Evolutionary Extreme Learning Machine for Ordinal Regression

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

Department of Management and Quantitative Methods, Universidad Loyola Andalucia, AYRNA Research Group
{dbecerra,mariano,fjmestud,acme}@etea.com

Abstract. This paper presents a novel method for generally adapting ordinal classification models. We essentially rely 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 assumption, this paper proposes an algorithm in two phases that takes advantage of the ordinal structure of the dataset and tries to translate this ordinal structure in the total ordered real line and then to rank the patterns of the dataset. The first phase 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. The results obtained in seven ordinal datasets are competitive in comparison with state-of-the-art algorithms in ordinal regression, but with much less computational time in datasets with many patterns.

Keywords: ordinal regression, ordinal classification, extreme learning machine, support vector machine, neural networks.

1 Introduction

Ordinal Regression (OR) is a supervised learning problem of predicting categories that have an ordered arrangement. The samples are labeled by a set of ranks with an ordering amongst different categories. In contrast to the nominal classification, there is an ordinal relationship throughout the categories and it is different from regression in that the number of ranks is finite and exact amounts of difference between ranks are not defined. In this way, ordinal classification lies somewhere between classification and regression.

OR problems are important, since they are common in our everyday life where many problems require classification of items into naturally ordered classes. Selecting the best route to work, where to stop, which product to buy, and where to live, are examples of daily ordinal decision-making. In this way, ordinal classification is one of the most important components in many applications.

Compared with general classification problems, much less effort has been devoted to ordinal classification learning. However, in the last decade an increasing number of publications report progress in the artificial learning of ordinal concepts [1,2,3,4,7,9,10,11,15,16,18].

T. Huang et al. (Eds.): ICONIP 2012, Part III, LNCS 7665, pp. 217–227, 2012.

Ordinal Classification opens a door that was not readily accessible to us in Nominal Classification: the possibility to somehow project the ordered classes onto a 1D real array. This is made possible by a reasonable a priori assumption [14]: the ordinal outputs to be classified must have a corresponding ordinality in the topology of the instance (or attribute's) space. Ordinal classes are generally expected to overlap in challenging classification problems. Noise and bad quality data are also expected. Yet in OR some degree of coherent gradual transition within the attibutes space is expected. By *coherent,* we mean that there is a correspondence with the a priori class labels provided by the dataset, and the inheretly continuous regressor.

We propose a method that performs OR, including both questionable and reliable patterns, with competitive results. This is basically done by helping the classifier understand the questionable patterns in a way that damps the noise made by them during the classification process.

The Evolutionary Ordinal Regression with Extreme Learning Machine (EOR-ELM) algorithm presented here has two stages. On *Phase 1*, the aim is to project each pattern on a one dimensional real interval in accordance with their classes. The ELM regressor, known to be fast and with good results, is used for this projection of patterns. ELM is used a number of times with different initial weights, in order to keep the one with most accuracy.

The second stage takes on projected values of this better ELM, and begins a sorting process where those patterns incorrectly classified, yet close to their correct class are reallocated in the hopes that this will help that better ELM the next time it performs a regression. An evolutionary criterion is used to reallocate such patterns. The distance to the new location takes place according to random additions or subtractions from the initial values of these patterns, as in random mutation. The new set of mutated and non-mutated projections undergoes ELM once more. Since the input weights for ELM were chosen in *Phase 1*, only the output weights are affected by this reallocation of projected patterns. Robustness is thus ensured, since the regressor remains not significantly altered.

Another important element to remark is how classes are assigned. Previously fixed intervals are not used in this case. Instead the separation between classes is obtained counting the number of patterns on each class. Thus, on this second stage the projections evolve, and so do the boundaries between classes.

Once the second stage is finished, due to a lack of improvement after a number of generations, the boundaries are fixed according to the best evolved classifier (that comes from the best evolved mutated projection).

Although the initial regression is applied to projected patterns within the interval $[0, 1]$, the system is allowed to take on as wide an interval as needed as it evolves in *Phase 2*. This allows for a greater flexibility in the classifier.

