

Learning conditionally lexicographic preference relations

Richard Booth,¹ Yann Chevaleyre,² Jérôme Lang² Jérôme Mengin³ and Chattrakul Sombattheera⁴

Abstract. We consider the problem of learning a user’s ordinal preferences on a multiattribute domain, assuming that her preferences are lexicographic. We introduce a general graphical representation called *LP-trees* which captures various natural classes of such preference relations, depending on whether the *importance* order between attributes and/or the *local preferences* on the domain of each attribute is conditional on the values of other attributes. For each class we determine the Vapnik-Chernovenkis dimension, the communication complexity of preference elicitation, and the complexity of identifying a model in the class consistent with a set of user-provided examples.

1 Introduction

In many applications, especially electronic commerce, it is important to be able to learn the preferences of a user on a set of alternatives that has a combinatorial (or multiattribute) structure: each alternative is a tuple of values for each of a given number of variables (or attributes). Whereas learning *numerical* preferences (*i.e.*, utility functions) on multiattribute domains has been considered in various places, learning *ordinal* preferences (*i.e.*, order relations) on multiattribute domains has been given less attention. Two streams of work are worth mentioning.

First, a series of very recent works focus on the learning of preference relations enjoying some preferential independencies conditions. Passive learning of separable preferences is considered by Lang & Mengin (2009), whereas passive (resp. active) learning of acyclic CP-nets is considered by Dimopoulos *et al.* (2009) (resp. Koriche & Zanuttini, 2009).

The second stream of work, on which we focus in this paper, is the class of *lexicographic* preferences, considered in Schmitt & Martignon (2006); Dombi *et al.* (2007); Yaman *et al.* (2008). These works only consider very simple classes of lexicographic preferences, in which both the importance order of attributes and the local preference relations on the attributes are *unconditional*. In this paper we build on these papers, and go considerably beyond, since we consider *conditionally lexicographic* preference relations.

Consider a user who wants to buy a computer and who has a limited amount of money, and a web site whose objective is to find the best computer she can afford. She prefers a laptop to a desktop, and this preference overrides everything else. There are two other attributes: colour, and type of optical drive (whether it has a simple DVD-reader or a powerful DVD-writer). Now, for a laptop, the next most important attribute is colour (because she does not want to be

seen at a meeting with the usual bland, black laptop) and she prefers a flashy yellow laptop to a black one; whereas for a desktop, she prefers black to flashy yellow, and also, colour is less important than the type of optical drive. In this example, both the importance of the attributes and the local preference on the values of some attributes may be conditioned by the values of some other attributes: the relative importance of colour and type of optical drive depends on the type of computer, and the preferred colour depends on the type of computer as well.

In this paper we consider various classes of lexicographic preference models, where the importance relation between attributes and/or the local preference on an attribute may depend on the values of some more important attributes. In Section 2 we give a general model for lexicographic preference relations, and define six classes of lexicographic preference relations, only two of which have already been considered from a learning perspective. Then each of the following sections focuses on a specific kind of learning problem: in Section 3 we consider preference elicitation, *a.k.a.* active learning, in Section 4 we address the sample complexity of learning lexicographic preferences, and in Section 5 we consider passive learning, and more specifically model identification and approximation.

2 Lexicographic preference relations: a general model

2.1 Lexicographic preferences trees

We consider a set \mathcal{A} of n attributes. For the sake of simplicity, all attributes we consider are binary (however, our notions and results would still hold in the more general case of attributes with finite value domains). The domain of attribute $X \in \mathcal{A}$ is $\underline{X} = \{x, \bar{x}\}$. If $U \subseteq \mathcal{A}$, then \underline{U} is the Cartesian product of the domains of the attributes in U . Attributes and sets of attributes are denoted by upper-case Roman letters (X, X_i, A etc.). An outcome is an element of $\underline{\mathcal{A}}$; outcomes are denoted by lower case Greek letters (α, β , etc.). Given a (partial) assignment $u \in \underline{U}$ for some $U \subseteq \mathcal{A}$, and $V \subseteq \mathcal{A}$, we denote by $u(V)$ the assignment made by u to the attributes in $U \cap V$.

In our learning setting, we assume that, when asked to compare two outcomes, the user, whose preferences we wish to learn, is always able to choose one of them. Formally, we assume that the (unknown) user’s preference relation on $\underline{\mathcal{A}}$ is a linear order, which is a rather classical assumption in learning preferences on multi-attribute domains (see, *e.g.*, Koriche & Zanuttini (2009); Lang & Mengin (2009); Dimopoulos *et al.* (2009); Dombi *et al.* (2007)). Allowing for indifference in our model would not be difficult, and most results would extend, but would require heavier notations and more details.

