Learning and Ensembling Lexicographic Preference Trees with Multiple Kernels

Kelwin Fernandes
INESC TEC, Porto, Portugal
Faculty of Engineering, University of Porto
Email: kafc@inesctec.pt

Jaime S. Cardoso
INESC TEC, Porto, Portugal
Faculty of Engineering, University of Porto
Email: jaime.cardoso@inesctec.pt

Hector Palacios
Universitat Pompeu Fabra,
Barcelona, Spain
Email: hector.palacios@upf.edu

***Abstract*—We study the problem of learning lexicographic preferences on multiattribute domains, and propose *Rankdom Forests* as a compact way to express preferences in learning to rank scenarios. We start generalizing Conditional Lexicographic Preference Trees by introducing multiple kernels in order to handle non-categorical attributes. Then, we define a learning strategy for inferring lexicographic rankers from partial pairwise comparisons between options. Finally, a Lexicographic Ensemble is introduced to handle multiple weak partial rankers, being *Rankdom Forests* one of these ensembles. We tested the performance of the proposed method using several datasets and obtained competitive results when compared with other lexicographic rankers.**

I. Introduction

Preference learning is an inductive learning task concerned about inferring underlying preference models given partially declared preferences [1]. In Artificial Intelligence, preferences are of critical importance, as they express agent's desires in a declarative fashion [2]. The choices of an agent that acts rationally are driven by an underlying preference model [2]. Therefore, being able to infer models that reflect user's preferences is a key issue of recommending decisions or actions [2], [3].

It is particularly interesting to learn ranks in combinatorial domains, where the options to rank are represented by a set of variables assigned to values. They are also studied in combinatorial preference aggregation [4], a well studied problem in computational social choice, an area dealing with the computational aspects of social choice problems like voting. The main challenge is to have a language that allows compact representation of a number of options which are exponential in the number of variables. Any chosen language would have pros and cons. More compact languages are usually less expressive and *vice versa*, and there are also complexity issues [5].

Learning to Rank in combinatorial domains [6] has become a trendy topic in recent years due to the growing number of applications involving the prediction of structured preference data instead of traditional classification and regression tasks. For example, search engines [7] receive a query and return a set of objects. Rankings are useful to present such objects according to their relevance. Good rankings can be obtained from the query and the previous interaction of the user with the engine's results, that can be translated into their preferences, characterized by a number of attributes. Recommender systems and subjective image quality assessment can benefit as well from learnt ranks [8], [9].

Learning to Rank methods can be divided, according to their input representation, in three types: pointwise, pairwise and listwise [10]. Pointwise methods rely on assigning a numerical (e.g. [0, 1]) or ordinal (e.g. *poor, fair, good* or *excellent*) score for each observation. Pointwise methods reduce the learning to rank problem to traditional regression[11], classification [12], and ordinal regression/classification [13] tasks. In contrast, Pairwise models rely on comparing pair of observations [6], [14]–[16]. Finally, the listwise paradigm deals directly with the ordering of the entire set of entities associated to a given query [17]. All these three approaches present advantages and disadvantages, both in terms of the accuracy and workload associated with the training set acquisition, and in terms of their adequacy to different scenarios.

The scope of this work is limited to the area of pairwise rankers, which can be further subdivided into scoring (SR) and non-scoring (NSR) rankers. Assuming that the learned ranker can be represented by a binary function $f(a, b)$ that decides the relative order between observations a and b, a SR decides which observation is better by comparing the score assigned by a pointwise (unary) function s with monotonous outcome, $f(a, b) = s(a) > s(b)$. Indeed, SR project the options into a linear space. Examples of these models are GBRank [18] and RankSVM [14]. Pointwise rankers and pairwise SR assume that the underlying preference model can be understood as a utility function. However, whilst it is always possible to transform a utility function into a preference model, it is not the case in the opposite direction [19]. In contrast, NSR cannot be directly obtained from pointwise scoring functions, but instead use the two observations together, enabling more expressive non-linearity.

Lexicographic orders (LO) express compactly the order between any pair of options by specifying a particular order of the variables and of their respective values. LO are widely accepted as a plausible representation of humans preferences [20], [21]. However, their adequacy depends on the assumption of having underlying lexicographic preferences (i.e. learning bias).

Several types of LO have been proposed in the last decade. Linear LO, and their learning strategies, have been studied in [15], [20]. Booth et al. [16] introduced Conditional Lexicographic Preference Trees (CLPT). CLPT are an extension of LO to express conditional local preferences, where the preference value of an attribute depends on the values of previous attributes, and conditional importance, where the ordering of the attributes depends on the values of previous attributes. Liu and Truszczynski [22] extended CLPT to allow partial rankings, focusing on formal properties of the language. CLPT can be understood as general decision trees, where linear LO is a particular case representable as decision lists. Bräuning and Hüllermeier studied CLPT from a Machine Learning perspective and demonstrated their adequacy in different problems [21].

The contributions of this work can be summarized as follows. First, we extend CLPT [16] to handle non-categorical attributes, like real-valued or structured, through learning multiple kernels. Second, we propose a learning strategy to induce CLPT encodings with multiple kernels from a partial pairwise preference set. Third, we propose a lexicographic ensemble strategy to aggregate (conditional) lexicographic rankers, called *Rankdom Forest*. Finally, we validate the advantages of the proposed model from a Machine Learning point of view using several datasets.

II. Languages for Representing Multiattribute Lexicographic Preferences

In this section we formalize three languages for encoding lexicographic preference models. Section II-A and II-B define linear and non-linear (i.e. conditional) lexicographic languages. Then, the proposed language is presented in Section II-C.