The EOR-ELM classifier turns out to be competitive against well-known ordinal classification methods, but most importantly, it provides us with an intuitive means to understand the data we are working with. It is a fast method, allowing for quick convergences in the evolutionary process explained in later sections.

Section 2 presents the nature of the problem, the regressor used (ELM), and how it is used for this particular kind of problem. Section 3 details how the experiments were carried out and their corresponding results. Section 4 ends the document with conclusions.

2 Problem Setup

2.1 The Approach

The center of our proposal is to turn a classification problem into a regression problem so that the class structures are reflected in the regression variable. Let us consider an ordinal classification problem with Q classes that we presume ordered by the class labels, i. e. $\mathcal{C}_1 \prec \mathcal{C}_2 \prec \ldots \prec \mathcal{C}_Q$, and the training set $\mathcal{D} = \left\{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in X \subset \mathbf{R}^k,\ y_i \in Y = \{\mathcal{C}_1, \ldots, \mathcal{C}_Q\}\right\}$ $(i = 1, \ldots . N)$ made by N patterns, where $\mathbf{x}$ is a characteristic random k-vector and y is the class it belongs to.

Our goal is to find a classifier that is capable of assigning, according to the best fit possible, a pattern to its class depending on its characteristics. It should also be designed to include the information related to the ordinality of the classes. Thus, the ordered structure of Y should be used to determine the classifier. This also implies that the order is somehow related to the distribution of patterns in the space of attributes X, and also to the topological distribution of the classes.

We assume the existence of a one-dimensional latent variable $z = \varphi(\mathbf{x})$ that is a function of the characteristics observed and takes on the underlying order mentioned in the previous paragraph.

While ordinality in classification datasets is allowing current researchers to present the problem as a regression where the intervals for each class are chosen one way or another (simple and intuitive), this procedure presents its own drawbacks. First of all, little has been said about how big these intervals must be. We do have the intuition that a given regressor with a given dataset should have an optimal interval for ordinal regression. However it is commonplace to see the interval (0,1) being chosen by default. This is not necessarily always convenient, and it can often make regression harder for a certain method. The second problem has to do with patterns falling close enough to their right interval.

As we can see in Figure 1, patterns may be preserving the ordinality that we are looking for, yet we might be classifying them incorrectly for the sole fact of being (close but) out of their correct interval. In other words, the regressor has done quite a good job ordering the patterns, yet we chose to evaluate it most negatively. In this manner, we are indeed presenting results that look clearly worse than they really are. It is the order of the classes that matters after all, and sticking to interval fitness might be making researchers lose touch with the real challenge (ordering patterns).

We will not use the value for z_j to see if it fits a certain interval. Instead, we will order and rank all the z_j. Then, knowing that class 1 in training had N_1 patterns, we look at the N_1 lowest values returned by the regressor for training. These values we assign to class 1 although they might not belong to it. We do

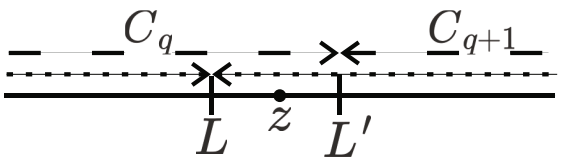

Fig. 1. The pattern projected as z belongs to class C_q. Still, a certain method has placed the boundary L before z, rendering this pattern as incorrectly classified. However, ordinality remains correct when z is immediate to L, since it is right next to its correct class. EOR-ELM proposes a way to define a boundary L' that solves this problem, considering z as correctly classified.

the same for the N_2 values after the lowest N_1 ones, and assign them to class 2, and so on with all the other classes.

