The complexity of unsupervised learning of lexicographic preferences

Hélène Fargier[1], **Pierre-François Gimenez**[2], Jérôme Mengin[1], Ngoc Bao Nguyen[3]

[1]IRIT, University of Toulouse
[2] CentraleSupélec, University of Rennes, IRISA
[3]INSA Toulouse

ANITI seminar, March 17th, 2023

Introduction

Preferences and recommendation

- Preferences learning is widely used in commercial systems, notably for recommendation and personalization (Netflix, Amazon, Youtube, social network, etc.)
- Multiple techniques have been developed in two main categories:
  - Collaborative filtering: "people like you enjoyed this item"
  - Content-based recommendation: "you seem to like horror movie by John Carpenter, here is another one"
- One underlying hypothesis: the order of magnitude of the number of user feedback (stars, purchases, views, etc.) is comparable to the order of magnitude of the number of items

CentraleSupélec

Preference in combinatorial domain

Combinatorial domain

- This hypothesis is not met with highly customizable items like cars, kitchens, computers or holiday trips
- These items are described as vectors of categorical features
- For example: 10^{16} different Renault "Master" cars, 2 millions Renault cars sold in 2022
- In reality, these features are constrained (no sunroof on a convertible car, no leather wheel on a lower-end car, etc.) $\Rightarrow$ we will ignore this issue for this study

Preferences learning

Unsupervised preferences learning

- User preferences are modelized by an order (total or not) on a set of items $\mathcal{X}$
- Most preference learning approaches are supervised: the learning data is a set of pairwise comparisons (A is preferred to B, etc.)
- This data can be costly to collect, in contrary to sales histories that are readily available
- Sales histories contain information about the preferences: they mostly contain items that rank high in the preferences
- Unsupervised preferences learning uses a set of items chosen by a user as training set

Previous work and goal

Previous work by [FGM18]

- We proposed greedy unsupervised learning algorithms for lexicographic preferences models
- The approach is based on the idea that the user won't always chose her most preferred item (because this item is not available, because of a desire of variety, etc.)
- We rely on the following assumption: **the more preferred an item is, the more likely it is chosen**, i.e. if $o \succ o'$, then $p(o) \geq p(o')$
- The learning algorithms have some convergence properties and have been experimentally assessed, but their theoretical properties have not been studied

The goal of this article is to study the time and sample complexity of unsupervised learning for several classes of lexicographic preferences models

CentraleSupélec

Outline

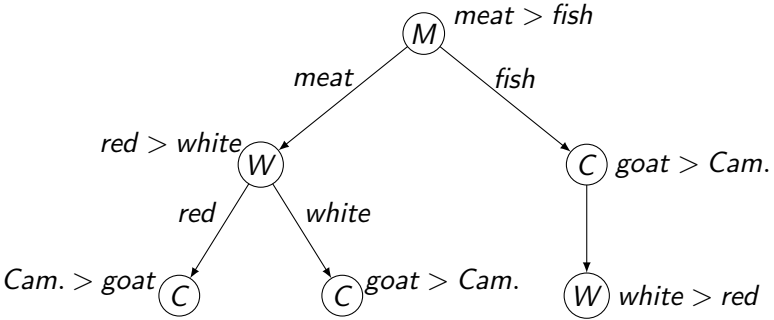
CentraleSupélec

Toy example

Imagine a user wants to book a dinner online. A menu is a tuple of values of three different attributes:

Main course (M)	meat or fish
Wine (W)	red or white
Cheese (C)	goat or Camembert

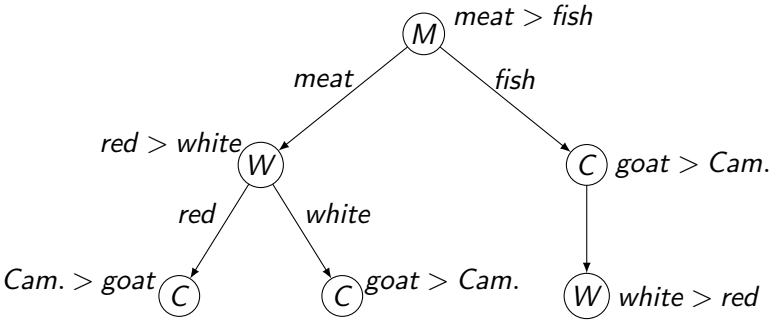
Lexicographic Preference Trees (LP-trees) [BCL+09]



LP-tree definition [BCL+09]