Lexicographic comparisons order pairs of outcomes (α, β) by looking at the attributes in sequence, according to their importance, until we reach an attribute X such that $\alpha(X) \neq \beta(X)$; α and β are

¹ University of Luxembourg. Also affiliated to Mahasarakham University, Thailand as Adjunct Lecturer. richard.booth@uni.lu

² LAMSADE, Université Paris-Dauphine, France. {yann.chevaleyre, lang}@lamsade.dauphine.fr

³ IRIT, Université de Toulouse, France. mengin@irit.fr

⁴ Mahasarakham University, Thailand. chattrakul.s@msu.ac.th

then ordered according to the *local preference* relation over the values of X . For such lexicographic preference relations we need both an *importance* relation, between attributes, and *local preference* relations over the domains of the attributes. Both the importance between attributes and the local preferences may be conditioned by the values of more important attributes. This conditionality can be expressed with the following class of trees:

Definition 1 A Lexicographic Preference Tree (or LP-tree) over \mathcal{A} is a tree such that:

- every node n is labelled with an attribute $\text{Att}(n)$ and a conditional preference table $\text{CT}(n)$ (details follow);
- every attribute appears once, and only once, on every branch;
- every non-leaf node labelled with attribute $\text{Att}(n) = X$ has either one outgoing edge, labelled by $\{x, \bar{x}\}$, or two outgoing edges, labelled respectively by x and \bar{x} .

$\text{Domin}(n)$ denotes the set of attributes labelling the ancestor nodes of n , called dominating attributes for n . $\text{Domin}(n)$ is partitioned between instantiated attributes and non-instantiated attributes: $\text{Inst}(n)$ (resp. $\text{NonInst}(n)$) is the set of attributes in $\text{Domin}(n)$ that have two children (resp. that have only one child). The conditional preference table associated with n consists of a subset $U \subseteq \text{NonInst}(n)$ (the attributes on which the preferences on X depend) and for each $u \in U$, a conditional preference entry of the form $u :>$, where $>$ is a linear order over the domain of $\text{Att}(n)$. In our case, because the domains are binary, these rules will be of the form $u : x > \bar{x}$ or $u : \bar{x} > x$. If U is empty, then the preference table is said to be maximally general and contains a single rule of the form $\top :>$. If $U = \text{NonInst}(n)$, then the preference table is said to be maximally specific.

Note that since the preferences on a variable X can vary between branches, the preference on X at a given node n with $\text{Att}(n) = X$ implicitly depends on the values of the variables in $\text{Inst}(n)$. The only variables on which the preferences at node n do not depend are the ones that are less important than X .⁵

Example 1 Consider three binary attributes: C (colour), with two values c (yellow) and \bar{c} (black); D (dvd device) with two values d (writer) and \bar{d} (read-only); and T (type) with values t (laptop) and \bar{t} (desktop). On the first LP-tree depicted next page, T is the most important attribute, with laptops unconditionally preferred to desktops; the second most important attribute is C in the case of laptops, with yellow laptops preferred to black ones, and D in the case of desktops. In all cases, a writer is always preferred to a read-only drive. The least important attribute is D for laptops, and C for desktops, black being preferred to yellow in this case.

If n is for instance the bottom leftmost leaf node, then $\text{Att}(n) = D$, $\text{Domin}(n) = \{T, C\}$, $\text{Inst}(n) = \{T\}$ and $\text{NonInst}(n) = \{C\}$. \square

Definition 2 Given distinct outcomes α and β , and a node n of some LP-tree σ with $X = \text{Att}(n)$, we say that n decides (α, β) if n is the (unique) node of σ such that $\alpha(X) \neq \beta(X)$ and for every attribute Y in $\text{Domin}(n)$, we have $\alpha(Y) = \beta(Y)$.

We define $\alpha \succ_{\sigma} \beta$ if the node n that decides (α, β) , is such that $\text{CT}(n)$ contains a rule $u :>$ with $\alpha(U) = \beta(U) = u$ and $\alpha(X) > \beta(X)$, where $u \in U$ for some $U \subseteq \text{NonInst}(n)$ and $X = \text{Att}(n)$.

Proposition 1 Given a LP-tree σ , \succ_{σ} is a linear order.

⁵ Relaxing this condition could be interesting, but can lead to a weak relation \succ_{σ} (see Def. 2): for instance, if we had attribute X more important than Y , and local preferences on X depend on Y , e.g. $y : x > \bar{x}$; $\bar{y} : \bar{x} > x$, one would not know how to compare $x\bar{y}$ and $\bar{x}y$.

Example 1 (continued) According to σ , the most preferred computers are yellow laptops with a DVD-writer, because $cdt \succ_{\sigma} \alpha$ for any other outcome $\alpha \neq cdt$; for any $x \in \underline{C}$ and any $z \in \underline{D}$ $xzt \succ_{\sigma} xz\bar{t}$, that is, any laptop is preferred to any desktop computer. And $cd\bar{t} \succ_{\sigma} \bar{c}d\bar{t}$, that is, a yellow desktop with DVD-writer is preferred to a black one with DVD-reader because, although for desktops black is preferred to yellow, the type of optical reader is more important than the colour for desktop computers. \square

LP-trees are closely connected to Wilson’s “Pre-Order Search Trees” (or POST) (2006). One difference is that in POSTs, local preference relations can be nonstrict, whereas here we impose linear orders, mainly to ease the presentation. More importantly, whereas in POSTs each edge is labelled by a single value, we allow more compact trees where an edge can be labelled with several values of the same variable; we need this because in a learning setting, the size of the representation that one learns is important.