A. Lexicographic Orders

As stated in [21], Lexicographic Orders (LO) can be defined as follows. Let us define an option $o \in \mathcal{O} = \mathcal{D}(A_1) \times \ldots \times \mathcal{D}(A_n)$, where $A = \{A_1, \ldots, A_n\}$ is the set of attributes and $\mathcal{D}(A_i)$ is the domain of the corresponding attribute A_i. For a given attribute subset $A' = \{A_{i_1}, \ldots, A_{i_k}\} \subseteq A$, $\mathcal{D}(A')$ is defined as $\mathcal{D}(A_{i_1}) \times \ldots \times \mathcal{D}(A_{i_k})$. Also, for an option $o \in \mathcal{O}$ and a subset $A' \subseteq A$, $\pi_{A'}[o]$ denotes the projection of o from $\mathcal{O}$ to $\mathcal{D}(A')$. For the sake of readability, we write o_k instead of $\pi_{\{A_k\}}[o]$ if $A' = \{A_k\}$ is a single attribute.

A lexicographic order on $\mathcal{O}$ is a total order $\succ$ defined in terms of [21]:

- The *attribute importance*, defined as the total order $\sqsupset$ on A.
- The *preferences on attribute values*, defined as a total order $\sqsupset_i$ on each attribute domain $\mathcal{D}(A_i)$.

Figure 1a illustrates a LO with three binary attributes. Hereafter, unless we said otherwise, we assume without loss of generality that $A_1 \sqsupset A_2 \sqsupset \ldots \sqsupset A_n$. Formally, o^* is preferred to o, $o^* \succ o$, if and only if there exists a $k \in \{1, \ldots, n\}$ such that

$$(o_k^* \sqsupset_k o_k) \ \wedge (\forall i | i \in 1 \leq i < k : o_i^* = o_i) \qquad (1)$$

It is important to notice that we refer to each object of the universe $\mathcal{O}$ as an option to recall the idea of being the preferred alternative in given set (pairs in our case). In other contexts, we might think about them as states or observations.

B. Conditional Lexicographic Preference Trees

Conditional Lexicographic Preference Trees (CLPT) extend linear LO by allowing to express conditional preferences on attribute values and conditional attribute importance based on the observed values of previous attributes [16].

1) Conditional Preferences on Attribute Values: Formally, conditional preferences introduce the notion of attribute dependency by refining the preference between attribute values $\sqsupset_k$ into $\sqsupset_k^{(o_1,\ldots,o_{k-1})}$, so it depends on $(o_1, \ldots, o_{k-1})$, the assignments of the attributes more important than k [21].

Then, for conditional preferences on attribute values, $o^* \succ o$ if and only if there exists a $k \in \{1, \ldots, n\}$ such that

$$\left(o_k^* \sqsupset_k^{(o_1,\ldots,o_{k-1})} o_k\right) \wedge (\forall i | 1 \leq i < k : o_i{}^* = o_i) \qquad (2)$$

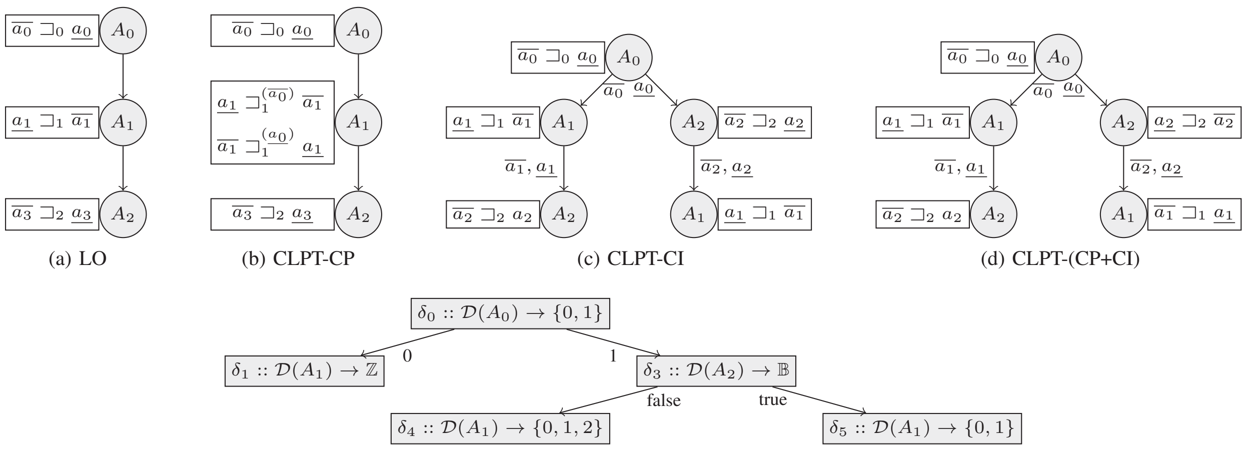
Figure 1b illustrates how the observed value on A_0 changes the preferred value on A_1. This concept was originally introduced in [16] to handle preferences on single attributes and extended in [21] to consider attribute tuples.

2) Conditional Attribute Importance: Observed values on important attributes define the order in which subsequent attributes are evaluated (their position in the hierarchy) [16]. Formally, the partial assignment $o_{i_1} \times \ldots \times o_{i_k}$ of the most important characteristics $(A_{i_1}, \ldots A_{i_k}) \in A^k$ determines the next attribute in the hierarchy $A_{i_{k+1}} \in A \setminus (A_{i_1}, \ldots A_{i_k})$. Figure 1c illustrates how the observed value on A_0 changes the relative importance of A_1 and A_2. Namely, if the observed value on A_0 is $\overline{a_0}$, A_1 is evaluated before A_2. Otherwise, A_2 is more important than A_1.

