Ordinal Classification with Decision Rules

Krzysztof Dembczyński[1], Wojciech Kotłowski[1], and Roman Słowiński[1,2]

[1] Institute of Computing Science, Poznań University of Technology,
60-965 Poznań, Poland
{kdembczynski, wkotlowski, rslowinski}@cs.put.poznan.pl
[2] Institute for Systems Research, Polish Academy of Sciences, 01-447 Warsaw, Poland

Abstract. We consider the problem of ordinal classification, in which a value set of the decision attribute (output, dependent variable) is finite and ordered. This problem shares some characteristics of multi-class classification and regression, however, in contrast to the former, the order between class labels cannot be neglected, and, in the contrast to the latter, the scale of the decision attribute is not cardinal. In the paper, following the theoretical framework for ordinal classification, we introduce two algorithms based on gradient descent approach for learning ensemble of base classifiers being decision rules. The learning is performed by greedy minimization of so-called threshold loss, using a forward stagewise additive modeling. Experimental results are given that demonstrate the usefulness of the approach.

1 Introduction

In the prediction problem, the aim is to predict the unknown value of an attribute y (called *decision attribute*, *output* or *dependent variable*) of an object using known joint values of other attributes (called *condition attributes*, *predictors*, or *independent variables*) $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. In the *ordinal classification*, it is assumed that $y = \{r_1, \ldots, r_K\}$, with r_k, $k \in \mathcal{K} = \{1, \ldots, K\}$, being K distinct and ordered class labels $r_K \succ r_{K-1} \succ \ldots \succ r_1$, where $\succ$ denotes the ordering relation between labels. Let us assume in the following, without loss of generality, that $r_k = k$. This problem shares some characteristics of multi-class classification and regression. A value set of y is finite, but in contrast to the multi-class classification, the order between class labels cannot be neglected. The values of y are ordered, but in contrast to regression, the scale of y is not cardinal. Such a setting of the prediction problem is very common in real applications. For example, in recommender systems, users are often asked to evaluate items on five value scale (see Netflix Prize problem [16]). Another example is the problem of email classification to ordered groups, like: "very important", "important", "normal", and "later".

The problem of ordinal classification is often solved by multi-class classification or regression methods. In recent years, however, some new approaches tailored for ordinal classification were introduced [13, 6, 7, 18, 17, 3, 14, 15]. In this paper, we take first a closer look at the nature of ordinal classification. Later

on, we introduce two novel algorithms based on gradient descent approach for learning ensemble of base classifiers. The learning is performed by greedy minimization of so-called threshold loss [17] using a forward stagewise additive modeling [12]. As a base classifier, we have chosen single decision rule which is a logical expression having the form: if [*conditions*], *then* [*decision*]. This choice is motivated by simplicity and ease in interpretation of decision rule models. Recently, one can observe a growing interest in decision rule models for classification purposes (e.g. such algorithms like SLIPPER [5], LRI [19], RuleFit [11], ensemble of decision rules [1, 2]).

Finally, we report experimental results that demonstrate the usefulness of the proposed approach for ordinal classification. In particular our approach is competitive to traditional regression and multi-class classification methods, and also to existing ordinal classification methods.

2 Statistical Framework for Ordinal Classification

Similarly to classification and regression, the task is to find a function $F(\mathbf{x})$ that predicts accurately an ordered label of y. The optimal prediction function (or Bayes optimal decision) is given by:

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} E_{y\mathbf{x}} L(y, F(\mathbf{x})) \tag{1}$$

where the expected value $E_{y\mathbf{x}}$ is over joint distribution of all variables $P(y, \mathbf{x})$ for the data to be predicted. $L(y, F(\mathbf{x}))$ is a loss or cost for predicting $F(\mathbf{x})$ when the actual value is y. $E_{y\mathbf{x}} L(y, F(\mathbf{x}))$ is called *prediction risk* or *expected loss*. Since $P(y, \mathbf{x})$ is generally unknown, the learning procedure uses only a set of training examples $\{y_i, \mathbf{x}_i\}_1^N$ to construct $F(\mathbf{x})$ to be the best possible approximation of $F^*(\mathbf{x})$. Usually, it is performed by minimization of *empirical risk*:

$$R_e = \frac{1}{N} \sum_{i=1}^{N} L(y_i, F(\mathbf{x}_i)).$$

Let us remind that the typical loss function in binary classification (for which $y \in \{-1, 1\}$) is 0-1 loss:

$$L_{0-1}(y, F(\mathbf{x})) = \begin{cases} 0 & \text{if } y = F(\mathbf{x}), \\ 1 & \text{if } y \neq F(\mathbf{x}), \end{cases} \tag{2}$$

and in regression (for which $y \in \mathbb{R}$), it is squared-error loss:

$$L_{se}(y, F(\mathbf{x})) = (y - F(\mathbf{x}))^2. \tag{3}$$

One of the important properties of the loss function is a form of prediction function minimizing the expected risk $F^*(\mathbf{x})$, sometimes called *population minimizer* [12]. In other words, it is an answer to a question: what does a minimization of expected loss estimate on a population level? Let us remind that the population minimizers for 0-1 loss and squared-error loss are, respectively:

$$F^*(\mathbf{x}) = \operatorname{sgn}\left(\Pr(y = 1|\mathbf{x}) - 0.5\right), \qquad F^*(\mathbf{x}) = E_{y|\mathbf{x}}(y).$$

