Rule Learning with Monotonicity Constraints

Wojciech Kotłowski WKOTLOWSKI@CS.PUT.POZNAN.PL
Institute of Computing Science, Poznań University of Technology, Poznań, 60-965, Poland

Roman Słowiński RSLOWINSKI@CS.PUT.POZNAN.PL
Institute of Computing Science, Poznań University of Technology, Poznań, 60-965, Poland
Systems Research Institute, Polish Academy of Sciences, Warsaw, 01-447, Poland

Abstract

In classification with monotonicity constraints, it is assumed that the class label should increase with increasing values on the attributes. In this paper we aim at formalizing the approach to learning with monotonicity constraints from statistical point of view. Motivated by the statistical analysis, we present an algorithm for learning rule ensembles. The algorithm first "monotonizes" the data using a nonparametric classification procedure and then generates a rule ensemble consistent with the training set. The procedure is justified by a theoretical analysis and verified in a computational experiment.

1. Introduction

Utilizing the prior knowledge is of fundamental importance in the learning process. A common type of such knowledge are monotone relationships between the input and output variables, which can often be elicited from a domain expert.

Monotone relationships are frequently encountered in the applications of machine learning. For example, in the customer satisfaction analysis (Greco et al., 2007), the global evaluation of the product by a customer should increase with increasing evaluations of the product on a set of attributes. In the house pricing problem (Potharst & Feelders, 2002), the selling price of the house should increase with e.g. lot size, number of rooms, and decrease with e.g. crime rate or pollution concentration in the area. Monotonicity also occurs in such domains as option pricing (Gamarnik, 1998), medical diagnosis (Sill, 1998), survey data (Cao-Van & De Baets, 2004) and many others. Moreover, such problems are widely considered under the name *multiple criteria sorting* within multiple criteria decision analysis (Greco et al., 2001). We will refer to such problems as to *ordinal classification with monotonicity constraints*.

There are at least two important reasons why knowledge about monotonicity should be exploited in the learning process. Firstly, monotonicity imposes constraints on the prediction function, hence decreasing the size of the hypothesis space, which in turn decreases the complexity of the model and in the light of the structural risk minimization it should lead to increase in the accuracy of predictions. Secondly, very often only the model consistent with domain knowledge is acceptable to the domain experts: behavior of the model contradicting the experts' knowledge may lead to rejection of the model's predictions regardless of their accuracy.

This led to the development of new algorithms and modification of existing ones in order to handle monotonicity constraints in the learning process. The examples of such algorithms are Dominance-based Rough Set Approach (DRSA) (Greco et al., 2001) along with decision rule induction (Dembczyński et al., 2008), monotone classification trees (Ben-David, 1995; Potharst & Feelders, 2002), monotone networks (Sill, 1998), instance-based methods (Ben-David et al., 1989; Cao-Van & De Baets, 2004) or isotonic separation (Chandrasekaran et al., 2005).

In this paper we aim at formalizing the approach to learning with monotonicity constraints from statistical point of view. We start with presenting the probabilistic model for the problem and showing how monotonicity relationships impose constraints on the distribution generating the data. Having the model, we present a method of learning with monotonicity constraints

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

based on the statistical analysis of the problem. Our method works in three subsequent phases. First, we decompose the general multi-class problem into a sequence of binary subproblems. Then, for each binary subproblem, we use nonparametric classification to remove inconsistencies from the data set (so called *data monotonization*). Finally, we apply rule learning algorithm on the monotonized data; the algorithm approximates the data in an exact way, i.e. without making any classification errors (thus can be thought of as a data compression), simultaneously achieving high margin distribution. The learning algorithm is based on the LPBoost scheme (Demiriz et al., 2001), therefore we call our method *LPRules*. The ideas standing behind all three phases are theoretically justified. Notice, that rule induction has already been considered with this kind of problems (Dembczyński et al., 2008), but here for the first time we motivate the algorithm from the statistical point of view.

In the last section, we test LPRules on ten real dataset, for which it is known that there exist monotone relationships between attribute values and class labels. We also compare our method with two existing approaches to learning with monotonicity constraints and two ordinary classification algorithms, which are all outperformed by LPRules in the experiment.

2. Problem Statement

Let $(x, y) \in X \times Y$ be the object-label pair generated according to distribution $P(x, y)$, where x is a vector of m attributes with *ordered* domains and y is a class label from a finite set of *ordered* labels $Y = \{1, \ldots, K\}$. We aim at finding the classifier $h\colon X \to Y$, that accurately predicts values of y. The accuracy is measured in terms of *loss function* $L(y, k)$, which is the penalty for predicting k when the actual value is y. The overall accuracy of the classifier h is defined as $\mathbb{E}L(y, h(x))$, the expected loss (*risk*) according to $P(x, y)$.

