Learning Conditional Lexicographic Preference Trees

Michael Bräuning and **Eyke Hüllermeier**[1]

Abstract. We introduce a generalization of lexicographic orders and argue that this generalization constitutes an interesting model class for preference learning in general and ranking in particular. We propose a learning algorithm for inducing a so-called conditional lexicographic preference tree from a given set of training data in the form of pairwise comparisons between objects. Experimentally, we validate our algorithm in the setting of multipartite ranking.

1 INTRODUCTION

Preference learning is an emerging subfield of machine learning that has received increasing attention in recent years [13]. A specific though important special case of preference learning is "learning to rank", that is, the learning of models that can be used to predict preferences in the form of rankings of a set of alternatives [7, 8]. Ranking problems are often reduced to problems of a simpler type, such as learning a value function that assigns scores to alternatives (with better alternatives having higher scores) or learning a binary predicate that compares pairs of alternatives [15]; while the former approach is close to regression, the latter is in the realm of classification learning.

Another approach to learning ranking functions is to proceed from specific model assumptions, that is, assumptions about the structure of the sought preference relations. This approach is less generic than the previous ones, as it strongly depends on the concrete assumptions made. On the other hand, it typically offers the advantage of being more easily understandable and interpretable. An example is the representation of preferences in the form of a CP-net [5]. Another example is lexicographic orders that are widely accepted as a plausible representation of (human) preferences [16], especially in complex decision making domains [1]. Here, the assumption is that the target ranking of a set of alternatives, each one described in terms of multiple attributes, can be represented as a lexicographic order.

From a machine learning point of view, assumptions of the above type can be seen as an inductive bias restricting the hypothesis space. Provided the bias is correct, this is clearly an advantage, as it may simplify the learning problem. On the other hand, an overly strong bias may prevent the learner from approximating the target ranking sufficiently well. For example, while being plausible in some situations, the assumption of a lexicographic order will be too restrictive for many applications.

In this paper, we therefore present a method for learning generalized lexicographic orders. While still being simple and easy to understand, the model class we consider relaxes some of the assumptions of a proper lexicographic order. More specifically, we increase flexibility thanks to two extensions of conventional lexicographic orders.

[1] Department of Mathematics and Computer Science, University of Marburg, Germany, email: {braeunim,eyke}@mathematik.uni-marburg.de

- First, we allow for *conditioning* [3, 4]: The importance of attributes as well as the preferences for the values of an attribute may depend on the values of other variables preceding that one in the underlying variable order.
- Second, we allow for *grouping* [17]: Several (one-dimensional) variables can be grouped into a single high-dimensional variable, and preferences can be specified on the cartesian product of the corresponding domains.

The remainder of this paper is organized as follows. In the next section, we give a brief overview of related work. In Section 3, we introduce generalized lexicographic orders and the notion of conditional lexicographic preference trees. In Section 4, we present an algorithm for learning such preference models from data. An experimental study is presented in Section 5, prior to concluding the paper in Section 6.

2 RELATED WORK

The use of lexicographic orders in preference modeling has already been considered in the seventies of the last century [10], whereas in machine learning, this type of structure has attracted attention only recently. Flach and Matsubara developed a lexicographic ranker called LexRank, using a linear preference ordering on attributes derived by the odds ratio [12, 11]. Experimentally, they show that LexRank is competitive to decision trees and naive Bayes in terms of ranking performance (AUC).

Further work on learning lexicographic orders was done by Schmitt and Martignon [16], Dombi et al. [9], and Yaman et al. [18]. However, these works are based on rather simplistic assumptions. More general models were studied by Booth et al. [3, 4], and in fact, important parts of our approach (such as conditional importance of attributes and conditional preferences on attribute values) is inspired by these models. Their work remains rather theoretical, however, without a practical realization in terms of an implementation of algorithms or an experimental study with real data.

3 GENERALIZED LEXICOGRAPHIC ORDERS

Formally, we proceed from an attribute-value representation of decision alternatives or objects, i.e., an object is represented as a vector

$$\boldsymbol{o} \in \mathcal{O} = \mathcal{D}(V) = \mathcal{D}(A_1) \times ... \times \mathcal{D}(A_n),$$

where $V = \{A_1, ..., A_n\}$ is the set of attributes (variables) and $\mathcal{D}(A_i)$ is the domain of attribute A_i. For a subset $A = \{A_{i_1}, \ldots, A_{i_k}\} \subset V$ of attributes we define $\mathcal{D}(A) = \mathcal{D}(A_{i_1}) \times ... \times \mathcal{D}(A_{i_k})$.