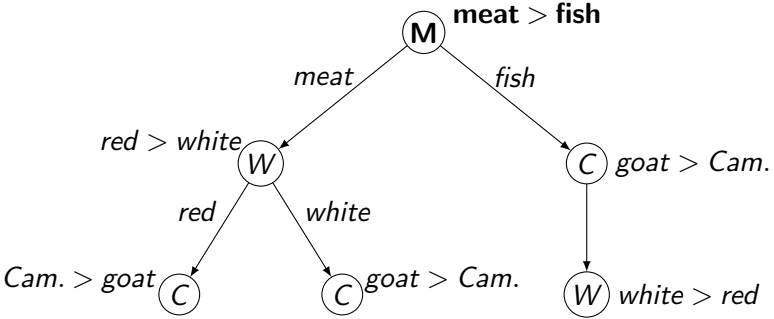
- Tree of attributes ordered by importance (root: most important)
- Edges can be labelled by a value or not
- Preferences rules associated to each node

LP-trees semantics



We would like to compare:
$meat - red - goat$ and $fish - white - goat$

LP-trees semantics

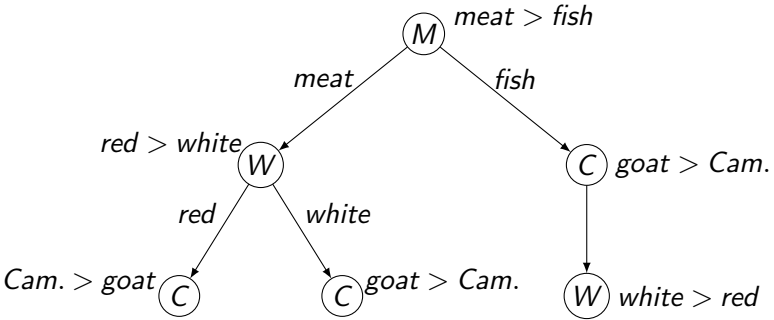


We would like to compare:

$meat - red - goat \succ fish - white - goat$

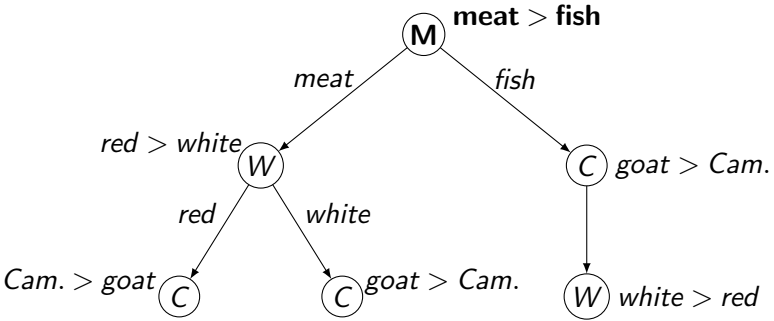
Any menu with meat is preferred to any menu with fish

LP-trees semantics



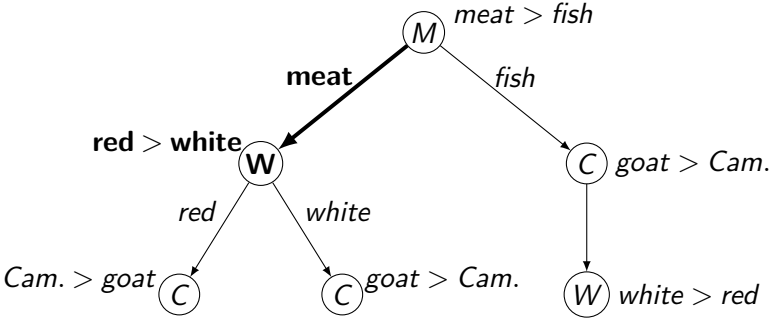
We would like to compare:
$meat - white - goat$ and $meat - white - Cam.$

LP-trees semantics



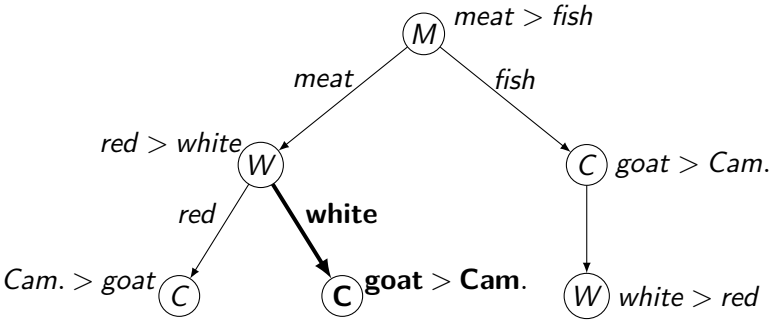
We would like to compare:
$meat - white - goat$ and $meat - white - Cam.$
Root node can't decide the comparison