Bayes classifier is a function h^* minimizing the expected loss, $h^* = \arg\min_h \mathbb{E}L(y, h(x))$. The minimum $L^* = \mathbb{E}L(y, h^*(x))$ is called *Bayes risk*. Since $P(x, y)$ is unknown, h^* is also unknown and the classifier is learned from a sample of n training examples $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ (training set). In order to minimize the expected loss, the learning procedure usually performs minimization of *empirical risk*:

$$\mathbb{E}_D L(y, h(x)) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i)),$$

where $\mathbb{E}_D$ denotes the empirical mean. It is reasonable to assume that $L(k, k) = 0$ and $L(y, k) > 0$ if $y \neq k$ for $y, k = 1, \ldots, K$. The loss should also be consistent with the order between class labels in the sense that the loss should not decrease, as the predicted value moves away from the true value. Hence, following (Lin & Li, 2007), we assume the loss matrix is *V-shaped*: for $k \leq y$ it holds $L(y, k-1) \geq L(y, k)$, while for $k \geq y$ it holds $L(y, k) \leq L(y, k+1)$.

Notice that until now, the proposed model bears close resemblance to the problem of ordinal regression, except that the loss function is in the spirit of (Lin & Li, 2007) and is different from more popular *rank loss* formulation (Herbrich et al., 1999). What distinguishes the problem is the presence of monotonicity constraints. To define them in terms of the probability distribution, we exploit the order properties in the set X by using a *dominance relation*. We say x *dominates* x', $x \succeq x'$ if x has greater or equal values on every attribute, $x_j \geq x'_j, j = 1, \ldots, m$. Dominance relation is a partial preorder. Let us call a function $h\colon X \to Y$ *monotone* if for any $x, x' \in X$ it holds $x \succeq x' \to h(x) \geq h(x')$.

Intuitively, the monotonicity constraints require that if $x \succeq x'$ then x should be assigned a class not lower than x'. In practice, those constraints are not always satisfied, leading to the situations referred to as *inconsistencies*. This suggests that the order relation $\succeq$ does not impose "hard" constraints, so that the constraints should rather be defined in a probabilistic setting. In (Kotłowski & Słowiński, 2008), a definition based on the *stochastic dominance relation* was proposed, which appears to be especially useful:

$$x \succeq x' \longrightarrow P(y \geq k|x) \geq P(y \geq k|x'), \qquad (1)$$

for each $x, x' \in X$, $k = 2, \ldots, K$. Expression (1) states that the probability of an event $\{y \geq k\}$ is a monotone function. We will say that a probability distribution is *monotonically constrained* if (1) is satisfied for every $x, x' \in X$. For the rest of the paper, we will assume that the distribution is of this kind.

Constraint (1) suggests using classifiers which are monotone functions. Indeed, almost every previous approach to classification with monotonicity constraints restricts learning to monotone classifiers. Since we aim at finding the classifier which is as close as possible to the Bayes classifier h^* (which is our "target function"), we would expect that the Bayes classifier is monotone. However, although the probability distribution is monotonically constrained, the monotonicity of the Bayes classifier does not always hold. For instance, the Bayes classifier for 0-1 loss (the mode of the distribution) is not monotone under stochastic dominance assumption. It appears that some specific constraints must be imposed on the loss function in

order to maintain the monotonicity of the Bayes classifier. Here we restrict to a special, but illustrative case, when the loss function has the form $L(y,k) = C(y-k)$, for some function $C\colon \mathbb{Z} \to Y$.

Let us call a function $C\colon \mathbb{Z} \to \mathbb{R}$ *convex* if for all $i, j \in \mathbb{Z}$ and for every $\lambda \in [0,1]$ such that $\lambda i + (1-\lambda)j \in \mathbb{Z}$, we have: $c(i\lambda + (1-\lambda)j) \leq \lambda c(i) + (1-\lambda)c(j)$. The following theorem holds:

Theorem 1. *Let $L(y,k) = C(y-k)$ be the V-shaped loss function. Then, the Bayes classifier h^* is monotone if and only if $C(k)$ is convex.*

The proof can be found in (Kotłowski & Słowiński, 2008). Theorem 1 immediately leads to the following:

Corollary 1. *Let $L(y,k) = |y-k|^p$ with $p \geq 0$, be the loss function and let $K \geq 3$. Then, the Bayes classifier is monotone if and only if $p \geq 1$.*

We assumed $K \geq 3$, since in case $K = 2$, every V-shaped loss is convex. Corollary 1 implies that 0-1 loss ($p \to 0$) does not lead to the monotone Bayes classifier under the stochastic dominance assumption, while absolute error loss ($p = 1$) and squared-error loss ($p = 2$) do ensure monotonicity.

