fig. 1 describes the algorithmic flow of the SAMME algorithm, where $w_n^{(m)}$ is the weight of the n-th pattern, at the m-th iteration of the ensemble model, and $o^{(m)}(\mathbf{x}_n)$ is the index of the maximum component of the corresponding predicted values

$$o^{(m)}(\mathbf{x}_n) = \arg\max \mathbf{f}^{(m)}(\mathbf{x}_n), \tag{4}$$

with $\mathbf{f}^{(m)}(\mathbf{x}_n)$ the m-th classifier, $I(\cdot)$ is the indicator function ($I(x) = 0$ if x is false, 1 otherwise) and $C(\mathbf{x})$ is the class predicted by the ensemble model for the test pattern $\mathbf{x}$. From Fig. 1, it is possible to recognise which is the main difference between SAMME and two-class AdaBoost. This difference resides in Step 6 of Fig. 1. A further $\log(J-1)$ term is added to guarantee the positiveness of the exponent $\alpha^{(m)}$ (and hence the increasing of the corresponding weight for the misclassified pattern) when the weighted error $e^{(m)} < (J-1)/J$, at each iteration m of the ensemble model. In the case of $J = 2$, the SAMME algorithm is equivalent to the original two-class AdaBoost because $\log(J-1) = 0$.

III. Cost Sensitive AdaBoost for Ordinal Regression

In ordinal regression problems exists an order relation between labels, such as $\mathcal{C}_1 \prec \mathcal{C}_2 \prec \ldots \mathcal{C}_J$, where $\prec$ denotes the given order between different ranks. To be compliant with the previous notation, a bijection between the labels set $\{\mathcal{C}_j\}_{j=1}^{J}$ and integer values $\{1, \ldots, J\}$ is established, that maintains the order, such as $\mathcal{C}_j \leftrightarrow j$.

Based on the approach of [37], designed to tackle combinatorial and imbalanced datasets with a cost-sensitive boosting classifier, a cost model that encodes the penalty of the misclassified patterns for ordinal regression problems is introduced in the ensemble model here proposed. The cost matrix $\mathcal{K} \in \mathbb{R}^J \times \mathbb{R}^J$ used to encode the penalty of the misclassified patterns is the Absolute cost matrix reported in Table I, for the particular case of a 5-class classification problem, where the element at position (i, j) represents the cost of classifying a pattern of class i as pattern of class j [1].

TABLE I: Example of different cost matrices.

Zero-one	Absolute cost	Quadratic cost
$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 2 & 3 \\ 2 & 1 & 0 & 1 & 2 \\ 3 & 2 & 1 & 0 & 1 \\ 4 & 3 & 2 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 4 & 9 & 16 \\ 1 & 0 & 1 & 4 & 9 \\ 4 & 1 & 0 & 1 & 4 \\ 9 & 4 & 1 & 0 & 1 \\ 16 & 9 & 4 & 1 & 0 \end{pmatrix}$

Three cost-sensitive variants of the SAMME algorithm are provided. To guarantee the equivalence to the stagewise additive modeling three different loss functions are used

1) $L_1(\mathbf{y}_n, \mathbf{f}(\mathbf{x}_n)) = \kappa_n \exp(-1/J \, \mathbf{y}_n^T \mathbf{f}(\mathbf{x}_n))$,
2) $L_2(\mathbf{y}_n, \mathbf{f}(\mathbf{x}_n)) = \exp(-\kappa_n/J \, \mathbf{y}_n^T \mathbf{f}(\mathbf{x}_n))$,
3) $L_3(\mathbf{y}_n, \mathbf{f}(\mathbf{x}_n)) = \kappa_n \exp(-\kappa_n/J \, \mathbf{y}_n^T \mathbf{f}(\mathbf{x}_n))$,

where κ_n represents the cost of misclassifying the n-th pattern. Each formulation affect the update rule of the error estimation and/or of the pattern weights at the m-th iteration of the ensemble model (where the weights used in the following iteration are determined). In particular

$$\begin{aligned}
1) \quad e^{(m)} &\leftarrow \frac{\sum_{n=1}^{N} \kappa_n^{(m)} w_n^{(m)} I(o^{(m)}(\mathbf{x}_n) \neq c_n)}{\sum_{n=1}^{N} \kappa_n^{(m)} w_n^{(m)}}, \\
2) \quad w_n^{(m+1)} &\leftarrow w_n^{(m)} \exp(\kappa_n^{(m)} \alpha^{(m)} I(o^{(m)}(\mathbf{x}_n) \neq c_n)), \\
3) \quad e^{(m)} &\leftarrow \frac{\sum_{n=1}^{N} \kappa_n^{(m)} w_n^{(m)} I(o^{(m)}(\mathbf{x}_n) \neq c_n)}{\sum_{n=1}^{N} \kappa_n^{(m)} w_n^{(m)}}, \\
w_n^{(m+1)} &\leftarrow w_n^{(m)} \exp(\kappa_n^{(m)} \alpha^{(m)} I(o^{(m)}(\mathbf{x}_n) \neq c_n)),
\end{aligned}$$

where

$$\kappa_n^{(m)} := \frac{(k_{c_n, o^{(m)}(\mathbf{x}_n)} + 1)}{J}, \tag{5}$$

[1]Please note that all the cost matrices in Table I are symmetric. It is important also to point out that asymmetric cost matrices are often encountered in practical applications as proposed in [38].

with $k_{c_n,o^{(m)}(\mathbf{x}_n)}$ the $(c_n, o^{(m)}(\mathbf{x}_n))$-element of the cost matrix, hence the cost of misclassifying pattern $\mathbf{x}_n$ of class c_n as pattern of the class $o^{(m)}(\mathbf{x}_n)$; J is introduced for robustness as normalization factor and 1 is added to avoid zeroing the equation. If compared with [37], where only one cost value is assigned to the misclassification of each pattern, the proposed model includes a cost schema, $\kappa_n^{(m)}$, whose values depend on the prediction of the m-th model.

For the details of the proof of equivalence with the stagewise additive modeling please refer to [31].

IV. Weighted Least Squares Estimation for Extreme Learning Machine

Extreme Learning Machine (ELM) is an efficient algorithm that determines the output weights of a Single Layer Feedforward Neural Network (SLFNN) using an analytical solution instead of the standard gradient descent algorithm [39]. ELM have been used to solve classification and regression problems in several domains ranging from computer vision [40], credit risk evaluation [41] or bioinformatics [42].

Traditionally, for a SLFNN, all the parameters for the different layers need to be tuned and there is a dependency among the different layers. The gradient descent algorithm is slow and is prone to converge to local minima. Furthermore, to achieve good generalization performance several iterative steps are necessary [33], [43], [44]. The ELM scheme proposed by Huang et. al. [43] overcomes these problems by randomly assigning weights to the input layers and analytically computing the weights for the output layer using a simple generalized inverse operation. The ELM framework has shown comparable classification performance, and faster run times in comparison to support vector machines [45], [46].

Let's note as $\mathbf{v}_s = (v_{s1}, v_{s2}, \ldots, v_{sK})$ the weight vector connecting the input nodes to the s-th basis function, for $s = 1, 2, \ldots, S$ and with $\boldsymbol{\beta}^j = (\beta_1^j, \ldots, \beta_S^j)$ the weight vector connecting the basis functions to the j-th output node for $j = 1, \ldots, J$.