Table 1. Commonly used loss functions and their population minimizers

Loss function	Notation	$L(y, F(\mathbf{x}))$	$F^*(\mathbf{x})$
Binary classification, $y \in \{-1, 1\}$:			
Exponential loss	L_{exp}	$\exp(-y \cdot F(\mathbf{x}))$	$\frac{1}{2}\log\frac{\Pr(y=1\|\mathbf{x})}{\Pr(y=-1\|\mathbf{x})}$
Deviance	L_{dev}	$\log(1+\exp(-2 \cdot y \cdot F(\mathbf{x})))$	$\frac{1}{2}\log\frac{\Pr(y=1\|\mathbf{x})}{\Pr(y=-1\|\mathbf{x})}$
Regression, $y \in \mathbb{R}$:			
Least absolute deviance	L_{lad}	$\|y - F(\mathbf{x})\|$	$\text{median}_{y\|\mathbf{x}}(y)$

Apart from 0-1 and squared error loss, some other important loss functions are considered. Their definitions and population minimizers are given in Table 1.

In ordinal classification, one minimizes prediction risk based on the $K \times K$ loss matrix:

$$L_{K\times K}(y, F(\mathbf{x})) = [l_{ij}]_{K\times K} \tag{4}$$

where $y, F(\mathbf{x}) \in \mathcal{K}$, and $i = y, j = F(\mathbf{x})$. The only constraints that (4) must satisfy in ordinal classification problem are the following, $l_{ii} = 0, \forall i$, $l_{ik} \geq l_{ij}, \forall\ k \geq j \geq i$, and $l_{ik} \geq l_{ij}, \forall\ k \leq j \leq i$. Observe that for

$$l_{ij} = 1, \quad \text{if } i \neq j, \tag{5}$$

loss matrix (4) boils down to the 0-1 loss for ordinary multi-class classification problem. One can also simulate typical regression loss functions, such as least absolute deviance and squared-error, by taking:

$$l_{ij} = |i - j|, \tag{6}$$

$$l_{ij} = (i - j)^2, \tag{7}$$

respectively. It is interesting to see, what are the population minimizers of the loss matrices (5)-(7). Let us observe that we deal here with the multinomial distribution of y, and let us denote $\Pr(y = k|\mathbf{x})$ by $p_k(\mathbf{x})$. The population minimizer is then defined as:

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} \sum_{k=1}^{K} p_k(\mathbf{x}) \cdot L_{K\times K}(k, F(\mathbf{x})). \tag{8}$$

For loss matrices (5)-(7) we obtain, respectively:

$$F^*(\mathbf{x}) = \arg\max_{k\in\mathcal{K}} p_k(\mathbf{x}), \tag{9}$$

$$F^*(\mathbf{x}) = \text{median}_{p_k(\mathbf{x})}(y) = \text{median}_{y|\mathbf{x}}(y), \tag{10}$$

$$F^*(\mathbf{x}) = \sum_{k=1}^{K} k \cdot p_k(\mathbf{x}) = E_{y|\mathbf{x}}(y). \tag{11}$$

In (11) it is assumed that the range of $F(\mathbf{x})$ is a set of real values.

The interesting corollary from the above is that in order to solve ordinal classification problem one can use any multi-class classification method that estimates $p_k(\mathbf{x})$, $k \in \mathcal{K}$. This can be, for example, logistic regression or gradient boosting machine [9]. A final decision is then computed according to (8) with respect to chosen loss matrix. For (5)-(7) this can be done by computing mode, median or average over y with respect to estimated $p_k(\mathbf{x})$, respectively. For loss matrix entries defined by (7) one can use any regression method that aims at estimating $E_{y|\mathbf{x}}(y)$. We refer to such an approach as *simple ordinal classifier*.

Let us notice that multi-class classification problem is often solved as K (one class against $K-1$ classes) or $K \times (K-1)$ (one class against one class) binary problems. However, taking into account the order on y, we can solve the ordinal classification by solving $K-1$ binary classification problems. In the k-th ($k = 1, \ldots, K-1$) binary problem, objects for which $y \leq k$ are labeled as $y' = -1$ and objects for which $y > k$ are labeled as $y' = 1$. Such an approach has been used in [6].

