Clustering Partial Lexicographic Preference Trees

Joseph Allen, Xudong Liu, Karthikeyan Umapathy, Sandeep Reddivari
School of Computing, University of North Florida
{n01045721, xudong.liu, k.umapathy, sandeep.reddivari}@unf.edu

Abstract
In this work, we consider the problem of clustering partial lexicographic preference trees (PLP-trees), intuitive and often compact representations of user preferences over multi-valued attributes. Due to the preordering nature of PLP-trees, we define a variant of Kendall’s $\tau$ distance metric to be used to compute distances between PLP-trees for clustering. To this end, extending the previous work by Li and Kazimipour (Li and Kazimipour 2018), we propose a polynomial time algorithm $PlpDis$ to compute such distances, and present empirical results comparing it against the brute-force baseline.

Introduction
Understanding the decision patterns of users and modeling their preferences is an important problem in artificial intelligence and has received a lot of attention in recent years. A myriad of models for the task of preference representation have been proposed, at the frontier of which are partial trees, present results on PLP-tree clustering as the distance measure, and point out future research directions.

Partial Lexicographic Preference Trees
Let $V = \{X_1, \ldots, X_n\}$ be a set of $n$ categorical attributes with each $X_i \in V$ having a finite domain of values $D_i = \{x_{i1}, \ldots, x_{im_i}\}$. The combinatorial domain $CD(V)$ over $V$ is the Cartesian product $D_1 \times \ldots \times D_n$. We call elements of $CD(V)$ alternatives. A PLP-tree over $V$ is an ordered labeled tree where every non-leaf node 1) is labeled with an attribute $X_i$ from $V$, 2) has a local preference $>_{\gamma}$ (a total order over $D_i$), and 3) has $|D_i|$ outgoing edges ordered from left to right according to $>_{\gamma}$. Additionally, each attribute from $V$ appears at most once in any branch of the tree. We denote a leaf node with $\Box$, indicating a bucket of alternatives.

We compute the distance between PLP-trees by considering the disagreements between the orders they represent. Let $T_1$ and $T_2$ be two PLP-trees (equivalently, two total preorders), and $\alpha$ and $\beta$ two distinct alternatives from $CD(V)$. The ordering of $\alpha$ and $\beta$ on $T_1$ and $T_2$ either strictly agree, strictly disagree, or partially disagree. Strict agreement oc-
curs when in both \( T_1 \) and \( T_2 \) either \( \alpha \succ \beta \), \( \beta \succ \alpha \), or \( \alpha \approx \beta \), where \( \succ \) means strict preference and \( \approx \) means equivalence. Strict disagreement occurs when in \( T_1 \) we have \( \alpha \succ \beta \) and in \( T_2 \) \( \beta \succ \alpha \), or vice versa. Partial disagreement occurs when in \( T_1 \) \( \alpha \approx \beta \) and in \( T_2 \) \( \alpha \neq \beta \) (\( \alpha \succ \beta \) or \( \beta \succ \alpha \)), or vice versa. Examples and detailed discussion on the semantics, classes, and notations of PLP-trees follow in the next three sections.

**An Example of PLP-trees**

Let us consider a domain of cars specified using four binary attributes: BodyType (\( B \)): sedan (\( s \)) and sport (\( r \)), Make (\( M \)): Honda (\( h \)) and Ford (\( f \)), Price (\( P \)): low (\( l \)) and high (\( g \)), and Transmission (\( T \)): automatic (\( a \)) and manual (\( m \)).

The PLP-tree in Figure 1e bears the following user preferences. The most important attribute for the user is BodyType, for which she prefers sedan to sport. Among sedan cars, the most important attribute is Make and the user prefers Honda over Ford. Similarly, among sport cars, the most important is Transmission and manual is better than automatic. On the other hand, among the automatic sport cars, her most important attribute to consider is Make and she likes Ford more than Honda.

\[
\begin{align*}
P & \top > g \\
B & \top > r \\
T & \top > a \\
M & \top > f \\
\end{align*}
\]

Figure 1: PLP-trees over the car domain

**Semantics of PLP-trees**

We see that non-leaf nodes in a PLP-tree are to partition the space of alternatives into groups of them. Non-leaf nodes further down in the PLP-tree refine this partitioning. In other words, given an alternative and a PLP-tree, one may traverse the tree from the root down to some leaf: when at a node, descend to the left child if the alternative has the preferred value on that node, or to the right child, otherwise. A leaf node, therefore, represents a group of alternatives descending to it.