CLPT can be graphically understood as a tree, where every node is labeled with an attribute A_k and a total ordering $\sqsupset_k$ of its domain $\mathcal{D}(A_k)$. Each node has an outgoing edge to a descendant node for each possible value $a_k \in \mathcal{D}(A_k)$. A CLPT can be used as a ranker by feeding an option pair $(o^*, o) \in \mathcal{O} \times \mathcal{O}$ to the root of the tree and propagating it through the structure. At a given node v with attribute A_k, the projections o_k^* and o_k are compared. If their scores are different (i.e. $o_k^* \neq o_k$), $\sqsupset_k^{(o_{v_1},\ldots,o_{v_{k-1}})}$ is used to decide the preferred option. Otherwise, the pair is propagated to the proper descendant through the edge o_k. If a pair is propagated to a leaf without a decision, the model rejects the pair (i.e. it decides not to answer). Since the model is allowed to abstain, the preference order induced by it might be partial [22], [23]. Figures 1b and 1c show two examples of CLPT with conditional preferences on attribute values and conditional attribute importance respectively. Figure 1d illustrates an example with combined importance in both, preferences and importance.

C. Conditional Lexicographic Preference Trees with Multiple Kernels

CLPT have been studied in the literature[16], [21], [22]: 1) assuming discrete attributes and 2) assuming that preferences


$\delta_0 :: \mathcal{D}(A_0) \rightarrow \{0,1\}$
0
1
$\delta_1 :: \mathcal{D}(A_1) \rightarrow \mathbb{Z}$
$\delta_3 :: \mathcal{D}(A_2) \rightarrow \mathbb{B}$
false
true
$\delta_4 :: \mathcal{D}(A_1) \rightarrow \{0,1,2\}$
$\delta_5 :: \mathcal{D}(A_1) \rightarrow \{0,1\}$

(a) LO (b) CLPT-CP (c) CLPT-CI (d) CLPT-(CP+CI)

(e) MKCLPT

Fig. 1. Examples of LO, CLPT with conditional preferences (CP), CLPT with conditional attribute importance (CI), CLPT with CP and CI, and MKCLPT encodings. Attributes (A_i) are assumed to be binary, $\mathcal{D}(A_i) = \{\overline{a_i}, \underline{a_i}\}$.

behave in a strictly lexicographic manner, which might impose a very restrictive bias. Thereby, we extend the CLPT proposed in [16] to: 1) handle infinite (countable and uncountable) value domains and 2) to reduce the lexicographic bias $A_i \sqsupset A_k$. In order to do this, we introduce the notion of a kernel δ,

$$\delta :: \mathcal{D}(A') \rightarrow \mathbb{T}$$

where $A' \subseteq A$ and $\mathbb{T}$ is a totally ordered set under the relation $\leq$. The kernel function can be considered as a weak scoring ranker, that takes as input the projection of o over A', $\pi_{A'}[o]$, and computes a score $\delta(\pi_{A'}[o])$. For readability, we assume that A' is known and write instead $\delta(o)$.

The language of CLPT with multiple kernels (MKCLPT) extends the language of CLPT by introducing a customized kernel to each node. Thereby, instead of defining local preferences (see (2)) on attribute values, we consider a linear order of the assigned local scores. In the same way, conditional importance (c.f. Section II-B2) is defined in terms of the assigned scores of previous attributes instead of the original attribute values. Consequently, MKCLPT defines a parametric language to encode pseudo-lexicographic preferences.

Kernel expressiveness may range from naïve functions that consider a total order of a single attribute domain (traditional CLPT) to very complex multiattribute scoring rankers (e.g. RankSVM, GBRank, etc) able to learn any ranking function. Although we allow kernels to return an infinite number of values, we encourage the use of kernels with finite ordinal range. The proper trade-off between kernel expressiveness and lexicographic prior depends on the underlying preference model. In this sense, expressive kernels induce shallow tree structures, reducing the lexicographic assumption on the original attribute space.

As done with CLPT, a pairwise ranker can be inferred from a given MKCLPT. The only difference relies on the use of the scores assigned by the node's kernel δ_v when comparing two options instead of the projection on the node's attribute. (see Figure 1e).

III. Learning CLPT with Multiple Kernels

There have been several attempts in the past to learn lexicographic models from partial pairwise annotations. In this sense, the training set $\mathcal{T}$ consists of object pairs $(o^*, o) \in \mathcal{O} \times \mathcal{O}$, being o^* preferred to o.

$$\mathcal{T} = \{(o^*,\ o) \mid o^* \in \mathcal{O} \wedge o \in \mathcal{O} \wedge o^* \succ o\} \subseteq \mathcal{O} \times \mathcal{O}$$

The learning goal is to find lexicographic models with high agreement with the pairwise preferences in $\mathcal{T}$, avoiding overfitting the training data.

Each aforementioned language defines a space of preference models with its proper expressiveness. From a machine learning perspective, a predictive model is a particular instantiation from the entire search space defined by the language, which is able to generalize the underlying preference model. Namely, the preferences induced by the model are highly consistent with the unknown preferences. The resulting model is a computationally tractable way to represent the space of all the possible combinatorial preferences. While from a Machine Learning point of view, we are interested in compact models able to generalize, from a social choice perspective, this can be used for combinatorial preference aggregation [5].

For instance, LexRank presents a greedy strategy to learn linear LO by learning unconditional attribute importance and preferences on attribute values for Boolean domains [15]. On the other hand, CLeRa [21] is a greedy algorithm to induce CLPT for options with discrete domains.