2.2 Classes of lexicographic preference trees

Some interesting classes of LP-trees can be obtained by imposing a restriction on the local preference relations and/or on the conditional importance relation. The local preference relations can be *conditional* (general case), but can also be *unconditional* (the preference relation on the value of any attribute is independent from the value of all other attributes), and can also be *fixed*, which means that not only is it unconditional, but that it is known from the beginning and doesn’t have to be learnt (this corresponds to the distinction in Schmitt & Martignon (2006) between lexicographic strategies with or without cue inversion). Likewise, the attribute importance relation can be *conditional* (general case), or *unconditional* (for the sake of completeness, it could also be *fixed*, but this case is not very interesting and we won’t consider it any further). Without loss of generality, in the case of fixed local preferences, we assume that the local preference on X_i is $x_i > \bar{x}_i$.

Definition 3 (conditional/unconditional/fixed local preferences)

- The class of LP-trees with unconditional local preferences (UP) is the set of all LP-trees such that for every attribute X_i there exists a preference relation $>_i$ (either $x_i > \bar{x}_i$ or $\bar{x}_i > x_i$) such that for every node n with $\text{Att}(n) = X_i$, the local preference table at n is $\top :>_i$.
- The class of LP-trees with fixed local preferences (FP) is the set of all LP-trees such that for every attribute X_i and every node n such that $\text{Att}(n) = X_i$, the local preference at n is $x_i > \bar{x}_i$.

In the general case, local preferences can be conditional (CP).

Obviously, $\text{FP} \subset \text{UP}$.

Definition 4 (conditional/unconditional importance) The class of LP-trees with an unconditional importance relation (UI) is the set of all linear LP-trees, i.e., for which every node n has only one child. In the general case, the importance relation can be conditional (CI).

We stress that the class FP should not be confused with the class UP. FP consists of all trees where the local preference relation on each X_i is $x_i \succ \bar{x}_i$, while UP consists of all trees where the local preference relation on each X_i is the same in all branches of the tree ($x_i \succ \bar{x}_i$ or $\bar{x}_i \succ x_i$). For instance, if we have two variables X_1, X_2 and require the importance relation to be unconditional (UI) then we have only two LP-trees in FP-UI, whereas we have 8 in UP-UI.

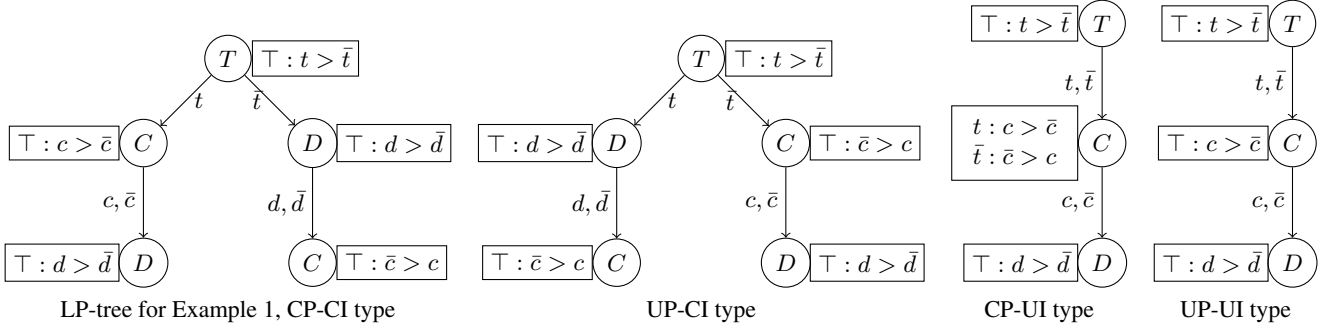


Figure 1. Examples of LP-trees

We can now combine a restriction on local preferences and a restriction on the importance relation. We thus obtain six classes of LP-trees, namely, CP-CI, CP-UI, UP-CI, UP-UI, FP-CI and FP-UI. For instance, CP-UI is defined as the class of all LP-trees with conditional preferences and an unconditional importance relation. The lexicographic preferences considered in Schmitt & Martignon (2006); Dombi *et al.* (2007); Yaman *et al.* (2008) are all of the FP-UI or UP-UI type. Note that the LP-tree on Example 1 is of the CP-CI type. Examples of LP-trees of other types are depicted above.