For the rest of the paper we will restrict the analysis to the absolute error loss, $L(y,k) = |y-k|$ (however, most of the results presented in this paper can be stated for a more general form of the loss function). Absolute error loss has a unique property: its minimizer (Bayes classifier) does not depend on a particular encoding of class labels. Indeed, it is known, that for a given x, the absolute error loss is minimized by the median of the conditional distribution $P(y|x)$. The median in the ordered set does not depend on the particular values of the elements in the set and takes only the order into account. Thus, although it seems that absolute error imposes a distance measure between the class labels (equal to the difference between their indices), it is invariant under arbitrary monotone (order-preserving) transformation of the labels.

3. Binary Decomposition

We introduce a method for learning with monotonicity constraints consisting of three subsequent phases: decomposition into binary subproblems, data monotonization and rule learning. The phases will be covered in detail in the next three sections. In this section we describe the first phase: decomposition of the general K-class problem into a sequence of $K-1$ binary subproblems.

To simplify further expressions we always assume $Y = \{-1, 1\}$ when concerning the binary problem. Suppose we have an access to the learning algorithm for binary classification problem, which can produce classifier $h(x) \in \{-1, 1\}$ with small 0-1 loss $L(k,y) = \mathbb{1}[y \neq h(x)]$ (for $y \in \{-1,1\}$), where $\mathbb{1}[C]$ is the indicator function, equal to 1 if predicate C holds and 0 otherwise. Let y be the class label and let us define $y_k = \operatorname{sgn}(y-k)$ for $k = 2, \ldots, K$. Therefore, $y_k = 1$ corresponds to the class "at least k", while $y_k = -1$ corresponds to the class "at most $k-1$". We train $K-1$ binary classifiers using the available algorithm, where the k-th classifier ($k = 2, \ldots, K$) $h_k(x)$ is trained on the dataset with original class labels y substituted by labels y_k. This dataset will be denoted D_k. We combine binary classifiers into a single K-class classifier in the following way:

$$h(x) = 1 + \sum_{k=2}^{K} \mathbb{1}[h_k(x) = 1]. \tag{2}$$

We show that the final classifier makes an error not greater then the sum of errors of the base classifiers:

Theorem 2. *It holds:*

$$|y - h(x)| \leq \sum_{k=2}^{K} \mathbb{1}[y_k \neq h_k(x)]$$

Proof. To abbreviate the notation, we denote $h_k(x)$ by h_k and $h(x)$ by h. We have:

$$\begin{aligned}\sum_{k=2}^{K} \mathbb{1}[y_k \neq h_k] &= \sum_{k=2}^{y} \mathbb{1}[h_k = -1] + \sum_{k=y+1}^{K} \mathbb{1}[h_k = 1] \\ &\geq (y-h)\mathbb{1}[h < y] + (h-y)\mathbb{1}[h > y] = |y-h|\end{aligned}$$

□

Thus, we see that in order to obtain accurate composite classifier h, all we need is an accurate learning algorithm for binary problems with small 0-1 loss.

In classification with monotonicity constraints we usually require that the classifier $h(x)$ must be monotone. It follows directly from (2) that if the binary classifiers $h_k(x)$ are all monotone, then so is $h(x)$.

4. Data Monotonization

Let us consider a nonparametric approach to classification with monotonicity constraints. By this we mean a problem of minimizing the empirical risk in the class of all monotone functions. The problem can be solved efficiently by realizing that we need to consider the values of function only at training points $x_i, i = 1, \ldots, n$. Thus, for the k-th binary problem we end up with the

following formulation:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} |y_{ik} - d_i| \\ \text{subject to} \quad & x_i \succeq x_j \rightarrow d_i \geq d_j \quad i,j = 1,\ldots,n \\ & d_i \in \{-1,1\} \qquad i = 1,\ldots,n, \end{aligned} \tag{3}$$

where y_{ik} are the labels from the dataset D_k and d_i are optimization variables. The problem has already been considered several times in various contexts (Dykstra et al., 1999; Chandrasekaran et al., 2005; Kotłowski et al., 2008; Kotłowski & Słowiński, 2008; Barile & Feelders, 2008). Although variables d_i are integer, the constraint matrix is totally unimodular, so that integer constraints can be relaxed and the problem can be solved efficiently via the linear programming. Moreover, it was shown that the complexity of the linear program can be significantly reduced (Kotłowski et al., 2008; Kotłowski & Słowiński, 2008).

Optimal values of variables d_i, denoted y'_{ik}, can be regarded as new class labels for the training objects. Thus, the problem can be interpreted as follows: relabel (reassign) the smallest number of objects to make the dataset monotone. The dataset with new labels y'_{ik} replacing the original labels y_{ik} will be denoted D'_k. The process of relabeling will be called *monotonization*. The main idea of how monotonization can be used is to apply the method to the original data, monotonize it and pass it to the learning algorithm. The reason behind this method is to remove inconsistencies in the data using the only available objective information about the problem: the monotonicity constraints. Later we justify the procedure from statistical point of view by showing how data monotonization influences the classifiers' performance.

