LEARNING THE TREE STRUCTURE OF A NESTED LOGIT CHOICE MODEL FROM DATA

Yilong Ju

A THESIS

in

Data Science

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Master of Science in Engineering

2019

Shivani Agarwal
Supervisor of Thesis Signature

Susan B. Davidson
Graduate Group Chairperson Signature

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my advisor, Prof. Shivani Agarwal. During the latter course of my master's program, she offered me an opportunity to conduct research in machine learning and ignited my passion for pursuing a higher degree. Without her encouragement and support, it would not have been possible for me to finish this thesis and to be admitted by a Ph.D. program in computer science.

Besides, I would like to thank Nicholas Johnson, who was a postdoc working with my advisor when I first joined the research project. I have gained a lot of insight and knowledge from the discussions we had. I really appreciate his guidance and suggestions.

Also, I am grateful to Saurav Bose, who was a master's student when the project started, for his contribution to implementing and improving several key algorithms. I had a really good time working with him.

I would also like to thank Arpit Agarwal and Mingyuan Zhang, who are current Ph.D. students working with Prof. Agarwal, for their help on the experiment design and shaping the writing of this thesis.

Last but not the least, I would like to thank my parents for providing a stable family environment and supporting me throughout my life.

ABSTRACT

Multinomial logit (MNL) models are widely used for modeling consumer choice behavior, but these models are somewhat restricted because of the underlying assumption, independence of irrelevant alternatives (IIA), which is often violated in real-world scenarios. The nested logit models are not restricted by the IIA assumption because the alternatives are assigned to leaves of a tree such that the IIA assumption holds for alternatives that are siblings but not necessarily between those that are not siblings. Thus, a proper tree structure is required for a nested logit model. However, the most common approach to learn the tree is to empirically propose several possible tree structures and perform statistical tests to select a best one, which becomes impractical when there is insufficient prior knowledge or when there are many alternatives. Addressing this issue, in this thesis, we propose an algorithm based on divisive clustering using some proxies of distance. The algorithm is evaluated on several synthetic and real-world data sets, and is compared to other algorithms. It turns out that our proposed algorithm has better performance in synthetic data sets when the underlying tree is wide, especially when there are insufficient training data. However, on real-world data sets, both our algorithm and an algorithm proposed by Benson et al. (2016) cannot produce a nested tree better than what an MNL model assumes.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

In this thesis, we study the problem of learning the underlying tree structures for *nested logit* models from data. In many applications, such as recommender systems, transportation systems, marketing, economics and consumer surveys, there is a need to develop statistical models to study how users make selections among a set of *choice alternatives* or *items*. *Discrete choice* models are developed for these purposes (McFadden et al., 1977), and *Multinomial logit* (MNL) models, one type of discrete choice models, are widely used in the applications mentioned above (Train, 1978; Berry, 1994; Lockshin et al., 2006; Yang et al., 2011; Elshiewy et al., 2017), since MNL models are tractable and easy to interpret.

Despite the popularity of MNL models, however, the way they are constructed indicates an underlying assumption, known as the *independence of irrelevant alternatives* (IIA) assumption: the probability of choosing one alternative over another is the same in any choice set containing these two alternatives. Unfortunately, this assumption is often violated in real world scenarios, making MNL models somewhat inaccurate.

Example 1. (McFadden, 1973) Consider a case where people choose the means of commuting to work. The choice set $S = \{\text{driving, red bus}\}$. We assume people are indifferent between them. Then, we know that the chance for each alternative to be chosen is 0.5. If a third alternative almost indistinguishable from the red buses, say blue buses, is added to the choice set, it is reasonable to assume that the probabilities of choosing driving, taking red buses and taking blue buses are 0.5, 0.25, 0.25. The ratio of the probabilities of choosing driving and taking red buses, however, is no longer 1:1, which violates the IIA assumption. Therefore, the MNL model does not apply and a more complex model is required.

As another type of discrete choice models, nested logit models are popular and powerful due to their ability to model the hierarchy of users' decision making by assigning the alternatives to the leaves of a proper tree structure, where similar alternatives are likely to be siblings, as they are typically substitutable for each other, and distinct alternatives will locate in different branches of the tree. In this way, nested logit models are not subject to the restriction imposed by the IIA assumption. The details will be discussed in Chapter 3. It can also be shown that MNL models are just a special case of nested logit models, since

the underlying tree of MNL models can be formed by assigning all alternatives in the universe to the children of the root.

Although with proper tree structures, nested logit models can be more accurate than MNL models, it typically takes a lot of effort to finally figure out the best tree structure because the number of possible tree structures grows rapidly with respect to the number of alternatives. For example, in Koppelman and Bhat, 2006, the best tree structure for the nested logit model is selected from 15 tree structures empirically proposed with statistical tests. Therefore, it is important to design algorithms able to automatically learn the tree structures for nested logit models from data.

Our results. We propose an algorithm `DivisR` based on divisive clustering that can recover a tree structure for a nested logit model from choice data. In the experiments, we evaluate the MNL models and the nested logit models with underlying tree structures recovered by our algorithm and the algorithm proposed in Benson et al. (2016) on both synthetic and real-world data sets, and compare the out-of-sample likelihood. It turns out that our algorithm outperforms the algorithm proposed in Benson et al. (2016) in wide synthetic trees, whose width is greater than the depth, but our algorithms do not present the same power when facing deep trees. As for the performance on real world choice data, the nested logit models learned by both algorithms can be only marginally better or even worse than the MNL model.

Roadmap. The thesis is organized as follows: In Chapter 2, we review some past research on recovering tree structures and especially those related to nested logit models. In Chapter 3, we introduce some background knowledge for the problem of learning nested trees. In Chapter 4, we present the problem statement. In Chapter 5, we describe the algorithm we propose. In Chapter 6, we evaluate our algorithm on both synthetic datasets and real world datasets and compare the results with those produced by other algorithms. In Chapter 7, we give a brief discussion and conclude the thesis.

CHAPTER 2

Related Work

Multinomial logit (MNL) models were formally discussed in McFadden (1973) and McFadden (1981) and more recently surveyed in Elshiewy et al. (2017). Due to restriction imposed by the assumption of independence of irrelevant alternatives (IIA), nested logit models are often used in real-world scenarios. These models can be more powerful than MNL models when using proper tree structures. Improper tree structures, however, may be even worse than MNL models, as shown in (Koppelman and Bhat, 2006; Benson et al., 2016). Typically, researchers empirically propose several possible tree structures and employ statistical tests, such as the Wald test, likelihood ratio test, and Lagrange multiplier test (Hausman and McFadden, 1984), to determine whether a nested tree structure will model the data significantly better than a flat tree that an MNL model assumes (Hensher and Ton, 2000; Das et al., 2012; Mojaverian et al., 2014). Unfortunately, it becomes impractical when there is insufficient prior knowledge or when there are many alternatives. In Koppelman and Bhat (2006), it is shown that even when there are only 6 choice alternatives in the universe, there can be 2,711 possible nesting structures. Thus, efficient algorithms should be explored.