Indexing the leaves from the left most one with 1, 2, \ldots, we can now define the preference relation induced by a PLP-tree. Let \( \alpha \) and \( \beta \) be two alternatives, and \( l_T(\alpha) \) and \( l_T(\beta) \) their leaf indices in PLP-tree \( T \). We define that \( \alpha \) is at least as good as \( \beta \) (denoted \( \alpha \succeq_T \beta \)) if \( l_T(\alpha) \leq l_T(\beta) \), that \( \alpha \) is strictly better than \( \beta \) (denoted \( \alpha \succ_T \beta \)) if \( l_T(\alpha) < l_T(\beta) \), and that \( \alpha \) is equivalent with \( \beta \) (denoted \( \alpha \approx_T \beta \)) if \( l_T(\alpha) = l_T(\beta) \). Clearly, relation \( \succeq_T \) is a total preorder that is total, reflective and transitive.

**Classifications of PLP-trees**

Exploring the tree structures and the local preferences on nodes, we introduce different classes of PLP-trees. (Our classification of PLP-trees is a slight variant of the proposed by Liu and Truszczynski (Liu and Truszczynski 2015).) When the tree structure is complete, i.e., every level is full, and the nodes per the same level are all labeled by the same attribute, we collapse the tree so that the tree is simplified to a linear path, dramatically reducing the size of the tree. The collapsed tree is called an *unconditional importance* tree, or UI tree. During this collapsing, one question is about what to do with the local preferences. If the preferences on the attributes per the same level are unanimous, the same preferences per the same level are different, then a conditional preference table (CPT) is created and used as the label. This resulting tree is called a UI and *unconditional preference* tree, or UI-UP tree, for short.) If, however, the preferences per the same level are different, then a conditional preference table (CPT) is created and used as the label. This resulting tree is called a UI and *conditional preference* tree, or UI-CP tree. The CPT consists of preference rules of form \( u : a > b \) that expresses that, conditioned on \( u \), the evaluations of parent attributes, the user prefers \( a \) to \( b \) on the labeling attribute. Examples of a UI-UP and UI-CP tree and their full versions over the car domain are shown in Figure 1b and Figure 1d, respectively. Other PLP-trees that are not UI trees are called *conditional importance* trees, or CI trees. Notably, given a PLP-tree \( T \) of any class and two alternatives \( \alpha \) and \( \beta \), deciding if \( \alpha \succ_T \beta \) or \( \alpha \approx_T \beta \) can be done in time linear in the size of \( T \).

**More Notations in PLP-trees**

Let \( T \) be a PLP-tree of any class in \{UI-UP, UI-CP, CI\}, and \( n \) a node in \( T \). We denote by \( A \) node \( n \)'s ancestor attributes to be the collection of attributes labeling the ancestors of \( n \), by \( B \) node \( n \)'s branching attributes to be the set of attributes labeling those ancestors with multiple children nodes, by \( P \) node \( n \)'s parent attributes to be the set of attributes forming the conditions in the CPT at \( n \).

For instance, in Figure 1d, for the node labeled by attribute \( T \), we have \( A = \{B \} \), \( B = \emptyset \), and \( P = \emptyset \). In Figure 1e, for the node labeled by attribute \( M \) and preference rule \( f > h \), we have \( A = \{B,T\} \), \( B = \{B,T\} \), and \( P = \emptyset \).
We say two nodes from two PLP-trees are consistent if they assign the same value to all common branching ancestor attributes. Let \( b[B] \) and \( b'[B'] \) denote the value assignments, \( b \) and \( b' \), to the branching ancestor attribute sets \( B \) and \( B' \) for two nodes \( n \) and \( n' \), respectively. Formally, \( n \) and \( n' \) are consistent if \( b[B \cap B'] = b'[B \cap B'] \). Clearly \( n \) and \( n' \) are consistent if \( B \cap B' = \emptyset \).

**Distance Between PLP-trees**

We see that the distance between PLP-trees involves partial disagreements that are not accounted for in the regular definition of Kendall’s \( \tau \). Thus, we first define a variant Kendall’s \( \tau \), called partial Kendall’s \( \tau \), denoted as \( \tau' \). Let \( SD_{T_1, T_2} \) and \( PD_{T_1, T_2} \) be the set of pairs of alternatives on which \( T_1 \) and \( T_2 \) strictly disagree and partially disagree, respectively. The metric \( \tau' \) is then the weighted sum:

\[
\tau'(T_1, T_2) = c_1 \times |SD_{T_1, T_2}| + c_2 \times |PD_{T_1, T_2}|
\]

where \( c_1 \) and \( c_2 \) are two constant coefficients that may be adjusted based on which of the disagreement type is favored over the other.