The ordinal classification problem can also be formulated from a value function perspective. Let us assume that there exists a latent value function that maps objects to scalar values. The ordered classes correspond to contiguous intervals on a range of this function. In order to define K intervals, one needs $K+1$ thresholds: $\theta_0 = -\infty < \theta_1 < \ldots < \theta_{K-1} < \theta_K = \infty$. Thus k-th class is determined by $(\theta_{k-1}, \theta_k]$. The aim is to find a function $F(\mathbf{x})$ that is possibly close to any monotone transformation of the latent value function and to estimate thresholds $\{\theta_k\}_1^{K-1}$. Then, instead of the loss matrix (4) one can use immediate-threshold or all-threshold loss [17] defined respectively as:

$$L^{imm}(y, F(\mathbf{x})) = L(1, F(\mathbf{x}) - \theta_{y-1}) + L(-1, F(\mathbf{x}) - \theta_y), \tag{12}$$

$$L^{all}(y, F(\mathbf{x})) = \sum_{k=1}^{y-1} L(1, F(\mathbf{x}) - \theta_k) + \sum_{k=y}^{K-1} L(-1, F(\mathbf{x}) - \theta_k). \tag{13}$$

In the above, $L(y, f)$ is one of the standard binary classification loss functions. When using exponential or deviance loss, (12) and (13) become continuous and convex functions that are easy to minimize.

There is, however, a problem with interpretation what does minimization of expected threshold loss estimate. Only in the case when 0-1 loss is chosen as the basis of (12) and (13), the population minimizer has a nice interpretable form. For (12), we have:

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} \sum_{k=1}^{K} p_k(\mathbf{x}) \cdot L_{0-1}^{imm}(k, F(\mathbf{x})) = \arg\max_{k \in \mathcal{K}} p_k(\mathbf{x}), \tag{14}$$

and for (13), we have:

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} \sum_{k=1}^{K} p_k(\mathbf{x}) \cdot L_{0-1}^{all}(k, F(\mathbf{x})) = \text{median}_{y|\mathbf{x}}(y). \tag{15}$$

An interesting theoretical result is obtained in [15], where (12) and (13) are used in derivation of the upper bound of generalization error for any loss matrix (4).

Threshold loss functions were already considered in building classifiers. In [17] the classifier was learned by conjugate gradient descent. Among different base loss functions, also deviance was used. In [18, 3, 15], a generalization of SVM (support vector machines) was derived. The algorithm based on AdaBoost [8] was proposed in [15]. In the next section, we present two algorithms based on forward stagewise additive modeling. The first one is an alternative boosting formulation for threshold loss functions. The second one is an extension of the gradient boosting machine [9].

Let us remark at the end of our theoretical considerations that (13) can also be formulated as a specific case of so-called rank loss [13, 7, 4]:

$$L_{rank}\big(y_1, y_2, F(\mathbf{x}_1), F(\mathbf{x}_2)\big) = L\big(\mathrm{sgn}(y_1 - y_2), F(\mathbf{x}_1) - F(\mathbf{x}_2)\big). \qquad (16)$$

This loss function requires that all objects are compared pairwise. Assuming that thresholds $\{\theta_k\}_1^{K-1}$ are values of $F(\mathbf{x})$ for some virtual objects/profiles and all other objects are compared only with these virtual profiles, one obtains (13). Rank loss was used in [13] to introduce a generalization of SVM for ordinal classification problems, and in [7], an extension of AdaBoost for ranking problems was presented. The drawback of this approach is the complexity of empirical risk minimization defined by rank loss that grows quadratically with the problem size (number of training examples). For this reason we do not use this approach in our study.

3 Ensemble of Decision Rules for Ordinal Classification

The introduced algorithms generating an ensemble of ordinal decision rules are based on forward stagewise additive modeling [12]. The decision rule being the base classifier is a logical expression having the form: *if* [*conditions*], *then* [*decision*]. If an object satisfies conditions of the rule, then the suggested decision is taken. Otherwise no action is performed. By conditions we mean a conjunction of expressions of the form $x_j \in S$, where S is a value subset of j-th attribute, $j \in \{1, \ldots, n\}$. Denoting set of conditions by Φ and decision by α, the decision rule can be equivalently defined as:

$$r(\mathbf{x}, \mathbf{c}) = \begin{cases} \alpha & \text{if } \mathbf{x} \in cov(\Phi), \\ 0 & \text{if } \mathbf{x} \notin cov(\Phi), \end{cases} \qquad (17)$$

where $\mathbf{c} = (\Phi, \alpha)$ is a set of parameters. Objects that satisfy Φ are denoted by $cov(\Phi)$ and referred to as cover of conditions Φ.