This tree recovery problem is closely related to the problem of learning the phylogenetic trees of a set of species, which is the fundamental problem in computation evolutionary biology. It has been receiving attention for a long time (Fitch and Margoliash, 1967; Buneman, 1971; Sattath and Tversky, 1977; Colonius and Schulze, 1981; Bandelt and Dress, 1986; Kannan et al., 1996; Agarwala et al., 1998; Brodal et al., 2001; Wu et al., 2008; Gambette et al., 2012). One possible solution to the problem is query (or oracle/experiment)-based algorithms. Both triplet queries and quartet queries are used in the literature, triplet queries lead to rooted trees (Kannan et al., 1996), while quartet queries lead to unrooted trees (Gambette et al., 2012). We focus on triplet queries because they produce rooted trees, which are the desired tree structures in the context of discrete choice. Typically, triplet queries reveal information on which pair of item has the least common ancestor (LCA) that is a descendant of the LCA's of the other two pairs, or all three pairs of items have the same LCA.

Aho et al. (1981) studied an algorithm for constructing a tree to satisfy a set of constraints on the LCA's. The constraints are of the form $(i, j) < (k, l)$, where $i \neq j$ and $k \neq l$, meaning the LCA of i and j is

a proper descendant of the LCA of k and l. They proved (in Theorem 3) that if all constraints are of the form $(i, j) < (i, k)$, then the algorithm can recover the tree in $O(n^2)$ time where n is the number of constraints, by partitioning the alternatives and recursing on each subset with the constraints only pertaining to the alternatives in the subset. Their algorithm reconstructs the tree structure in a top-down hierarchical manner, but in order to obtain the constraints, one must have prior knowledge of the true tree structure, otherwise an arbitrary set of LCA constraints cannot guarantee the reconstruction of the ground truth.

Kannan et al. (1996) proposed an $O(n^2)$ experiment-based algorithm for unbounded degree trees, where n is the number of leaves. It computes the equivalence relation with experiments, determines the order on the equivalence classes with a binary insertion sort, and recurses on each equivalence class. The experiment can be done by examine the dissimilarity matrix or biological experiments on DNAs. They did not mention, however, the implementation of the experiments in the context of discrete choice.

Benson et al. (2016) also proposed a similar oracle-based $O(n^2)$ algorithm for inducing the nested tree structures, where n is the number of alternatives in the universe. They implemented the oracle using χ^2 tests and presented an alternative greedy algorithm due to its simplicity and its effectiveness on sparse, real-world datasets. The algorithm was evaluated on synthetic and real-world data sets and improves the likelihood on some of the datasets, compared to MNL models. However, their algorithm is sometimes too greedy and the nests formed does not guarantee an improvement of likelihood.

Emamjomeh-Zadeh and Kempe (2018) proposed two algorithms for representing hierarchical clustering with binary trees: a simple one with $O(n \log n)$ queries, which assumes all the queries responses are correct, and a robust version, which can recover the correct hierarchical clustering with probability at lest $1 - \delta$, using $O(n \log n + n \log(1/\delta))$ queries, and only increase the number of queries by a constant factor depending on the correctness probability p. Nevertheless, they only considered the case of binary trees.

Distance(dissimilarity)-based algorithms are more favorable for our task since in most of the time we can find a proxy for the distances between each pair of alternatives and reconstruct the tree structure based on them. In the context of learning the phylogenetic trees, the algorithms are able to find a tree in which the species are represented as leaves and fit a tree metric corresponding to the distances (Sattath and Tversky, 1977; Colonius and Schulze, 1981; Agarwala et al., 1998). The two main types of tree metrics,

namely the ultrametric and the additive tree metric, impose some constraints on the distances between the leaves. In nested logit models, however, once the tree structure is identified, the distances do not matter anymore. Instead, we focus on learning the edge weights based on the choice probabilities learned from data.

Therefore, we focus on divisive clustering algorithms since they are distance-based, but there are little constraints on the metric. And unlike agglomerative clustering which always generates binary trees, they produce unbounded degree trees. The divisive clustering algorithm splits the alternatives into several subsets and recurses on each one of them. It is also more efficient than the agglomerative clustering, since it does not generate a complete hierarchy for every single leaf (Aggarwal, 2013). k-means (MacQueen, 1967) and k-medoids (Reynolds et al., 2004) are common methods for splitting the alternatives. k-medoids is preferred as the mean of the alternatives does not make sense unless we train some embedding on the alternatives. Typically, the splits are greedy, which means we do not make any change to previous clusterings. Depending on the value of k and distance metrics, the outcomes can be different. Hence we need to find the optimal k at each level. We use the k that generalizes best to new data, using validation sets. We will introduce the details in the next chapter.

CHAPTER 3

Preliminaries

3.1. Multinomial Logit (MNL) Models

We briefly introduce the standard multinomial logit model. More detailed explanation can be found in McFadden (1973) Bierlaire (1998). In discrete choice models, each alternative is usually associated with a real number, known as *utility*. When making selections, the choice c is made by user a such that $c = \text{argmax}_{c' \in S} V_{ac'}$ in the given choice set $S \subseteq \mathcal{U}$, where $V_{ac'}$ is the utility of alternative c' assigned by user a and $\mathcal{U}$ is the universe of alternatives. Utilities can also be associated with the attributes of the alternatives, such as cost and quality, as well as users' attributes, like gender and income. Besides, in real world, there is always uncertainty involved in the decision making process. Manski (1975) identified four different sources of uncertainty: unobserved alternative attributes, unobserved users' attributes, measurement error and proxy, or instrument variables. Hence *random utility models* (RUM) are popular because of the ability to capture some level of uncertainty.

In these models, a random noise ε_{ac} is added to the deterministic utility V_{ac}. Hence the overall utility, or utility $U_{ac} = V_{ac} + \varepsilon_{ac}$. User a selects an alternative c such that $c = \text{argmax}_{c' \in S} U_{ac'}$. By assuming ε_{ac}, $c \in \mathcal{U}$ follows an *i.i.d.* Gumbel distribution, whose density function is $f(\varepsilon) = e^{-\varepsilon - e^{-\varepsilon}}$, we obtain an MNL model. In this thesis, we assume the utilities are only associated with the alternatives themselves, irrelevant of any alternative or user attributes, i.e. $\exists f : \mathcal{U} \to \mathbb{R}$, $V_c = f(c)$, where V_c is the deterministic utility of an alternative $c \in \mathcal{U}$ and the overall utility $U_c = V_c + \varepsilon_c$. It follows that the probability of any user choosing alternative c from a set of alternatives S is

$$P_{c|S} = \Pr\big(U_c \geq \max_{c' \in S, c' \neq c} U_{c'}\big) = \frac{e^{V_c}}{\sum_{c' \in S} e^{V_{c'}}}. \tag{3.1}$$

Sometimes we use the weight of alternative c, $w_c = e^{V_c}$, for simplicity. Hence, Eq. 3.1 can be simplified to

$$P_{c|S} = \frac{w_c}{\sum_{c' \in S} w_{c'}},$$

3.2. The Independence of Irrelevant Alternatives (IIA) Assumption

In Chapter 1, we briefly mentioned the IIA assumption and how it make the MNL model fail. From Eq. (3.1), we can verify that

$$\frac{P_{c_1|S}}{P_{c_2|S}} = \frac{e^{V_{c_1}} / \sum_{c' \in S} e^{V_{c'}}}{e^{V_{c_2}} / \sum_{c' \in S} e^{V_{c'}}} = \frac{e^{V_{c_1}}}{e^{V_{c_2}}}.$$

The relative probability of choosing c_1 over c_2 only depends on V_{c_1} and V_{c_2}, regardless of the choice set S. As we observed from Example 1, the IIA assumption may impose restrictions on the MNL models.