3 Exact learning with queries

Our aim in this paper is to study how we can learn a LP-tree that fits well a collection of examples. We first consider *preference elicitation*, a.k.a. active learning, or learning by queries. This issue has been considered by Dombi *et al.* (2007) in the FP-UI case (see also Koriche & Zanuttini (2009) who consider the active learning of CP-nets). The setting is as follows: there is some unknown target preference relation $>$, and a *learner* wants to learn a representation of it by means of a lexicographic preference tree. There is a *teacher*, a kind of oracle to which the learner can submit queries of the form (α, β) where α and β are two outcomes: the teacher will then reply whether $\alpha > \beta$ or $\beta > \alpha$ is the case. An important question in this setting is: how many queries does the learner need in order to completely identify the target relation $>$? More precisely, we want to find the communication complexity of preference elicitation, i.e., the worst-case number of requests to the teacher to ask so as to be able to elicit the preference relation completely, assuming the target can be represented by a model in a given class. The question has already been answered in Dombi *et al.* (2007) for the FP-UI case. Here we identify the communication complexity of eliciting lexicographic preferences trees in all five other cases, when all attributes are binary. (We restrict to the case of binary attributes for the sake of simplicity. The results for nonbinary attributes would be similar.) We know that a lower bound of the communication complexity is the log of the number of preference relations in the class. In fact, this lower bound is reached in all 6 cases:

Proposition 2 *The communication complexities of the six problems above are as follows, when all attributes are binary:*

	FP	UP	CP
UI	$\Theta(n \log n)$ Dombi <i>et al.</i> (2007)	$\Theta(n \log n)$	$\Theta(2^n)$
CI	$\Theta(2^n)$	$\Theta(2^n)$	$\Theta(2^n)$

Proof (Sketch) In the four cases FP-UI, UP-UI, FP-CI and UP-CI, the local preference tables are independent of the structure of

the importance tree. There are $n!$ unconditional importance trees, and $\prod_{k=0}^{n-1} (n-k)^{2^k}$ conditional ones. Moreover, when preferences are not fixed, there are 2^n possible unconditional preference tables. For the CP-CI case, a complete conditional importance tree contains $\sum_{k=0}^{n-1} 2^k = 2^n - 1$ nodes, and at each node there are two possible conditional preference rules. The elicitation protocols of Dombi *et al.* (2007) for the UI-FP case can easily be extended to prove that the lower bounds are reached in the five other cases. For instance, for the FP-CI case, a binary search, using $\log_2 n$ queries, determines the most important variable – for example, with four attributes A, B, C, D the first query could be $(ab\bar{c}\bar{d}, \bar{a}bcd)$, if the answer were $>$ we would know the most important attribute is A or B ; then for each of its possible values we apply the protocol for determining the second most important variable using $\log_2(n-1)$ queries, etc. When the local preferences are not fixed, at each node a single preliminary query gives a possible preference over the domain of every remaining variable. We then get the following exact communication complexities:

	FP	UP	CP
UI	$\log(n!)$	$n + \log(n!)$	$2^n - 1 + \log(n!)$
CI	$g(n) = \sum_{k=0}^{n-1} 2^k \log(n-k)$	$n + g(n)$	$2^n - 1 + g(n)$

Finally, $\log(n!) = \Theta(n \log n)$ and $g(n) = \Theta(2^n)$, from which the table follows. \square

4 Sample complexity of some classes of LP-trees

We now turn to supervised learning of preferences. First, it is interesting to cast our problem as a classification problem, where the training data is a set of pairs of distinct outcomes (α, β) , each labelled either $>$ or $<$ by a user: $((\alpha, \beta), >)$ and $((\alpha, \beta), <)$ respectively mean that the user prefers α to β , or β to α . (Recall that we assume that the user is always able to choose between two outcomes.) We want to learn a LP-tree σ that orders well the examples, e.g. such that $\alpha \succ_{\sigma} \beta$ for every example $((\alpha, \beta), >)$, and such that $\beta \succ_{\sigma} \alpha$ for every example $((\alpha, \beta), <)$. In this setting, the Vapnik-Chervonenkis (VC) dimension of a class of LP-trees is the size of the largest set of pairs (α, β) that can be ordered correctly by some LP-tree in the class, whatever the labels ($>$ or $<$) associated with each pair. In general, the higher this dimension, the more examples will be needed to correctly identify a LP-tree.

Proposition 3 *The VC dimension of any class of irreflexive, transitive relations over a set of n binary attributes is strictly less than 2^n .*

Proof (Sketch) Consider a set \mathcal{E} of 2^n pairs of outcomes over \mathcal{A} , and the corresponding undirected graph over \mathcal{A} : it has as many edges as vertices, so it has at least one cycle; if all edges of this cycle are directed so as to obtain a directed cycle, the transitive closure of the corresponding relation over \mathcal{A} will not be irreflexive. Hence \mathcal{E} cannot be shattered with transitive irreflexive relations. \square

Proposition 4 *The VC dimension of the class of CP-CI LP-trees and of CP-UI LP-trees over n binary attributes, is equal to $2^n - 1$.*

Proof (Sketch) It is possible to build a LP-tree over n attributes with 2^k nodes at the k -th level, for $0 \leq i \leq n - 1$: this is a tree for CP-CI trees, it has $2^n - 1$ nodes. Such a tree can shatter a set of $2^n - 1$ examples: take one example for each node, the local preference relation that is applicable at each node can be used to give both labels to the corresponding example. The upper bound follows from Prop. 3. \square

This result is rather negative, since it indicates that a huge number of examples would in general be necessary to have a good chance of closely approximating an unknown target relation. This important number of necessary examples also means that it would not be possible to learn in reasonable - that is, polynomial - time. However, learning CP-CI LP-trees is not hopeless in practice: decision trees have a VC dimension of the same order of magnitude, yet learning them has had great success experimentally.