An *assignment* or *instantiation* of a subset $A \subseteq V$ of attributes is an element $\boldsymbol{a} \in \mathcal{D}(A)$; an assignment is called *complete* if $A =$

V, otherwise it is called *partial*. For an object $\boldsymbol{o} \in \mathcal{O}$ and a subset $A \subset V$, we denote by $\boldsymbol{o}[A]$ the projection of $\boldsymbol{o}$ from $\mathcal{D}(V)$ to $\mathcal{D}(A)$; if $A = \{A_k\}$ is a single attribute, we also write $\boldsymbol{o}[k]$ instead of $\boldsymbol{o}[\{A_k\}]$.

A lexicographic order on $\mathcal{O}$ is a total order $\succ$ defined in terms of

- a total order $\sqsupset$ on V, i.e., a ranking of the attributes,
- a total order $\sqsupset_i$ on each attribute domain $\mathcal{D}(A_i)$.

More specifically, $\boldsymbol{o}^* \succ \boldsymbol{o}$ (suggesting that $\boldsymbol{o}^*$ is preferred to $\boldsymbol{o}$) if and only if there exists a $k \in \{1, \ldots, n\}$ such that

$$\big(\boldsymbol{o}^*[k] \sqsupset_k \boldsymbol{o}[k]\big) \wedge \Big((A_i \sqsupset A_k) \Rightarrow \big(\boldsymbol{o}^*[i] = \boldsymbol{o}[i]\big)\Big)$$

for all $i \in \{1, \ldots, n\}$. The relations $\sqsupset_i$ indicate preference on individual attributes: $a \sqsupset_i b$ means that, for $a, b \in \mathcal{D}(A_k)$, a is preferred to b as a value for attribute A_i. Moreover, the relation $\sqsupset$ reflects the importance of attributes: $A_i \sqsupset A_j$ means that attribute A_i is more important than A_j, whence the former is considered prior to the latter. Without loss of generality, we shall subsequently assume that $A_1 \sqsupset A_2 \sqsupset \cdots \sqsupset A_n$ (unless otherwise stated).

3.1 Conditional preferences on attribute values

Conventional lexicographic orders assume that preferences $\sqsupset_k$ on attribute domains are independent of each other. Needless to say, this assumption is often violated in practice. For example, although it is possible that a person prefers red wine to white wine *in general*, it is also plausible that her preference for wine may depend on the main dish: red is preferred to white in the case of meat, whereas white is preferred to red in the case of fish.

In order to capture attribute dependencies of that type, the preferences relations $\sqsupset_k$ can be conditioned on the values of the attributes A_j preceding A_k in the order $\sqsupset$ [3, 4]. That is, $\sqsupset_k$ is now replaced by a set of strict orders

$$\Big\{ \sqsupset_k^{(a_1,\ldots,a_{k-1})} \mid (a_1, \ldots, a_{k-1}) \in \mathcal{D}(\{A_1, \ldots, A_{k-1}\}) \Big\}$$

Moreover, the order relation $\succ$ on $\mathcal{O}$ is then defined as follows: $\boldsymbol{o}^* \succ \boldsymbol{o}$ for $\boldsymbol{o}^* = (a_1^*, \ldots, a_n^*)$ and $\boldsymbol{o} = (a_1, \ldots, a_n)$ if and only if there exists a $k \in \{1, \ldots, n\}$ such that

$$\Big(\forall\, i \in \{1, \ldots, k-1\} : a_i^* = a_i\Big) \wedge \big(a_k^* \sqsupset_k^{(a_1,\ldots,a_{k-1})} a_k\big).$$