Given that reasoning directly in a lexicographic manner with infinite-valued features returns a ranker with zero probability of non-abstention, leaving the entire decision to the most important attribute, previous efforts on learning CLPT preprocess real-valued features using global discretization techniques (e.g. equal frequency binning) [21]. However, these discretization techniques behave in a global fashion, preventing the model

```
procedure Fit-MKCLPT($\mathcal{T}, A, k, \delta$)
    if $\mathcal{T} = \emptyset \lor A = \emptyset$ then
        return MKCLPT_Dummy()
    end if

    $A^* \leftarrow$ Select-k-Features$(A, k)$
    $\delta_n \leftarrow$ Fit-Kernel$(\delta, \mathcal{T}, A^*)$
    $A_c \leftarrow$ Consumed$(\delta_n)$
    children $\leftarrow \{\}$

    for each s in Range$(\delta_n)$ do
        $\mathcal{T}_s \leftarrow \{(o^*, o) \mid (o^*, o) \in \mathcal{T} \land \delta_n(o^*) = \delta_n(o) = s\}$
        children[s] $\leftarrow$ Fit-MKCLPT$(\mathcal{T}_s,\ A - A_c,\ k,\ \delta)$
    end for

    return MKCLPT_Node$(\delta_n$, children$)$
end procedure
```

Fig. 2. Pseudocode of the proposed algorithm for fitting MKCLPT

to decide internally the best discretization technique to handle local preferences.

We propose an extension of the CLeRa algorithm to induce MKCLPT from a training set $\mathcal{T}$ (see Figure 2). In our proposal, kernels are learned in a local manner, instead of traditional global discretization schemes learned as a preprocessing step. In order to increase coherence between our extension and the original CLeRa algorithm, this section follows of [21].

As usually done in decision trees (DT) learning, our approach presents a greedy top-down strategy. Thereby, similarly to the information gain measure used in DT, we use a performance estimator that can be used as a heuristic for selecting the best topology [21], [23].

Thus, (3)-(4) measure respectively the number of concordant and discordant pairs between the induced preference model and the training set.

$$C(\succ, \mathcal{T}) = |\{(o^*, o) \in \mathcal{T} | o^* \succ o\}| \tag{3}$$

$$D(\succ, \mathcal{T}) = |\{(o^*, o) \in \mathcal{T} | o \succ o^*\}| \tag{4}$$

Then, motivated by [23], (5)-(6) define correctness (CR) and completeness (CP), that express respectively the agreement degree between two preference models and the degree of abstention.

$$CR(\succ, \mathcal{T}) = \frac{C(\succ, \mathcal{T}) - D(\succ, \mathcal{T})}{C(\succ, \mathcal{T}) + D(\succ, \mathcal{T})} \tag{5}$$

$$CP(\succ, \mathcal{T}) = \frac{C(\succ, \mathcal{T}) + D(\succ, \mathcal{T})}{|\mathcal{T}|} \tag{6}$$

Finally, we introduce the following performance measure, $0 \le \mathcal{C}_{\mathcal{RP}} \le 1$, motivated by the area under the correctness-completeness curve (see (7)). Since correctness range from -1 to +1, we introduce a scale transformation in order to be consistent with traditional AUC interpretations.

$$\mathcal{C}_{\mathcal{RP}}(\succ, \mathcal{T}) = \frac{(CR(\succ, \mathcal{T}) + 1)}{2} \cdot CP(\succ, \mathcal{T}) \tag{7}$$

A. Initialization and Termination

We begin considering the entire training set $\mathcal{T}$ and the complete set of attributes A. The recursion stops when there are no remaining attributes ($A' = \emptyset$) or training pairs ($\mathcal{T}' = \emptyset$). In this case, the function returns a dummy node that always rejects (i.e. full abstention).

B. Creating a node

In [21], a node is created by enumerating the exhaustive search on the attribute tuples and total orders of attribute values (with some heuristics to reduce computational time). The best node is chosen by maximizing correctness, using completeness to break ties. We generalize this process by learning a kernel that creates a latent attribute that encodes local preferences. At any given time, the kernel function has access to the remaining (non used) attributes. By using kernels able to handle non-categorical data types, we provide to CLPT nodes a way to reason about complex attributes. Moreover, we may introduce combinations of attributes in a more compact way than the total order of tuples proposed in [21] by using kernels that consider multiple attributes.

The kernel is assumed to be a weak scoring ranker. Thereby, we might use the scores directly in the lexicographic comparison without learning a total order on these values.

Although the decision process regarding the best kernel configuration depends on the kernel's learning strategy, maximizing correctness might encourage overfitted kernels that reject most comparisons when the kernel has high expressiveness. Thereby, in the evaluation of kernels we maximize $\mathcal{C}_{\mathcal{RP}}$, using correctness and completeness to break ties (in that order).

C. Recursion

After the best kernel has been identified and learned, the training pairs (o^*, o) predicted by the current node are removed from the training set. Then, a child is created for each possible score s in the image of δ. Each child is recursively trained using the training set defined in (8) and the attributes $A' \setminus A_\delta$, where A_δ are the attributes consumed by the kernel function.

$$\mathcal{T}_s = \{(o^*, o) \mid (o^*, o) \in \mathcal{T} \ \land \ \delta(o^*) = \delta(o) = s\} \tag{8}$$

D. Randomization

In order to introduce variability, we apply the random subspace method in the selection of attributes (features) to be considered by the kernel. This method has been traditionally used in the construction of Random Forests [24]. Thereby, each kernel is trained using $k < |A'|$ randomly chosen features. During the feature selection process, a fitness value is assigned to each attribute by fitting a kernel using it individually. Then, the roulette wheel selection strategy is employed to decide the best subset of features. The probabilitiy of a given features to be chosen is defined by

$$p_i = \frac{\mathcal{C}_{\mathcal{RP}}(\succ_{\delta i}, \mathcal{T})}{\sum\limits_{a \in A'} \mathcal{C}_{\mathcal{RP}}(\succ_{\delta a}, \mathcal{T})} \tag{9}$$

where $\succ_{\delta a}$ is the preference model induced by fitting a single node using the remaining training pairs and the feature a.

IV. Lexicographic Ensemble

Given the discrete nature of the comparisons performed by lexicographic preference models, coarse rankings (i.e. partial orders) are produced by them when compared with orders induced by SR with continuous output. Therefore, it is relevant to study a way to aggregate multiple NSR.