As for trees with unconditional preferences, Schmitt & Martignon (2006) have shown that the VC dimension of UP-UI trees over n binary attributes is exactly n . Since every UP-UI tree is equivalent to a CP-UI tree, the VC dimension of UP-UI trees over n binary attributes is at least n .

5 Model identifiability

We now turn to the problem of identifying a model of a given class \mathcal{C} , given a set \mathcal{E} of examples. As before, an example is an ordered pair (α, β) , meaning that the user prefers α to β - this is not restrictive since we assume that the user's preference relation is a linear order.

Definition 5 *A LP-tree σ is said to be consistent with example (α, β) if $\alpha \succ_{\sigma} \beta$. Given a set of examples \mathcal{E} , σ is consistent with \mathcal{E} if it is consistent with every example of \mathcal{E} .*

The aim of the learner is to find some LP-tree in \mathcal{C} consistent with \mathcal{E} . Dombi *et al.* (2007) have shown that this problem can be solved in polynomial time for the class of binary LP-trees with unconditional importance and unconditional, fixed local preferences (FP-UI). In order to prove this, they exhibit a simple greedy algorithm that, given a set of examples \mathcal{E} and a set P of unconditional local preferences for all attributes, outputs an importance relation on variables (that is, an LP-tree of the FP-UI class) consistent with \mathcal{E} , provided there exists one. We will prove in this section that the result still holds for most of our classes of LP-trees, except one.

5.1 A greedy algorithm

We now present a generic algorithm able to generate LP-trees of various types. Given a set \mathcal{E} of examples, algorithm 1 constructs a tree that satisfies the examples if such a tree exists, iteratively adding nodes from the root to the leaves. Depending on how the two functions $\text{chooseAttribute}(n)$ and $\text{generateLabels}(n, X)$ are defined,

Algorithm 1 GenerateLPtree

INPUT: \mathcal{A} : set of attributes; \mathcal{E} : set of examples over \mathcal{A} ;
OUTPUT: LP-tree consistent with \mathcal{E} , or FAILURE;

1. $T \leftarrow \{\text{unlabelled root node}\}$;
 2. while T contains some unlabelled node:
 - (a) choose unlabelled node n of T ;
 - (b) $(X, CT) \leftarrow \text{chooseAttribute}(n)$;
 - (c) if $X = \text{FAILURE}$ then STOP and return FAILURE;
 - (d) label n with X and CT ;
 - (e) $L \leftarrow \text{generateLabels}(n, X)$; (*labels for edges below n*)
 - (f) for each $l \in L$:
 - add new unlabelled node to T , attached to n with edge labelled with l ;
 3. return T .
-

the output tree will have conditional/unconditional/fixed preference tables, and conditional/unconditional attribute importance.

At a given currently unlabelled node n , step 2b considers the set $\mathcal{E}(n) = \{(\alpha, \beta) \in \mathcal{E} \mid \alpha(\text{Domin}(n)) = \beta(\text{Domin}(n))\}$, that is, the set of all examples in \mathcal{E} for which α and β coincide on all variables above $\text{Att}(n)$ in the branch from the root to n . $\mathcal{E}(n)$ is the set of examples that correspond to the assignments made in the branch so far and that are still undecided. The helper function $\text{chooseAttribute}(n)$ returns an attribute $X \notin \text{Domin}(n)$ and a conditional table CT . Informally, the attribute and the local preferences are chosen so that they do not contradict any example of $\mathcal{E}(n)$: for every $(\alpha, \beta) \in \mathcal{E}(n)$, if $\alpha(X) \neq \beta(X)$ then CT must contain a rule of the form $u : \alpha(X) > \beta(X)$, with $u \in \underline{U}$, $U \subseteq \text{NonInst}(n)$ and $\alpha(U) = \beta(U) = u$. If we allow the algorithm to generate conditional preference tables, the function $\text{chooseAttribute}(n)$ may output any (X, CT) satisfying the above condition. However, if we want to learn a tree with unconditional or fixed preferences we need to impose appropriate further conditions; we will say (X, CT) is:

UP-choosable if CT is of the form $\top :>$ (a single unconditional rule);

FP-choosable if CT is of the form $\top : x > \bar{x}$.

If no attribute can be chosen with an appropriate table without contradicting one of the examples, $\text{chooseAttribute}(n)$ returns FAILURE and the algorithm stops at step 2c.

Otherwise, the tree is expanded below n with the help of the function generateLabels , unless $\text{Domin}(n) \cup \{X\} = \mathcal{A}$, in which case n is a leaf and $\text{generateLabels}(n, X) = \emptyset$. If this is not the case, $\text{generateLabels}(n, X)$ returns labels for the edges below the node just created: it can return two labels $\{x\}$ and $\{\bar{x}\}$ if we want to split with two branches below X , or one label $\{x, \bar{x}\}$ when we do not want to split. If we want to learn a tree of the UI class, clearly we never split. If we want to learn a tree with possible conditional importance, we create two branches, unless the examples in $\mathcal{E}(n)$ that are not decided at n all have the same value for X , in which case we do not split. An important consequence of this is that the number of leaves of the tree built by our algorithm never exceeds $|\mathcal{E}|$.