During the training process, ELM determines the parameters $\boldsymbol{\beta}^j$, for all j values, by minimizing the Least Squared Error (LSE) function:

$$\text{LSE} = \sum_{n=1}^{N} \sum_{j=1}^{J} (f_j(\mathbf{x}_n) - y_n^j)^2, \tag{6}$$

where $f_j(\mathbf{x}_n)$ is the estimated output corresponding to the n-th input pattern and the j-th class. It is defined as:

$$f_j(\mathbf{x}_n) = \sum_{s=1}^{S} \beta_s^j \phi(\mathbf{x}_n; \mathbf{v}_s), \quad n \in \{1, \ldots, N\}, \tag{7}$$

where $\phi(\mathbf{x}_n; \mathbf{v}_s)$ is the activation function. According to [47] the concurrent minimization of the training error and the norm of the weight parameters, allows better generalization performance for the network. Hence the minimization problem has the following form

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^S \times \mathbb{R}^J} \left(\|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|^2, \ \|\boldsymbol{\beta}\| \right) \tag{8}$$

where $\|\cdot\|$ is the L2 norm, $\mathbf{H}$ is the hidden layer output matrix of the SLFN:

$$\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_S) = \\ = \begin{pmatrix} \phi_1(\mathbf{x}_1; \mathbf{v}_1) & \ldots & \phi_S(\mathbf{x}_1; \mathbf{v}_S) \\ \ldots & \ldots & \ldots \\ \phi_1(\mathbf{x}_N; \mathbf{v}_1) & \ldots & \phi_S(\mathbf{x}_N; \mathbf{v}_S) \end{pmatrix} \in \mathbb{R}^N \times \mathbb{R}^S \tag{9}$$

$$\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N)^T \in \mathbb{R}^N \times \mathbb{R}^J, \tag{10}$$

and

$$\boldsymbol{\beta} = (\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^J) \in \mathbb{R}^S \times \mathbb{R}^J. \tag{11}$$

The ELM algorithm starts choosing the activation function $\phi(\mathbf{x}, \mathbf{v})$ and the number of basis functions S. Generally, the sigmoidal function is the one selected in the ELM framework although other types of basis functions could be also considered [48], [49]. In the first step, arbitrary weights are assigned to the input weight vectors $\mathbf{v}_s$. The problem of minimizing the training error reduces to solve the linear system

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{Y}. \tag{12}$$

Therefore the output weights $\boldsymbol{\beta}$ are approximated by the Moore-Penrose generalized inverse [43], [44], to guarantee better generalization performance [50],

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{Y}, \tag{13}$$

where

$$\mathbf{H}^{\dagger} = \begin{cases} \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} & \text{for } N < S, \\ \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H} \right)^{-1} \mathbf{H}^T & \text{otherwise,} \end{cases} \tag{14}$$

and $C \in \mathbb{R}$ is a user-specified parameter that promotes generalization performance.

Traditionally Boosting algorithms proceed by continuously minimizing the Weighted Least Square Error (WLSE) between the estimated outputs and its true target. In the field of ELM, several adaptations of the original AdaBoost algorithm have been proposed for regression and classification problems [34], [35], [36]. These approaches use the AdaBoost algorithm to generate M training subsets from the training set, and then train one ELM regressor/classifier for each of training subsets, hence M regressors/classifiers are finally obtained.

In this work, the weights distribution is employed to directly estimate the $\boldsymbol{\beta}$ parameters instead of using it to generate M different sub-datasets. The generation of these M sub-datasets is unnecessary if the WLSE is adopted. Therefore, the goal is to find the parameter matrix $\boldsymbol{\beta}$ which minimizes the WLSE for all n patterns in the training set with weight w_n, i.e.:

$$\text{WLSE} = \sum_{n=1}^{N} \sum_{j=1}^{J} w_n (f_j(\mathbf{x}_n) - y_n^j)^2. \tag{15}$$

As before, to improve the generalization performance, the norm of the weights need to be minimized concurrently. Therefore the problem can be formulated as

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^S \times \mathbb{R}^J} \left((\mathbf{H}\boldsymbol{\beta} - \mathbf{Y})^T \mathbf{W} (\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}), \ \|\boldsymbol{\beta}\| \right) \tag{16}$$

AdaBoost(ELM) Algorithm:
Require: Training dataset (D)
Require: Size of the ensemble (M)
Require: Regularization Parameter (C)
Require: Width Gaussian Kernel (k)
Ensure: ELM Ensemble model
$w_n^{(1)} \leftarrow 1/N, \forall n = 1, \ldots, N$ {Initialization of the patterns weights}
Estimation of $\mathbf{\Omega}_{\mathrm{ELM}}$
Initialization of the parameters of the ensemble model
for $m = 1, \ldots, M$ **do**
  $\mathbf{f}^{(m)}(\mathbf{x}) := \mathbf{K}(\mathbf{x})^T \left(\frac{\mathbf{I}}{C} + \mathbf{W}^{(m)}\mathbf{\Omega}_{\mathrm{ELM}}\right)^{-1} \mathbf{W}^{(m)}\mathbf{Y}$ {Computation of the kernelized output function}
  $e^{(m)} \leftarrow \sum_{n=1}^{N} w_n^{(m)} I(o^{(m)}(\mathbf{x}_n) \neq c_n) / \sum_{n=1}^{N} w_n^{(m)}$ {Computation of the error of the weighted ELM model}
  $\alpha^{(m)} \leftarrow \log \frac{1-e^{(m)}}{e^{(m)}} + \log(J-1)$
  $w_n^{(m+1)} \leftarrow w_n^{(m)} \exp(\alpha^{(m)} I(o^{(m)}(\mathbf{x}_n) \neq c_n)), \forall n = 1, \ldots, N$ {Updating of the weights}
  $w_n^{(m+1)} \leftarrow w_n^{(m+1)} / \sum_{n=1}^{N} w_n^{(m+1)}, \forall n = 1, \ldots, N$ {Normalization of the weights}
end for
Output: $C(\mathbf{x}) = \arg\max_j \sum_{m=1}^{M} \alpha^{(m)} I(o^{(m)}(\mathbf{x}) = j)$
return Ensemble model

Fig. 2: AdaBoost(ELM) training algorithm framework

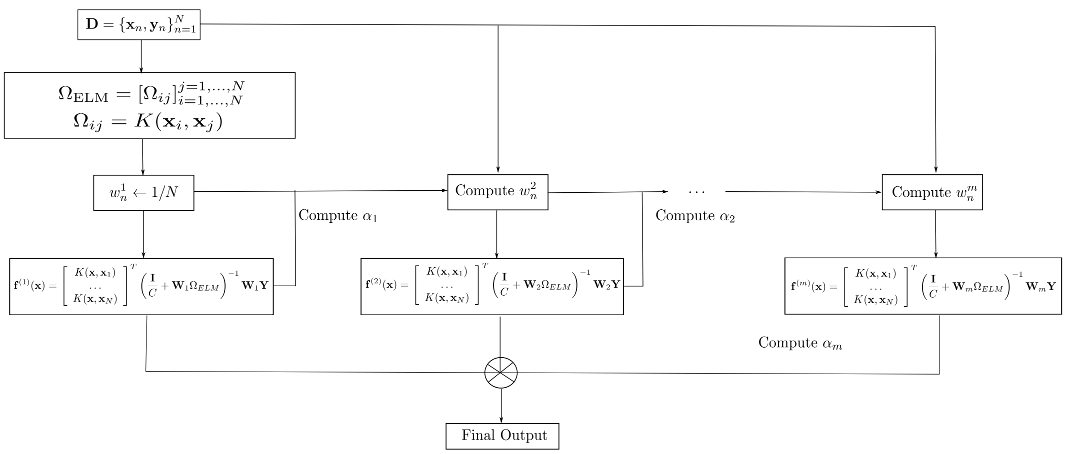


Fig. 3: Graphical illustration of the AdaBoost(ELM)

where $\mathbf{W}$ is a diagonal matrix of dimension $N \times N$ defined as:

$$\mathbf{W} = \begin{pmatrix} w_1 & 0 & \ldots & 0 \\ 0 & w_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & w_N \end{pmatrix} \in \mathbb{R}^N \times \mathbb{R}^N. \quad (17)$$

The optimal $\boldsymbol{\beta}$ value is computed as critical point of the first order derivative of the weighted error function, hence solution of the following linear system

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\beta}} \left[(\mathbf{H}\boldsymbol{\beta} - \mathbf{Y})^T \mathbf{W} (\mathbf{H}\boldsymbol{\beta} - \mathbf{Y})\right] &= 0 \\ \frac{\partial}{\partial \boldsymbol{\beta}} \Big[(\boldsymbol{\beta}^T \mathbf{H}^T \mathbf{W} \mathbf{H} \boldsymbol{\beta} - \boldsymbol{\beta}^T \mathbf{H}^T \mathbf{W} \mathbf{Y} - \mathbf{Y}^T \mathbf{W} \mathbf{H} \boldsymbol{\beta} + & \\ + \mathbf{Y}^T \mathbf{W} \mathbf{Y})\Big] &= 0 \\ (\mathbf{H}^T \mathbf{W} \mathbf{H} \boldsymbol{\beta})^T + \boldsymbol{\beta}^T \mathbf{H}^T \mathbf{W} \mathbf{H} - (\mathbf{H}^T \mathbf{W} \mathbf{Y})^T - \mathbf{Y}^T \mathbf{W} \mathbf{H} &= 0 \\ 2\boldsymbol{\beta}^T \mathbf{H}^T \mathbf{W} \mathbf{H} - 2\mathbf{Y}^T \mathbf{W} \mathbf{H} &= 0. \end{aligned}$$