Interestingly, sometimes nonparametric approach is enough to classify with high accuracy. For instance, if $P(x,y)$ has density, one can show (Devroye et al., 1996) that empirical risk minimization in the class of monotone functions is strongly consistent. Unfortunately, the speed of convergence can be very slow, especially when m (the dimension of the attribute space) is high. Moreover, one needs to remember large part of the dataset D'_k to classify. Thus, to solve those issues we "compress" the monotonized data with rule ensemble, as it is described in the next section.

5. Learning Rule Ensemble

In the last phase, we use monotonized data to train a classifier, which decreases training error down to 0. The classifier can be though of as a compression of the monotonized data. We assume the classifier has the form of rule ensemble.

Rules are simple and interpretable logical statements of the form: "if *conditions* then *response*", which can be treated as simple classifiers that give a constant response to examples satisfying the condition part, and abstain from the response for other examples. Let X_j be the set of all possible values (domain) for the j-th attribute. Condition part of the rule consist of *elementary conditions* of the form $x_j \geq s_j$ or $x_j \leq s_j$ for some $s_j \in X_j$. If all conditions are satisfied, rule response is 1 or -1, otherwise rule returns 0. Thus, rule will be treated as a function, which has one of the two following forms:

$$\begin{aligned} r(x) &= \mathbb{1}[x_{j_1} \geq s_{j_1} \wedge x_{j_2} \geq s_{j_2} \wedge \ldots \wedge x_{j_q} \geq s_{j_q}] \\ r(x) &= -\mathbb{1}[x_{j_1} \leq s_{j_1} \wedge x_{j_2} \leq s_{j_2} \wedge \ldots \wedge x_{j_q} \leq s_{j_q}] \end{aligned}$$

One can show that only that kinds of rules are acceptable if we do not want to violate monotonicity constraints; indeed, both kinds of rules are monotone functions. The first kind will be called *positive rule*, while the second kind – *negative rule*.

The rule ensemble $f(x)$ is a convex combination of rules treated as base classifiers:

$$f(x) = \sum_{t=1}^{T} a_t r_t(x),$$

where $a_t \geq 0$ and $\sum_t a_t = 1$. Object x is classified to the class indicated by the sign of $f(x)$, i.e. $h(x) = \operatorname{sgn}(f(x))$. The convex combination has a very simple interpretation as a voting procedure: positive rules vote for the positive class, while negative rules – for the negative class and x is assigned to the class with a higher vote. One can show that if all rules are monotone then $f(x)$ is a monotone function.

Consider k-th binary problem. As soon as the data are monotonized, we train rule ensemble $f_k(x)$ on D'_k such that it makes no errors on the dataset. In other words, $y'_{ik} f_k(x_i) > 0$ for all $i = 1,\ldots,n$, and we say that such rule ensemble *separates* D'_k. To prove the validity of our procedure, we must show that a monotone rule ensemble separating the data always exists. It appears that the existence of a separating rule ensemble is strictly related to the monotonicity of the dataset.

Theorem 3. *There exists a monotone rule ensemble $f(x)$ separating a dataset $D = \{(x_1,y_1),\ldots,(x_n,y_n)\}$, $y_i \in \{-1,1\}$, if and only if D is monotone, i.e. for each $i,j = 1,\ldots,n$, we have $x_i \succeq x_j \rightarrow y_i \geq y_j$.*

Proof. From the monotonicity of $f(x)$ it follows that $x_i \succeq x_j \rightarrow f(x_i) \geq f(x_j) \rightarrow \operatorname{sgn}(f(x_i)) \geq$

$\operatorname{sgn}(f(x_j))$. Moreover, $f(x)$ separates D so that we have $\operatorname{sgn}(y_i f(x_i)) = \operatorname{sgn}(y_j f(x_j)) = 1$. Since $y_i \in \{-1, 1\}$, it follows that $y_i \geq y_j$, so D is monotone.

Assume D is monotone. Then, for each x_i such that $y_i = 1$ we construct a rule $r(x) = \mathbb{1}[x \succeq x_i]$ with $a_i = 1/n$; this is a valid positive rule. Similarly, for each x_i with $y_i = -1$ we construct a rule $r(x) = -\mathbb{1}[x \preceq x_i]$ with $a_i = 1/n$. Since D is monotone, positive rules have nonzero response only for the objects with $y_j = 1$, while negative rules – only for the objects with $y_j = -1$. Hence, rule ensemble makes no errors on D. □

Since D'_k is monotone, we are guaranteed that a separating monotone rule ensemble exists. Moreover, to create separating rule ensemble we *must* first monotonize the data.