The general scheme of the algorithm is presented as Algorithm 1. In this procedure, $F_m(\mathbf{x})$ is a real function being a linear combination of m decision rules $r(\mathbf{x}, \mathbf{c})$, $\{\theta_k\}_1^{K-1}$ are thresholds and M is a number of rules to be generated. $L^{all}(y_i, F(\mathbf{x}))$ is an all-threshold loss function. The algorithm starts with $F_0(\mathbf{x}) = 0$ and $\{\theta_k\}_1^{K-1} = 0$. In each iteration of the algorithm, function $F_{m-1}(\mathbf{x})$ is

Algorithm 1: Ensemble of ordinal decision rules

input : set of training examples $\{y_i, \mathbf{x}_i\}_1^N$,
M – number of decision rules to be generated.
output: ensemble of decision rules $\{r_m(\mathbf{x})\}_1^M$,
thresholds $\{\theta_k\}_1^{K-1}$.
$F_0(\mathbf{x}) := 0$; $\{\theta_{k0}\}_1^{K-1} := 0$;
for $m = 1$ *to* M **do**
  $(\mathbf{c}, \{\theta_k\}_1^{K-1}) := \arg\min_{(\mathbf{c}, \{\theta_k\}_1^{K-1})} \sum_{i=1}^N L^{all}(y_i, F_{m-1}(\mathbf{x}_i) + r(\mathbf{x}_i, \mathbf{c}))$;
  $r_m(\mathbf{x}, \mathbf{c}) := r(\mathbf{x}, \mathbf{c})$;
  $\{\theta_{km}\}_1^{K-1} := \{\theta_k\}_1^{K-1}$;
  $F_m(\mathbf{x}) := F_{m-1}(\mathbf{x}) + r_m(\mathbf{x}, \mathbf{c})$;
end
$ensemble = \{r_m(\mathbf{x}, \mathbf{c})\}_1^M$; $thresholds = \{\theta_{kM}\}_1^{K-1}$;

augmented by one additional rule $r_m(\mathbf{x}, \mathbf{c})$. A single rule is built by sequential addition of new conditions to Φ and computation of α. This is done in view of minimizing

$$L_m = \sum_{i=1}^{N} L^{all}(y_i, F_{m-1}(\mathbf{x}_i) + r(\mathbf{x}_i, \mathbf{c})) =$$
$$= \sum_{\mathbf{x}_i \in cov(\Phi)} \left(\sum_{k=1}^{y_i - 1} L(1, F_{m-1}(\mathbf{x}_i) + \alpha - \theta_k) + \sum_{k=y_i}^{K-1} L(-1, F(\mathbf{x}_i)_{m-1} + \alpha - \theta_k) \right)$$
$$+ \sum_{\mathbf{x}_i \notin cov(\Phi)} \left(\sum_{k=1}^{y_i - 1} L(1, F_{m-1}(\mathbf{x}_i) - \theta_k) + \sum_{k=y_i}^{K-1} L(-1, F(\mathbf{x}_i)_{m-1} - \theta_k) \right) \quad (18)$$

with respect to Φ, α and $\{\theta_k\}_1^{K-1}$. A single rule is built until L_m cannot be decreased.

Ordinal classification decision is computed according to:

$$F(\mathbf{x}) = \sum_{k=1}^{K} k \cdot I \left(\sum_{m=1}^{M} r_m(\mathbf{x}, \mathbf{c}) \in [\theta_{k-1}, \theta_k) \right), \quad (19)$$

where $I(a)$ is an indicator function, i.e. if a is true then $I(a) = 1$, otherwise $I(a) = 0$. Some other approaches are also possible. For example, in experiments we have used a procedure that assigns intermediate values between class labels in order to minimize squared error.

In the following two subsections, we give details of two introduced algorithms.

3.1 Ordinal Decision Rules based on Exponential Boosting (ORDER-E)

The algorithm described in this subsection can be treated as generalization of AdaBoost [8] with decision rules as base classifiers. In each iteration of the

algorithm, a strictly convex function (18) defined using the exponential loss L_{exp} is minimized with respect to parameters Φ, α and $\{\theta_k\}_1^{K-1}$. In iteration m, it is easy to compute the following auxiliary values that depend only on $F_{m-1}(\mathbf{x})$ and Φ:

$$A_{km} = \sum_{\mathbf{x}_i \in cov(\Phi)} I(y_i > k) e^{-F_{m-1}(\mathbf{x}_i)} \quad B_{km} = \sum_{\mathbf{x}_i \in cov(\Phi)} I(y_i \leq k) e^{F_{m-1}(\mathbf{x}_i)}$$
$$C_{km} = \sum_{\mathbf{x}_i \notin cov(\Phi)} I(y_i > k) e^{-F_{m-1}(\mathbf{x}_i)} \quad D_{km} = \sum_{\mathbf{x}_i \notin cov(\Phi)} I(y_i \leq k) e^{F_{m-1}(\mathbf{x}_i)}$$