Finally, the weighted least squares solution can be approximate by the generalized form:

$$\hat{\boldsymbol{\beta}} = \begin{cases} \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{W}\mathbf{H}\mathbf{H}^T\right)^{-1} \mathbf{W}\mathbf{Y} & \text{for } N < S, \\ \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{W} \mathbf{H}\right)^{-1} \mathbf{H}^T \mathbf{W} \mathbf{Y} & \text{otherwise.} \end{cases} \quad (18)$$

The output function of the m-th ELM classifier is defined as (just for the case $N < S$)

$$\begin{aligned} \mathbf{f}^{(m)}(\mathbf{x}) &= \mathbf{h}(\mathbf{x})\hat{\boldsymbol{\beta}} \\ &= \mathbf{h}(\mathbf{x})\mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{W}\mathbf{H}\mathbf{H}^T\right)^{-1} \mathbf{W}\mathbf{Y}, \end{aligned} \quad (19)$$

where $\mathbf{h}(\mathbf{x})$ is a mapping function that corresponds to the basis functions outputs in the neural network literature or it is unknown to users in the kernel machines literature. Therefore, the output function can be kernelized, as suggested in [44], as

$$\mathbf{f}^{(m)}(\mathbf{x}) = \mathbf{K}(\mathbf{x})^T \left(\frac{\mathbf{I}}{C} + \mathbf{W}\mathbf{\Omega}_{\mathrm{ELM}}\right)^{-1} \mathbf{W}\mathbf{Y}, \quad (20)$$

where $\mathbf{K}(\mathbf{x}) : \mathbb{R}^K \to \mathbb{R}^N$ is the vector of kernel functions $\mathbf{K}(\mathbf{x})^T = [K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_N)]$. The Gaussian kernel function here considered is

$$K(\mathbf{x}, \mathbf{x}_i) = \exp(-k||\mathbf{x} - \mathbf{x}_i||^2), \quad i = 1, \ldots, N \tag{21}$$

where $k \in \mathbb{R}$ is the kernel parameter. Similarly the kernel matrix $\mathbf{\Omega}_{\mathrm{ELM}} = [\Omega_{i,j}]_{i,j=1,\ldots,N}$ is defined element by element as

$$\Omega_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j). \tag{22}$$

The algorithm proposed is named AdaBoost based on ELM (AdaBoost(ELM)) and is described in Fig. 2 and Fig. 3.

To tackle Ordinal Regression problems the AdaBoost(ELM) algorithm has been extended to include the cost model introduced in Section III. In particular three new algorithms are generated, namely AdaBoost for Ordinal Regression based on ELM and Cost model i (AdaBoost(ELM).ORC[i]), with $i = 1, 2, 3$. They differ from the algorithm in Figure 2 in the update schema of the error estimation and/or of the patterns weights. In particular the following modifications apply

- AdaBoost(ELM).ORC1:

$$6 : e^{(m)} \leftarrow \frac{\sum_{n=1}^{N} \kappa_n^{(m)} w_n^{(m)} I(o^{(m)}(\mathbf{x}_n) \neq c_n)}{\sum_{n=1}^{N} \kappa_n^{(m)} w_n^{(m)}}$$

- AdaBoost(ELM).ORC2:

$$8 : w_n^{(m+1)} \leftarrow w_n^{(m)} \exp(\kappa_n^{(m)} \alpha^{(m)} I(o^{(m)}(\mathbf{x}_n) \neq c_n)) \quad \forall n = 1, \ldots, N$$

- AdaBoost(ELM).ORC3:

$$6 : e^{(m)} \leftarrow \frac{\sum_{n=1}^{N} \kappa_n^{(m)} w_n^{(m)} I(o^{(m)}(\mathbf{x}_n) \neq c_n)}{\sum_{n=1}^{N} \kappa_n^{(m)} w_n^{(m)}}$$

$$8 : w_n^{(m+1)} \leftarrow w_n^{(m)} \exp(\kappa_n^{(m)} \alpha^{(m)} I(o^{(m)}(\mathbf{x}_n) \neq c_n)) \quad \forall n = 1, \ldots, N$$

where $\kappa_n^{(m)}$ is the cost factor computed as described in Section III.

V. Experimental Framework

In this section, the experimental study performed to validate the new algorithms is presented. In Section V-A details of the datasets selected for the experimentation are provided. Section V-B gives the measures employed to evaluate the performance of the algorithms. Instead, Section V-C is dedicated to a description of the algorithms chosen for the comparison and their relevant parameters. Finally, the description of the statistical tests used to validate the obtained results (see Section V-D) is provided.

A. *Ordinal regression datasets*

Sixteen datasets have been selected from the UCI [51] and the *mldata.org* repositories and one synthetic dataset (the *toy* dataset) has been included in the test sets. The latter dataset was created as suggested in [52]: 300 example patterns $\mathbf{x} = (x_1, x_2)$ were generated uniformly at random in the unit square $[0, 1] \times [0, 1] \subset \mathbb{R}^2$. To each pattern a class y from the set $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5\}$ has been assigned according to:

$$\mathcal{O}(y) \quad = \quad \min\{j : \theta_{j-1} < 10(x_1 - 0.5)(x_2 - 0.5) + \varepsilon < \theta_j\}$$

where $\mathcal{O}(y)$ represents the rank of the patterns, θ_j is the threshold for the j-th class, according to the values

$$(\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5) = (-\infty, -1, -0.1, 0.25, 1, \infty),$$

and $\varepsilon \sim N(0; 0.125^2)$ simulates the possible existence of error in the assignment of the true class to $\mathbf{x}$.

TABLE II: Characteristics of the sixteen datasets used for the experiments: number of patterns (Size), total number of inputs (#In.), number of classes (#Out.), and number of patterns per-class (NPPC)

Dataset	Size	#In.	#Out.	NPPC
ERA	1000	4	9	(92,142,181,172,158,118,88,31,18)
ELS	488	4	9	(2,12,38,100,116,135,62,19,4)
LEV	1000	4	5	(93,280,403,197,27)
SWD	1000	10	4	(32,352,399,217)
automobile	205	71	6	(3,22,67,54,32,27)
balance-scale	625	4	3	(288,49,288)
car	1728	21	4	(1210,384,69,65)
contact-lenses	24	6	3	(15,4,4)
eucalyptus	736	91	5	(180,107,130,214,105)
newthyroid	215	5	3	(30,150,35)
pasture	36	25	3	(12,12,12)
squash-stored	52	51	3	(23,21,8)
squash-unstored	52	52	3	(24,24,4)
tae	151	54	3	(49,50,52)
toy	300	2	5	(35,87,79,68,31)
winequality-red	1599	11	6	(10,53,681,638,199,19)

Table II summarizes the properties of the selected datasets. It shows, for each dataset, the number of patterns (Size), the total number of inputs (#In.), the number of classes (#Out.) and the number of patterns per-class (NPPC). Their descriptions (available in the web sites) lead to the conclusion that they are ordinal datasets since the class labels show an ordinal nature.

The datasets considered are partitioned by using a hold-out cross-validation procedure. Concretely, 30 different stratified random splits of the datasets have been considered, with 75% and 25% of the instances in the training and test sets respectively (30 hold-outs).