LP-trees semantics



We would like to compare:
$meat - white - goat$ and $meat - white - Cam.$
Among meat menus, any menu with red wine is preferred to any menu with white wine

LP-trees semantics



We would like to compare:
$meat - white - goat \succ meat - white - Cam.$

CentraleSupélec

The LP-trees languages we study

Linear LP-trees with univariate nodes

One attribute per node, one branch only

Linear LP-trees with multivariate nodes

Multiple attributes per node, one branch only

LP-trees with multivariate nodes and a bounded number of leaves

Multiple attributes per node, at most l branches

All these models represent total orders

Rank

Rank

- These models all represent a total relation $\succ$ over $\underline{\mathcal{X}}$, a set of items
- The **rank** of an item $o \in \underline{\mathcal{X}}$, denoted as $rank(\succ, o)$, is defined as 1 + the number of elements preferred to o.
  - The rank of the preferred item is 1
  - The rank of the second preferred item is 2
  - ...
  - The rank of the least preferred item is $|\underline{\mathcal{X}}|$

Sample complexity

PAC-learning

How many examples are enough to probably learn a good model?

- "enough": we look for an upper bound
- "probably": we cannot rule out rare unrepresentative training set
- "good": we don't need to learn the optimal model, but we want to be close

Sample complexity definition

Let n be the number of attributes, ϕ^* the unknown target LP-tree, $\hat{\phi}$ the LP-tree that maximize a criteria over the training set. We are looking for a function S such as, if there are at least $S(n, \delta, \epsilon)$ examples in the training set, then

$$Pr(dist(\phi^*, \hat{\phi}) \leq \epsilon) \geq 1 - \delta$$

But what distance should we use?

Learning model

Ranking loss

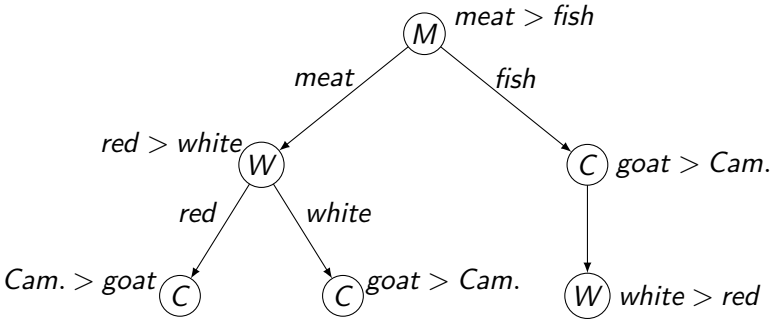
- To compare two models, we need a similarity measure but classic measures weight the same way the order on the preferred and the least preferred items
- In unsupervised learning, we use the ranking loss, the normalized difference of the mean rank of items drawn from p (preferred items are more weighted):

$$rloss_p(\succ^*, \succ') = \frac{1}{\underline{\mathcal{X}}} \left(\sum_{o \in \underline{\mathcal{X}}} p(o) rank(\succ^*, o) - \sum_{o \in \underline{\mathcal{X}}} p(o) rank(\succ', o) \right)$$

- Since $\succ^*$ (the ground truth) is unknown, the learning process seeks to minimize the mean rank of items drawn from the empirical distribution p_S according to $\succ'$:

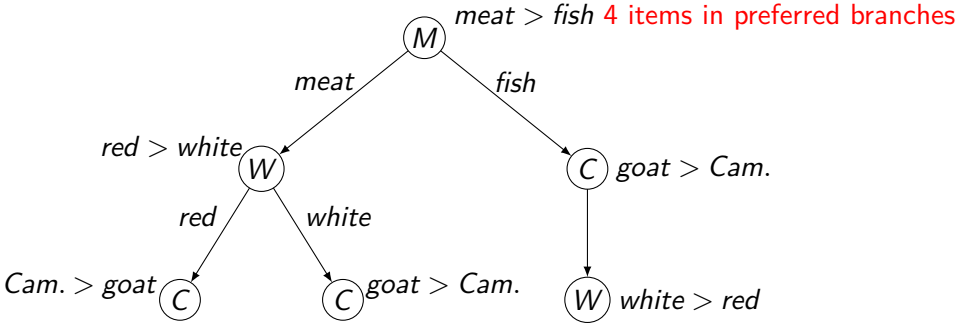
$$\hat{\phi} = argmin_{\succ'} \frac{1}{\underline{\mathcal{X}}} \sum_{o \in \underline{\mathcal{X}}} p_S(o) rank(\succ', o)$$