5.2 Some examples of GenerateLPtree

Throughout this subsection we assume the algorithm checks the attributes for choosability in the order $T \rightarrow C \rightarrow D$.

Example 2 Suppose \mathcal{E} consists of the following five examples:

1: $(tcd, t\bar{c}\bar{d})$; 2: $(\bar{t}\bar{c}d, \bar{t}c\bar{d})$; 3: $(t\bar{c}\bar{d}, t\bar{c}d)$; 4: $(\bar{t}\bar{c}\bar{d}, \bar{t}\bar{c}d)$; 5: $(\bar{t}\bar{c}d, \bar{t}c\bar{d})$

Let's try using the algorithm to construct a UP-UI tree consistent with \mathcal{E} . At the root node n_0 of the tree we first check if there is table CT such that (T, CT) is UP-choosable. By the definition of UP-choosability, CT must be of the form $\{\top : >\}$ for some total order $>$ of $\{t, \bar{t}\}$. Now since $\alpha(T) = \beta(T)$ for all $(\alpha, \beta) \in \mathcal{E}(n_0) = \mathcal{E}$, $(T, \{\top : >\})$ is choosable for any $>$ over $\{t, \bar{t}\}$. Thus we label n_0 with e.g. T and $\top : t > \bar{t}$ (we could have chosen $\top : \bar{t} > t$ instead). Since we are working in the UI-case the algorithm generates a single edge from n_0 labelled with $\{t, \bar{t}\}$ and leading to a new unlabelled node n_1 . We have $\mathcal{E}(n_1) = \mathcal{E}(n_0)$ since no example is decided at n_0 . So C is not UP-choosable at n_1 , owing to the opposing preferences over C exhibited for instance in examples 1,2. However $(D, \{\top : d > \bar{d}\})$ is UP-choosable, thus the algorithm labels n_1 with D and $\{\top : d > \bar{d}\}$. Example 5 is decided at n_1 . At the next node the only remaining attribute C is not UP-choosable because for instance we still have $1, 2 \in \mathcal{E}(n_2)$. Thus the algorithm returns FAILURE. Hence there is no UP-UI tree consistent with \mathcal{E} .

However, if we allow conditional preferences, the algorithm *does* successfully return the CP-UI tree depicted on Fig. 1, because C is choosable at node n_1 , with the conditional table $\{t : c > \bar{c} ; \bar{t} : \bar{c} > c\}$. All examples are decided at n_0 or n_1 , and the algorithm terminates with an arbitrary choice of table for the last node, labelled with D . \square

This example raises a couple of remarks. Note first that the choice of T for the root node is really a completely uninformative choice in the UP-UI case, since it does not decide any of the examples. In the CP-UI case, the greater importance of T makes it possible to have the local preference over C depend on the values of T . Second, in the CP-UI case, the algorithm finishes the tree with an arbitrary choice of table for the leaf node, since all examples have been decided above that node; this way, the ordering associated with the tree is complete, which makes the presentation of our results easier. In a practical implementation, we may stop building a branch as soon as all corresponding examples have been decided, thus obtaining an incomplete relation.

Example 3 Consider now the following examples:

1: $(tcd, t\bar{c}\bar{d})$; 2b: $(tcd, t\bar{c}\bar{d})$; 3b: $(\bar{t}\bar{c}d, \bar{t}c\bar{d})$; 4: $(\bar{t}\bar{c}\bar{d}, \bar{t}\bar{c}d)$; 5: $(\bar{t}\bar{c}d, \bar{t}c\bar{d})$

We will describe how the algorithm can return the CP-CI tree of Ex. 1, depicted on Fig. 1, if we allow conditional importance and preferences. As in the previous example, T is CP-choosable at the root node n_0 with a preference table $\{\top : t > \bar{t}\}$. Since we are now in the CI-case, `generateLabels` generates an edge-label for each value t and \bar{t} of T . Thus two edges from n_0 are created, labelled with t, \bar{t} resp., leading to two new unlabelled nodes m_1 and n_1 .

We have $\mathcal{E}(m_1) = \{(\alpha, \beta) \in \mathcal{E} \mid \alpha(T) = \beta(T) = t\} = \{1, 2b\}$, so `chooseAttribute` returns $(C, \{\top : c > \bar{c}\})$, and m_1 is labelled with C and $\top : c > \bar{c}$. Since only example 2b remains undecided at m_1 , no split is needed, one edge is generated, labelled with $\{c, \bar{c}\}$, leading to new node m_2 with $\mathcal{E}(m_2) = \{2b\}$. The algorithm successfully terminates here, labelling m_2 with D and $\top : d > \bar{d}$.