B. *Performance measures for Ordinal Regression*

In this study, ordinal regression datasets are considered. In these domains, two measures are widely used because of their simplicity and successful application. Therefore, two evaluation metrics have been considered which quantify the accuracy of N predicted ordinal labels for a given dataset $\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_N\}$, with respect to the true targets $\{y_1, y_2, \ldots, y_N\}$. Namely they are:

- **Accuracy rate** (*Acc*): It is the number of successful hits (correct classifications) relative to the total number of classifications. It has been by far the most commonly

TABLE III: Parameter specification for the methods considered (C: regularization parameter; k: width of the Gaussian functions;M: number of models in the ensemble;S: number of basis functions). The criteria for selecting the best configuration was the MAE performance

Algorithm	Ref.	Parameters
ASAOR	[23]	There is no hyperparameters to be considered
MCOSvm	[24]	C: Best $\in \{10^3, 10^2, \ldots, 10^{-3}\}$; k: Best $\in \{10^3, 10^2, \ldots, 10^{-3}\}$; Gaussian Kernel
ORBoost-All	[29]	$M = 25$; S Best $\in \{5, 10, 15, 20, 30, 40\}$; Sigmoidal Basis Function
ORBoost-LR	[29]	$M = 25$; S Best $\in \{5, 10, 15, 20, 30, 40\}$; Sigmoidal Basis Function
ELMOR	[53]	S Best $\in \{10 + i10\}$, $i = 0, \ldots, 19$; Sigmoidal Basis Function
AdaBoost(ELM)	-	$M = 25$; C: Best $\in \{10^3, 10^2, \ldots, 10^{-3}\}$; k: Best $\in \{10^3, 10^2, \ldots, 10^{-3}\}$; Gaussian Kernel
AdaBoost(ELM).ORC1	-	$M = 25$; C: Best $\in \{10^3, 10^2, \ldots, 10^{-3}\}$; k: Best $\in \{10^3, 10^2, \ldots, 10^{-3}\}$; Gaussian Kernel
AdaBoost(ELM).ORC2	-	$M = 25$; C: Best $\in \{10^3, 10^2, \ldots, 10^{-3}\}$; k: Best $\in \{10^3, 10^2, \ldots, 10^{-3}\}$; Gaussian Kernel
AdaBoost(ELM).ORC3	-	$M = 25$; C: Best $\in \{10^3, 10^2, \ldots, 10^{-3}\}$; k: Best $\in \{10^3, 10^2, \ldots, 10^{-3}\}$; Gaussian Kernel

used metric to assess the performance of classifiers for years [3]. The mathematical expression of Acc is:

$$Acc = \frac{1}{N} \sum_{n=1}^{N} I\left(\hat{y}_n = y_n\right), \quad (23)$$

where $I(\cdot)$ is the zero-one loss function and N is the number of patterns of the dataset.

- **Mean Absolute Error** (MAE): It is the average deviation of the prediction from the true targets, i.e.:

$$MAE = \frac{1}{N} \sum_{n=1}^{N} \left|\mathcal{O}(\hat{y}_n) - \mathcal{O}(y_n)\right|, \quad (24)$$

where $\mathcal{O}(\mathcal{C}_j) = j, 1 \leq j \leq J$, i.e. $\mathcal{O}(y_n)$ is the rank of pattern $\mathbf{x}_n$ according to the encoding scheme used.

These measures aim to evaluate different aspects that can be taken into account when an ordinal regression problem is considered: (a) Acc measures that patterns are generally well classified, and (b) MAE measures that the classifier tends to predict a class as closely as possible to the real class without taking into account the relative sizes of the classes.

Additionally, the time required to estimate the parameters of each method has been also considered. The time (T) is the simplest way to measure the practical efficiency of a method. The average time elapsed (in seconds) is analyzed by every method, considering cross-validation time, training and test time.

C. Comparison Methods

The models proposed have been evaluated comparing their results to the results of ensemble models for ordinal regression and one extreme learning approach for ordinal data. All of them have been already mentioned in the Introduction section.

- Ensemble approaches for Ordinal regression:
  - **A Simple Approach to Ordinal Regression (ASAOR)** [23] is a meta classifier that allows standard classification algorithms to be applied to ordinal class problems. In the current work, the C4.5 method available in Weka [54] is used as the underlying classification algorithm, since this is the one initially employed by the authors.
  - **Multi-Class Ordinal Support vector machines (MCOSvm)** [24] is an enhanced ensemble method for ordinal regression. As proposed in [24], weighted SVMs are used as base classifiers. Specific weights are assigned to each pattern in such a way that errors of more than one rank are heavier penalized. Therefore the weight of a training pattern differs for each binary SVM.
  - **Ordinal Regression Boosting (ORBoost)** [29] is a thresholded ensemble model for ordinal regression problems. The model consists of a weighted ensemble of confidence functions and an ordered vector of thresholds. ORBoost can be used with any base learners for confidence functions. In the presented experimental study, a standard feedforward neural network is used as the underlying classification model. Two boosting approaches are considered:
    - ∗ ORBoost with all margins (ORBoost-All).
    - ∗ ORBoost with left-right margins (ORBoost-LR).
- ELM models for Ordinal regression:
  - **Extreme Learning Machine for Ordinal Regression (ELMOR)** [53]. For this experimental study the single model proposed in [53] is employed. The other two multiple model approaches have not been considered for efficiency reasons.

Table III presents the parameters configuration of the different models proposed. In the case of ensemble models the same size has been considered for all the methods $M = 25$. However, for the iterative neural network ensemble algorithms (ORBoost.LR and ORBoost.All), the number of basis functions S, were selected by considering the following values, $S \in \{5, 10, 20, 30, 40\}$ while for the ordinal ELM algorithm (ELMOR), it is necessary to consider a more extensive set of possible number of basis functions, in this case $S \in \{10+i10\}$ with $i = 0, \ldots, 19$, given that the method relies on random projections. For the ensemble kernel methods (MCOSvm and AdaBoost(ELM) algorithm and its ordinal variants), the regularization parameter, C, and the width of the Gaussian kernel, k, were selected by considering the following set of values, C and $k \in \{10^3, 10^2, \ldots, 10^{-3}\}$. The hyperparameters were adjusted using a grid search with a 5-fold cross-validation considering just the training set. Despite this, the optimal number of basis functions for the ELMOR could be also determined using the approach proposed in [55].

D. Statistical Tests for Performance Comparison

In the presented experimental study, the hypothesis testing techniques are used to provide statistical support for the analysis of the results. Concretely, nonparametric tests have been used, due to the fact that the initial conditions that guarantee the reliability of the parametric tests may not be satisfied, causing the statistical analysis to lose credibility [56].

Throughout the study, the Friedman test is used to detect statistical differences among the methods. Holm *post hoc* procedure will be used to find out which methods are distinctive among the multiple comparisons performed [56].

VI. Results and Analysis

In this section, the different experimental studies carried out with the cost-sensitive boosting proposals are detailed. In particular, the aims are multiple:

1) To compare the generalization performance of the approaches proposed to recent ensemble and ELM algorithms for ordinal regression (Section VI-A).
2) To test the time complexity of the models proposed compared to the above-mentioned methods (Section VI-B).
3) To show the influence of the hyperparameters in the overall performance (Section VI-C)

A. Comparison between the models proposed and ensemble and ELM algorithms for ordinal regression

For the sake of simplicity, only the graphical and the summary of the statistical results achieved are included, whereas the complete results can be found online[2].

TABLE IV: Summary of results in Acc and MAE for the generalization set: Mean results over all the datasets, mean ranking and Holm statistical test results (using as the control method the one with the best mean ranking) for $\alpha = 0.10$

Acc generalization results					
Algorithm	$\overline{Acc}$	$\overline{R}_{\text{Acc}}$	z-statistic	p-value	α_{Adjusted}
$\text{ELMOR}_{\bullet}$	64.12	7.68	5.06	0.00	0.01
$\text{AdaBoost(ELM)}_{\bullet}$	66.36	6.84	4.19	3.0E-5	0.01
$\text{ASAOR}_{\bullet}$	66.12	5.68	3.00	2.6E-3	0.01
$\text{ORBoost}-\text{All}_{\bullet}$	69.58	5.28	2.58	9.8E-3	0.02
AdaBoost(ELM).ORC1	69.68	4.46	1.74	0.08	0.03
ORBoost-LR	70.32	4.28	1.54	0.12	0.03
AdaBoost(ELM).ORC2	70.34	4.18	1.45	0.14	0.05
MCOSvm	71.36	3.78	1.03	0.30	0.10
$\text{AdaBoost(ELM).ORC3}_{+}$	71.88	2.78	-	-	-
MAE generalization results					
Algorithm	$\overline{MAE}$	$\overline{R}_{\text{MAE}}$	z-statistic	p-value	α_{Adjusted}
$\text{ELMOR}_{\bullet}$	0.49	8.43	6.51	0.00	0.01
$\text{AdaBoost(ELM)}_{\bullet}$	0.42	7.06	5.09	0.00	0.01
$\text{ASAOR}_{\bullet}$	0.40	5.62	3.61	3.0E-4	0.01
$\text{AdaBoost(ELM).ORC1}_{\bullet}$	0.37	5.12	3.09	1.9E-3	0.02
$\text{ORBoost}-\text{All}_{\bullet}$	0.36	5.03	3.00	2.6E-3	0.03
$\text{ORBoost}-\text{LR}_{\bullet}$	0.36	4.40	2.35	0.01	0.03
AdaBoost(ELM).ORC2	0.36	3.78	1.71	0.08	0.05
MCOSvm	0.34	3.40	1.32	0.18	0.10
$\text{AdaBoost(ELM).ORC3}_{+}$	0.34	2.12	-	-	-