2.2 Extreme Learning Machine (ELM)

Huang et al. [12], is the reference for the description of ELM. The regression problem can be formulated as an attempt to find solutions for $\mathbf{w}_i = (w_{i1}, \ldots, w_{in})$ and β_i using the following system of equations:

$$f(\mathbf{x}_j) = t_j, \quad j = 1, 2, \ldots, N \tag{1}$$

where

$$f(\mathbf{x}_j) = \sum_{i=1}^{m} \beta_i g(< \mathbf{w}_i, \mathbf{x}_j > + b_i), \quad j = 1, 2, \ldots, N, \tag{2}$$

where g is the activation function and the symbols $<>$ indicate an ordinary scalar product. This system can also be expressed more concisely as $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$, where $\mathbf{H}$ is the hidden layer's output matrix of the neural network given by:

$$\begin{aligned} &\mathbf{H}(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m, b_1, b_2, \ldots, b_m, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_N) = \\ &= \begin{pmatrix} g(< \mathbf{w}_1, \mathbf{x}_1 > + b_1) & \ldots & g(< \mathbf{w}_m, \mathbf{x}_1 > + b_m) \\ \vdots & \ddots & \vdots \\ g(< \mathbf{w}_1, \mathbf{x}_N > + b_1) & \ldots & g(< \mathbf{w}_m, \mathbf{x}_N > + b_m) \end{pmatrix} \end{aligned} \tag{3}$$

with

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_m \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}. \tag{4}$$

Each column on matrix $\mathbf{H}$ is made of the values of the corresponding hidden layer node, evaluated for each one of the patterns $\mathbf{x}_i$ in the training set.

The ELM algorithm randomly selects the values for $\mathbf{w}_i = (w_{i1}, \ldots, w_{in})$ and b_i, and then obtains corresponding values for $\beta_1, \ldots, \beta_m$, from the generalized linear model. This is done by calculating the minimum quadratic solution of the linear system, given by:

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T} \tag{5}$$

where $\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$ is the generalized Moore-Penrose inverse matrix.

In short, the corresponding algorithm for this method is as follows:

ELM Algorithm 1. *Given a training set* $D = \{(\mathbf{x}_i, t_i) : \mathbf{x}_i \in \mathbf{R}^n, t_i \in \mathbf{R}, i = 1, 2, \ldots, N\}$, *the activation function* $g(t)$, *and* m *neurons in the hidden layer:*
Step 1: Assign arbitrary input weights for w *and bias* b.
Step 2: Calculate the hidden layer output matrix $\mathbf{H}$.
Step 3: Calculate the output weights $\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}$.

The ELM algorithm has been shown to have a good generalisation capability while it significantly reduces the time needed to train the neural network. For more details on this method the reader can see: [12] and [13].

2.3 The EOR-ELM Algorithm

The EOR-ELM procedure is enumerated as follows:

1. Before the first phase, partition the dataset making sure all classes are almost (at least) equally represented to be used as training and test subsets. Let N be the total number of patterns in the training set. The first N_1 patterns belong to class 1, the next N_2 belong to class 2, and so on until the last N_Q patterns that correspond to class Q. The following will refer to this set.
2. *Phase 1:* A real value is assigned to each one of the patterns. Although these values must be coherent with the order of the classes, there is no a priori information about the order within each class. In order to comply with both interclass labelling and intraclass uncertainty, let us consider the accumulated values $S_0 = 0$, $S_q = \sum_1^q N_i$ and $m_q = \frac{1}{2N}(S_{q-1} + S_q)$ where $z_i = m_{q(i)}$, for $q(i) = j$ if $S_{j-1} < i \leqslant S_j$ is assigned. The class that corresponds to each pattern can still be identified according to the arrangement defined in step 1. Thus, the greater z_i correspond to the highest labelled classes (interclass ordinality), and all z_i inside a class are the same (intraclass uncertainty). Indeed, we intend to perform a regression.
3. Invoke a regressor (we use ELM) to be trained according to the new training subset $\{(\mathbf{x}_i, z_i) \;\; i = 1, \ldots, N\}$. ELM will be used M times, each one with different random input weights w. Let φ_w be the regressor and let each pattern be transformed according to $\hat{z}_i = \varphi_w(\mathbf{x}_i)$. Once these outputs are sorted in increasing order, let $r(i)$ be the rank for $\hat{z}_i$. $C(w) = \frac{1}{N}\sum \delta(q(i), q(r(i)), 0)$ where

$$\delta(i, j, k) = \begin{cases} 1 & |i - j| \leqslant k \\ 0 & |i - j| > k \end{cases}$$

is calculated. Thus C becomes the CCR of the classifier, based on regressor φ_w, that assigns patterns according to the order of its output. Of all the M regressors ran, the one with the highest C is chosen, and called φ.