On the other branch from n_0 , we have $\mathcal{E}(n_1) = \{3b, 4, 5\}$. Due to the opposing preferences on their restriction to C exhibited by examples 3b,5, C is not choosable. Thus we have to consider D instead. Here we see that $(D, \{\top : d > \bar{d}\})$ can be returned by `chooseAttribute` thus n_1 is labelled with D , and $\top : d > \bar{d}$. The only undecided example that remains on this branch is 4, so the branch is finished with a node labelled with C and $\top : \bar{c} > c$. \square

5.3 Complexity of model identification

The greedy algorithm above solves five learning problems. In fact, the only problem that cannot be solved with this algorithm, as will be shown below, is the learning of a UP-CI tree without initial knowledge of the preferences.

Proposition 5 *Using the right type of labels and the right choosability condition, the algorithm returns, when called on a given set \mathcal{E} of examples, a tree of the expected type, as described in the table below, consistent with \mathcal{E} , if such a tree exists:*

learning problem	chooseAttribute	generateLabels
CP-CI	no restriction	split possible
CP-UI	no restriction	no split
UP-UI	UP-choosable	no split
FP-CI	FP-choosable	split possible
FP-UI	FP-choosable	no split

Proof (Sketch) The fact that the tree returned by the algorithm has the right type, depending on the parameters, and that it is consistent with the set of examples is quite straightforward. We now give the main steps of the proof of the fact that the algorithm will not return failure when there exists a tree of a given type consistent with \mathcal{E} .

Note first that given any node n of some LP-tree σ , labelled with X and the table CT , then (X, CT) is candidate to be returned by `chooseAttribute`. So if we know in advance some LP-tree σ consistent with a set \mathcal{E} of examples, we can always construct it using the greedy algorithm, by choosing the "right" labels at each step.

Importantly, it can also be proved that if at some node n `chooseAttribute` chooses another attribute Y , then there is some other LP-tree σ' , of the same type as σ , that is consistent with \mathcal{E} and extends the current one; more precisely, σ' is obtained by modifying the subtree of σ rooted at n , taking up Y to the root of this subtree. Hence the algorithm cannot run into a dead end. \square

This result does not hold in the UP-CI case, because taking an attribute upwards in the tree may require using a distinct preference rule, which may not be correct in other branches of the LP-tree.

Example 4 Consider now the following examples:

1b: $(tcd, t\bar{c}\bar{d})$; 2b: $(tcd, t\bar{c}\bar{d})$; 4b: $(\bar{t}\bar{c}\bar{d}, \bar{t}c\bar{d})$; 5: $(\bar{t}\bar{c}d, \bar{t}c\bar{d})$

The UP-CI tree depicted on Fig. 1 is consistent with $\{1b, 2b, 4b, 5\}$. However, if we run the greedy algorithm again, trying this time to enforce unconditional preferences, it may, after labelling the root node with T again, build the t branch first: it will choose C with preference $\top : c > \bar{c}$ and finish this branch with $(D, \top : d > \bar{d})$; when building the \bar{t} branch, it cannot choose C first, because the preference has already been chosen in the t branch and would wrongly decide 5; but D cannot be chosen either, because 4b and 5 have opposing preferences on their restrictions to D . \square

Proposition 6 *The problems of deciding if there exists a LP-tree of a given class consistent with a given set of examples over binary attributes have the following complexities:*

	FP	UP	CP
UI	P (Dombi et al., 2007)	P	P
CI	P	NP-complete	P

Proof (Sketch) For the CP-CI, CP-UI, FP-CI, FP-UI and UP-UI cases, the algorithm runs in polynomial time because it does not have

more than $|\mathcal{E}|$ leaves, and each leaf cannot be at depth greater than n ; and every step of the loop except (2b) is executed in linear time, whereas in order to choose an attribute, we can, for each remaining attribute X , consider the relation $\{(\alpha(X), \beta(X)) \mid (\alpha, \beta) \in \mathcal{E}(n)\}$ on \underline{X} : we can check in polynomial time if it has cycles, and, if not, extend it to a total strict relation over \underline{X} .

For the UP-CI case, one can guess a set of unconditional local preference rules P , of size linear in n , and then check in polynomial time (FP-CI case) if there exists a tree T such that (T, P) is consistent with \mathcal{E} ; thus the problem is in **NP**. Hardness comes from a reduction from WEAK SEPARABILITY – the problem of checking if there is a CP-net without dependencies *weakly consistent* with a given set of examples – shown to be **NP**-complete by Lang & Mengin (2009). More precisely, a set of examples \mathcal{E} is weakly separable if and only if there exists a (non ambiguous) set of unconditional preference rules that contains, for every $(\alpha, \beta) \in \mathcal{E}$, a rule $X, \top \text{ :>}$ such that $\alpha(X) > \beta(X)$. To prove the reduction, given a set of examples $\mathcal{E} = \{(\alpha_1, \beta_1), \dots, (\alpha_m, \beta_m)\}$, built on a set of attributes $\{X_1, \dots, X_n\}$, we introduce m new attributes P_1, \dots, P_m . For each example $e_i = (\alpha_i, \beta_i) \in \mathcal{E}$ we create a new example

$$e'_i = (\alpha_i \bar{p}_1 \dots \bar{p}_{i-1} p_i \bar{p}_{i+1} \dots \bar{p}_m, \beta_i \bar{p}_1 \dots \bar{p}_{i-1} p_i \bar{p}_{i+1} \dots \bar{p}_m).$$