$\bullet$ Statistical differences are found
$+$ Control Method

Fig. 4 is the star plot representation of generalization performance of the comparison of the different methodologies. This star plot represents the performance as the distance from the center; hence a higher area determines the best average performance where the goal is to maximize the metric (Acc) and lower area determines the best average performance where the goal is to minimize (MAE). The plot allows to visualize the performance of the algorithms comparatively for each dataset. As can be seen in Fig. 4, the AdaBoost(ELM).ORC3 is the most promising methodology following by the MCOSvm method. From the analysis of the results (Table IV), it can be concluded that the AdaBoost(ELM).ORC3 model produces the best mean ranking in Acc and MAE ($\overline{R_{\text{Acc}}} = 2.78$ and $\overline{R_{\text{MAE}}} = 2.12$), reporting also the best mean accuracy and mean absolute error ($\overline{Acc} = 71.88\%$ and $\overline{MAE} = 0.34$).

To determine the statistical significance of the rank differences observed for each method in the different datasets, a non-parametric Friedman test [57] has been completed with the ranking of Acc and MAE in the generalization set of the best models as test variables. The test shows that the effect of the method used for classification is statistically significant at a significance level of 10%.

Based on this rejection, the Holm post-hoc test was used to compare all classifiers with a control method [58]. For the experiments carried out, the control method selected is the one reporting the best mean ranking in Acc and MAE, the AdaBoost(ELM).ORC3. The results of the Holm test for $\alpha = 0.10$ can be seen in Table IV. By using a level of significance $\alpha = 0.10$, AdaBoost(ELM).ORC3 is significantly better than ELMOR, AdaBoost(ELM), ASAOR and ORBoost-All using Acc as variable test, and significantly better than ELMOR, AdaBoost(ELM), ASAOR, AdaBoost(ELM).ORC1, ORBoost-All and ORBoost-LR using MAE as variable test.

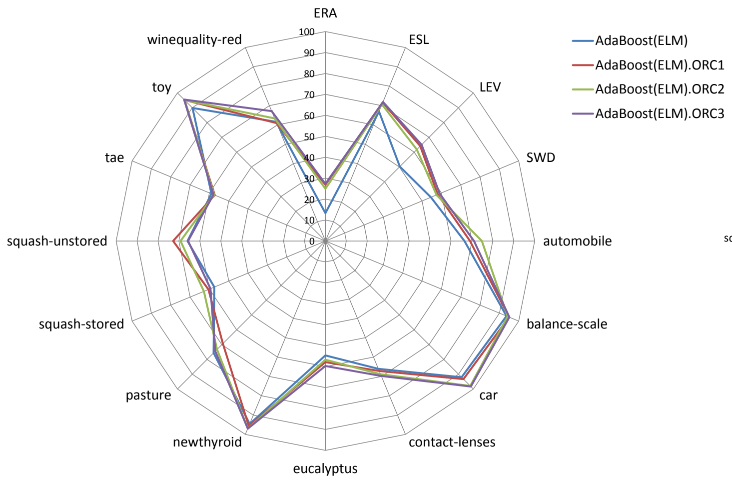
As can be seen in Table IV, the AdaBoost(ELM).ORC3 algorithm is competitive when compared to the most promising ensemble methods for ordinal regression. Furthermore, it is much more efficient than most of them. This justifies its proposal.

B. Time complexity analysis

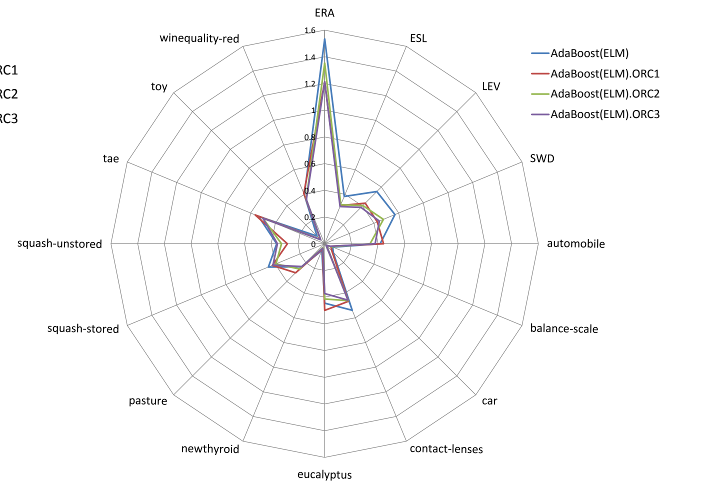
In this section, the computational time and complexity of the proposed methods are analyzed and compared to the already existing ensemble models for ordinal regression already presented in the experimental section.

The computational complexity of the SAMME algorithm is conditioned by the choice of its base classifier. In the proposed ELM model the computation of the kernel matrix has a quadratic complexity in N, where N is the size of the dataset. However the kernel matrix is initialized at the beginning of the ensemble and not recomputed. In each iteration of model, the most time consuming task is the inversion of a $N \times N$ matrix and the multiplication of it with a matrix of dimension $N \times J$. The computational complexity of the multiplication of the two matrices is $O(N^2J)$, while the complexity of inverting the matrix of dimension N is $O(N^3)$ (if the Gauss–Jordan elimination algorithm is used), where N is the number of training patterns and J is the number of classes. Hence the computational complexity of the AdaBoost(ELM) algorithm is $O((N^3 + N^2J)M)$, where M is the size of its ensemble [31].

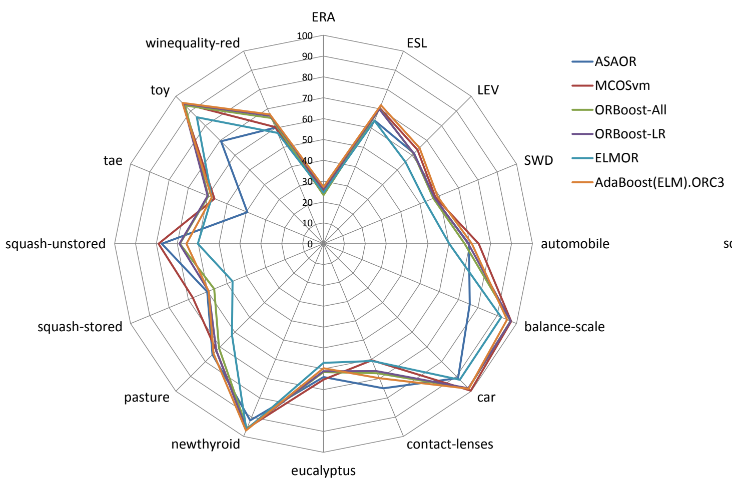
[2]http://www.esa.int/gsp/ACT/cms/projects/ResultsAdaboostELM.zip



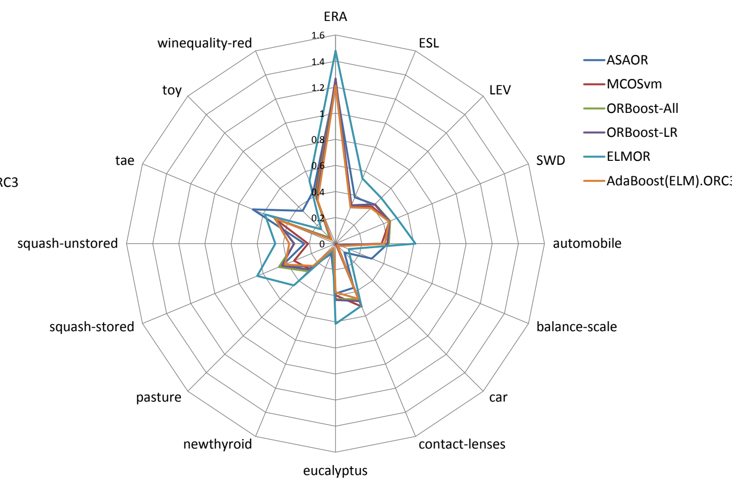
(a) Acc results: Comparison to AdaBoost models



(b) MAE results: Comparison to AdaBoost models



(c) Acc results: AdaBoost(ELM).ORC3 versus state-of-the-art models



(d) MAE results: AdaBoost(ELM).ORC3 versus state-of-the-art models

Fig. 4: Radar illustration of the results on Acc (Figure 4a and 4c) and MAE (Figure 4b and 4d)

The time recorded included cross-validation, training and test, and it is shown in Table V. The number of hyperparameters of each method is decisive for the final time spent in running the algorithms, given that they have to be adjusted using a time-consuming cross-validation process (see Section V-C for further details).