3.2 Conditional attribute importance

Going one step further, one may assume that the values of the first attributes in the attribute order $\sqsupset$ do not only influence the preferences on the values of the attributes that follow, but also the importance of the attributes themselves [3, 4]. Thus, we are no longer dealing with a lexicographic order in the sense that $\sqsupset$ defines a *sequence* of the attributes V according to their importance. Instead, we are dealing with a *tree-like* structure. This structure is defined by the following (choice) function:

$$A = C\Big((A_{i_1}, A_{i_2}, \ldots, A_{i_k}), (a_{i_1}, a_{i_2}, \ldots, a_{i_k})\Big),$$

where $(A_{i_1}, A_{i_2}, \ldots, A_{i_k}) \in V^k$ is a sequence of attributes (such that $A_{i_j} \neq A_{i_k}$ for $j \neq k$) and $a_{i_j} \in \mathcal{D}(A_{i_j})$ for all $j \in \{1, \ldots, k\}$. Moreover, $A \in V \setminus \{A_{i_1}, \ldots, A_{i_k}\}$ is the most important attribute given that $A_{i_j} = a_{i_j}$ for all $j \in \{1, \ldots, k\}$.

3.3 Variable grouping

Another extension consists of grouping several variables, that is, to allow the expression of preferences on *attribute tuples* instead of single attributes only [17]. Formally, this means selecting an index set $\mathcal{I} \subseteq \{1, \ldots, n\}$ and defining a total order relation $\sqsupset_\mathcal{I}$ on the cartesian product $\mathcal{D}(V_\mathcal{I})$ of the domains $\mathcal{D}(A_i)$, $i \in \mathcal{I}$.

Note that the possibility of variable grouping significantly increases the expressivity of the model class. In particular, taking $\mathcal{I} = \{1, \ldots, n\}$, it is possible to define every order on $\mathcal{D}(V)$, that is, to sort the set of alternatives in any way. Since this level of expressivity is normally not desirable, it is reasonable to restrict to variable grouping of order g_{max}, meaning to impose the constraint $|\mathcal{I}| \leq g_{max}$ for a fixed $g_{max} \leq n$.

3.4 Conditional lexicographic preference trees

Combining the generalizations discussed above, we end up with what we call a Conditional Lexicographic Preference Tree (CLPT). Graphically, this is a tree structure in which

- every node is labeled with a subset of attributes $V_\mathcal{I}$ and a total order on the cartesian product $\mathcal{D}(V_\mathcal{I})$ of the corresponding attribute domains $\mathcal{D}(A_i)$, $i \in \mathcal{I}$;
- there is one outgoing edge (descendant node) for each value $\boldsymbol{o}[V_\mathcal{I}] \in \mathcal{D}(V_\mathcal{I})$;
- every attribute $A_i \in V$ occurs at most once on each branch from the root of the tree to a leaf node (i.e., the index sets $\mathcal{I}$ along a branch are disjoint).

We call a CLPT *complete* if every attribute $A_i \in V$ occurs exactly once on each branch from the root of the tree to a leaf node (i.e., the index sets $\mathcal{I}$ along a branch form a partition of $\{1, \ldots, n\}$).

A (complete) CLPT can be thought of a defining an order relation on $\mathcal{O}$ through recursive refinement of a weak order $\succeq$, that is, by refining an order relation with tie groups in a recursive manner (in the following, $\sim$ and $\succ$ denote, respectively, the symmetric and asymmetric part of $\succeq$):

- One starts with a single equivalence class (tie group), i.e., $\boldsymbol{o}^* \sim \boldsymbol{o}$ for all $\boldsymbol{o}^*, \boldsymbol{o} \in \mathcal{O}$.
- Let the root of the CLPT be labeled with the attribute set $V_\mathcal{I}$, and let $\sqsupset_\mathcal{I}$ denote the corresponding order on $\mathcal{D}(V_\mathcal{I})$. The current order $\succeq$ is then refined by letting $\boldsymbol{o}^* \succ \boldsymbol{o}$ whenever $\boldsymbol{o}^*[V_\mathcal{I}] \sqsupset_\mathcal{I} \boldsymbol{o}[V_\mathcal{I}]$; otherwise, if $\boldsymbol{o}^*[V_\mathcal{I}] = \boldsymbol{o}[V_\mathcal{I}]$, then $\boldsymbol{o}^*$ and $\boldsymbol{o}$ remain tied.
- Thus, a linear order of tie groups (equivalence classes) is produced.
- Each equivalence class (represented by a value $\boldsymbol{a} \in \mathcal{D}(V_\mathcal{I})$) is then recursively refined by the subtree the objects of this equivalence class are passed to.