These values are then used in computation of the parameters. The optimal values for thresholds $\{\theta_k\}_1^{K-1}$ are obtained by setting the derivative to zero:

$$\frac{\partial L_m}{\partial \theta_k} = 0 \Leftrightarrow \theta_k = \frac{1}{2} \log \frac{B_k \cdot \exp(\alpha) + D_k}{A_k \exp(-\alpha) + C_k}, \tag{20}$$

where parameter α is still to be determined. Putting (20) into (18), we obtain the formula for L_m:

$$L_m = 2 \sum_{k=1}^{K-1} \sqrt{(B_k \cdot \exp(\alpha) + D_k)(A_k \cdot \exp(-\alpha) + C_k)}. \tag{21}$$

which now depends only on single parameter α. The optimal value of α can be obtained by solving

$$\frac{\partial L_m}{\partial \alpha} = 0 \Leftrightarrow \sum_{k=1}^{K-1} \frac{B_k \cdot C_k \cdot \exp(\alpha) - A_k \cdot D_k \cdot \exp(-\alpha)}{\sqrt{(B_k \cdot \exp(\alpha) + D_k)(A_k \cdot \exp(-\alpha) + C_k)}} = 0 \tag{22}$$

There is, however, no simple and fast exact solution to (22). That is why we approximate α by a single Newton-Raphson step:

$$\alpha := \alpha_0 - \nu \cdot \frac{\partial L_m}{\partial \alpha} \cdot \left(\frac{\partial^2 L_m}{\partial^2 \alpha} \right)^{-1} \Bigg|_{\alpha = \alpha_0} \tag{23}$$

computed around zero, i.e. $\alpha_0 = 0$. Summarizing, a set of conditions Φ is chosen which minimizes (21) with α given by (23). One can notice the absence of thresholds in the formula for L_m (21). Indeed, thresholds are necessary only for further classification and can be determined once, at the end of induction procedure. However, L_m (21) is not additive anymore, i.e. it is not the sum of losses of objects due to implicit dependence between objects through the (hidden) thresholds values.

Another boosting scheme for ordinal classification has been proposed in [14]. Similar loss function has been used, although expressed in terms of margins (therefore called "left-right margins" and "all-margins" instead of "immediate-thresholds" and "all-thresholds"). The difference is that in [14] optimization over parameters is performed sequentially. First, a base learner is fitted with $\alpha = 1$.

Then, the optimal value of α is obtained, using thresholds values from previous iterations. Finally, the thresholds are updated. In section 4, we compared this boosting strategy with our methods, showing that such a sequential optimization does not work well with decision rule as a base learner.

3.2 Ordinal Decision Rules based on Gradient Boosting (ORDER-G)

The second algorithm is an extension of the gradient boosting machine [9]. Here, the goal is to minimize (18) defined by deviance loss L_{dev}. Φ is determined by searching for regression rule that fits pseudoresponses $\tilde{y_i}$ being negative gradients:

$$\tilde{y_i} = -\frac{\partial L_{dev}^{all}(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\bigg|_{F(\mathbf{x}_i)=F_{m-1}(\mathbf{x}_i)} \tag{24}$$

with $\{\theta_{km-1}\}_1^{K-1}$ determined in iteration $m-1$. The regression rule is fit by minimization of the squared-error loss:

$$\sum_{\mathbf{x}_i \in cov(\Phi)} (\tilde{y_i} - F_{m-1}(\mathbf{x}_i) - \tilde{\alpha})^2 + \sum_{\mathbf{x}_i \notin cov(\Phi)} (\tilde{y_i} - F_{m-1}(\mathbf{x}_i))^2. \tag{25}$$

The minimum of (25) is reached for

$$\tilde{\alpha} = \sum_{\mathbf{x}_i \in cov(\Phi)} (\tilde{y_i} - F_{m-1}(\mathbf{x}_i)) / \sum_{\mathbf{x}_i \in cov(\Phi)} 1. \tag{26}$$

The optimal value for α is obtained by setting

$$\frac{\partial L_m}{\partial \alpha} = 0$$

with Φ already determined in previous step. However, since this equation has no closed-form solution, the value of α is then approximated by a single Newton-Raphson step, as in (23). Finally, $\{\theta_{km}\}_1^{K-1}$ are determined by

$$\frac{\partial L_m}{\partial \theta_{km}} = 0.$$

Once again, since there is no closed-form solution, θ_{km} is approximated by a single Newton-Raphson step,

$$\theta_{km} = \theta_{km-1} - \frac{\partial L_m}{\partial \theta_{km}} \cdot \left(\frac{\partial^2 L_m}{\partial^2 \theta_{km}} \right)^{-1} \Bigg|_{\theta_{km}=\theta_{km-1}},$$

with Φ and α previously determined.