3.3. Nested Logit Models

Nested logit models do not make the IIA assumption (McFadden, 1981). The issue in **Example 1** can be resolved by assigning the alternatives to the leaves of a rooted tree, as shown in Figure 3.1. The number next to an edge is the probabilities of choosing the edge, which corresponds to the alternative or the set of alternatives at the bottom end of the edge. Then, if the choice set $S = \{$driving, red bus$\}$, a user first makes selections between driving or buses, each with probability 0.5. Then, since red bus is the only option under node buses, the overall probabilities of choosing driving or red bus are 0.5 and 0.5. When the choice set $S' = \{$driving, red bus, blue bus$\}$, after choosing buses at the first level, the user needs to make choice between red bus and blue bus, where the probability is 0.5 for each alternative. Thus, the probabilities of choosing driving, taking red buses and taking blue buses are 0.5, 0.25, 0.25, which is consistent with our expectation. General nested logit models have a similarity parameter μ which depicts how the utilities of alternatives under the same nested are correlated. Similar to the work of Benson et al. (2016), we only consider the case when $\mu = 1$, which means the utilities are not correlated. As such, the problem is computationally simplified and we can focus on recovering the tree structures.

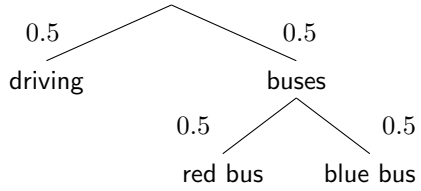


Figure 3.1: Choice of the means of commuting to work

As we can see, with proper nested tree structures, nested logit models can work in more general cases than MNL models do. And they are probably able to predict users' selections with a higher likelihood. A common approach to identifying such nested tree structures is to empirically propose several possible tree structures and to select the best one based on significant tests on the likelihoods (Hensher and Ton, 2000; Koppelman and Bhat, 2006; Das et al., 2012; Mojaverian et al., 2014). However, this approach becomes impractical when there is insufficient prior knowledge on the tree structures or there are a large amount of alternatives. Thus, algorithms that can automatically learn nested tree structures from *choice data* should be explored. Choice data are data sets comprised of observations of users' selections from various choice sets. We will formly define them in Chapter 5.

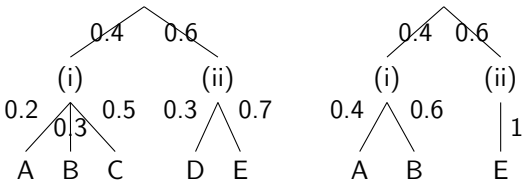
We follow the set representation of nested logit trees in Benson et al. (2016): Leaf nodes are represented as singleton sets consisting of a single alternative, while internal nodes or *nests* are sets whose elements are their children. Then, the root can represent the whole tree. For example, the tree in Fig. 3.1 can be represented by $\{\{\text{driving}\}, \{\{\text{red bus}\}, \{\text{blue bus}\}\}\}$. Let $T[\mathcal{U}]$ be the tree structure for the universe of alternatives $\mathcal{U}$. For a subset of alternatives $S \subseteq \mathcal{U}$, the corresponding tree structure, the *induced tree* $T[S|\mathcal{U}]$ is defined as follows (Emamjomeh-Zadeh and Kempe, 2018): keep the leaves of $T[\mathcal{U}]$ that are in S, remove other leaves, then remove any internal nodes without children, and merge the internal node or root who has only one child with the child. We omit $\mathcal{U}$ when the context is clear.

3.4. Choice Probabilities in Nested Logit Models

Following Benson et al. (2016), the selection of a choice alternative c from a choice set S can be regarded as a series of selections in the induced tree $T[S]$, level by level. Suppose c is located at depth d of $T[S]$. Then, at a level l, the selection of nest c_l is actually made among S_l, the set of nests at the level l of $T[S]$. Then, the probability of choosing c_l is $P_{c_l|S_l} = e^{V_{c_l}} / \sum_{c' \in S_l} e^{V_{c'}}$, where V_{c_l} is the utility associated with nest c_l. Then, we can obtain a series of selections when traversing the edges from the root to c, and the choice probabilities $P_{c_l|S_l}$, $l = 1, 2, ..., d_i$. Thus, the probability of choosing c from S is $P_{c|S} = \prod_{l=1}^{d} P_{c_l|S_l}$. Let $P_{c|S,T}$ denote the probability of choosing c from a choice set $S \subseteq \mathcal{U}$, when the underlying tree structure is T. When the context is clear, we omit the T. And it is obvious that $P_{c|S,T} = P_{c|S,T[S]}$. Next, we give an numeric example to show how the choice probabilities are calculated in a nested logit model.

Example 2. Consider a tree T structure shown in Figure 3.2(a). Items in the universe $\mathcal{U}$ ($= \{A, B, C, D, E\}$ in this case) are assigned to the leaves of T. The numbers labeled on the edges are choice probabilities, which are just the normalized values of e^{V_c} within the same level of the tree, where V_c is the utility of a node c. Thus, based on the choice probability we introduced above, we can calculate them as such, for example: users have a probability of 0.6 to choose Branch (ii), then they have a probability of 0.3 to choose D and a probability of 0.7 to choose E, i.e. they have an overall probability of 0.18 to choose D and 0.42 to choose E. Thus, if all choices are available, the actual probability of choosing A, B, C, D, E are $0.08, 0.12, 0.2, 0.18, 0.42$, respectively.

The nested logit model can violate the IIA assumption when only a subset of choice alternatives are available. In such cases, we consider the induced tree $T[S]$. For example, let $S = \{A, B, E\}$, and the induced tree is shown in Figure 3.2(b). Since the utilities for each edge remain the same, calculating the new choice probabilities is equivalent to normalizing the remaining edge probabilities. Then, the edge probabilities corresponding to alternative A and alternative B become $0.2/(0.2 + 0.3) = 0.4$ and $0.3/(0.2 + 0.3) = 0.6$. For E, it becomes 1 since it is the only choice in the nest (branch (ii)). Then, the overall probability of choosing A, B, E are $0.16, 0.24, 0.6$, respectively. Note that we still have $P_{A|S}/P_{B|S} = 0.16/0.24 = 0.08/0.12 = P_{A|\mathcal{U}}/P_{B|\mathcal{U}}$. Because A and B are siblings, the relative probability of choosing A over B is unchanged. However, $P_{A|S}/P_{E|S} = 0.16/0.6 \neq 0.08/0.42 = P_{A|\mathcal{U}}/P_{E|\mathcal{U}}$. These observations show that the IIA assumption holds within a nest, but not necessarily between nests. In the next chapter, we formally define the problem that our algorithm aims to solve.