Note that, if the CLPT is complete, the order relation $\succeq$ eventually produced is a total order $\succ$.

4 LEARNING CLPTs

In this section, we outline a method for inducing a CLPT from training data

$$\mathcal{T} = \big\{(\boldsymbol{o}_i^*, \boldsymbol{o}_i)\big\}_{i=1}^N \tag{1}$$

that consists of a set of object pairs $(\boldsymbol{o}_i^*, \boldsymbol{o}_i) \in \mathcal{O}^2$, suggesting that $\boldsymbol{o}_i^*$ is preferred to $\boldsymbol{o}_i$. Roughly speaking, this means finding a CLPT whose induced order relation $\succeq$ on $\mathcal{O}$ is as much as possible in

agreement with the pairwise preferences in $\mathcal{T}$ (without overfitting the training data). The induced order relation $\succeq$ is a total order $\succ$ if the CLPT is complete.

4.1 Performance and evaluation measures

In order to evaluate the predictive performance of a CLPT, there is a need to compare the order relation $\succeq$ (with asymmetric part $\succ$) induced by this model with a ground truth order $\succ *$. As will be seen below, the same measures can be used to fit a CLPT to a given set of training data (1) during the training phase. In this case, the "ground truth" is not a total order but a set of pairwise comparisons between objects. Since a total order $\succ^*$ can be decomposed into (a quadratic number of) such comparisons, too, we can assume (without loss of generality) that we compare $\succeq$ with a set $\mathcal{T}$ of pairs $(\boldsymbol{o}^*, \boldsymbol{o}) \in \mathcal{O}^2$, suggesting that $\boldsymbol{o}^*$ should be ranked higher than $\boldsymbol{o}$.

Inspired by the corresponding notions introduced in [6], we define two performance measures of *correctness* and *completeness*, respectively, as follows:

$$\mathrm{CR}(\succeq, \mathcal{T}) = \frac{|C| - |D|}{|C| + |D|}, \tag{2}$$

$$\mathrm{CP}(\succeq, \mathcal{T}) = \frac{|C| + |D|}{|\mathcal{T}|}, \tag{3}$$

where

$$C = \Big\{(\boldsymbol{o}^*, \boldsymbol{o}) \in \mathcal{T} \,|\, \boldsymbol{o}^* \succ \boldsymbol{o}\Big\},$$
$$D = \Big\{(\boldsymbol{o}^*, \boldsymbol{o}) \in \mathcal{T} \,|\, \boldsymbol{o} \succ \boldsymbol{o}^*\Big\}.$$

Note that $\mathrm{CR}(\succeq, \mathcal{T})$ assumes values between -1 (complete disagreement) and $+1$ (complete agreement), while $\mathrm{CP}(\succeq, \mathcal{T})$ ranges between 0 (no comparisons) and 1 (full comparison).

4.2 A greedy learning procedure

We implement an algorithm for learning a CLPT as a (greedy) search in the space of tree structures based on the greedy algorithms presented by Schmitt and Martignon [16] as well as Booth et al. [3, 4]. This is done by constructing the tree from the root to the leaves in a recursive manner. In each step of the recursion, a new node is created with an associated subset $V_\mathcal{I}$ of attributes, where $|V_\mathcal{I}| \leq g_{max}$, and a total order $\sqsupset_\mathcal{I}$ on $\mathcal{D}(V_\mathcal{I})$.

4.2.1 Creating a node

The problem to be solved in each recursion is the following: Given a set of pairwise comparisons $\mathcal{T}$ and a set $V' \subseteq V$ of attributes still available, select a most suitable subset $V_\mathcal{I} \subseteq V'$ and an order $\sqsupset_\mathcal{I}$. Following a greedy strategy, we choose $(V_\mathcal{I}, \sqsupset_\mathcal{I})$ so as to maximize correctness (2), using completeness (3) as a second criterion to break ties.