The properties of the rank in a LP-tree



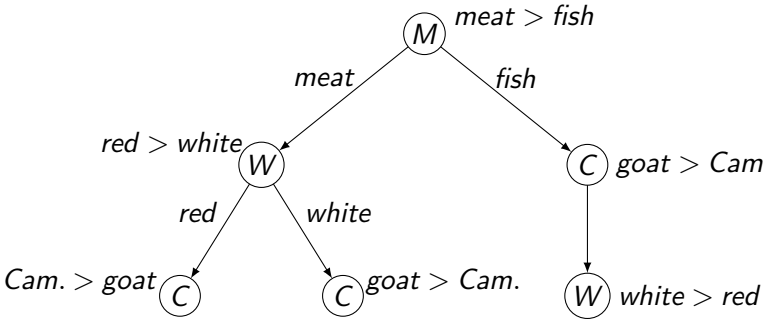
The rank of *fish*/*goat*/*red* is: $1 +$

The properties of the rank in a LP-tree



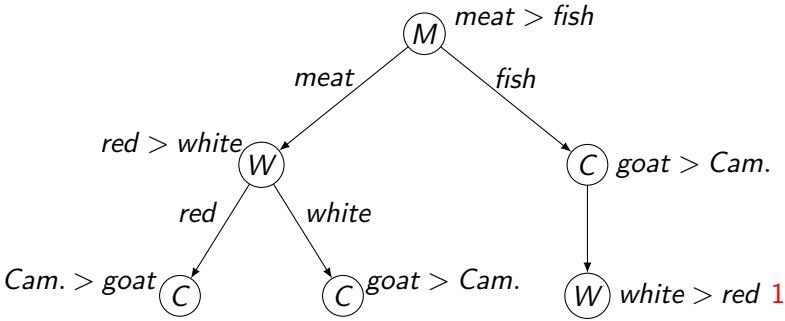
The rank of *fish*/*goat*/*red* is: $1 + 4 +$

The properties of the rank in a LP-tree



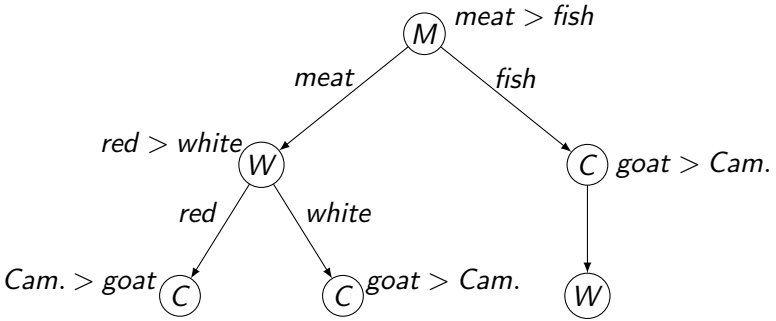
The rank of *fish*/*goat*/*red* is: $1 + \ 4 + 0 +$

The properties of the rank in a LP-tree



The rank of *fish*/*goat*/*red* is: $1 + 4 + 0 + 1 = 6$

The properties of the rank in a LP-tree



The rank of *fish*/*goat*/*red* is: $1 + 4 + 0 + 1 = 6$

$\Rightarrow$ The rank of an element o can be decomposed into the sum of contributions of each node of one branch.

Rank decomposition

Formally:

$$rank(\phi, o) = 1 + \sum_{N \in Nodes(\phi, o)} |\underline{Desc(N)}| \times r(N, o[Var(N)])$$

where

- $Nodes(\phi, o)$ be the set of nodes in the branch compatible with o
- $|\underline{Desc(N)}|$ is the product of the sizes of the domain of the attributes below the node N
- $r(N, o[Var(N)])$ is the local rank (starting at 0) of the value of o for the attribute in N

The mean rank is:

$$\frac{1}{\underline{\mathcal{X}}} \sum_{o \in \underline{\mathcal{X}}} p_{\mathcal{S}}(o) rank(\phi, o) =$$

$$\frac{1}{\underline{\mathcal{X}}} \left(1 + \sum_{N \in Nodes(\phi)} |\underline{Desc(N)}| \times p_{\mathcal{S}}(inst(N)) \times E_{p_{\mathcal{S}}}[r(N, o[Var(N)]) \mid inst(N)] \right)$$

Sample complexity

At least $S(n, \delta, \epsilon)$ examples $\Longrightarrow Pr(rloss(\phi^*, \hat{\phi}) \leq \epsilon) \geq 1 - \delta$