4. *Phase 2:* An iterated process to improve the regression selected on step 3 is started. Let $k = 0$.
5. Let $z_i^k = \hat{z}_i$. From these values, the interclass boundaries are definde as $L_0^k = z_{(1)}^k$, $L_q^k = \frac{z_{(S_q)}^k + z_{(S_{q+1})}^k}{2}$, $L_Q^k = z_{(N)}^k$ y $m_q^k = \frac{L_q^k + L_{q-1}^k}{2}$, where $z_{(i)}$ is the i-th ranked pattern.
6. The values

$$z_i = z_i^k + \left(\delta\left(q\left(i\right), q\left(r\left(i\right)\right), 1\right) - \right.$$
$$\left. -\delta\left(q\left(i\right), q\left(r\left(i\right)\right), 0\right)\right) \delta\left(i, r\left(i\right), n\right) \left(m_{q(i)}^k - z_i^k\right) \upsilon_i,\ \upsilon_i \in U\left(0,1\right)$$

are obtained. Here, incorrectly classified borderline (but only n or less units away from the boundary of their class) patterns are randomly reallocated towards the center of the interval of the class they belong to. Incorrectly classified patterns beyond this point are not reallocated, nor are correctly classified patterns. The aim here is to try and reallocate only those patterns close to their correct classification, counting on the inherent continuity of the regressor to not significantly alter those patterns correctly classified (see Figure 2).

7. Accuracy for training is obtained from doing regression to this new z_i class-mutated dataset. The new ELM regressor retains the input weights w obtained in step 3, only changing the output weights β.
8. The accuracies of the original and mutated regressor are compared. The best one is kept, and the other discarded.
9. If the new regressor is chosen, we return to step 5 with $k = k+1$ y $\hat{z}_i = \varphi(\mathbf{x}_i)$. Otherwise we go back to step 6.
10. The stop condition is simply the lack of improvement in accuracy for more than G generations of comparing regressors.
11. The final classifier is

$$\phi\left(\mathbf{x}\right) = q \ \text{ if } L_{q-1} < \varphi\left(\mathbf{x}\right) \leqslant L_q$$

where φ and the boundaries L are the obtained for the regressor at the stop condition, except for $L_1 = -\infty$ y $L_Q = \infty$.

12. The efficiency of the regressor is verified on the test subset, this time only using the boundaries obtained to differentiate classes.

3 Experiments

3.1 Ordinal Classification Datasets and Experimental Design

Up to the author's knowledge, there are no public specific datasets repositories for ordinal classification. The most used dataset repository in the literature is the *ordinal regression benchmark datasets* provided by Chu et. al [4]. However,

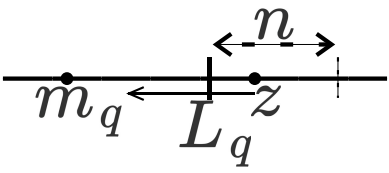

Fig. 2. When and incorrectly classified z ranks among the first n patterns past L_q it can be randomly mutated towards the center of the interval of the class it belongs to (m_q)

Table 1. Datasets used for the experiments

Dataset	Size	#Input	#Classes	Classes Distribution
automobile	205	71	6	(3,22,67,54,32,27)
balance-scale	625	4	3	(288,49,288)
ERA	1000	4	9	(92,142,181,172,158,118,88,31,18)
LEV	1000	4	5	(93,280,403,197,27)
newthyroid	215	5	3	(150,35,30)
SWD	1000	10	4	(32,352,399,217)
tae	151	54	3	(49,50,52)

the benchmark datasets provided by Chu et. al, are not real ordinal classification datasets but regression problems. These datasets are turned from a regression problem into a classification problem by discretizing the target variable into r different bins, with equal frequency or equal width, so each bin is labeled as a different ordinal class.

We have collected a set of real ordinal classification datasets which are publicly available at the UCI repository [8] and at the *mldata.org* datasets repository [17] (see Table 1 for datasets description).