The algorithm we are using for learning rule ensemble is a version of LPBoost (Demiriz et al., 2001) with rule used as a base classifier; therefore, our method will be called *LPRules*. Using LPBoost schema has several important advantages. First of all, LPBoost is based on linear programming which produces very sparse optimal solutions. In our case this results in a small number of nonzero coefficients a_t. Sparsity of the ensemble is very desired as it improves the comprehensibility of the model. Secondly, LPBoost always converges to the optimal solution in a finite number of steps. By increasing the penalty for misclassification in the algorithm we are sure that the algorithm converges to the solution with no training errors if such exists; however, from Theorem 3 we know that such a solution *always* exists, so that we end up with the following corollary:

Corollary 2. *LPBoost always returns monotone rule ensemble separating the monotonized dataset D'_k.*

Finally, LPBoost is known to directly maximize the minimum margin, which guarantees a good out of sample accuracy of the ensemble. This is considered in details in the next section.

6. Generalization Bound

Let us consider the binary-class case $Y = \{-1, 1\}$ and let $f(x)$ be some real-valued function, such that object x is classified according to the sign of $f(x)$, $h(x) = \operatorname{sgn}(f(x))$. We call value $yf(x)$ a *margin* for x. It is a measure of prediction confidence and is positive if and only if x is correctly classified.

The margin theorem (Schapire et al., 1998; Koltchinskii & Panchenko, 2002) bounds the expected risk of $h(x) = \operatorname{sgn}(f(x))$ in terms of its margin distribution. Here we cite the result obtained by (Koltchinskii & Panchenko, 2002):

Theorem 4. *Let $f(x) = \sum_j a_j b_j(x)$ be the ensemble of classifiers $b_j(x) \in \{-1, 1\}, b_j \in \mathcal{B}$, where a_j are non-negative and $\sum_j a_j = 1$; let $h(x) = sgn(f(x))$. Then, with probability $1 - \delta$, for every $\gamma \in (0, 1]$ the following inequality holds:*

$$\mathbb{E}L(y, h(x)) \leq \mathbb{E}_D \mathbb{1}[yf(x) \leq \gamma] + M\left(\sqrt{\frac{d}{n\gamma^2}} + \sqrt{\frac{\log\frac{1}{\delta}}{n}}\right), \tag{4}$$

where d is the Vapnik-Chervonenkis dimension of the class $\mathcal{B}$ and M is some universal constant.

Thus, with high probability the expected 0-1 loss is bounded by the fraction of objects from the training set with margin smaller than γ plus some complexity term which increases with decreasing value of γ.

We would like to apply Theorem 4 to our learning algorithm, but there are three problems which need to be addressed. Firstly, rules return values in the set $\{0, 1\}$, rather than $\{-1, 1\}$, but this is only a matter of rescaling the coefficient and does not cause any trouble. Secondly, we must extend the theorem to the multi-class problem with absolute error loss. We do it by bounding the loss of the composite classifier in terms of sum of the losses of binary classifiers. Finally, the ensemble is learned on the monotonized data, which are not i.i.d. anymore. We will by-pass this problem by noticing that the ensemble learned on monotonized data makes on the original data no more margin errors than the number of relabeled objects.

Moreover, we do not want simply to give a generalization bound for our method, rather we want to show the benefit we gain from learning on the monotonized data. It follows that in the presence of monotonicity constraints, the risk of the ensemble separating the data with large margin remains close to the accuracy of Bayes classifier.

Theorem 5. *Assume $P(x, y)$ is monotonically constrained. Let $h(x)$ be the final classifier and let $f_k(x)$, $k = 2, \ldots, K$, be the k-th rule ensemble trained on the monotonized data set D'_k., Then, with probability at least $1 - \delta$ for every $\gamma_2, \ldots, \gamma_K$:*

$$\mathbb{E}\, L(y, h(x)) - L^* \leq \tag{5}$$
$$M\left(2(K-1)\sqrt{\frac{\log\frac{2(K-1)}{\delta}}{n}} + \sqrt{\frac{m}{n}}\sum_{k=2}^{K}\frac{1}{\gamma_k}\right).$$

for some universal constant M.

Before we give the proof, notice that Theorem 5 relates the *regret*, i.e. difference between the risk of the

composite classifier and the lowest possible risk, to the fraction of margin errors on the monotonized data set plus some complexity terms which depend on γ_k. Due to monotonicity constraints and data monotonization we are able to compare the behavior of our algorithm to the Bayes classifier, in opposite to ordinary margin theorems, in which one compares the fraction of margin errors to the expected accuracy of the ensemble.