While simple voting schemes (e.g. weighted votes) can be considered in general classification, preserving consistency (i.e. symmetry and transitivity) when combining multiple non-scoring rankers is not trivial, even if each individual ranker is consistent [23]. Formally, if n rankers r_i with induced preference relations $\succ_i$ are merged, the ensemble R formed by them with preference relation $\succ$ must preserve (10) and (11). Namely, the directed graph induced by the underlying preference models must be acyclic (DAG).

$$a \succ b \equiv b \prec a \tag{10}$$

$$a_0 \succ a_1 \ \wedge \ \ldots \ \wedge a_{k-1} \succ a_k \implies a_0 \succ a_k \tag{11}$$

A previous attempt on merging lexicographic preference models was proposed in [25] from a computational social choice perspective. Thus, several constraints over the voting scheme were introduced that, in general, turn the problem intractable. However, from a Machine Learning point of view, we are interested in building an aggregation, without concerning about social choice properties.

A Lexicographic Ensemble (LE) comprises a sequence of (weak) lexicographic rankers $R = \{r_i \mid i \in [0 \ldots n)\}$ with a hierarchic voting rule. Namely, the predicted outcome for the pair (a, b) is specified by the estimator (r_i) with highest priority that doesn't abstain $(\neg a \sim_i b)$. Formally, the pair (a, b) is predicted as $\oplus \in \{\prec, \succ\}$ if and only if (12) holds, otherwise, $a \sim b$.

$$a \oplus b \equiv \underset{i \in [0 \ldots n)}{\exists} \left(a \oplus_i b \wedge \left(\underset{j \in [0 \ldots i)}{\forall} a \sim_j b \right) \right) \tag{12}$$

The proposed voting scheme can be understood as recursively appending a copy of the estimators with lower priority to the leaves (tail for linear LO) of the estimators with higher priority.

The learning process is reduced to find the best linear arrangement (i.e. permutation) of the base estimators. Although this scheme presents a *dictatorship*, from a Machine Learning perspective this is acceptable since estimators with higher priority are better informed than estimators in the tail of the arrangement. This process can be done by enumerating all the possible permutations and choosing the one that maximizes the metric of interest (e.g. correctness, completeness, $\mathcal{C}_{\mathcal{RP}}$). Since this becomes rapidly intractable for large ensembles, two strategies are proposed to address this task.

The first idea relies on sorting the estimators in decreasing order by their performance in a training (or validation) set using either correctness (with completeness to break ties), $\mathcal{C}_{\mathcal{RP}}$ (with correctness and completeness to break ties) or Borda count (number of wins minus number of losses) as evaluation metric. Given that the base estimators are weak and may abstain, only pairs of non-rejected predictions are considered when computing the winner between a pair of rankers in the Borda count strategy.

The second idea consists in modeling the problem as learning an unconditional lexicographic order with fixed preferences (correct prediction over incorrect prediction) of the estimators. Thus, any lexicographic learning strategy like LexRank [15] can be used to optimize the final order.

TABLE I
Datasets used in the experimental evaluation

Dataset	Options	Features	Pairs
Lenses	24	4	155
Hepatitis	155	19	3,936
Echocardiogram	131	9	5,418
Parkinson [27]	195	22	7,056
SPECT Heart	267	22	17,270
Ionosphere	351	34	28,350
Ecoli	336	7	37,831
Arrhythmia	451	14	68,990
Blood Transfusion [28]	748	4	101,460
Breast Cancer Wisconsin [29]	683	9	106,116
Mammographic Mass [30]	830	5	172,081
Tic-Tac-Toe	958	9	207,832
Banknote Authentication	1372	4	464,820
Car Evaluation	1728	6	682,721
Wine Quality (Red) [31]	1599	11	821,581
Chess	3196	36	2,548,563

V. Rankdom Forest

A Rankdom Forest is the aggregation using a Lexicographic Ensemble of weak Conditional Lexicographic Preference Trees. As done by traditional Random Forests [24], our proposal uses feature randomization to introduce variability in the model construction (c.f. Section III-D).

Also, in order to induce weak rankers, the maximum depth for each base estimator may be limited. This parameter configuration can be optimized through cross validation. Encouraging shallow trees increases the probability of abstention on the top of the ensemble (i.e. those with highest priority), enabling estimators in the tail of the ensemble to contribute in the construction of a more complete preference relation.

VI. Experiments and Results

In this section we detail the experimental evaluation of the proposed method to induce MKCLPT against two state-of-the-art methods to induce lexicographic rankers: CLeRa [21] algorithm to induce CLPT and LexRank [15] to induce linear LO. These three methods were assessed using 16 real-life datasets from the UCI [26] repository (c.f. Table I).

We used a 10-fold cross validation assessment strategy on the space of options. In this sense, we consider for the training stage preferences whose two options belong to the training set. Then, we validate the results on two different test sets. The first test set, hereinafter referred as train-test (TR-TS) contains preference pairs with one option in the training test and one in the test set. The second test set, referred as test-test (TS-TS) contains pairs with both options in the test set. TR and TS sets are important in different

TABLE II
COUNTS OF WINS FOR EACH PAIR OF MODELS IN TERMS OF CR AND CP. THE NUMBERS INDICATE THE NUMBER OF TIMES THE MODEL IN THE ROW OBTAINED A BETTER PERFORMANCE THAN THE MODEL IN THE COLUMN.

Train-Test						
Model	**Correctness**			**Completeness**		
	MKCLPT	CLPT	LO	MKCLPT	CLPT	LO
MKCLPT	-	**14**	**15**	-	4	2
CLPT	2	-	**9**	**12**	-	0
LO	1	7	-	**14**	**16**	-
Test-Test						
MKCLPT	-	**13**	**12**	-	4	2
CLPT	3	-	4	**12**	-	0
LO	4	**12**	-	**13**	**16**	-

TABLE III
AVERAGE RELATIVE DIFFERENCE (%) FOR EACH PAIR OF MODELS IN TERMS OF CR AND CP. THE MODELS IN THE COLUMN ARE USED AS REFERENCE TO MEASURE THE GAIN OF THE MODELS IN THE ROWS.