Clearly, computing \( \tau'(T_1, T_2) \) boils down to counting the pairs in \( SD_{T_1, T_2} \) and \( PD_{T_1, T_2} \). We can compute the number of strict disagreements between two PLP-trees, \( SD_{T_1, T_2} \), using the \( LpDis \) algorithm (Li and Kazimipour 2018). To compute \( |SD_{T_1, T_2}| \), for every non-leaf node \( n \) in \( T_1 \) we traverse every non-leaf node \( n' \) in \( T_2 \) and accumulate \( |SD_{n,n'}| \). Equivalently, we compute \( \sum_{n\in T_1, n'\in T_2} |SD_{n,n'}| \). Due to space constraint, we refer the reader to the \( LpDis \) paper for details about the traversal and the equations used to compute \( |SD_{n,n'}| \).

We now present the equation for computing \( |PD_{T_1, T_2}| \), the novel contribution in our \( PlpDis \) algorithm. To compute \( |PD_{T_1, T_2}| \), we extend the \( LpDis \) traversal down to the leaves in both trees and compute the number of alternative pairs decided both at a leaf node in one tree and at a non-leaf node in the other, which we denote by \( |P_n \cap P_{n'}| \). Thus, we have \( |PD_{T_1, T_2}| = \sum_{n\in T_1, n'\in T_2} |P_n \cap P_{n'}| \), where exactly one of \( n \) and \( n' \) is a leaf node. To be decided at a node means that the preference relation over a pair of alternatives is encoded by that node of the tree. Since leaf nodes always encode the relation \( = \) and non-leaf nodes encode the relation \( \succ \), according to the local preference \( \succ \), computing \( |P_n \cap P_{n'}| \) gives the number of partial disagreements. For all distinct pairs of \( n \) and \( n' \) from both trees (w.l.o.g., \( n \) is a leaf), if the attribute labeling \( n' \), \( V_{n'} \), is not an ancestor of \( n \) and the branching attributes in both trees are consistent, we have:

\[
|P_n \cap P_{n'}| = \left( \frac{|D_{V_{n'}}|}{2} \right) \times \prod_{x \in A \setminus B} |D_x| \times \prod_{y \in (A \setminus (V_n \cup V_{n'}))} |D_y|^2
\]

The first term computes the number of distinct pairs of values in the domain of \( V_{n'} \), and the other two terms adjust for possible values of ancestor and descendant (including missing) attributes, respectively.

Clearly, our algorithm \( PlpDis \) to compute \( \tau'(T_1, T_2) \) runs in time polynomial in the size of \( T_1 \) and \( T_2 \). This follows from the fact that in the worst case we compare every pair of nodes between the two trees.

**Clustering PLP-trees**

With \( PlpDis \) defined, we may now reason about collections of PLP-trees through clustering. Because \( PlpDis \) does not satisfy the triangle inequality \( d(x, z) \leq d(x, y) + d(y, z) \), we focus on clustering algorithms that utilize distances between pairs of PLP-trees, i.e., those that take a distance or similarity matrix as input.

These methods included spectral clustering, affinity propagation, and agglomerative clustering (or AGNES, for short). Spectral clustering (Ng, Jordan, and Weiss 2002) takes the similarity matrix as input, computes the spectrum of it to reduce dimensionality, and clusters the result in fewer dimensions with another algorithm such as K-means. Affinity propagation (Frey and Dueck 2007) passes messages between data points to find representative data points, called exemplars, to perform clustering. Lastly, AGNES (Rokach and Maimon 2005) is a bottom-up hierarchical clustering method that starts with viewing each data point per se as a cluster and repeatedly finds two closest clusters and merges them. Based on how the distance is measured between two clusters of data points, AGNES provides three different variants. Given two clusters \( C_i \) and \( C_j \) of PLP-trees, we define min distance to be \( d_{min}(C_i, C_j) = \min_{T_i \in C_i, T_j \in C_j} \tau'(T_i, T_j) \), max distance \( d_{max}(C_i, C_j) = \max_{T_i \in C_i, T_j \in C_j} \tau'(T_i, T_j) \), and average distance \( d_{avg}(C_i, C_j) = \sum_{T_i \in C_i, T_j \in C_j} \tau'(T_i, T_j) \).  

AGNES is called single-linkage, if \( d_{min} \) is used as the distance measure; complete-linkage, if \( d_{max} \) is used; and average-linkage, if \( d_{avg} \) is used.