Proof. We start with a transformation of rules (which can abstain from the response) to classifiers with output in $\{-1, 1\}$ and with well-defined VC dimension. Let $f(x) = \sum_{t=1}^{T} a_t r_t(x)$. With each rule $r_t(x)$ we associate a classifier $g_t(x) = 2r_t(x) - 1$ for positive rules and $g_t(x) = 2r_t(x) + 1$ for negative rules. Notice that $g_t(x) \in \{-1, 1\}$ and $g_t(x)$ are axis-parallel cones. The class of axis-parallel cones has VC dimension m (Devroye et al., 1996). We add to the ensemble one more cone $g_0(x) = \operatorname{sgn}\left(\sum_{t\in T^+} a_t - \sum_{t\in T^-} a_t\right)$ which covers the whole X and the subsets T^+ and T^- correspond to the positive and negative rules, respectively. Then, one can easily show that $\sum_{t=0}^{T} c_t g_t(x) = \sum_{t=1}^{T} a_t r_t(x)$, where $c_t = a_t/2$ for $t \geq 1$ and $c_0 = \frac{1}{2}|\sum_{t\in T^+} a_t - \sum_{t\in T^-} a_t|$. Hence, we see that each rule ensemble has an equivalent cone ensemble. Finally, one can show that $\ell = \sum_t c_t \leq \sum_t a_t = 1$, which means that weights c_t must be divided by ℓ to be normalized to 1. But this means that if the margin of the normalized cone ensemble equals to z, then the margin of rule ensemble equals to z/ℓ. Thus, the fraction of margin errors for the normalized cone ensemble upperbounds the fraction of margin errors for the normalized rule ensemble. Thus, we can prove the theorem for an ensemble of cones, and it will also hold for a rule ensemble.

The k-th ensemble $f_k(x)$ makes no errors on the monotonized dataset D'_k. Since D_k and D'_k differ only on the relabeled objects, the ensemble makes on D_k a margin error $\mathbb{E}_{D_k}\mathbb{1}[y_k h_k(x) \leq \gamma_k]$ not greater than $\mathbb{E}_{D_k}\mathbb{1}[y_k \neq y'_k]$. Since it equals (up to a constant n^{-1}) to the objective function of the monotonization problem (3), it is minimized by values y'_{ik} in the class of all monotone functions. But according to Theorem 1 Bayes classifier $h^*_k(x)$ is monotone, which means that:

$$\mathbb{E}_{D_k}\mathbb{1}[y_k h_k \leq \gamma_k] \leq \mathbb{E}_{D_k}\mathbb{1}[y_k \neq h^*_k],$$

where we abbreviate $h_k(x)$ as h_k. Using Hoeffding's bound (Devroye et al., 1996), for every $\delta' \in (0, 1)$:

$$P\left(\mathbb{E}_{D_k}\mathbb{1}[y_k \neq h^*_k] - \mathbb{E}\mathbb{1}[y_k \neq h^*_k] \geq \sqrt{\frac{\log\frac{1}{\delta'}}{2n}}\right) \leq \delta'.$$

From Theorem 4 we have with probability at most δ'':

$$\mathbb{E}L(y_k, h_k) \geq \mathbb{E}_{D_k}\mathbb{1}[y_k f_k \leq \gamma] + M\left(\sqrt{\frac{d}{n\gamma_k^2}} + \sqrt{\frac{\log\frac{1}{\delta''}}{n}}\right).$$

By setting $\delta' = \delta'' = \frac{\delta}{2(K-1)}$, we have with probability at most $\delta' + \delta'' = \frac{\delta}{K-1}$:

$$\mathbb{E}L(y_k, h_k) \geq \mathbb{E}\mathbb{1}[y_k \neq h^*_k] + M\left(\sqrt{\frac{d}{n\gamma_k^2}} + 2\sqrt{\frac{\log\frac{2(K-1)}{\delta}}{n}}\right) \quad (6)$$

To bound the risk in a general K-class case, notice that with probability at most δ there exists $k \in \{2, \ldots, K\}$ such that (6) holds. But this means that with probability at most δ:

$$\sum_{k=2}^{K} \mathbb{E}L(y_k, h_k) \geq \sum_{k=2}^{K} \mathbb{E}\mathbb{1}[y_k \neq h^*_k] + M\left(\sum_{k=2}^{K}\sqrt{\frac{d}{n\gamma_k^2}} + 2(K-1)\sqrt{\frac{\log\frac{2(K-1)}{\delta}}{n}}\right) \quad (7)$$

Using the fact that $|y - h^*(x)| = \sum_{k=2}^{K} \mathbb{1}[y_k \neq h^*_k(x)]$, we have $L^* = \sum_{k=2}^{K} \mathbb{E}\mathbb{1}[y_k \neq h^*_k]$. From Theorem 2 we have $\mathbb{E}|y - h(x)| \leq \sum_{k=2}^{K} \mathbb{E}\mathbb{1}[y_k h_k(x) < 0]$. Plugging those expression into (7) proves the thesis. □

7. Experimental Results

We verified the efficiency of our method in the computational experiment and compared it with two existing approaches to classification with monotonicity constraints and two ordinary classification algorithms. The experiment is conducted on several real datasets, for which it is known that there exist monotone relationships between attribute values and class labels.