Notice that the scheme presented here is valid not only for L_{dev}, but for any other convex, differentiable loss function used as a base loss function in (18).

4 Experimental Results

We performed two experiments. Our aim was to compare simple ordinal classifiers, ordinal decision rules and approaches introduced in [3, 14]. We also wanted to check, how the introduced approaches works on Netflix Prize dataset [16]. As a comparison criteria we chose zero-one error (ZOE), mean absolute error (MEA) and root mean squared error (RMSE). The former two were used in referred papers. RMSE was chosen because of Netflix Prize rules.

The simple ordinal classifiers were based on logistic regression, LogitBoost [10, 9] with decision stumps, linear regression and additive regression [9]. Implementations of these methods were taken from Weka package [20]. In the case of logistic regression and LogitBoost, decisions were computed according to the analysis given in section 2. In order to minimize, ZOE, MAE and RMSE a final decision was computed as a mode, median or average over the distribution given by these methods, respectively. We used three ordinal rule ensembles. The first one is based on ORBoost-All scheme introduced in [14]. The other two are ORDER-E and ORDER-G introduced in this paper. In this case, a final decision was computed according to (19) in order to minimize ZOE and MAE. For minimization of RMSE, we have assumed that the ensemble constructs $F_M(\mathbf{x})$ which is a monotone transformation of a value function defined on an interval $[1, 5] \subseteq \mathbb{R}$. In classification procedure, values of $F_M(\mathbf{x})$ are mapped to $[1, 5] \subseteq \mathbb{R}$ by:

$$F(\mathbf{x}) = \sum_{k=1}^{K} \Big(k + \frac{F_M(\mathbf{x}) - (\theta_k + \theta_{k-1})/2}{\theta_k - \theta_{k-1}} \Big) \cdot I(F_M(\mathbf{x}) \in [\theta_{k-1}, \theta_k)),$$

where $\theta_0 = \theta_1 - 2 \cdot (\theta_2 - \theta_1)$ and $\theta_K = \theta_{K-1} + 2 \cdot (\theta_{K-1} - \theta_{K-2})$. These methods were compared with SVM with explicit constraints and SVM with implicit constraints introduced in [3] and with ORBoost-LR and ORBoost-All with perceptron and sigmoid base classifiers introduced in [14].

In the first experiment we used the same datasets and settings as in [3, 14] in order to compare the algorithms. These datasets were discretized by equal-frequency bins from some metric regression datasets. We used the same $K = 10$, the same "training/test" partition ratio, and also averaged the results over 20 trials. We report in Table 2 the means and standard errors for ZOE and MEA as it was done in the referred papers. In the last column of the table we put the best result found in [3, 14] for a given dataset. The optimal parameters for simple ordinal classifiers and ordinal rule ensembles were obtained in 5 trials without changing all other settings.

Second experiment was performed on Netflix Prize dataset [16]. We chose 10 first movies from the list of Netflix movies, which have been evaluated by at least 10 000 and at most 30 000 users. Three types of error (ZOE, MEA and RMSE) were calculated. We compared here only simple ordinal classifiers with ORDER-E and ORDER-G. Classifiers were learned on Netflix-training dataset and tested on Netflix-probe dataset (all evaluations from probe dataset were removed from training dataset). Ratings on 100 movies, selected in the same

Table 2. Experimental results on datasets used in [3, 14]. The same data preprocessing is used that enables comparison of the results. In the last column, the best results obtained by [1)]SVM with explicit constraints [3], [2)]SVM with implicit constraints [3], [3)]ORBoost-LR [14], and [4)]ORBoost-All [14] are reported. Two types of error are considered (zero-one and mean-absolute). Best results are marked in bold among all compared methods and among methods introduced in this paper.