(a) The original tree T (b) A tree induced from T for S

Figure 3.2: An example of calculating choice probabilities for a nested logit model

CHAPTER 4

Problem Statement

4.1. The Problem

Based on all the previous discussion, we formally define the problem of learning nested tree structure from data. Suppose we have a data set of users' choice behavior: $\mathcal{D} = \{(S_1, c_1), (S_2, c_2), ..., (S_m, c_m)\}$, where S_i is the choice set, from which a selection c_i is made, $i \in [m]$. m is the number of observations. Let $\mathcal{U}$ be the universe of alternatives, and T be a tree, each of whose leaves is assigned an alternative in $\mathcal{U}$. Let $\mathcal{T}_\mathcal{U}$ denote the set of all possible nested trees for all alternatives in $\mathcal{U}$, and let $\mathsf{ALLH}(\mathcal{D}, T, \mathcal{V})$ denote the average log-likelihood of tree T with node utilities $\mathcal{V}$ evaluated on the data $\mathcal{D}$, averaged over number of observations m, where $\mathcal{V} = \{V_x | x$ is an internal node or an leaf node (alterative) in $T\}$. We will discuss the calculation of $\mathsf{ALLH}(\mathcal{D}, T, \mathcal{V})$ in detail in Section 5.5. Then, the goal is to find a tree T such that with utilities $\hat{\mathcal{V}}$ estimated by maximum likelihood estimation, it has the largest likelihood when predicting the users choices in data $\mathcal{D}$, i.e.

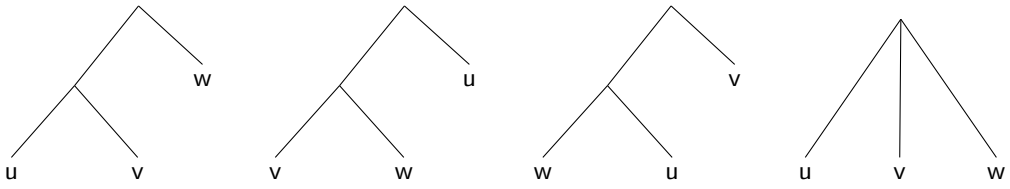
$$T = \underset{T' \in \mathcal{T}_\mathcal{U}}{\operatorname{argmax}} \max_{\mathcal{V}} \mathsf{ALLH}(\mathcal{D}, T', \mathcal{V}).$$

4.2. The Queries

In order to be able to infer a best tree structure from $T_\mathcal{U}$, first we should make it clear how to exploit any information about the tree we could obtain. In those query-based (or oracle/experiment-based) algorithms (e.g. Kannan et al., 1996; Benson et al., 2016; Emamjomeh-Zadeh and Kempe, 2018), a query reveals for 3 alternatives u, v, and w that which pair of them out of 3 possible pairs is the closest pair. Thus, one single query can rule out 2 out of 4 possible formations for any 3 alternatives, and two queries will reveal the only possible formation. The 4 possible formations are illustrated in Fig. 4.1.

4.3. Implementing the Queries

When implementing the query, in the context of reconstructing phylogenetic trees, Kannan et al. (1996) used some biological experiments on the double strands of DNA of two species to determine the distance



(a) u and v are the closest pair

(b) v and w are the closest pair

(c) w and u are the closest pair

(d) All 3 alternatives are equally close

Figure 4.1: Possible formations of 3 alternatives in a tree

between them, from which the closest pair of species (out of 3) can be found. As for Benson et al. (2016), their implementation is more related to choice data and bases on the following observation, which are also shown in Example 1 and Example 2:

Observation 1. Assume there is no alternative in the tree with 0 utility,

1. If the relative probability of choosing u over v given choice set $S = \{u, v\}$, denoted by $P_{uv|S}$ is the same as that given the choice set $S' = \{u, v, w\}$, $P_{uv|S'}$, then the LCA of u and v, denoted by LCA(u, v) is located no higher than LCA(u, w) and LCA(v, w) do. This corresponds to the cases in Fig. 4.1(a) or 4.1(d).

2. If $P_{uv|S} < P_{uv|S'}$, then LCA(v, w) is located lower (or deeper) than LCA(w, u) and LCA(u, v) do. This corresponds to the case in Fig. 4.1(b).

3. If $P_{uv|S} > P_{uv|S'}$, then LCA(w, u) is located lower than LCA(u, v) and LCA(v, w) do. This corresponds to the case in Fig. 4.1(c).

4. From 1, 2, and 3, it turns out that if and only if $P_{uv|S} = P_{uv|S'}$, u and v are siblings.

We give a brief explanation for the observation. For Case 1, it is obvious because the IIA assumption holds within a nest. For Case 2 (Case 3 is similar), suppose the edge probability in Fig. 4.1(b) for u is p_u, for the nest $\{\{v, w\}\}$ is $p_{v,w}$, and in the nest is p_v for v and p_w for w, as shown in Fig. 4.2. Note that these edge probabilities are calculated from the utilities for each node - e.g. $p_u = e^{V_u}/(e^{V_u} + e^{V_{v,w}})$, where V_u is the utility of u and $V_{v,w}$ is the utility of the nest $\{\{v, w\}\}$. Thus,

$$P_{uv|S} = \frac{\frac{p_u}{p_u+p_{v,w}}}{\frac{p_{v,w}}{p_u+p_{v,w}}} = \frac{p_u}{p_{v,w}} < \frac{p_u}{p_{v,w}} \cdot \frac{p_v + p_w}{p_v} = \frac{\frac{p_u}{p_u+p_{v,w}}}{\frac{p_{v,w}}{p_u+p_{v,w}} \cdot \frac{p_v}{p_v+p_w}} = P_{uv|S'}$$

Note that it is reasonable to postulate that no alternative has a utility of 0, and hence $p_w > 0$. Otherwise we can remove it from the universe.