Train-Test						
Model	**Correctness**			**Completeness**		
	MKCLPT	CLPT	LO	MKCLPT	CLPT	LO
MKCLPT	–	**14.75**	**14.10**	–	−0.82	−1.53
CLPT	−10.24	–	−0.41	**0.90**	–	−0.72
LO	−9.70	**0.84**	–	**1.63**	**0.73**	–
Test-Test						
MKCLPT	–	**15.95**	**11.39**	–	−0.83	−1.97
CLPT	−10.28	–	−3.71	**0.92**	–	−1.14
LO	−6.79	**4.21**	–	**2.11**	**1.18**	–

TABLE IV
AVERAGE PERFORMANCE IN TERMS OF CR AND CP

Model	**Train-Test**		**Test-Test**	
	CR	**CP**	**CR**	**CP**
MKCLPT	**0.8165**	0.9773	**0.7764**	0.9730
CLPT	0.7268	0.9859	0.6880	0.9817
LO	0.7309	**0.9931**	0.7114	**0.9931**

contexts and applications. While the TR-TS set measures the performance of the models when comparing new (unseen) options and old (with partial annotations) options, the TS-TS set measures their performance on purely new options. Moreover, all the experiments were repeated 10 times (100 executions in total). In order to facilitate the reproducibility of the proposed experiments we have made available the source code, training/test partitions and extended results[1].

A. Kernels Used in the MKCLPT Learning Process

For experimental evaluation we considered the following kernels.

1) Binning: a given feature is discretized using equal-frequency binning. The optimal number of bins ($n \in \{2, \dots, 6\}$) and the feature to be used in the binning are decided at learning time. Then, a score is assigned to each bin according to its quality measured using the Borda count.

2) Clustering: observations are clustered using K-means with the number of centroids ($k \in \{2, 3, 4\}$) decided at learning time by maximizing $\mathcal{C}_{\mathcal{RP}}$. Centroids are indexed by Borda count. Then, each observation is assigned to its closest centroid.

3) RankSVM: a scoring ranker is trained. Then, scores are discretized using equal-frequency binning (5 bins). Given that the scores assigned by a scoring ranker are already monotonous, the final bins preserve the preference order. Notice that RankSVM [14] can be considered a SR given that the decision rule $\omega^T(a-b) > 0$ can be transformed into a scoring function since $\omega^T(a-b) > 0 \equiv \omega^T a > \omega^T b \equiv s(a) > s(b)$.

4) Multiple Kernel: in order to merge multiple kernels, a meta-kernel that simultaneously fits a sequence of base kernels and decides the best transformation based on $\mathcal{C}_{\mathcal{RP}}$ is used.

B. Assessment of the Individual Languages and Learning Strategies

Tables II and III show a pairwise comparison of the three methods. MKCLPT obtained better correctness results than CLPT and LO in most of the datasets. This behavior was consistent in both test sets. Although the proposed method abstains in a higher degree than the others state-of-the-art methods, the improvement in terms of correctness is higher than the decrease in terms of completeness. Table IV shows the average absolute correctness and completeness for each method and test set. An extended version of the results are presented in the online complimentary content.

C. Topology Comparison of the MKCLPT and CLPT Models

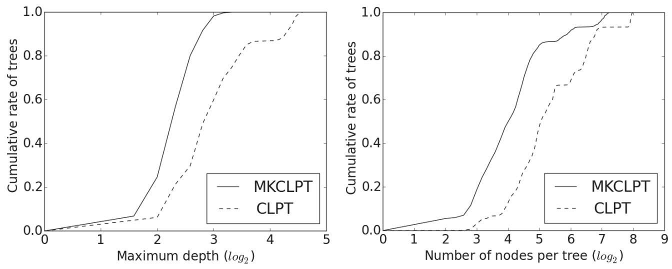
As shown in Figure 3a, the proposed MKCLPT language and its training algorithm induces more compact topologies than its non-kernelized counterpart (CLPT and CLeRa). This can be observed by the fact that the curve generated by MKCLPT completely dominates the curve induced by CLPT.

Also, the learning strategy detailed in Section III tends to locate expressive kernels (e.g. RankSVM) near the root of the trees and a higher number of kernels with limited expressiveness (e.g. binning) in the leaves. Therefore, given that the number of nodes increases exponentially on the height of the tree, it is natural to observe a higher number of nodes using Binning than RankSVM (see Figure 3b). However, given that MKCLPT follows a Pareto rule, a vast majority of the predictions are performed by the underrepresented RankSVM kernel (see Figures 3b and 3c). In this sense, MKCLPT is a NSR able to reduce the lexicographic bias of traditional CLPT by inducing shallower topologies with expressive kernels near the root.

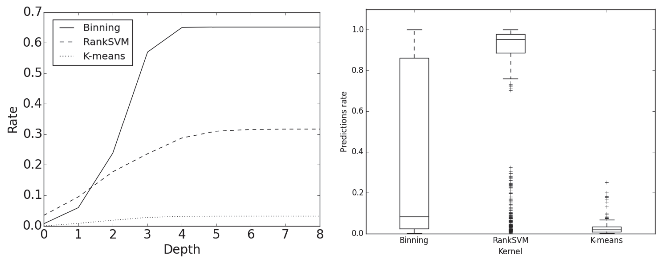
D. Assessment of the Lexicographic Ensemble Methods

We validated the proposed Lexicographic Ensemble by comparing the single MKCLPT model trained in Section VI-B with ensembles consisting of 5 randomized MKCLPT. The number of estimators in the ensemble is relatively low since the MKCLPT produced relatively high completeness values ($> 97\%$). Therefore, it is expected to take a few number of steps in the hierarchy to produce a final answer. We used 75% of the features when creating each node following the randomization rule explained in Section III-D.