The selection of an attribute subset $V_\mathcal{I}$ can be done through exhaustive search if its size is sufficiently limited, i.e., if the upper bound g_{max} is small. Otherwise, a complete enumeration of all possibilities may become too expensive. Moreover, for each candidate subset $V_\mathcal{I}$, a total order $\sqsupset_\mathcal{I}$ needs to be determined. Again, all such orders can be tried if $\mathcal{D}(V_\mathcal{I})$ is not too large. Otherwise, heuristic ranking procedures such as Borda count can be used (counting the number of "wins" and "losses" of each value $\boldsymbol{a} \in \mathcal{D}(V_\mathcal{I})$ in the training data $\mathcal{T}$ and sorting according to the difference).

4.2.2 Limiting the number of candidate subsets

In order to avoid a complete enumeration of all candidate subsets $V_\mathcal{I}$ of size $\leq g_{max}$, we combine a greedy search with a kind of lookahead procedure: We provisionally create a node by selecting a single attribute instead of a subset, i.e., we tentatively set g_{max} to 1; apart from that, exactly the same selection procedure (as outlined above) is applied. This step is repeated g_{max} times, thereby producing a subtree of depth g_{max}. Let $V^* \subseteq V$ denote the subset of attributes that occur in this subtree, i.e., that are chosen in at least one of the nodes. Then, as candidate subsets $V_\mathcal{I}$, we only try subsets V^*, i.e., subsets $V_\mathcal{I} \subseteq V^*$ such that $|V_\mathcal{I}| \leq g_{max}$. Obviously, the underlying assumption is that an attribute that has not been chosen in any of the g_{max} steps is not important at this point.

4.2.3 Recursion

Once an optimal subset $V_\mathcal{I}$ has been chosen, the training examples $(\boldsymbol{o}^*, \boldsymbol{o})$ with $\boldsymbol{o}^*[V_\mathcal{I}] \neq \boldsymbol{o}[V_\mathcal{I}]$ are removed from $\mathcal{T}$ (since they are sorted at this node). Moreover, for each value $\boldsymbol{a} \in \mathcal{D}(V_\mathcal{I})$, a data set

$$\mathcal{T}_{\boldsymbol{a}} = \Big\{(\boldsymbol{o}^*, \boldsymbol{o}) \in \mathcal{T} \,|\, \boldsymbol{o}^*[V_\mathcal{I}] = \boldsymbol{o}[V_\mathcal{I}] = \boldsymbol{a}\Big\}$$

is created and passed to the corresponding successor node (together with $V' \setminus V_\mathcal{I}$ as the attributes that have not been used so far). The same recursive procedure is then applied to each of these successor nodes.

4.2.4 Initialization and termination

The learning procedure is called with the original training set $\mathcal{T}$ and the full set V of attributes as candidates. The recursion terminates if no attribute is left ($V' = \emptyset$) or if the set of training examples is empty ($\mathcal{T} = \emptyset$).

4.2.5 CLeRa

We call the algorithm outlined above *CLeRa*, which is short for Conditional Lexicographic Ranker. The CLPT induced by CLeRa can be used to compare new object pairs $\{\boldsymbol{o}^*, \boldsymbol{o}\} \subset \mathcal{O}$. To this end, the tuple is submitted to the root and propagated through the tree until either a leaf node is reached or a node at which $\boldsymbol{o}^*[V_\mathcal{I}] \neq \boldsymbol{o}[V_\mathcal{I}]$; in this case, $\boldsymbol{o}^* \succ \boldsymbol{o}$ is decided if $\boldsymbol{o}^* \sqsupset_\mathcal{I} \boldsymbol{o}$ and $\boldsymbol{o} \succ \boldsymbol{o}^*$ if $\boldsymbol{o} \sqsupset_\mathcal{I} \boldsymbol{o}^*$. Otherwise, if $\boldsymbol{o}^*[V_\mathcal{I}] = \boldsymbol{o}[V_\mathcal{I}]$ in all nodes traversed by the two objects, then $\boldsymbol{o}^* \sim \boldsymbol{o}$.

Given not only a pair but a complete set of objects to be ranked, the pairwise comparison realized by the CLPT can be embedded in any standard sorting algorithm, such as insertion sort. Note that, since $\boldsymbol{o}^* \sim \boldsymbol{o}$ is possible in a pairwise comparison, the result of the sorting procedure will in general only be a weak order $\succeq$.