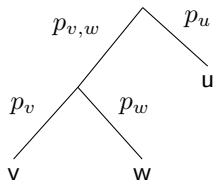


Figure 4.2: Case 2 with edge probabilities

Thus, the key is to determine whether two relatives probabilities are equal in different choice sets. In Benson et al. (2016), it is suggested to use the χ^2 test to determine whether the relative probabilities of choosing u over v differ significantly among a set of different choice sets. Let $\mathcal{S}_{uv}$ be the collection of all unique choice sets containing items u and v and the corresponding choice is either u or v, i.e. $\mathcal{S}_{uv} = \{S_i \mid c_i \in \{u, v\}, u, v \in S_i, i \in [m]\}$. Suppose $\mathcal{S}^*_{uv} \subseteq \mathcal{S}_{uv}$ is the collection of choice sets of our interest. In a χ^2 independence test (See, e.g. Agresti (2018)), the test statistic is

$$\sum_{S \in \mathcal{S}^*_{uv}} \frac{(X_{uv|S} - N_{uv|S}\hat{P}_{uv})^2}{N_{uv|S}\hat{P}_{uv}} + \frac{(X_{vu|S} - N_{vu|S}\hat{P}_{vu})^2}{N_{vu|S}\hat{P}_{vu}} \sim \chi^2_{|\mathcal{S}^*_{uv}|-1}, \tag{4.1}$$

Where $\hat{P}_{uv}$ is the estimated relative probability of choosing u over v, $N_{uv|S} = N_{vu|S}$, $X_{uv|S} = N_{uv|S} - X_{uv|S}$, and $\hat{P}_{vu} = 1 - \hat{P}_{uv}$. And $X_{uv|S}$ and $N_{uv|S}$ are defined as

$$X_{uv|S} = \sum_{i=1}^{m} \mathbf{1}\{c_i = u, S_i \in \mathcal{S}^*_{uv}\},$$
$$N_{uv|S} = \sum_{i=1}^{m} \mathbf{1}\{c_i \in \{u, v\}, S_i \in \mathcal{S}^*_{uv}\},$$

where $\mathbf{1}\{\cdot\}$ is the binary indicator function. Then, the estimated relative probability of choosing u over v is

$$\hat{P}_{uv} = \frac{\sum_{S \in \mathcal{S}^*_{uv}} X_{uv|S}}{\sum_{S \in \mathcal{S}^*_{uv}} N_{uv|S}},$$

Then, when $\mathcal{S}^*_{uv} = \{S_1, S_2\}$, the test result will tell us whether the relative probabilities of choosing u over v are the same in these two specific choice sets, and when $\mathcal{S}^*_{uv} = \mathcal{S}_{uv}$, we will know whether the

relative probabilities are the same among all relevant choice sets. If the left hand side in Eq. 4.1 is significantly large with respect to some significance level α, then the relative probabilities of choosing u over v vary among choice sets and hence u and v are not siblings.

In the next chapter, we first introduce the proxies of distance we designed for the proposed algorithm `DivisR`, which is based on divisive clustering. In the divisive clustering algorithm, the objects are divided into some number of clusters based on the distances between the objects, using some partition algorithm. Then, the algorithm recurses on each cluster to generate subclusters and so on, until every object is assigned to a leaf node of the tree. We use k-medoids as the partition algorithm. Besides, the value of k should also be carefully determined to optimize the result. Therefore, we will also briefly introduce the k-medoids algorithm and the criteria of finding optimal k. Then, we present our algorithm for learning nested tree structure from data and introduce how to use maximum likelihood estimation (MLE) to learn the deterministic utilities for each node. Finally, we perform a complexity analysis of the algorithm.

CHAPTER 5

Learning Tree Structure with the Divisive Clustering

5.1. Proxies for Distance

Inspired by the idea of χ^2 test, we can define 3 proxies for distance between each pair of alternatives, which can further support the divisive clustering algorithm we propose.

1. χ^2 distance: In our algorithm, the χ^2 test we just introduced is not necessarily performed. Instead, the test statistic itself can serve as a proxy of the distance between a pair of choice alternatives. Let $\mathrm{Chisq}(u, v)$ denote the left hand side of Eq. (4.1), i.e.

$$\mathrm{Chisq}(u, v) = \sum_{S \in \mathcal{S}_{uv}} \frac{(X_{uv,S} - \hat{P}_{uv} N_{uv,S})^2}{\hat{P}_{uv} N_{uv,S}} + \frac{(X_{vu,S} - \hat{P}_{vu} N_{uv,S})^2}{\hat{P}_{vu} N_{uv,S}},$$

   It can be immediately observed that if u and v are siblings, $\mathrm{Chisq}(u, v)$ should be small; otherwise, $\mathrm{Chisq}(u, v)$ is a large number, since the relative probabilities of choosing u over v in different choice sets will differ from each other. The more distant u and v are located, the more relative probabilities will be affected by the change of choice sets, and hence the more likely this value is to be large.

2. Sum of squared log probability ratio

$$\mathrm{SSLPR}(u, v) = \sum_{S \in \mathcal{S}_{uv}} \left(\log \hat{P}_{uv} - \log \hat{P}_{uv|S} \right)^2.$$

   This is a heuristic way of defining the distance between u and v. It also has the similar property, as discussed above.

3. Sum of absolute log probability ratio

$$\mathrm{SALPR}(u, v) = \sum_{S \in \mathcal{S}_{uv}} |\log \hat{P}_{uv} - \log \hat{P}_{uv|S}|.$$

   This is another heuristic way of defining the distance. The only difference is how to aggregate the differences between relative choice probabilities in different choice sets.

Based on one of these proxies for distance and the universe $\mathcal{U}$, we can calculate a distance matrix $D_{\mathcal{U}}$ whose entries d_{uv}, $u, v \in \mathcal{U}$ is calculated using one of the distance functions presented above. For a subset of alternatives $S \subseteq \mathcal{U}$, the distance matrix D_S is simply the submatrix of $D_{\mathcal{U}}$, obtained by only keeping the rows and columns corresponding to the alternatives in S. If u and v never appear in the same choice set in data $\mathcal{D}$, the distance between them is set to infinity, which means there is no evidence to assign u and v into the same cluster.

5.2. k-Medoids Clustering

k-medoids are methods in which the representatives of clusters are just original alternatives. It is proper to use k-medoids since it does not make sense to take the mean of the alternatives in clusters, unless we embed the alternatives into a continuous vector space. We briefly introduce the k-medoids clustering algorithm here (See Reynolds et al., 2004; Aggarwal, 2013 for more details).

We know that k ranges from 2 to $|S|$, where $|S|$ is the cardinality of S. S being partitioned into $|S|$ clusters means that each alternative in S belongs to its own cluster. Then, to partition a set of alternatives S into k clusters: (1) k alternatives are randomly selected as the medoids of the k clusters. (2) Each alternative is assigned to the cluster corresponding to the closest medoid. (3) Update the medoids of each cluster – the new medoid is the alternative which has the minimum average distance to other alternatives in the same cluster. (4) Repeat Step (2) and (3) until convergence - the medoids stop changing after update.

5.3. Criterion of Choosing the Best k

While partitioning the current set of alternatives into k new clusters, we do bookkeeping of the whole structure of the tree T_k (Also discussed in Section 5.4), learn the utilities with maximum likelihood estimation (MLE) on the training dataset $\mathcal{D}_{\text{train}}$, and calculate the average log-likelihood (ALLH) of tree T_k evaluated on the validation data $\mathcal{D}_{\text{valid}}$. Then, we select the k that maximizes the average log-likelihood of the recovered tree structure evaluated on the validation set. Formally,

$$\hat{\mathcal{V}}_k = \underset{\mathcal{V}}{\text{argmax}}\ \text{ALLH}(\mathcal{D}_{\text{train}}, T_k, \mathcal{V}),$$

$$k^* = \underset{k}{\text{argmax}}\ \text{ALLH}(\mathcal{D}_{\text{valid}}, T_k, \hat{\mathcal{V}}_k).$$

We choose k^* in a greedy manner, which means we do not change the value of k^* for higher levels once determined.