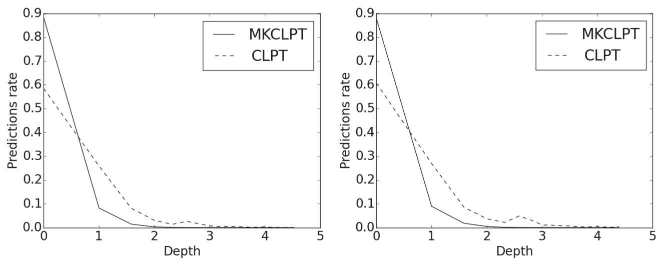
[1]http://vcmi.inescporto.pt/reproducible_research/ijcnn2016/RankdomForest



(a) Cumulative maximum depth and number of nodes per tree.



(b) Cumulative number of kernels per depth and prediction rate per kernel.



(c) Predictions done by depth. **Left:** Train-test set, **Right:** Test-test set.

Fig. 3. Topology analysis

TABLE V
AVERAGE PERFORMANCE OF THE ENSEMBLES IN TERMS OF CORRECTNESS AND COMPLETENESS.

Ensemble	Train-Test		Test-Test	
	CR	CP	CR	CP
$\mathcal{C}_{\mathcal{RP}}$	0.7862	0.9958	0.7424	0.9956
Borda	0.7881	0.9958	0.7437	0.9956
CR	**0.7883**	0.9958	0.7442	0.9956
LexRank	0.7878	0.9958	**0.7459**	0.9956

TABLE VI
COUNTS OF WINS AND TIES FOR EACH ENSEMBLE MODELS IN TERMS OF CORRECTNESS AND COMPLETENESS.

Ensemble	Correctness				Completeness		
	MKCLPT	CLPT	LO	LE	MKCLPT	CLPT	LO
Train-Test							
$\mathcal{C}_{\mathcal{RP}}$	4	**16**	**14**	5	**16**	**12**	3
Borda	5	**16**	**15**	4	**16**	**12**	3
CR	5	**16**	**15**	7	**16**	**12**	3
LexRank	5	**16**	**15**	1	**16**	**12**	3
Test-Test							
$\mathcal{C}_{\mathcal{RP}}$	4	**12**	**11**	6	**16**	**12**	3
Borda	5	**14**	**10**	5	**16**	**12**	3
CR	6	**14**	**10**	7	**16**	**12**	3
LexRank	5	**14**	**11**	5	**16**	**12**	3

TABLE VII
AVERAGE RELATIVE DIFFERENCE (%) BETWEEN EACH LE STRATEGY AND BASE LEXICOGRAPHIC MODEL IN TERMS OF CR AND CP.

Ensemble	Correctness				Completeness		
	MKCLPT	CLPT	LO	LE	MKCLPT	CLPT	LO
Train-Test							
$\mathcal{C}_{\mathcal{RP}}$	-3.2183	9.1312	8.3927	-0.4529	1.9132	1.0712	0.3419
Borda	-2.7968	9.5616	8.8072	-0.0824	1.9132	1.0712	0.3419
CR	**-2.7741**	**9.5883**	**8.8369**	**-0.0367**	1.9132	1.0712	0.3419
LexRank	-2.8385	9.5164	8.7632	-0.1364	1.9132	1.0712	0.3419
Test-Test							
$\mathcal{C}_{\mathcal{RP}}$	-3.7773	9.0306	4.6674	-0.5214	2.3551	1.4845	0.3089
Borda	-3.3340	9.4438	5.0503	-0.2930	2.3551	1.4845	0.3089
CR	-3.2482	9.5393	5.1382	-0.1941	2.3551	1.4845	0.3089
LexRank	**-3.0632**	**9.7629**	**5.3451**	**-0.0123**	2.3551	1.4845	0.3089

Tables V, VI and VII show the behavior of the proposed ensemble method using the four aforementioned heuristics (i.e. $\mathcal{C}_{\mathcal{RP}}$, Borda, correctness and LexRank). All the methods behaved in a similar way with slightly different results, being the one based on Borda count the method with highest correctness. Since the abstantion degree doesn't depend on the order of the base estimators, all the methods obtained the same completeness score.

The proposed aggregation strategy was able to increase the completeness of the base MKCLPT model to 99.58% in the train-test set and 99.56% in the test-test set. Moreover, the attained completeness values are higher than the ones obtained by the rest of the models. The best average results in the train-test and test-test sets were achieved by the CR and LexRank rules respectively.

Since increasing completeness requires to introduce more risky predictions, the average correctness decreased when compared with the base MKCLPT model. However, as can be seen in the results, the behavior of the ensemble strategies is better in terms of correctness than the other methods in the literature.

VII. Conclusions and Future Work

Learning to rank is a inductive learning technique that focuses on creating predictive models able to generalize underlying preference models on combinatorial domains. Being a plausible representation of human preferences [20], conditional and unconditional Lexicographic Orders (LO) are languages (and predictive models) for representing compactly and reasoning about multi-attribute preferences. Moreover, since LO are non-scoring rankers, they are able to represent preference models that can not be expressed as a utility function [19]. However, LO may pose a very strong bias when dealing with preference functions that don't behave in a lexicographic way. While other Learning to Rank methods are able to deal with non-categorical features, traditional LO methods require to introduce global preprocessing techniques to transform the attributes domain [21].

We overcome these limitations introducing Conditional Lexicographic Preference Trees with Multiple Kernels, MKCLPT, a new parametric language to reason about lexicographic preferences. MKCLPT is a kernelized version of conditional lexicographic orders, allowing: 2) to decrease the dependence on the assumption of having an underlying lexicographic preference model, and 2) to deal with non-categorical attributes.