We now define the the Dunn index (\( DI \)) and the Davies-Bouldin index (\( DBI \)) which we’ll use to evaluate cluster quality. The Dunn index is a worst case measurement of the ratio of the minimum intercluster distance to the maximum intracluster distance:

\[
DI = \min_{1 \leq i \leq k} \left\{ \min_{j \neq i} \left( \frac{d_{avg}(C_i, C_j)}{\max_{1 \leq l \leq k} |avg(C_l)|} \right) \right\}
\]

where \( k \) is the number of clusters and \( avg(C_l) \) is the intracluster distance for cluster \( l \), \( avg(C_l) = \frac{1}{|C_l|} \sum_{T_i \in C_l} \tau'(T_i, T_j) \). The Davies-Bouldin index is similar, but differs in that it finds a worst case ratio for every cluster and takes the average:

\[
DBI = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left( \frac{avg(C_i) + avg(C_j)}{d_{avg}(C_i, C_j)} \right)
\]

Intuitively, both cluster indices measure the extent to which a set of clusters minimize overall intracluster distances while also maximizing intercluster distances.

**Experimental Results**

To test \( PlpDis \)-powered clustering, we applied the selected algorithms to the task of clustering PLP-forests, consisting of random mixtures of UI-UP, UI-CP, and CI PLP-trees, learned from a Car Evaluation dataset using Liu
Table 1: The $DI$ to $DBI$ ratio for each clustering

<table>
<thead>
<tr>
<th>Forest Size</th>
<th>100</th>
<th>500</th>
<th>1,000</th>
<th>2,500</th>
<th>5,000</th>
<th>10,000</th>
</tr>
</thead>
<tbody>
<tr>
<td># Clusters</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>14</td>
<td>17</td>
<td>24</td>
</tr>
<tr>
<td>Spectral</td>
<td>0.090</td>
<td>0.386</td>
<td>0.387</td>
<td>0.352</td>
<td>DNF</td>
<td>DNF</td>
</tr>
<tr>
<td>Aff. Prop.</td>
<td>0.337</td>
<td>0.376</td>
<td>0.342</td>
<td>0.317</td>
<td>0.217</td>
<td>0.238</td>
</tr>
<tr>
<td>AGNES-A</td>
<td>2.628</td>
<td>2.354</td>
<td>2.172</td>
<td>1.555</td>
<td>1.196</td>
<td>1.323</td>
</tr>
<tr>
<td>AGNES-S</td>
<td>2.876</td>
<td>2.157</td>
<td>2.172</td>
<td>2.857</td>
<td>1.888</td>
<td>2.034</td>
</tr>
<tr>
<td>AGNES-C</td>
<td>3.393</td>
<td>2.894</td>
<td>2.245</td>
<td>1.176</td>
<td>0.884</td>
<td>0.886</td>
</tr>
</tbody>
</table>

and Truszczynski’s greedy algorithm (Liu and Truszczynski 2019). In the experiment we learned forests of sizes 100, 500, 1,000, 2,500, 5,000, and 10,000 with each tree learning from 100 examples. To evaluate cluster quality we used a ratio of $DI$ to $DBI$, $DI/DBI$, since higher is better for $DI$ and lower is better for $DBI$. In addition to this quantitative measure, to visually assess cluster quality we constructed KNN-graphs from the distance matrix and then colored vertices according to each algorithm’s cluster assignments. The reasoning for this representation follows from the fact that the distance matrix itself can be seen as a fully-connected graph. By considering only the $K$ nearest neighbors for each tree instead, we see that clusters emerge based on connectedness. After testing multiple values of $K$, we set $K = 10$ since it provided the best visualizations of the clusters for this experiment. Note that since affinity propagation is the only algorithm that doesn’t take the number of clusters $k$ as input, we simply run it first and provide the $k$ it finds to the other algorithms, but $k$ can also be chosen via search.

The results of these tests are included in Table 1, where the best score is bolded in each column. Note that “DNF” in the table for spectral clustering indicates that the algorithm did not finish within a set timeout threshold of 20 minutes. In Figure 2 we show the KNN-graphs from the test with 1000 trees and 8 clusters for each clustering algorithm. Clearly the visual quality of the clusters in these graphs supports the numerical results. Overall, all three variants of AGNES perform better compared to spectral clustering and affinity propagation. AGNES consistently detects the highly-connected vertices (similar trees) in the KNN-graphs, but the other methods do not. This trend continues for all tested forest sizes, with single-linkage taking the lead for the largest forest sizes (2,500+).

Future Work

Clustering preference models provides significant insights into decision making and can support many essential applications, such as understanding demographical distributions of user preferences and how those preferences change over time. To this end, we introduced a polynomial algorithm $Plp\!\!Dis$ that computes Kendall’s $\tau$ distance between PLP-trees, and presented empirical results using it in combination with existing clustering methods. For future directions, we plan to improve the scalability of our implementation to handle larger numbers of models. We also intend to explore other preferential datasets as well as visualization techniques to improve the interactivity of the clustering results.

References


1www.unf.edu/~xudong.liu/preflearnlib.html

Figure 2: 1000 PLP-tree KNN-Graphs (8 clusters)