TABLE V: Computational time results in seconds (cross-validation, training and test) for the *toy* dataset and all the methods: average and standard deviation over the 30 holdouts.

Computational Time ($Mean_{SD}$)	
ORBoost-All	216.92 (160.54)
ORBoost-LR	215.92 (76.35)
MCOSvm	27.4 (0.90)
AdaBoost(ELM).ORC1	10.6 (0.5)
AdaBoost(ELM).ORC2	10.6 (0.5)
AdaBoost(ELM).ORC3	10.5 (0.3)
AdaBoost(ELM)	10.4 (0.4)
ELMOR	1.2 (0.6)
ASAOR	0.15 (0.04)

As can be seen, the ensemble models proposed are the methods with the lowest computational time, together with MCOSvm, ELMOR and ASAOR. The differences in time of these methods are not significant if they are compared to the differences with the ORBoost-All and ORBoost-LR methods. A simplified version of the proposed ensemble model, with a neural network as base classifier and without the regularization parameter and the kernel functions, has a single hyperparameter to be tuned (the number of hidden nodes) and doesn't require the computation of the kernel matrix. This results in a more computational efficient model (is gained approximately one order of magnitude) but less performing. For this reason, the base classifier with its kernel version and with the regularization parameter is the one proposed in this paper.

Furthermore, note that software implementations can affect these times. For example, the ASAOR Weka implementation was written in Java and the remaining methods were run using a common Matlab framework proposed in Gutierrez et al. [59].

In general, the most efficient algorithms are the ones based on ELM. Both are trained without iterative tuning. Despite this, the lowest computation time is achieved by the ASAOR algorithm. The reason of that is that the ASAOR algorithm has not any hyperparameters to be optimized by cross-validation unlike ELMOR and AdaBoost(ELM) approaches (they have, respectively, the number of basis functions S, and the kernel and regularization parameters (C, k) as hyperparameters). The efficiency of the models proposed and their good performance justify their proposal.

C. *Influence of the hyperparameters*

The proposed algorithms rely on three hypeparameters that need to be set: the size of the ensemble M, the regularization coefficient, C and the width of the Gaussian kernel k. A study has been performed to analyze the sensitivity of the model, in terms of Accuracy and MAE, with respect to the three hyperparameters. The algorithm considered is the one achieving the best results, AdaBoost(ELM).ORC3, on the *toy* problem. The hyperparameters are compared 2-by-2 fixing the value of the third one to the best value achieved in the cross validation process. In particular the best set of values used in this particular case is

$$(M^*, C^*, k^*) = (25, 10, 1). \tag{25}$$

While (C^*, k^*) are result of the cross-validation process, $M^* = 25$ has been considered as competitive trade-off between efficiency, diversity and accuracy [60]. Several runs of the AdaBoost(ELM).ORC3 model have been performed for values of the three hyperparameters ranging in the sets

$$\begin{aligned} M &\in \{10, \ldots, 50\} \\ C, k &\in \{10^{-3}, \ldots, 10^{3}\}. \end{aligned} \tag{26}$$

Results are reported in Fig. 5 and Fig. 6, where also the solution of the cross validation process is drawn in the contour lines plot for comparison. As expected the model is less sensitive to the size of its ensemble: the significant variations in performance are determined by the (C, k) parameters. The most critical parameter is the width of the Gaussian kernel k. The accuracy of the model has a very sensitive behavior with respect to the parameter k, with a drop down up to 80% of the overall model performance.

VII. Conclusions

The presented work extends the class of boosting algorithms for ordinal regression. In particular it enlarges the family of models that employ Extreme Learning Machine (ELM) as a base classifier. It differs from the already existing techniques in the way of addressing the training at each iteration of the ensemble. Instead of generating at each step a new training dataset according to the new set of patterns weights, the weights are used into the definition of the training problem, solving the derived Weighted Least Squares Problem (WLSP) in a close form and maintain the original training dataset during all the iterations cycle. Moreover, in order to be applied to Ordinal Regression problems, three cost models have been proposed that affect the way in which the weights are redistributed among the patterns.

After introducing the existing boosting algorithms, in particular those using ELM as base classifier, more attention has been given to the description of the Stagewise Additive Modeling using a Multi-class Exponential loss function (SAMME) algorithm, being the version of the AdaBoost method adopted in the proposed algorithms. The SAMME algorithm has been extended, in order to address ordinal regression problems, including three cost models and using an ELM as base classifier that determines the linear parameters of the kernel ELM method using the analytic solution of the WLSP. This led to the definition of four new algorithms, namely AdaBoost(ELM) for nominal classification and AdaBoost(ELM).ORC1, AdaBoost(ELM).ORC2 and AdaBoost(ELM).ORC3 for ordinal regression.

Ordinal regression datasets available in the community and one synthetic dataset (the toy dataset) have been used as benchmark test sets, four algorithms from the state-of-the-art ensemble models for ordinal regression (ASAOR, MCOSvm, ORBoost-All, ORBoost-LR) and one extreme learning approach for ordinal data (ELMOR) have been used for comparison and the model performance has been evaluated using the Accuracy and Mean Absolute Error (MAE) measures. Finally the models have been compared also in terms of computational efficiency, non parametric statistical tests have been performed to validate the results and an analysis of the influence of the hyperparameters on the selected metrics has also been included.

From the results of these tests the AdaBoost(ELM).ORC3 algorithm is the method, among the one proposed in this article, with the most effective cost model. The algorithm reaches competitive results in terms of performance with the state of the art ensemble models, achieving the best mean ranking in accuracy and in mean absolute error. Furthermore, the models proposed outperforms in efficiency the selected ensemble models for ordinal regression but the ASAOR algorithm. It's comparable performances with the state-of-the-art algorithms and its efficiency justify its proposal.

The adaptation of the algorithms proposed to the incremental learning paradigm will be considered as future work. Indeed the Adaboost algorithm has already been adapted to the incremental learning paradigm [61] for nominal classification [62], [63] but not for ordinal regression problems.

References

[1] A. K. Jain, R. P. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience, 2000.

[3] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed., ser. Data Management Systems. Morgan Kaufmann (Elsevier), 2005.

[4] V. Cherkassky and F. M. Mulier, *Learning from Data: Concepts, Theory, and Methods*. Wiley-Interscience, 2007.

[5] B.-Y. Sun, J. Li, D. D. Wu, X.-M. Zhang, and W.-B. Li, "Kernel discriminant learning for ordinal regression," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 6, pp. 906–910, 2010.

[6] C.-W. Seah, I. W. Tsang, and Y.-S. Ong, "Transductive ordinal regression," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 7, pp. 1074–1086, 2012.

[7] P. McCullagh, "Regression models for ordinal data," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 42, no. 2, pp. 109–142, 1980.