For this method, we perform 10 times a holdout validation and 3 repetitions for each holdout (obtaining a total of $10 \times 3 = 30$ different results). Each holdout is a stratified random division of the data, where approximately 75% of the instances are used for the training set and 25% of them for the test set (maintaining the original distribution of classes for both sets). For the deterministic methods (all of them except EOR-ELM), we perform 30 times a stratified holdout validation using 75% of the instances for the training set and 25% of them for the generalization set, what implies a total of 30 different results. The partitions are the same for all the deterministic methods.

In this way, a total of 30 error measures has been obtained for all the methods compared, which guarantees a proper statistical significance of the results.

3.2 Machine Learning Methods Used for Comparison Purposes

For comparison purposes, different state-of-the-art methods have been included in the experimentation. These methods are the following:

- **Gaussian Processes for Ordinal Regression (GPOR)** by Chu et. al [4], presents a probabilistic kernel approach to ordinal regression based on Gaussian processes where a threshold model that generalizes the *probit* function is used as the likelihood function for ordinal variables.
- **Support Vector Ordinal Regression (SVOR)** by Chu et. al [5,6], proposes two new support vector approaches for ordinal regression. Here, multiple thresholds are optimized in order to define parallel discriminant hyperplanes for the ordinal scales. The first approach with explicit inequality constraints on the thresholds, derive the optimal conditions for the dual problem, and adapt the SMO algorithm for the solution, and we will refer to it as SVOR-EX. In the second approach, the samples in all the categories are allowed to contribute errors for each threshold, therefore there is no need of including the inequality constraints in the problem. This approach is named a SVOR with implicit constraints (SVOR-IM).

Regarding the algorithms' hyper-parameters, the following procedure has been applied. For the Support Vector algorithms, i.e. SVOR-EX and SVOR-IM, the corresponding hyper-parameters (regularization parameter, C, and width of the Gaussian functions, γ), were adjusted using a grid search with a 10-fold cross-validation, considering the following ranges: $C \in \{10^3, 10^1, \ldots, 10^{-3}\}$ and $\gamma \in \{10^3, 10^0, \ldots, 10^{-3}\}$. All the methods were configured to use the Gaussian kernel.

EOR-ELM was run under $M = 300, n = 5, G = 500$. Same order variations of these parameters do not return significant differences in the output.

3.3 Ordinal Classification Evaluation Metrics

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 $\{y_1, y_2, \ldots, y_N\}$:

1. Mean Zero-one Error (MZE) is simply the fraction of incorrect predictions on individual samples. Accuracy (CCR) measures the correct ones against the entire dataset:

$$MZE = \frac{1}{N} \sum_{i=1}^{N} I\left(\hat{y}_i \neq y_i\right), \tag{6}$$

$$CCR = 1 - MZE, \tag{7}$$

   where $I(\cdot)$ is the zero-one loss function and N is the number of patterns of the dataset.
2. Mean Absolute Error (MAE) is the average deviation of the prediction from the true targets, i.e.:

$$MAE = \frac{1}{N} \sum_{i=1}^{N} \left|\mathcal{O}(\hat{y}_i) - \mathcal{O}(y_i)\right|, \tag{8}$$

   where $\mathcal{O}(\mathcal{C}_q) = q, 1 \leq q \leq Q$, i.e. $\mathcal{O}(y_i)$ is the order of class label y_i.

These measures are aimed to evaluate two different aspects that can be taken into account when an ordinal regression problem is considered: whether the patterns are generally well classified (CCR) and whether the classifier tends to predict a class as close to the real class as possible (MAE).