Zero-one error (ZOE)						
Dataset	Logistic Regression	LogitBoost with DS	ORBoost-All with Rules	ORDER-E	ORDER-G	Best result from [3, 14]
Pyrim.	**0.754±0.017**	0.773±0.018	0.852±0.011	**0.754±0.019**	0.779±0.018	**0.719±0.066**[2]
CPU	0.648±0.009	0.587±0.012	0.722±0.011	0.594±0.014	**0.562±0.009**	0.605±0.010[4]
Boston	0.615±0.007	0.581±0.007	0.653±0.008	**0.560±0.006**	0.581±0.007	**0.549±0.007**[3]
Abal.	**0.678±0.002**	0.694±0.002	0.761±0.003	0.710±0.002	0.712±0.002	0.716±0.002[3]
Bank	**0.679±0.001**	0.693±0.001	0.852±0.002	0.754±0.001	0.759±0.001	0.744±0.005[1]
Comp.	0.489±0.001	0.494±0.001	0.593±0.002	**0.476±0.002**	0.479±0.001	**0.462±0.001**[1]
Calif.	0.665±0.001	**0.606±0.001**	0.773±0.002	0.631±0.001	0.609±0.001	**0.605±0.001**[3]
Census	0.707±0.001	**0.665±0.001**	0.793±0.001	0.691±0.001	0.687±0.001	0.694±0.001[3]
Mean absolute error (MAE)						
Dataset	Logistic Regression	LogitBoost with DS	ORBoost-All with Rules	ORDER-E	ORDER-G	Best result from [3, 14]
Pyrim.	1.665±0.056	1.754±0.050	1.858±0.074	**1.306±0.041**	1.356±0.063	**1.294±0.046**[2]
CPU	0.934±0.021	0.905±0.025	1.164±0.026	0.878±0.027	**0.843±0.022**	0.889±0.019[4]
Boston	0.903±0.013	0.908±0.017	1.068±0.017	**0.813±0.010**	0.828±0.014	**0.747±0.011**[2]
Abal.	**1.202±0.003**	1.272±0.003	1.520±0.008	1.257±0.002	1.281±0.004	1.361±0.003[2]
Bank	**1.445±0.003**	1.568±0.003	2.183±0.005	1.605±0.005	1.611±0.004	**1.393±0.002**[2]
Comp.	0.628±0.002	0.619±0.002	0.930±0.005	**0.583±0.002**	0.588±0.002	0.596±0.002[2]
Calif.	1.130±0.004	0.957±0.001	1.646±0.007	0.955±0.003	**0.897±0.002**	0.942±0.002[4]
Census	1.432±0.003	1.172±0.002	1.669±0.006	**1.152±0.002**	1.166±0.002	1.198±0.002[4]

way for each movie, were used as condition attributes. For each method, we tuned its parameters to optimize its performance, using 10% of training set as a validation set; to avoid favouring methods with more parameters, for each algorithm we performed the same number of tuning trials. The results are shown in Table 3.

The results from both experiments indicate that ensembles of ordinal decision rules are competitive to other methods used in the experiment:

- From the first experiment, one can conclude that ORBoost strategy does not work well with decision rule as a base learner, and that simple ordinal classifiers and ordinal decision rules perform comparably to approaches introduced in [3, 14].
- The second experiment shows that especially ORDER-E outperforms other methods in RMSE for most of the movies and in MAE for half of the movies. However, this method was the slowest between all tested algorithms. ORDER-G is much more faster than ORDER-E, but it obtained moderate results.
- In both experiments logistic regression and LogitBoost perform well. It is clear that these algorithms achieved the best results with respect to ZOE. The reason is that they can be tailored to multi-classification problem with zero-one loss, while ordinal decision rules cannot.

Table 3. Experimental results on 10 movies from Netflix Prize data set. Three types of error are considered (zero-one, mean-absolute and root mean squared). For each movie, best results are marked in bold.

Zero-one error (ZOE)						
Movie #	Linear Regression	Additive Regression	Logistic Regression	LogitBoost with DS	ORDER-E	ORDER-G
8	0.761	0.753	0.753	**0.714**	0.740	0.752
18	0.547	0.540	0.517	**0.493**	0.557	0.577
58	0.519	0.496	0.490	**0.487**	0.513	0.496
77	0.596	0.602	0.583	**0.580**	0.599	0.605
83	0.486	0.486	0.483	**0.398**	0.462	0.450
97	0.607	0.607	0.591	**0.389**	0.436	0.544
108	0.610	0.602	0.599	**0.593**	0.613	0.596
111	0.563	0.561	0.567	**0.555**	0.572	0.563
118	0.594	0.596	0.532	0.524	**0.511**	0.551
148	0.602	0.610	0.593	0.536	**0.522**	0.573
Mean absolute error (MAE)						
Movie #	Linear Regression	Additive Regression	Logistic Regression	LogitBoost with DS	ORDER-E	ORDER-G
8	1.133	1.135	1.115	1.087	**1.013**	1.018
18	0.645	0.651	**0.583**	0.587	0.603	0.613
58	0.679	0.663	0.566	**0.543**	0.558	0.560
77	0.831	0.839	0.803	0.781	**0.737**	0.755
83	0.608	0.614	0.519	**0.448**	0.500	0.502
97	0.754	0.752	0.701	**0.530**	0.537	0.654
108	0.777	0.776	**0.739**	**0.739**	0.768	**0.739**
111	0.749	0.766	0.720	0.715	**0.693**	0.705
118	0.720	0.734	0.626	0.630	**0.596**	0.658
148	0.747	0.735	0.688	0.626	**0.604**	0.659
Root mean squared error (RMSE)						
Movie #	Linear Regression	Additive Regression	Logistic Regression	LogitBoost with DS	ORDER-E	ORDER-G
8	1.332	1.328	1.317	1.314	**1.268**	1.299
18	0.828	0.836	**0.809**	0.856	0.832	0.826
58	0.852	0.847	0.839	**0.805**	0.808	0.817
77	1.067	1.056	1.056	1.015	**0.999**	1.043
83	0.775	0.772	0.737	0.740	**0.729**	0.735
97	0.968	0.970	0.874	0.865	**0.835**	0.857
108	0.984	0.993	**0.969**	0.979	0.970	0.989
111	0.985	0.992	0.970	0.971	**0.967**	0.986
118	0.895	0.928	0.862	0.860	**0.836**	0.873
148	0.924	0.910	0.900	0.863	**0.838**	0.893