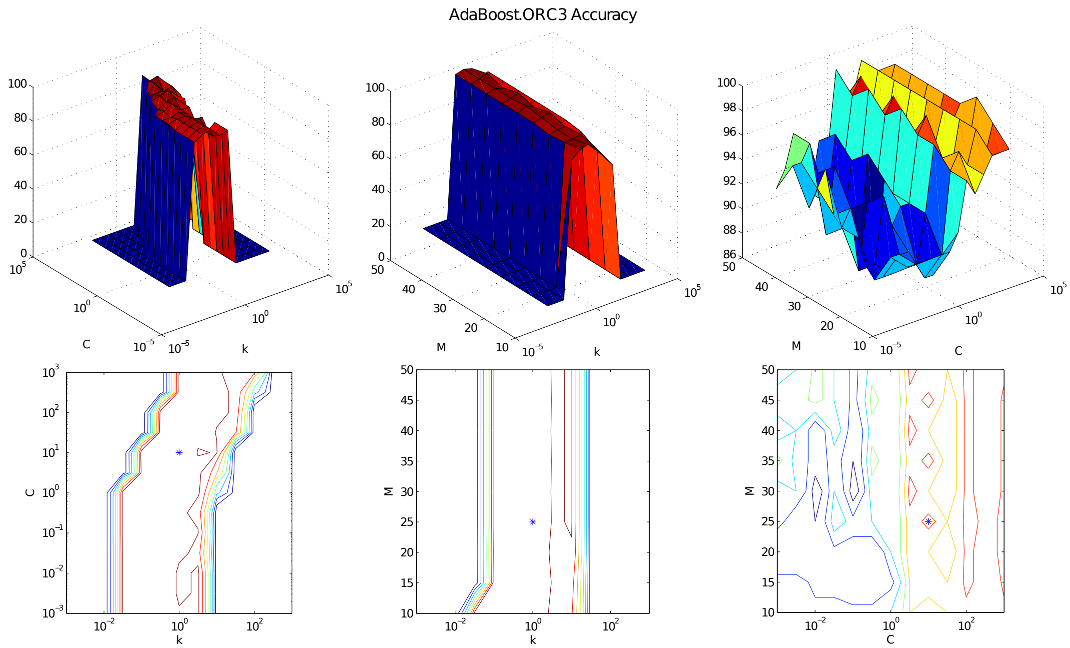


Fig. 5: Hypeparameters study on Acc for the AdaBoost(ELM).ORC3 algorithm and the parameters: M (ensemble size), C (regularization coefficient), k (width of the Gaussian kernel).

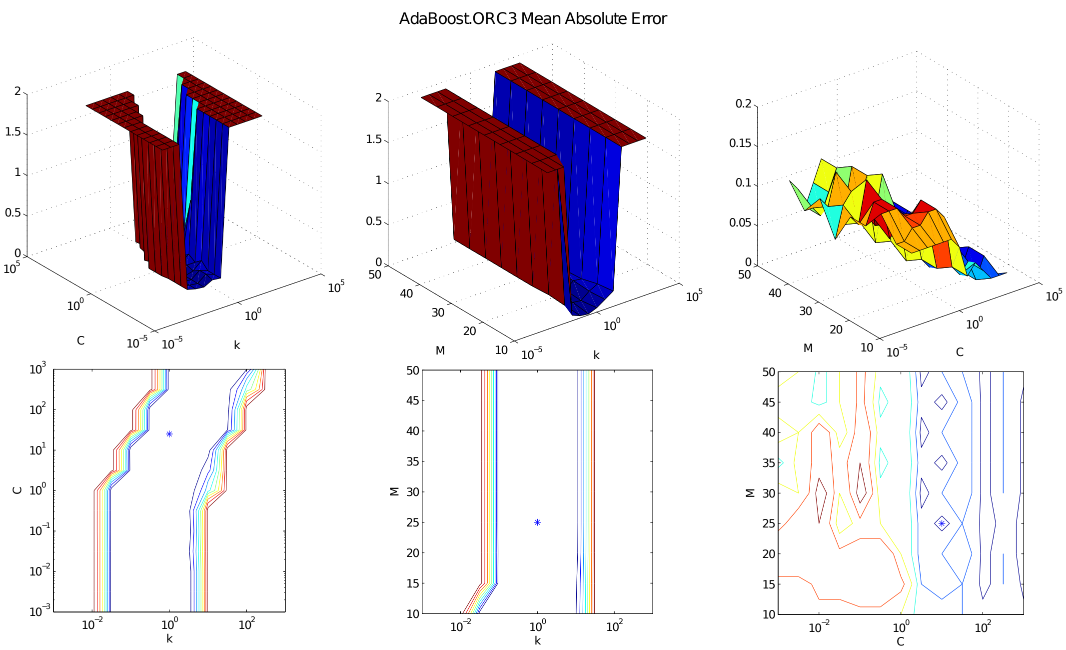


Fig. 6: Hypeparameters study on MAE for the AdaBoost(ELM).ORC3 algorithm and the parameters: M (ensemble size), C (regularization coefficient), k (width of the Gaussian kernel).

[8] J. A. Anderson, "Regression and ordered categorical variables," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 46, no. 1, pp. 1–30, 1984.

[9] M. J. Mathieson, "Ordinal models for neural networks," in *Proceedings of the Third International Conference on Neural Networks in the Capital Markets*, ser. Neural Networks in Financial Engineering, J. M. A.-P. N. Refenes, Y. Abu-Mostafa and A. Weigend, Eds. World Scientific, 1996, pp. 523–536.

[10] F. Fernández-Navarro, P. Campoy, M. De la Paz, C. Hervás-Martínez, and X. Yao, "Addressing the EU sovereign ratings using an ordinal regression approach," *IEEE Transaction on Cybernetics*, vol. 43, no. 6, pp. 2228–2240, 2013.

[11] R. Herbrich, T. Graepel, and K. Obermayer, "Large margin rank boundaries for ordinal regression," in *Advances in Large Margin Classifiers*, A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, Eds. Cambridge, MA: MIT Press, 2000, pp. 115–132.

[12] W. Chu and S. S. Keerthi, "Support Vector Ordinal Regression," *Neural Computation*, vol. 19, no. 3, pp. 792–815, 2007.

[13] W. Chu and Z. Ghahramani, "Gaussian processes for ordinal regression," *J. Mach. Learn. Res.*, vol. 6, pp. 1019–1041, Dec. 2005.

[14] W. Deng and L. Chen, "Color image watermarking using regularized extreme learning machine," *Neural Network World*, vol. 20, no. 3, pp. 317–330, 2010.

[15] D. Becerra-Alonso, M. Carbonero-Ruz, F. J. Martínez-Estudillo, and A. C. Martínez-Estudillo, "Evolutionary extreme learning machine for ordinal regression," in *Neural Information Processing*, ser. Lecture Notes

in Computer Science, 2012, vol. 7665, pp. 217–227.

[16] Q.-Y. Zhu, A. K. Qin, P. N. Suganthan, and G.-B. Huang, "Evolutionary extreme learning machine," *Pattern Recognition*, vol. 38, no. 10, pp. 1759–1763, 2005.

[17] Y. Liu, X. Yao, and T. Higuchi, "Ensembles with negative correlation learning," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 380–387, 2000.

[18] A. Chandra and X. Yao, "Divace: Diverse and accurate ensemble learning algorithm," in *Proceedings of the Fifth International Conference on intelligent Data Engineering and Automated learning*, vol. 3177. Exeter, UK: Lectures Notes and Computer Science, Springer, Berlin, 2005, pp. 619–625.

[19] X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active learning from stream data using optimal weight classifier ensemble," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 40, no. 6, pp. 1607–1621, 2010.

[20] D. Hernandez-Lobato, G. Martinez-Mu??oz, and A. Suarez, "Empirical analysis and evaluation of approximate techniques for pruning regression bagging ensembles," *Neurocomputing*, vol. 74, no. 12-13, pp. 2250 – 2264, 2011.

[21] A. L. Coelho and D. S. Nascimento, "On the evolutionary design of heterogeneous bagging models," *Neurocomputing*, vol. 73, no. 16-18, pp. 3319 – 3322, 2010.

[22] Z. Qi, Y. Xu, L. Wang, and Y. Song, "Online multiple instance boosting for object detection," *Neurocomputing*, vol. 74, no. 10, pp. 1769 – 1775, 2011.

[23] E. Frank and M. Hall, "A simple approach to ordinal classification," in *ECML'01*, 2001, pp. 145–156.

[24] W. Waegeman and L. Boullart, "An ensemble of weighted support vector machines for ordinal regression," *International Journal of Computer Systems Science and Engineering*, vol. 3, no. 1, pp. 47–51, 2009.