Table 2. CCR, MAE and computational time (in seconds), for each dataset and method

Dataset	Method	CCR	MAE	$time/s$
automobile	GPOR	0.4000±0.0633	1.0153±0.1060	26.55
	SVOREX	**0.6788**±0.0608	**0.3788**±0.0821	3.88
	SVORIM	0.6596±0.0573	0.4019±0.0776	**3.77**
	EOR-ELM	0.6677±0.0674	0.3812±0.0789	11.12
balance-scale	GPOR	**0.9726**±0.0095	0.0273±0.0095	978.88
	SVOREX	0.9994±0.0020	**0.0006**±0.0020	111.33
	SVORIM	0.9994±0.0020	**0.0006**±0.0020	38.55
	EOR-ELM	0.9657±0.0197	0.0474±0.0242	**18.89**
ERA	GPOR	0.2812±0.0253	1.2188±0.0732	1356.78
	SVOREX	0.2856±0.0286	1.1804±0.0646	2223.56
	SVORIM	0.2520±0.0184	1.2032±0.0504	3428.78
	EOR-ELM	**0.2965**±0.0186	**1.1535**±0.0525	**36.43**
LEV	GPOR	0.6080±0.0275	0.4172±0.0302	1643.78
	SVOREX	0.6140±0.0293	0.4136±0.0308	1966.33
	SVORIM	0.6124±0.0341	0.4168±0.0340	2137.00
	EOR-ELM	**0.6320**±0.0292	**0.4040**±0.0363	**28.22**
newthyroid	GPOR	0.9463±0.0320	0.0537±0.0320	54.67
	SVOREX	**0.9556**±0.0199	**0.0444**±0.0199	2.89
	SVORIM	0.9538±0.0200	0.0462±0.0200	**1.56**
	EOR-ELM	0.9259±0.0437	0.0926±0.0350	15.07
SWD	GPOR	0.5724±0.0279	0.4464±0.0347	701.56
	SVOREX	0.5660±0.0254	0.4500±0.0301	1026.78
	SVORIM	0.5708±0.0219	0.4448±0.0242	2270.22
	EOR-ELM	**0.5880**±0.0117	**0.4440**±0.0263	**31.40**
tae	GPOR	0.3132±0.0487	0.9736±0.1574	**4.89**
	SVOREX	0.6079±0.0518	0.4421±0.0605	12.44
	SVORIM	0.5974±0.0496	0.4552±0.0582	14.00
	EOR-ELM	**0.6271**±0.0529	**0.4291**±0.0729	13.68

3.4 Results

The model presented in this paper (EOR-ELM), is compared with the ones in Subsection 3.2. The experiments have been carried out by following the experimental design described in Subsections 3.1 and 3.3. Results considering mean and standard deviation in CCR and MAE are showed in Table 2. The best statistical result for each dataset is in bold face. EOR-ELM returns better results in 4 of the 7 datasets. It is particularly competitive in datasets where good classification is hard to obtain. Although EOR-ELM spends most of its computational

Table 3. CCR improvement from *Phase 1* to *Phase 2*

Dataset	*Phase 1*	*Phase 2*
automobile	0.6526	0.6677
balance-scale	0.9586	0.9657
ERA	0.2817	0.2965
LEV	0.6096	0.6320
newthyroid	0.9174	0.9259
SWD	0.5607	0.5880
tae	0.5824	0.6271

time on *Phase 1*, 3 out of 7 datasets were classified faster when EOR-ELM was used.

The relevance of *Phase 2* (steps 4 to 11 in section 2.3) becomes apparent when the increase in accuracy with respect to *Phase 1* is taken into account: a 2.32% increase for automobile, 0.74% for balance-scale, 5.26% for ERA, 3.67% for LEV, 0.92% for newthyroid, 4.87% for SWD, and 7.67% for tae. ELM is a good classifier on its own; EOR-ELM is good at refining the best results provided by ELM (see Table 3). This also indicates the degree of usefulness this method has for each one of the datasets: little use when accuracy is already too close to perfect, and greater on particularly difficult datasets.

4 Conclusions

A novel method for supervised ordinal classification was presented. The approach essentially relies on the assumption that the ordinal structure of the set of class labels is also reflected in the topology of the instance space. A continuous function that projected the instance space onto a real 1D interval (while preserving the ordinality of data) was approximated. This projection of data was carried out in two phases: first, the instance space was projected. Second, the thresholds that define the non-overlapping intervals were determined in an evolutionary process where misclassified patterns near the boundary of their correct class are mutated. The base regressor considered in the process was the Extreme Learning Machine algorithm. This algorithm was able to efficiently project the feature space. The evolutionary tuning process later carried out allowed us to improve the performance of the algorithm. The experimental results obtained show that our algorithm is competitive in comparison with state-of-the-art algorithms in ordinal regression, but with a significant improvement in computational time for datasets with many patterns.