5.4. Algorithm Description

Let $\mathcal{D}$ denote the data set, which has m observations. Let $\mathcal{U}$ denote the universe of alternatives, S denote an arbitrary set of alternatives. Similar to the set representation of tree, we can express the result of k-medoids as follows: let $\mathcal{Y}^{(k)}(S) = \{S_1^{(k)}, S_2^{(k)}, ..., S_k^{(k)}\}$, where $S_i^{(k)} = \{S_{i1}^{(k)}, S_{i2}^{(k)}, ...\} \subseteq S$ is a set of internal nodes or leaves (alternatives), $i \in [k]$. Also, $\bigcup_{i=1}^{k} S_i^{(k)} = S$ and $S_i^{(k)} \cap S_j^{(k)} = \phi, \forall i, j \in [k], i \neq j$. Then, the updated tree structure after each k-medoids clustering, T_k, can be obtained by replacing S with $\mathcal{Y}^{(k)}(S)$. Here, we denote a leaf by a singleton to make the notation consistent.

Then, our algorithm for recovering nested tree structures from choice data, `Divisive Reconstruction`, or `DivisR` in short, is shown in Algorithm 1. Given $\mathcal{D}$ and $\mathcal{U}$, we first split the data into training and test set, denoted by $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ respectively. Then, we calculate the distance matrix $D_{\mathcal{U}}$ using only $\mathcal{D}_{\text{train}}$. With the data and the distance matrix ready, the nested tree $T =$`DivisR`$(\mathcal{D}_{\text{train}}, \mathcal{U})$. The algorithm first uses k-medoids clustering for each k ranging from 2 to $|S|$, where S is the input set of alternatives (hence $S = \mathcal{U}$ in the first call of the algorithm). Then, a best k^* is selected based on the average log-likelihood on validation set (or cross validation). And then repeat the procedure on each of the k^* subset until no further partition can be performed.

5.5. Estimation of Edge Probabilities

After the tree structure T is recovered, as discussed in Section 3.4, for a data point i, the selection of a choice alternative c_i from a choice set S_i can be regarded as a series of selections, level by level. Suppose c_i is located at depth d_i of T. At each level l, the selection of nest c_{il} is made among S_{il}, the set of nests at the level l in $T[S_i]$. Then, the probability of choosing c_i is $P_{c_i|S_i} = \prod_{l=1}^{d_i} P_{c_{il}|S_{il}}$. Hence the likelihood over all observations $\mathcal{D}$ and the learned tree structure T is

$$\mathcal{L}(\mathcal{D}, T, \mathcal{V}) = \prod_{i=1}^{m} P_{c_i|S_i} = \prod_{i=1}^{m} \prod_{l=1}^{d_i} P_{c_{il}|S_{il}} = \prod_{i=1}^{m} \prod_{l=1}^{d_i} \frac{e^{V_{c_{il}}}}{\sum_{c \in S_{il}} e^{V_c}},$$

Algorithm 1 `DivisR`$(\mathcal{D}, S, (T))$

Input: data $\mathcal{D}$, set of alternatives S, (optional: current tree structure T)
if T is **not** provided **then**
Initialize tree structure $T = S$, which means all alternatives belong to one single cluster.
Extract the distance matrix D_S from $D_{\mathcal{U}}$,
for $k = 2$ **to** $|S|$ **do**
Perform k-medoids clustering. Save the result to $\mathcal{Y}^{(k)}(S) = \{S_1^{(k)}, S_2^{(k)}, ..., S_k^{(k)}\}$,
Obtain the current tree structure T_k by replacing S with $\mathcal{Y}^{(k)}(S)$,
We can further split $\mathcal{D}$ into training data $\mathcal{D}_{\text{train}}$ and validation data $\mathcal{D}_{\text{valid}}$ (or use cross validations),
Estimate the utilities for each node $\hat{\mathcal{V}}_k = \underset{\mathcal{V}}{\text{argmax}}\ \text{ALLH}(\mathcal{D}_{\text{train}}, T_k, \mathcal{V})$,
end for
Select the optimal number of clusters using a validation set.
$k^* = \underset{k \in 2,3,...,|S|}{\text{argmax}}\ \text{ALLH}(\mathcal{D}_{\text{valid}}, T_k, \hat{\mathcal{V}}_k)$,
Create an empty tree $T_r = \{\}$ to save the returned subtrees.
for $l = 1$ **to** k^* **do**
if $|S_l^{(k^*)}| > 2$ **then** recurse the algorithm on each cluster of alternatives
The returned results are siblings of each other.
$T_r = T_r \cup \texttt{DivisR}(\mathcal{D}, S_l^{(k^*)}, T_{k^*})$
else if $|S_l^{(k^*)}| = 2$ **then** there is only one way to assign these alternatives to a tree
return $\{\{S_{l1}^{(k^*)}\}, \{S_{l2}^{(k^*)}\}\}$
else we return the alternative itself in the base case
return $S_l^{(k^*)}$
end if
end for
return T_r

To compare its value among datasets with different numbers of observations, we take its average. The average log-likelihood (ALLH) of tree T evaluated on data $\mathcal{D}$ is defined as

$$\text{ALLH}(\mathcal{D}, T, \mathcal{V}) = \log\left(\mathcal{L}(\mathcal{D}, T, \mathcal{V})\right)^{\frac{1}{m}} = \frac{1}{m}\sum_{i=1}^{m}\sum_{l=1}^{d_i}\left(V_{c_{il}} - \log\sum_{c \in S_{il}} e^{V_c}\right).$$

ALLH is convex over $\mathbb{R}^N$ since it is the sum of a linear term and a log-sum-exponential term (Benson et al., 2016). When learning the utilities V_x for a node x, we simply use gradient descent with an adaptive step size chosen by backtracking line search (Boyd and Vandenberghe, 2004), which is simple and tends to work pretty well in practice. It is easy to find that

$$\frac{\partial \text{ALLH}}{\partial V_x} = \frac{1}{m}\sum_{i=1}^{m}\sum_{l=1}^{d_i}\left(\mathbf{1}\{c_{il} = x\} - \mathbf{1}\{x \in S_{il}\} \cdot \frac{e^{V_x}}{\sum_{c \in S_{il}} e^{V_c}}\right), \tag{5.1}$$

For the step size η, given two hyperparameters $\alpha \in (0, 0.5)$ and $\beta \in (0, 1)$,

$$\begin{aligned} \texttt{while} \quad & \mathsf{ALLH}(\mathcal{D}, T, \mathcal{V} + \eta\nabla\mathcal{V}) > \mathsf{ALLH}(\mathcal{D}, T, \mathcal{V}) + \alpha\eta\nabla\mathsf{ALLH}(\mathcal{D}, T, \mathcal{V})^T\nabla\mathcal{V}, \\ \texttt{update} \quad & \eta := \beta\eta, \end{aligned}$$

The idea is that we choose the step size that reduce ALLH enough, compared to the linear extrapolation as shown in the right hand side of the `while` condition. We used $\eta = 1$, $\alpha = 0.2$ and $\beta = 0.8$ in the implementation.