Datasets: ESL (*employee selection*), SWD (*social workers decisions*), LEV (*lecturers evaluation*) and ERA (*Employee Rejection/Acceptance*) contain surveys data and were taken from (Ben-David, 1995). Datasets: Housing (Asuncion & Newman, 2007), Windsor (Anglin & Gencay, 1996) and DenBosch (Daniels & Kamp, 1999) are related to house pricing. We discretized the continuous price variable in Housing and Windsor into four levels containing equal number of objects (quartiles of the price distribution), similarly as in (Potharst & Feelders, 2002). The other three datasets: Car, CPU, Balance, were taken from UCI repository. A more detailed description of the data is given in Table 1.

Two algorithms used for comparison are Ordinal Learning Model (OLM) (Ben-David et al., 1989), the first method proposed for dealing with monotone data, and Isotonic Separation (IS) (Chandrasekaran et al., 2005), an instance-based algorithm which proved to be very efficient in practice. Moreover, we add two ordinary methods, decision tree J48 and Support Vector

Table 1. Description of data sets used in experiments.

Data set	#attributes	#objects	#classes
ESL	4	488	8
SWD	10	1000	4
LEV	4	1000	5
ERA	4	1000	8
Housing	8	506	4
Windsor	11	546	4
DenBosch	9	119	2
Car	6	1728	4
CPU	6	209	4
Balance	4	625	3

Machines (SVM) with linear kernel. Unfortunately, both are designed to minimize 0-1 error, and using them directly on multi-class problems led to very poor results in terms of the mean absolute error. Therefore, we decided to improve its performance and use it in the ordinal setting by combining it with a simple approach to ordinal classification proposed by (Frank & Hall, 2001). This approach works in a very similar fashion as transformation of the main problem into $K-1$ binary problems described in Section 3. The implementations of the methods were taken from Weka (Witten & Frank, 2005).

The error measure was the mean absolute error, estimated in a 10-fold cross validation, repeated 10 times to improve the replicability of the experiment. To avoid underestimation, the standard deviation of error was estimated conservatively, by taking into account the dependence between the subsequent testing samples in the repeated cross-validation, as described in (Nadeau & Bengio, 2003). The results of both average accuracy and standard deviation are given in Table 2. For each dataset, the best method, and all methods within one standard error below the best, are marked with bold.

Judging by the number of times LPRules is among the best classifiers, we can conclude that our algorithm outperforms the other methods tested in the experiment. In particular, an improvement in accuracy was obtained over the regular classifiers which do not take into account the domain knowledge. A similar thing can be observed also for IS algorithm. This supports our hypothesis, that adapting the learning algorithm to deal with monotonicity constraints results in improved prediction accuracy. The exception was OLM, which achieved surprisingly poor results, but this may be due to the particular implementation that was used.

A separate issue is a matter of comprehensibility of our approach. A reasonable measure of comprehensibility would be the size of the ensemble, i.e. the number of rules with nonzero coefficient a_t. In Table 3 an average number of rules per binary subproblem is shown. In most cases only a few rules are needed, which shows that our method produces very concise models, simple to interpret by a domain expert.

Table 2. Experimental results: for each dataset, and each classifier, two values are given: mean absolute error (above) and standard deviation of error (below, starting with ±). For each dataset, the best method and all methods within one standard error below the best are marked with bold.

Dataset	OLM	IS	LPRules	J48	SVM
DenBosch	0.282	**0.183**	**0.168**	**0.172**	**0.202**
	±0.039	±0.037	±0.034	±0.032	±0.036
ESL	0.371	**0.328**	**0.323**	0.369	0.355
	±0.024	±0.023	±0.024	±0.022	±0.023
SWD	**0.452**	**0.442**	**0.435**	**0.442**	**0.435**
	±0.017	±0.018	±0.017	±0.016	±0.016
LEV	0.427	**0.398**	**0.396**	0.415	0.444
	±0.018	±0.017	±0.016	±0.018	±0.016
ERA	1.256	1.271	1.263	**1.217**	1.271
	±0.031	±0.034	±0.033	±0.032	±0.029
Housing	0.527	**0.286**	**0.274**	0.332	0.314
	±0.032	±0.02	±0.021	±0.023	±0.025
CPU	0.29	0.099	**0.073**	0.1	0.371
	±0.035	±0.02	±0.018	±0.019	±0.03
Balance	0.224	0.19	**0.063**	0.271	0.137
	±0.02	±0.017	±0.009	±0.021	±0.017
Windsor	0.576	0.52	**0.516**	0.565	**0.491**
	±0.028	±0.028	±0.026	±0.025	±0.026
Car	0.084	0.045	**0.03**	0.09	0.078
	±0.01	±0.006	±0.004	±0.008	±0.007

Table 3. Average numbers of rules per binary problem.