Acknowledgments. This work was supported in part by the Spanish Inter-Ministerial Commission of Science and Technology under Project TIN2011-22794, the European Regional Development fund, and the "Junta de Andaluc ía" (Spain), under Project P2011-TIC-7508.

References

1. Baccianella, S., Esuli, A., Sebastiani, F.: Evaluation Measures for Ordinal Regression. In: Proceedings of the Ninth International Conference on Intelligent Systems Design and Applications (ISDA 2009), pp. 283–287 (2009)
2. Cardoso, J.S., Pinto da Costa, J.F.: Learning to Classify Ordinal Data: The Data Replication Method. Journal of Machine Learning Research 8, 1393–1429 (2007)
3. Cardoso, J.S., Pinto da Costa, J.F., Cardoso, M.J.: Modelling Ordinal Relations with SVMs: an Application to Objective Aesthetic Evaluation of Breast Cancer Conservative Treatment. Neural Networks 18(5-6), 808–817 (2005)
4. Chu, W., Ghahramani, Z.: Gaussian Processes for Ordinal Regression. Journal of Machine Learning Research 6, 1019–1041 (2005)
5. Chu, W., Sathiya Keerthi, S.: New Approaches to Support Vector Ordinal Regression. In: ICML 2005: Proceedings of the 22nd International Conference on Machine Learning, pp. 145–152 (2005)
6. Chu, W., Sathiya Keerthi, S.: Support Vector Ordinal Regression. Neural Computation 19(3), 792–815 (2007)
7. Pinto da Costa, J.F., Alonso, H., Cardoso, J.S.: The Unimodal Model for the Classification of Ordinal Data. Neural Networks 21(1), 78–91 (2008)
8. Frank, A., Asuncion, A.: UCI Machine Learning Repository (2010), http://www.ncbi.nlm.nih.gov
9. Frank, E., Hall, M.: A Simple Approach to Ordinal Classification. In: Flach, P.A., De Raedt, L. (eds.) ECML 2001. LNCS (LNAI), vol. 2167, pp. 145–156. Springer, Heidelberg (2001)
10. Hawkes, R.K.: The Multivariate Analysis of Ordinal Measures. The American Journal of Sociology 76(5), 908–926 (1971)
11. Herbrich, R., Graepel, T., Obermayer, K.: Large Margin Rank Boundaries for Ordinal Regression. In: Smola, A.J., Bartlett, P.L., Schölkopf, B., Schuurmans, D. (eds.) Advances in Large Margin Classifiers, pp. 115–132. MIT Press, Cambridge (2000)
12. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. In: IEEE International Conference on Neural Networks - Conference Proceedings, pp. 985–990 (2004)
13. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme Learning Machine: Theory and Applications. Neurocomputing 70(1-3), 489–501 (2006)
14. Hühn, J.C., Hüllermeier, E.: Is an Ordinal Class Structure Useful in Classifier Learning? International Journal of Data Mining, Modelling and Management 1(1), 45–67 (2008)
15. Kramer, S., Widmer, G., Pfahringer, B., de Groeve, M.: Prediction of Ordinal Classes Using Regression Trees. In: Ohsuga, S., Raś, Z.W. (eds.) ISMIS 2000. LNCS (LNAI), vol. 1932, pp. 426–434. Springer, Heidelberg (2000)
16. Li, L., Lin, H.-T.: Ordinal Regression by Extended Binary Classification. Advances in Neural Information Processing Systems 19, 865–872 (2007)
17. Sonnenburg, S.: Machine Learning Data Set Repository (2011), http://mldata.org/
18. Tutz, G., Hennevogl, W.: Random Effects in Ordinal Regression Models. Computational Statistics & Data Analysis 22(5), 537–557 (1996)