After the best utilities $\hat{\mathcal{V}}$ are estimated using T and $\mathcal{D}_{\text{train}}$, we evaluate them on the test data $\mathcal{D}_{\text{test}}$.

5.6. Complexity Analysis

For an input data set with m observations and choice set with n alternatives, `DivisR` consists of the calculation of distance matrix, $|n| - 1$ runs of k-medoids clustering, and $|n| - 1$ MLE evaluations. In addition, it is a recursive algorithm. Let t_d be the time required to calculate the distance between 2 alternatives, t_k be the time required until the convergence of k-medoids clustering, and t_{MLE} be the time consumption of MLE until the convergence of the MLE.

Then, We need $O(n^2 t_d)$ to calculate the distance matrix, since there are n alternatives in total. It is only calculated once, however, because we directly use the submatrix (as mentioned in Section 5.1). For the k-medoids, we need t_k. For the MLE evaluation, we need t_{MLE}. Let $T(n)$ be the total time require for `DivisR` and $t(n)$ be the time required for the recursive component. Then, let n_1, n_2, ..., n_{k^*} be the sizes of k^* clusters produced by k-medoids, where k^* is the optimal number of clusters and $n_1 + n_2 + \cdots + n_{k^*} = n$. Note that when $n \leq 2$, there is only 1 possible clustering. Then, $T(n) = t(n) + O(n^2 t_d)$, and

$$t(n) = \begin{cases} t(n_1) + t(n_2) + \cdots t(n_{k^*}) + O(\sum_{k=2}^{n} t_k + t_{MLE}), & n \geq 3 \\ 0, \quad o.w. \end{cases} \tag{5.2}$$

where t_k is of order of $O(k)$, since all operations involved in it are only summations of the distances and taking minimums for each one of the k clusters. Then, $\sum_{k=2}^{n} t_k$ has an order of $O(n^2)$. As for t_{MLE}, it has an order of $O(mn^2/\varepsilon)$, where ε is the convergence condition of the MLE. This is indicated

by Eq.(5.1) - the time grows with the number of edge weights that needed to update, the number of observations and the depth of the tree, and the gradient descent with backtracking line search has at least a linear rate of convergence (Boyd and Vandenberghe, 2004). Then, combining the fact that the k-medoids generally takes less time than the MLE does in our experiments. Eq.(5.2) can be simplified as

$$t(n) = \begin{cases} t(n_1) + t(n_2) + \cdots t(n_{k^*}) + O(\frac{mn^2}{\varepsilon}), & n \geq 3 \\ 0, & o.w. \end{cases} \tag{5.3}$$

This is no better than $O(mn^2/\varepsilon)$, which corresponds to the case where the alternatives are divided into $k*$ even clusters. In the worst case, the input for one of the subproblem decrease linearly and the time required is the summation of a series of quadratic terms, which is of order of $O(mn^3/\varepsilon)$. It will struggle when there are hundreds and thousands of alternatives in the universe, but usually the algorithm is efficient since typically n is not large.

In the next chapter, we introduce how we design the experiments and how to measure the performance of the algorithms, followed by the results and discussions.

CHAPTER 6

Experiments

We evaluate the MNL model (the flat tree), Benson et al. (2016)'s algorithm `GreedyReconstructTree` or `greedyR` in short and our algorithm `DivisR` with 3 proposed distance metrics (`Chisq`, `SSLPR` and `SALPR`) on 6 synthetic data sets and 3 real world data sets. For `greedyR`, there is an extra step that finds the best significant level α using 5-fold cross validation for small data sets, and using a 20% validation set for large data sets (LastFM Artist).

A total of 30 experiments were run for each algorithms, except for the LastFM Artist data (5 runs, due to its huge size of universe of alternatives). For synthetic data, in each experiment, we run each algorithm on 1,000, 10,000, 50,000 and 100,000 training examples and evaluate the recovered trees on 10,000 test examples, all generated from the same true tree with specified edge probabilities (utilities). Hence in total the algorithm is executed 120 times for each dataset. The true tree structures are described in Section 6.2. For real-world data, in each experiment, we first split the data into training and test set randomly, with 80% and 20% examples, respectively. Then, we run each algorithm on 1%, 10%, 50% and 100% of the training examples and evaluate the recovered trees on the test set. We control the randomness by setting random seeds when generating synthetic data or splitting training/test sets for real-world data. Thus, all algorithms run on the same data, though the data used in each experiment are different.

We compare these results by log-likelihoods on test data and the average pairwise editing distances between the recovered trees, which reflect the stability of the algorithm (shown in Appendix A). For synthetic data sets, we also compute the average editing distances between the recovered trees and true trees, which reflect the quality of the recovered trees. For some real-world data sets like SFWork, and Meath Election, we also manually specified some tree structures which were learned from previous research and proved to be the best.

In the rest of the section, we first introduce the editing distance. Then, we describe the synthetic data sets and real-world data sets used for experiment. Finally, we present the results.

Table 6.1: Summary of synthetic data sets

Dataset name	# of items	Height	max # of training data	# of test data
Five-item tree	5	2	100,000	10,000
Nine-item tree	9	3	100,000	10,000
Wide tree	17	3	100,000	10,000
Imbalanced tree	9	4	100,000	10,000
Small deep tree	5	4	100,000	10,000
Large deep tree	9	8	100,000	10,000

6.1. Editing Distance between Trees

We now introduce a metric of distance between trees, the editing distance (Zhang and Shasha, 1989). Three types of editing operations are defined:

- **Change**. To change one node label to another.

- **Delete**. To delete a node. (All children of the deleted node b become children of the parent a.)

- **Insert**. To insert a node. (A subset of siblings among the children of a become the children of b)

Assume each editing operation incurs a unit cost. The editing distance between two trees is defined as the minimum total cost of converting one to another by a sequence of editing operations. For example, the two trees shown in Fig.6.1 have an editing distance of 4 (delete C, change E to F, insert G as the sibling of D and E, insert a node as the parent of F and G).

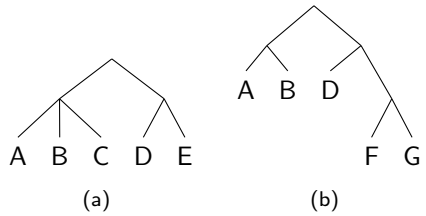


Figure 6.1: An example of calculating editing distance

We use Tim Henderson's implementation in the experiments (Henderson, 2018).

6.2. Synthetic Datasets

Experiments were evaluated on 6 different synthetic data sets. A summary of these synthetic data sets is shown in Table 6.1.