5 EXPERIMENTAL RESULTS

We evaluate our approach on 15 benchmark data sets from the Statlog and the UCI repository [2]. These data sets, which define binary or ordinal classification problems, were pre-processed as follows: numerical attributes and attributes with more than five values were discretized into four values using equal frequency binning. Moreover, instances with missing values were neglected.

The learning problem we consider is multipartite ranking [14]: Given a set of test instances $X \subset \mathcal{O}$, the goal is to predict a ranking

$\succeq$ that agrees with the (ordered) class labels of these instances. Formally, this agreement is measured in terms of the so-called C-index, which can be seen as an extension of the AUC:

$$C = \frac{1}{\sum_{i<j} n_i n_j} \sum_{1 \leq i < j \leq m} \sum_{(\boldsymbol{o}, \boldsymbol{o}^*) \in X_i \times X_j} \mathbb{I}(\boldsymbol{o}^* \succ \boldsymbol{o}) + \frac{1}{2}\mathbb{I}(\boldsymbol{o}^* \sim \boldsymbol{o}),$$

where $X_i \subseteq X$ denotes the set of instances with class labels y_i, and these class labels are assumed to have the order $y_1 < y_2 < \cdots < y_m$. The training data consists of a set of labeled instances, just like in classification. Since CLeRa is learning from pairwise comparisons of the form $(\boldsymbol{o}^*, \boldsymbol{o})$, it first extracts such comparisons from the original data by looking at the class information: A preference $(\boldsymbol{o}^*, \boldsymbol{o})$ is generated for each pair $(\boldsymbol{o}^*, y_j)$ and $(\boldsymbol{o}, y_i)$ of labeled instances in the (original) training data such that $y_i < y_j$.

The ranking performance of CLeRa (with maximum grouping size of $g_{max} = 2$) is compared with LexRank, which was implemented as proposed by Flach and Matsubara [12, 11]; therefore, this method was only applied to binary (two-class) problems but not to problems with more than two classes.[2] We applied naive Bayes (NB) and decision tree (J48) learning as additional baselines, using the standard implementations in Weka (trees are not pruned) and sorting instances according to the estimated probability of the positive class; note that these methods are not applicable to the multi-class case either.

Table 1. Average performance in terms of C-index based on a 10-fold cross-validation.

Dataset	CLeRa	LexRank	J48	NB
Red Wine	0.7827	0.8011	0.7378	0.8110
Census Income	0.7952	0.5776	0.7401	0.8607
Credit Approval	0.9201	0.9229	0.8517	0.9061
Mammographic Mass	0.8831	0.8960	0.8524	0.8999
Mushroom	1.000	0.9865	1.0000	0.9484
SPECT Heart	0.674	0.6590	0.5106	0.7409
Ionosphere	0.9198	0.5748	0.8059	0.9061
MAGIC Gamma Telescope	0.8218	0.7263	0.7841	0.8241
Breast Cancer Wisconsin	0.9837	0.9901	0.9793	0.9909
German Credit	0.6285	0.4523	0.6251	0.7835
Car Evaluation	0.9198	n/a	n/a	n/a
Nursery	0.9052	n/a	n/a	n/a
Tic-Tac-Toe Endgame	0.7728	n/a	n/a	n/a
Vehicle	0.7554	n/a	n/a	n/a
Cardiocraphic	0.9551	n/a	n/a	n/a

The results of a 10-fold cross-validation are given in Table 1. Since CLeRa produced a completeness of 1 or extremely close to 1 throughout, these values are not reported here. Overall, the performance of the methods is quite comparable. In particular, CLeRa and LexRank produce quite similar results on many data sets. In some cases, however, the results are strongly in favor of CLeRa (Census Income, Ionosphere, MAGIC Gamma Telescope, German Credit). Probably, this is because the bias imposed by the assumption of a standard lexicographic order is inadequate for these data sets, and hence our extensions (conditional attribute importance, conditional value preferences, variable grouping) clearly pay off.

6 CONCLUSIONS AND FUTURE WORK

Lexicographic orders constitute an interesting model class for preference learning, which allows for representing rankings of a set of objects in a very compact and comprehensible way. Yet, as we have argued in this paper, this model class may not be flexible enough for many real-world applications. Therefore, we have proposed to weaken the assumptions underlying a lexicographic order in various directions, allowing for conditional attribute importance, conditional preferences on attribute values, and variable grouping. Moreover, we have proposed an algorithm called CLeRa, which learns preference models in the form of conditional lexicographic preference trees from training data in the form of pairwise comparisons between objects.