If \mathcal{E} is weakly separable, we can build a UP-CI LP-tree consistent with $\mathcal{E}' = \{e'_i \mid e_i \in \mathcal{E}\}$: the m top levels of the tree are labelled with the P_i s; below that there remains no more than one example on each branch, and the importance order on that branch can be chosen to decide well the corresponding example. For the converse, it is not hard to see that the restriction to X_1, \dots, X_n of the tables of a UP-CI tree consistent with \mathcal{E}' is weakly compatible with \mathcal{E} . \square

5.4 Complexity of model approximation

In practice, it is often the case that no structure of a given type is consistent with all the examples at the same time. It is then interesting to find a structure that is consistent with as many examples as possible. In the machine learning community, the associated learning problem is often referred to as *agnostic learning*. Schmitt & Martignon (2006) have shown that finding a UI-UP LP-tree, with a fixed set of local preferences, that satisfies as many examples from a given set as possible, is **NP**-complete, in the case where all attributes are binary. We extend these results here.

Proposition 7 *The complexities of finding a LP-tree in a given class, which wrongly classifies at most k examples of a given set \mathcal{E} of examples over binary attributes, for a given k , are as follows:*

	FP	UP	CP
UI	NP-c. (Schmitt & Martignon (2006))	NP-c.	NP-c.
CI	NP-c.	NP-c.	NP-c.

Proof (Sketch) These problems are in **NP** because in each case a witness is the LP-tree that has the right property, and such a tree need not have more nodes than there are examples. For the UP-CI case, the problem is already **NP**-complete for $k = 0$, so it is **NP**-hard. **NP**-hardness of the other cases follow from successive reductions from the case proved by Schmitt & Martignon (2006). \square

6 Conclusion and future work

We have proposed a general, lexicographic type of models for representing a large family of preference relations. We have defined six interesting classes of models where the attribute importance as well as the local preferences can be conditional, or not. Two of these classes

correspond to the usual unconditional lexicographic orderings. Interestingly, classes where preferences are conditional have an exponential VC dimension.

We have identified the communication complexity of the five classes for which it was not previously known, thereby generalizing a previous result by Dombi *et al.* (2007).

As for passive learning, we have proved that a greedy algorithm like the ones proposed by Schmitt & Martignon (2006); Dombi *et al.* (2007) for the class of unconditional preferences can identify a model in another four classes, thereby showing that the model identification problem is polynomial for these classes. We have also proved that the problem is **NP**-complete for the class of models with conditional attribute importance but unconditional local preferences. On the other hand, finding a model that minimizes the number of mistakes turns out to be **NP**-complete in all cases.

Our LP-trees are closely connected to decision trees. In fact, one can prove that the problem of learning a decision tree consistent with a set of examples can be reduced to a problem of learning a CP-CI LP tree. It remains to be seen if CP-CI trees can be as efficiently learnt in practice as decision trees.

In the context of machine learning, usually the set of examples to learn from is not free of errors in the data. Our greedy algorithm is quite error-sensitive and therefore not robust in this sense; it will even fail in the case of a collapsed version space. Robustness toward errors in the training data is clearly an important property of real world applications.

As future work, we intend to test our algorithms, with appropriate heuristics to guide the choice of variables at each stage. A possible heuristic would be the mistake rate if some unconditional tree is built below a given node (which can be very quickly done). Another interesting aspect would be to study mixtures of conditional and unconditional trees, with e.g. the first two levels of the tree being conditional ones, the remaining ones being unconditional (since it is well-known that learning decision trees with only few levels can be as good as learning trees with more levels).

Acknowledgements We thank the reviewers for very helpful comments. This work has received support from the French Ministry of Foreign Affairs and the Commission on Higher Education, Ministry of Education, Thailand. A preliminary version of this paper has been presented at the Preference Learning workshop 2009.

References

- Dimopoulos, Y., Michael, L., & Athienitou, F. 2009. Ceteris Paribus Preference Elicitation with Predictive Guarantees. *Proc. IJCAI'09*, pp 1890-1895.
- Dombi, J., Imreh, C., & Vincze, N. 2007. Learning Lexicographic Orders. *European J. of Operational Research*, **183**, pp 748-756.
- Koriche, F., & Zanuttini, B. 2009. Learning conditional preference networks with queries. *Proc. IJCAI'09*, pp 1930-1935.
- Lang, J., & Mengin, J. 2009. The complexity of learning separable ceteris paribus preferences. *Proc. IJCAI'09*, pages 848-853.
- Schmitt, M., & Martignon, L. 2006. On the Complexity of Learning Lexicographic Strategies. *J. of Mach. Learning Res.*, **7**, pp 55-83.
- Wilson, N. 2006. An Efficient Upper Approximation for Conditional Preference. *In Proc. ECAI 2006*, pp 472-476.
- Yaman, F., Walsh, T., Littman, M., & desJardins, M., 2008. Democratic Approximation of Lexicographic Preference Models. *In: Proc. ICML'08*, pp 1200-1207.