The result

Our upper bound on the sample complexity shows that it belongs in:

- $O(\frac{1}{\epsilon^2})$
- $O(\ln \frac{1}{\delta})$
- $O(\ln n)$
- $O(l^2)$
- $O(d^{4k})$,

where l =number of leaves, d = domain size,
k = max number of attributes per node, in practice $k < 4$

This complexity is low and shows that unsupervised learning is efficient

Sample Complexity : Sketch of Proof

$$
\begin{aligned}
\text{rloss}(\phi^*, \hat{\phi}) &= \frac{1}{|\underline{\mathcal{X}}|}\left(E_p[\text{rank}(\phi^*, \cdot)] - E_p[\text{rank}(\hat{\phi}, \cdot)]\right) \\
&\leq \frac{1}{|\underline{\mathcal{X}}|}\Big(|E_p[\text{rank}(\phi^*, \cdot)] - E_{p_{\mathcal{S}}}[\text{rank}(\phi^*, \cdot)]| \\
&\qquad + E_{p_{\mathcal{S}}}[\text{rank}(\phi^*, \cdot)] - E_{p_{\mathcal{S}}}[\text{rank}(\hat{\phi}, \cdot)] \\
&\qquad + |E_{p_{\mathcal{S}}}[\text{rank}(\hat{\phi}, \cdot)] - E_p[\text{rank}(\hat{\phi}, \cdot)]|\Big) \\
&\leq \frac{2}{|\underline{\mathcal{X}}|} \max_{\phi \in \text{LPT}_I^k} |E_p[\text{rank}(\phi, \cdot)] - E_{p_{\mathcal{S}}}[\text{rank}(\phi, \cdot)]|
\end{aligned}
$$

Sample Complexity : Sketch of Proof

$$
\begin{aligned}
&|E_p[\mathrm{rank}(\phi,\cdot)] - E_{p_{\mathcal{S}}}[\mathrm{rank}(\phi,\cdot)]| \\
&\leq \left(\max_{\substack{N \in \mathrm{nodes}(\phi) \\ v \in \underline{\mathrm{Var}(N)}}} |p(v \wedge \mathrm{inst}(N)) - p_{\mathcal{S}}(v \wedge \mathrm{inst}(N))| \right) \times \frac{d^k(d^k+1)}{2} \times \sum_{N \in \mathrm{nodes}(\phi)} |\underline{\mathrm{Desc}(N)}| \\
&\leq \left(\max_{V \subseteq \mathcal{X}, v \in \underline{V}} |p(v) - p_{\mathcal{S}}(v)| \right) \times \frac{d^k(d^k+1)}{2} \times \sum_{N \in \mathrm{nodes}(\phi)} |\underline{\mathrm{Desc}(N)}|
\end{aligned}
$$

where $d =$ bound on the domain size of the attributes.

Hoeffding's inequality is used to bound $|p(v) - p_{\mathcal{S}}(v)|$

Time complexity for computing ϕ^*

Linear lexicographic preference trees with univariate nodes
Order attributes in order of non-increasing

$$\text{Score}(p, X) = E_p^*(X) \,/\, (|\underline{X}| - 1)$$

$\Rightarrow$ can be done in $O(n \log n |\mathcal{S}| d \log d)$

Time complexity for computing ϕ^*

Linear lexicographic preference trees with multivariate nodes
For every partition of $\mathcal{X}$:

- Order parts in order of non-increasing

$$\text{Score}(p, V) = E_p^*(V) \,/\, (|\underline{V}| - 1)$$

$\phi^* =$ best partition

$\Rightarrow$ exponential number of partitions

Time complexity for computing ϕ^*

Non-linear lexicographic preference trees $\Rightarrow$???

CentraleSupélec

Conclusion and future work

Conclusion

- Unsupervised learning relies on examples of chosen items and not on comparisons
- We focus on theoretical analysis of the time and sample complexity of unsupervised learning of three classes of lexicographic preferences trees

Future work

- We conjecture that the unsupervised learning of the optimal LP-tree is NP-hard
- We are working on another unsupervised learning methods based on two-part MDL (minimum description length) to learn LP-tree and CP-net (another graphical preference model)

CentraleSupélec


Richard Booth, Yann Chevaleyre, Jérôme Lang, Jérôme Mengin, and Chattrakul Sombattheera.
Learning various classes of models of lexicographic orderings.
Preference learning, page 1, 2009.


Hélène Fargier, Pierre Francois Gimenez, and Jérôme Mengin.
Learning lexicographic preference trees from positive examples.
In *Proceedings AAAI 2018*, pages 2959–2966. ACM, 2018.