First experimental results in the setting of multipartite ranking are quite promising and show CLeRa to be competitive with other methods. In a direct comparison with an existing lexicographic ranker, the benefit of our extensions are becoming quite obvious.

Important topics of future work can be found both on the theoretical and practical side. In particular, we are currently studying formal properties of our generalized model class, such as its expressiveness and means for regularization and complexity control. Practically, there is certainly scope for improving our current algorithm, for example by devising a suitable procedure for estimating an optimal value g_{max} for the oder of variable grouping. Moreover, improving the computational efficiency of CLeRa would be desirable, too. Last but not least, we are of course interested in real applications for which (generalized) lexicographic models appear to be an adequate representation.

[2] The red wine data actually has a target attribute with values between 1 and 10; it was binarized by thresholding at the median.

REFERENCES

[1] M. Ahlert, 'Aggregation of lexicographic orderings', *Homo Oeconomicus*, **25(3/4)**, 301–317, (2008).

[2] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[3] R. Booth, Y. Chevaleyre, J. Lang, J. Mengin, and C. Sombattheera, 'Learning various classes of models of lexicographic orderings', in *Workshop on Preferencce Learning at ECML-2009*, (2009).

[4] R. Booth, Y. Chevaleyre, J. Lang, J. Mengin, and C. Sombattheera, 'Learning conditionally lexicographic preference relations', in *Proc. ECAI 2010*, pp. 269–274, Amsterdam, The Netherlands, (2010). IOS Press.

[5] C. Boutilier, R.I. Brafman, C. Domshlak, H.H. Hoos, and D. Poole, 'CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements', *Journal of Artificial Intelligence*, **21**, 135–191, (2004).

[6] W. Cheng, M. Rademaker, B. De Baets, and E. Hüllermeier, 'Predicting partial orders: Ranking with abstention', in *Proc. ECML-2010*, (2010).

[7] William Cohen, Robert Schapire, and Yoram Singer, 'Learning to order things', *Journal of Artificial Intelligence Research*, **10**, 243–270, (1999).

[8] O. Dekel, C. Manning, and Y. Singer, 'Log-linear models for label ranking', in *Advances in Neural Information Processing Systems*, (2003).

[9] J. Dombi, C. Imreh, and N. Vincze, 'Learning lexicographic orders', *European Journal of Operational Research*, **183**(2), 748–756, (2007).

[10] P.C. Fishburn, 'Lexicographic orders, utilities and decision rules: A survey', *Management Science*, **20**(11), 1442–1471, (July 1974).

[11] P.A Flach and E.T. Matsubara, 'On classification, ranking and probability estimation', in *Probabilistic, Logical and Relational Learning - A Further Synthesis*, (2007).

[12] P.A. Flach and E.T. Matsubara, 'A simple lexicographic ranker and probability estimator', in *Proceedings of the 18th European conference on Machine Learning*, ECML '07, pp. 575–582, Berlin, Heidelberg, (2007). Springer-Verlag.

[13] J. Fürnkranz and E. Hüllermeier, *Preference Learning*, Springer-Verlag, Berlin, Heidelberg, 2010.

[14] J. Fürnkranz, E. Hüllermeier, and S. Vanderlooy, 'Binary decomposition methods for multipartite ranking', in *Proceedings ECML/PKDD–2009, European Conference on Machine Learning and Knowledge Discovery in Databases*, Bled, Slovenia, (2009).

[15] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker, 'Label ranking by learning pairwise preferences', *Artificial Intelligence*, **172**, 1897–1917, (2008).

[16] M. Schmitt and L. Martignon, 'On the complexity of learning lexicographic strategies', *J. Mach. Learn. Res.*, **7**, 55–83, (December 2006).
[17] N. Wilson, 'Efficient inference for expressive comparative preference languages', in *Proc. IJCAI-09*, (2009).
[18] F. Yaman, T.J. Walsh, M.L. Littman, and M. desJardins, 'Democratic approximation of lexicographic preference models', in *Proc. ICML-08*, pp. 1200–1207, Helsinki, Finland, (2008).