We proposed a learning strategy to induce CLPT highly consistent with a partially annotated training set. As shown in the experimental evaluation, the proposed language and learning strategy performed better in terms of correctness than CLPT (trained using CLeRa algorithm [21]) and LO (trained using LexRank [15]). However, MKCLPT was more prune to abstain than other rankers. Thereby, we defined an aggregation technique to build ensembles of lexicographic rankers that induces more complete orders. Through the proposed experiments we validated that the aggregation was able to increase the completeness of the induced rankers with a controlled correctness reduction. On the other hand, the computational cost of learning a MKCLPT node is higher than learning a CLPT node, given that kernels are learned locally.

Combining learned randomized MKCLPT and lexicographic aggregation techniques, we defined Rankdom Forest, which can be understood as the non-scoring ranking analog of Random Forest used for classification. As we show, Rankdom Forests offer promising results when reasoning about partially annotated preference models, leading to future work in different directions. For instance, we are considering the inclusion of a regularization factor in the allowed completeness of each level in order to balance the prediction rate through the hierarchy. Also, we intend to explore general voting rules by allowing inconsistencies and introducing optimization techniques to minimize them.

Acknowledgement

This work is co-financed by: NORTE-01-0145-FEDER-000016 of the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project SFRH/BD/93012/2013.

References

[1] J. Fürnkranz and E. Hüllermeier, *Preference learning*. Springer, 2010.
[2] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker, "Label ranking by learning pairwise preferences," *Artificial Intelligence*, vol. 172, no. 16, pp. 1897–1916, 2008.
[3] X. Liu and M. Truszczynski, "Aggregating conditionally lexicographic preferences using answer set programming solvers," in *Algorithmic Decision Theory*. Springer, 2013, pp. 244–258.
[4] J. Lang and L. Xia, "Voting in combinatorial domains," *Handbook of Computational Social Choice*, 2014.
[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," 2004.
[6] W. W. C. R. E. Schapire and Y. Singer, "Learning to order things," in *Advances in Neural Information Processing Systems 10: Proceedings of the 1997 Conference*, vol. 10. MIT Press, 1998, p. 451.
[7] D. Sculley, "Combined regression and ranking," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 979–988.
[8] Y. Shi, M. Larson, and A. Hanjalic, "List-wise learning to rank with matrix factorization for collaborative filtering," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 269–272.
[9] F. Gao, D. Tao, X. Gao, and X. Li, "Learning to rank for blind image quality assessment," 2013.
[10] T.-Y. Liu, "Learning to rank for information retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.
[11] D. Cossock and T. Zhang, "Subset ranking using regression," in *Learning theory*. Springer, 2006, pp. 605–619.
[12] P. Li, Q. Wu, and C. J. Burges, "Mcrank: Learning to rank using multiple classification and gradient boosting," in *Advances in neural information processing systems*, 2007, pp. 897–904.
[13] K. Crammer, Y. Singer *et al.*, "Pranking with ranking." in *NIPS*, vol. 14, 2001, pp. 641–647.
[14] R. Herbrich, T. Graepel, and K. Obermayer, "Large margin rank boundaries for ordinal regression," *Advances in neural information processing systems*, pp. 115–132, 1999.
[15] P. Flach and E. T. Matsubara, "A simple lexicographic ranker and probability estimator," in *ECML*. Springer, 2007, pp. 575–582.
[16] R. Booth, Y. Chevaleyre, J. Lang, J. Mengin, and C. Sombattheera, "Learning conditionally lexicographic preference relations." in *ECAI*, 2010, pp. 269–274.
[17] J. Xu and H. Li, "Adarank: a boosting algorithm for information retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 391–398.
[18] Z. Zheng, K. Chen, G. Sun, and H. Zha, "A regression framework for learning ranking functions using relative relevance judgments," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 287–294.
[19] T. Rader, "The existence of a utility function to represent preferences," *The Review of Economic Studies*, pp. 229–232, 1963.
[20] M. Schmitt and L. Martignon, "On the complexity of learning lexicographic strategies," *The Journal of Machine Learning Research*, vol. 7, pp. 55–83, 2006.
[21] M. Bräuning and E. Hüllermeier, "Learning conditional lexicographic preference trees," *Workshop on Preference learning: problems and applications in AI, ECAI*, pp. 11–15, 2012.
[22] X. Liu and M. Truszczynski, "Learning partial lexicographic preference trees over combinatorial domains," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
[23] W. Cheng, M. Rademaker, B. De Baets, and E. Hüllermeier, "Predicting partial orders: ranking with abstention," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 215–230.
[24] T. K. Ho, "The random subspace method for constructing decision forests," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 8, pp. 832–844, 1998.
[25] J. Lang, J. Mengin, and L. Xia, "Aggregating conditionally lexicographic preferences on multi-issue domains," in *Principles and Practice of Constraint Programming*. Springer, 2012, pp. 973–987.
[26] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[27] M. A. Little, P. E. McSharry, S. J. Roberts, D. A. Costello, I. M. Moroz *et al.*, "Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection," *BioMedical Engineering OnLine*, vol. 6, no. 1, p. 23, 2007.
[28] I.-C. Yeh, K.-J. Yang, and T.-M. Ting, "Knowledge discovery on RFM model using bernoulli sequence," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5866–5871, 2009.
[29] K. P. Bennett and O. L. Mangasarian, "Robust linear programming discrimination of two linearly inseparable sets," *Optimization methods and software*, vol. 1, no. 1, pp. 23–34, 1992.
[30] M. Elter, R. Schulz-Wendtland, and T. Wittenberg, "The prediction of breast cancer biopsy outcomes using two cad approaches that both emphasize an intelligible decision process," *Medical Physics*, vol. 34, no. 11, pp. 4164–4172, 2007.
[31] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009.