- It is worth noticing, that regression algorithms resulted in poor accuracy in many cases.
- We have observed during the experiment that ORDER-E and ORDER-G are sensitive to parameters setting. We plan to work on some simple method for parameters selection.

5 Conclusions

From the theoretical analysis, it follows that ordinal classification problem can be solved by different approaches. In our opinion, there is still a lot to do in order to establish a theoretic framework for ordinal classification. In this paper, we introduced a decision rule induction algorithm based on forward stagewise additive modeling that utilizes the notion of threshold loss function. The experiment

indicates that ordinal decision rules are quite promising. They are competitive to traditional regression and multi-class classification methods, and also to existing ordinal classification methods. Let us remark that the algorithm can also be used for other base classifiers like decision trees instead of decision rules. In this paper, we remained with rules because of their simplicity in interpretation. It is also interesting that such a simple classifier works so well as a part of the ensemble.

References

1. Błaszczyński, J., Dembczyński, K., Kotłowski, W., Słowiński, R., Szeląg, M.: Ensembles of Decision Rules. *Foundations of Computing and Decision Sciences*, **31** (2006) 21–232
2. Błaszczyński, J., Dembczyński, K., Kotłowski, W., Słowiński, R., Szeląg, M.: Ensembles of Decision Rules for Solving Binary Classification Problems in the Presence of Missing Values. *Lecture Notes in Artificial Intelligence*, **4259** (2006) 224–234
3. Chu, W., Keerthi, S. S.: New approaches to support vector ordinal regression. In *Proc. of 22nd International Conference on Machine Learning* (2005) 321–328
4. Clémençon, S., Lugosi, G., Vayatis, N.: Ranking and empirical minimization of U-statistics. (to appear)
5. Cohen, W., Singer, Y.: A simple, fast, and effective rule learner. In *Proc. of 16th National Conference on Artificial Intelligence* (1999) 335–342
6. Frank, E., Hall, M.: A simple approach to ordinal classification. *Lecture Notes in Computer Science*, **2167** (2001) 145–157
7. Freund, Y., Iyer, R., Schapire, R., Singer, Y.: An efficient boosting algorithm for combining preferences. *J. of Machine Learning Research*, **4** (2003) 933–969.
8. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an
application to boosting. *J. of Computer and System Sciences*, **55** 1 (1997) 119–139
9. Friedman, J.: Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, **29** 5 (2001) 1189–1232
10. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, **28** 2 (2000) 337–407
11. Friedman, J., Popescu, B.: Predictive learning via rule ensembles. Research report, Dept. of Statistics, Stanford University (2005)
12. Hastie, T., Tibshirani, R., Friedman, J.: *Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer (2003)
13. Herbrich, R., Graepel, T., Obermayer, K.: Regression models for ordinal data: A machine learning approach. Technical report TR-99/03, TU Berlin (1999)
14. Lin, H.T., Li, L.: Large-margin thresholded ensembles for ordinal regression: Theory and practice. *Lecture Notes in Artificial Intelligence*, **4264** (2006) 319–333
15. Lin, H.T., Li, L.: Ordinal regression by extended binary classifications. *Advances in Neural Information Processing Systems*, **19** (2007) 865–872
16. Netflix prize, http://www.netflixprize.com.
17. Rennie, J., Srebro, N.: Loss functions for preference levels: Regression with discrete ordered labels. In *Proc. of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling* (2005)

18. Shashua, A., Levin, A.: Ranking with large margin principle: Two approaches. *Advances in Neural Information Processing Systems*, **15** (2003)
19. Weiss, S., Indurkhya, N.: Lightweight rule induction. In *Proc. of 17th International Conference on Machine Learning* (2000) 1135–1142
20. Witten, I., Frank, E.: *Data Mining: Practical machine learning tools and techniques, 2nd Edition.* Morgan Kaufmann (2005)