Data set	# rules	Data set	# rules
ESL	3	Windsor	19.1
SWD	6.7	DenBosch	4.7
LEV	3.2	Car	5.2
ERA	2.6	CPU	4.1
Housing	26.6	Balance	32.2

8. Summary

We presented LPRules, a rule learning algorithm for classification problem in the presence of monotonicity constraints. The algorithm is based on the statistical analysis of the problem, which relates monotonicity constraints to the constraints imposed on the probability distribution. LPRules consists of three phases: first, a general multi-class problem is decomposed into a sequence of binary subproblems; next, the data for each binary subproblem is monotonized using a non-parametric approach exploiting the class of all monotone functions; finally, a rule ensemble is generated using LPBoost method, so that it makes no errors on the monotonized data. The ideas standing behind all

three phases were theoretically justified. In particular, we proved that in the presence of monotonicity constraints, the risk of the ensemble separating the data with large margin remains close to the accuracy of Bayes classifier. The performance of our method was successfully verified in the computational experiment.

References

Anglin, P., & Gencay, R. (1996). Semiparametric estimation of a hedonic price function. *Journal of Applied Econometrics*, *11*, 633–648.

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Barile, N., & Feelders, A. (2008). Nonparametric monotone classification with MOCA. *Proc. of International Conference on Data Mining (ICDM'08)*.

Ben-David, A. (1995). Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, *19*, 29–43.

Ben-David, A., Sterling, L., & Pao, Y.-H. (1989). Learning and classification of monotonic ordinal concepts. *Computational Intelligence*, *5*, 45–49.

Cao-Van, K., & De Baets, B. (2004). An instance-based algorithm for learning rankings. *Proceedings of Benelearn* (pp. 15–21).

Chandrasekaran, R., Ryu, Y. U., Jacob, V. S., & Hong, S. (2005). Isotonic separation. *INFORMS Journal on Computing*, *17*, 462–474.

Daniels, H., & Kamp, B. (1999). Applications of MLP networks to bond rating and house pricing. *Neural Computation and Applications*, *8*, 226–234.

Dembczyński, K., Greco, S., Kotłowski, W., & Słowiński, R. (2008). Ensemble of decision rules for ordinal classification with monotonicity constraints. *LNAI*, *5009*, 260–267.

Demiriz, A., Bennett, K. P., & Shawe-Taylor, J. (2001). Linear programming boosting via column generation. *Machine Learning Journal*, *46*, 225–254.

Devroye, L., Györfi, L., & Lugosi, G. (1996). *A probabilistic theory of pattern recognition*. Springer.

Dykstra, R., Hewett, J., & Robertson, T. (1999). Nonparametric, isotonic discriminant procedures. *Biometrica*, *86*, 429–438.

Frank, E., & Hall, M. (2001). A simple approach to ordinal classification. *Proc. 12th European Conference on Machine Learning* (pp. 145–157). Springer.

Gamarnik, D. (1998). Efficient learning of monotone concepts via quadratic optimization. *Conference on Computational Learning Theory* (pp. 134–143).

Greco, S., Matarazzo, B., & Słowiński, R. (2001). Rough sets theory for multicriteria decision analysis. *European J. of Operational Research*, *129*, 1–47.

Greco, S., Matarazzo, B., & Słowiński, R. (2007). Customer satisfaction analysis based on rough set approach. *Zeitschrift für Betriebswirtschaft*, *16*, 325–339.

Herbrich, R., Graepel, T., & Obermayer, K. (1999). *Regression models for ordinal data: A machine learning approach* (Technical Report). Technical University of Berlin.

Koltchinskii, V., & Panchenko, D. (2002). Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, *30*, 1–50.

Kotłowski, W., Dembczyński, K., Greco, S., & Słowiński, R. (2008). Stochastic dominance-based rough set model for ordinal classification. *Information Sciences*, *178*, 3989–4204.

Kotłowski, W., & Słowiński, R. (2008). Statistical approach to ordinal classification with monotonicity constraints. *Preference Learning ECML/PKDD 2008 Workshop*.

Lin, H.-T., & Li, L. (2007). Ordinal regression by extended binary classifications. *Advances in Neural Information Processing Systems*, *19*, 865–872.

Nadeau, C., & Bengio, Y. (2003). Inference for the generalization error. *Mach. Learn.*, *52*, 239–281.

Potharst, R., & Feelders, A. J. (2002). Classification trees for problems with monotonicity constraints. *SIGKDD Explorations*, *4*, 1–10.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, *26*, 1651–1686.

Sill, J. (1998). Monotonic networks. *Advances in Neural Information Processing Systems* (pp. 661–667). Denver, USA: The MIT Press.

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques, 2nd edition*. Morgan Kaufmann, San Francisco.