[25] F. Fernández-Navarro, P. Gutierrez, C. Hervás-Martínez, and X. Yao, "Negative correlation ensemble learning for ordinal regression," *IEEE Transaction on Neural Networks and Learning Systems*, vol. 24, no. 11, pp. 1836–1849, 2013.

[26] Y. Liu and X. Yao, "Negatively correlated neural networks can produce best ensembles," *Australian Journal of Intelligent Information Processing Systems*, vol. 4, no. 3, pp. 176–185, 1997.

[27] ——, "Ensemble learning via negative correlation," *Neural Networks*, vol. 12, no. 10, pp. 1399–1404, 1999.

[28] M. Pérez-Ortiz, P. Gutiérrez, and C. Hervás-Martínez, "Projection-based ensemble learning for ordinal regression," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1–1, 2013.

[29] H.-T. Lin and L. Li, "Large-margin thresholded ensembles for ordinal regression: theory and practice," in *Proceedings of the 17th international conference on Algorithmic Learning Theory*, ser. ALT'06. Springer-Verlag, 2006, pp. 319–333.

[30] ——, "Combining ordinal preferences by boosting," in *Proceedings of the ECML/PKDD 2009*, ser. Workshop on Preference Learning, 2009, pp. 69–83.

[31] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class adaboost," *Statistics and Its Interface*, vol. 2, no. 1, pp. 349–360, 2009.

[32] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[33] G.-B. Huang, D. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[34] G. Wang and P. Li, "Dynamic adaboost ensemble extreme learning machine," in *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, vol. 3, 2010, pp. V3–54–V3–58.

[35] H.-X. Tian and Z.-Z. Mao, "An ensemble elm based on modified adaboost.rt algorithm for predicting the temperature of molten steel in ladle furnace," *Automation Science and Engineering, IEEE Transactions on*, vol. 7, no. 1, pp. 73–80, 2010.

[36] J.-H. Zhai, H.-Y. Xu, and X.-Z. Wang, "Dynamic ensemble extreme learning machine based on sample entropy," *Soft Computing*, vol. 16, no. 9, pp. 1493–1502, 2012.

[37] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007.

[38] H.-T. Lin and L. Li, "Reduction from cost-sensitive ordinal ranking to weighted binary classification," *Neural Computation*, vol. 24, no. 5, pp. 1329–1367, 2012.

[39] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *IEEE International Conference on Neural Networks - Conference Proceedings*, vol. 2, 2004, pp. 985–990.

[40] C. Zhaohu, R. Xuemei, and C. Qiang, "Camera calibration based on extreme learning machine," in *Proceedings of the 2012 International Conference on Communication, Electronics and Automation Engineering*, ser. Advances in Intelligent Systems and Computing. Springer Berlin Heidelberg, 2013, vol. 181, pp. 115–120.

[41] H. Zhou, Y. Lan, Y. C. Soh, G.-B. Huang, and R. Zhang, "Credit risk evaluation with extreme learning machine," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, 2012, pp. 1064–1069.

[42] J. Sánchez-Monedero, M. Cruz-Ramírez, F. Fernández-Navarro, J. C. Fernández, P. A. Gutiérrez, and C. Hervás-Martínez, "On the suitability of extreme learning machine for gene classification using feature selection," in *ISDA '10: Proceedings of the 2010 International Conference on Intelligent Systems Design and Applications*. Cairo, Egypt: IEEE Computer Society, 2010, pp. 507–512.

[43] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489 – 501, 2006.

[44] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[45] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 1999.

[46] R. Zhang, G.-B. Huang, N. Sundararajan, and P. Saratchandran, "Multi-category classification using an extreme learning machine for microarray gene expression cancer diagnosis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 3, pp. 485–495, 2007.

[47] P. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *Information Theory, IEEE Transactions on*, vol. 44, no. 2, pp. 525–536, 1998.

[48] J. Cao, Z. Lin, and G.-b. Huang, "Composite function wavelet neural networks with extreme learning machine," *Neurocomputing*, vol. 73, no. 7-9, pp. 1405–1416, 2010.

[49] F. Fernández-Navarro, C. Hervás-Martínez, J. Sánchez-Monedero, and P. A. Gutierrez, "MELM-GRBF: A modified version of the extreme learning machine for generalized radial basis function neural networks," *Neurocomputing*, vol. 74, no. 16, pp. 2502–2510, 2011.

[50] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

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

[52] J. S. Cardoso and J. F. Pinto da Costa, "Learning to classify ordinal data: The data replication method," *J. Mach. Learn. Res.*, vol. 8, pp. 1393–1429, Dec. 2007.

[53] W.-Y. Deng, Q.-H. Zheng, S. Lian, L. Chen, and X. Wang, "Ordinal extreme learning machine," *Neurocomputing*, vol. 74, no. 1–3, pp. 447–456, 2010.

[54] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *Special Interest Group on Knowledge Discovery and Data Mining Explorer Newsletter*, vol. 11, pp. 10–18, November 2009.

[55] A. Castaño, F. Fernández-Navarro, and C. Hervás-Martínez, "PCA-ELM: A robust and pruned extreme learning machine approach based on principal component analysis," *Neural Processing Letters*, vol. 37, no. 3, pp. 377–392, 2013.

[56] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.

[57] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[58] Y. Hochberg and A. Tamhane, *Multiple Comparison Procedures*. John Wiley & Sons, 1987.

[59] P. A. Gutiérrez, M. Pérez-Ortiz, F. Fernández-Navarro, J. Sánchez-Monedero, and C. Hervás-Martínez, "An experimental study of different ordinal regression methods and measures," in *Hybrid Artificial Intelligent Systems*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7209, pp. 296–307.

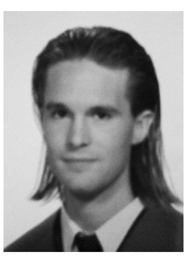
[60] G. Brown, J. L. Wyatt, and P. Tiňo, "Managing diversity in regression ensembles," *J. Mach. Learn. Res.*, vol. 6, pp. 1621–1650, Dec. 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1046920.1194899

[61] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1901–1914, 2011.

[62] H. Mohammed, J. Leander, M. Marbach, and R. Polikar, "Can AdaBoost.M1 learn incrementally? A comparison to Learn++ under different combination rules," in *Artificial Neural Networks – ICANN 2006*, ser. Lecture Notes in Computer Science, S. Kollias, A. Stafylopatis, W. Duch, and E. Oja, Eds. Springer Berlin Heidelberg, 2006, vol. 4131, pp. 254–263.

[63] M. D. Muhlbaier, A. Topalis, and R. Polikar, "Learn++.NC: Combining Ensemble of Classifiers With Dynamically Weighted Consult-and-Vote for Efficient Incremental Learning of New Classes," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 152–168, 2009.

Annalisa Riccardi was born in Monza, Italy, in 1983. She received a Bachelor Degree in Mathematics in 2005 and a Master Degree in Mathematics in 2008 from University of Milan, Italy. In 2012 she received the PhD degree in Applied Mathematics from the University of Bremen, Germany, working in the Optimization and Optimal Control Research Group, with a research focused on Multidisciplinary Design Optimization. She is currently a research fellow in Applied Mathematics and Computer Science at the Advanced Concept Team of the European Space Agency. Her main research interests are in the areas of global and local optimization techniques, multiobjective optimization, multidisciplinary optimization, neural networks, ordinal regression and parallel computing.



Francisco Fernández-Navarro was born in Córdoba, Spain, in 1984. He received the MsC degree in Computer Science from the University of Crdoba, Spain, in 2008, a MsC in Artificial Intelligence from the University of Málaga, Spain, in 2009 and the Ph. D. degree in Computer Science and Artificial Intelligence from the University of Málaga, Spain, in 2011. Currently, he is a Research Fellow in Computational Management at the European Space Agency (ESA), ESTEC. His current interests include radial basis functions neural networks, ordinal regression methods, imbalanced classification, evolutionary computation and hybrid algorithms.

Sante Carloni was born in Naples, Italy, in 1977. He obtained a B.S. in theoretical physics in 2000 from the University of Salerno (Italy) and in 2003 he achieved a PhD in physics of gravitation and astrophysics in the same university. He is currently Research fellow in Fundamental Physics in the Advanced Concepts Team of the European Space Agency. His main interests are in physics of gravitation and astrophysics. Recently his interests have extended to include the study of the geometrical foundations of machine learning algorithms.