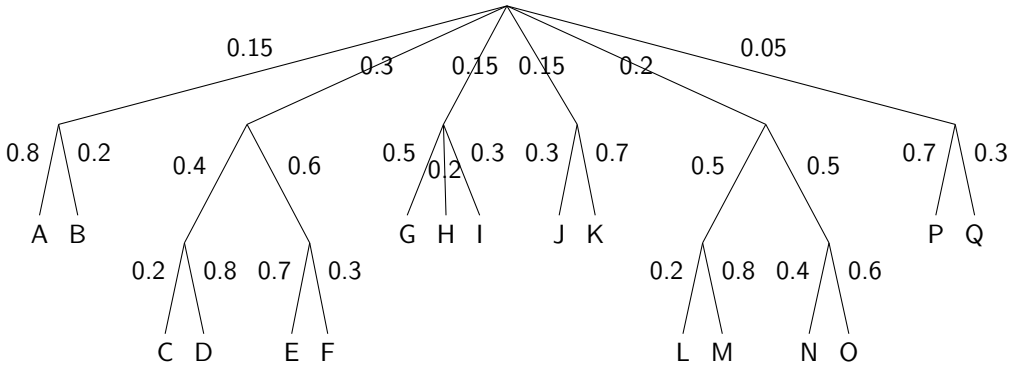


Figure 6.2: Wide tree

Generating Synthetic Data. For synthetic data, the whole tree structure and all edge probabilities are known. These data are generated one by one. For each data point, first, a choice set size s is chosen uniformly at random from the range $[2, L]$. Then, s items are uniformly randomly selected from the universe. According to the true tree, the edge probabilities of the selected items are calculated as shown in Section 3.4. Finally, a selection was made randomly based on the overall probabilities: For example, after calculation, if the choice set $S = \{A, B, C\}$ with overall probabilities $\{0.2, 0.3, 0.5\}$, then the choice has a probabilities of 0.2 to be A, 0.3 to be B and 0.5 to be C. Repeats m times to generate a dataset containing m data points. For 5-item tree and the small deep tree, $L = 5$. For others, $L = 4$.

We generated diverse trees, ranging from very simple trees to complicated trees, from wide trees to deep trees, in order to find out the strengths and weaknesses of each algorithm involved in this thesis.

- **Five-item tree.** See Fig. 3.2(a).
- **Nine-item tree.** This tree is used in Benson et al. (2016). We use the same tree structure and edge probabilities See Fig. 2 in Benson et al. (2016)).
- **Wide tree.** See Fig. 6.2.
- **Imbalanced tree.** See Fig. 6.3.
- **Small-deep tree.** See Fig. 6.4.
- **Large deep tree.** See in Appendix. The large deep tree has a structure similar to that of the small deep tree. It just has more levels and 9 items in total.

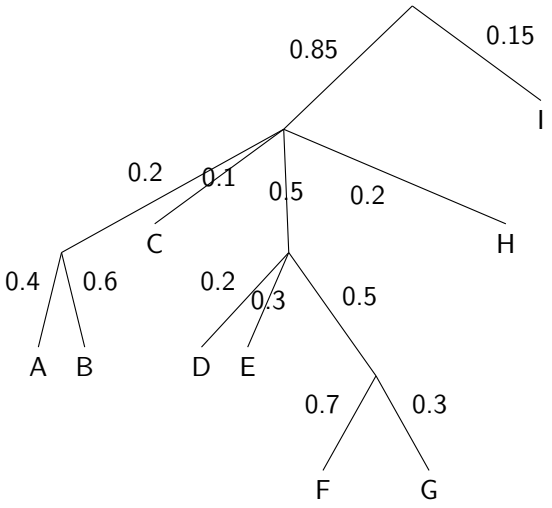


Figure 6.3: Imbalanced tree

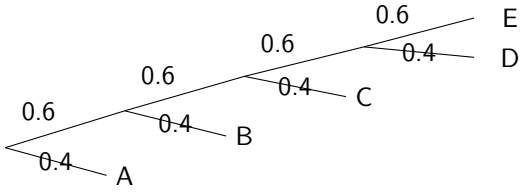


Figure 6.4: Small deep tree

6.3. Real-world Datasets

Experiments were evaluated on 3 different real-world data sets. A summary of these synthetic data sets is shown in Table 6.2.

Table 6.2: Summary of real-world data sets

Dataset name	# of items	# of training data	# of test data
SFWork	6	4029	1000
Meath Election	14	2471	617
LastFMArtist	270	2533	633

- **SFWork.** This dataset contains the transportation preferences of people in San Francisco on commuting to work (Koppelman and Bhat, 2006). Each observation is comprised of a subset of six possible transportation options - walking, biking, driving alone, taking public transit, ride sharing (2 people) and ride sharing (3+ people) - and commuter's corresponding choice. We also test the

best nested tree proposed in the research of Koppelman and Bhat (2006), as shown in Fig. 6.5.

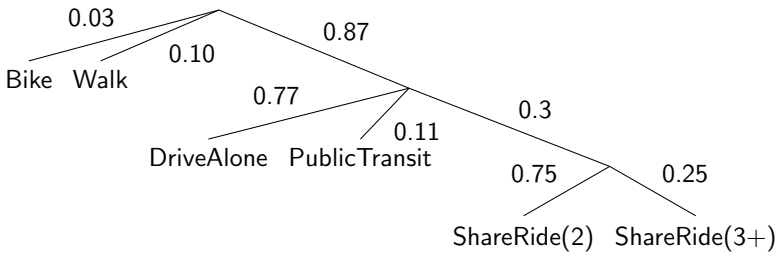


Figure 6.5: Nested tree by Koppelman and Bhat (2006)

- **Meath Election.** The Meath Election data sets include a complete record of votes for two separate elections held in Dublin, Ireland in 2002 (Mattei and Walsh, 2013). The data set we used contains 64,081 votes over 14 candidates. Many of them are partial votes over the candidate set. We collect choice data from the 3166 data points with complete ranking of 14 candidates. First, a random subset of candidates with size between 2 and 14 is selected for each observation. Then, the choice is the candidate ranked first in the set of remaining candidates. Besides the learned nested trees, we also tested the tree learned in Gormley and Murphy (2006) (the `Empirical 2` tree) and 2 empirical trees. The `Empirical 1` tree is generated by grouping candidates from 2 major parties into 2 nests, respectively, and grouping the rest into the third nest, while the `Empirical 3` tree is generated by grouping candidates from 2 major parties into 2 nests, respectively, and each of the rest candidate forming its own nest. The tree plot can be found in Appendix B. In addition, to further explore if there are any better experiment setting and the performance of the uniform 2-MNL on this data set, we also performed other experiments, which are shown in Appendix C.

- **LastFM Artist.** This dataset is collected from Last.fm Celma Herrada (2009), comprised of the listening records of about 1,000 users. Based on the chronological sequence of the songs a user listen to, we extract a data point in the same way as Benson et al. (2016):

  1. If the user has played 3 songs in a row from 3 different artists
  2. The next song (the 4th song) is one of these 3 songs

  Thus, all choice sets are of size 3, from which a user chooses an artist (not a song). We also eliminate the position bias of the artists by adding suffixes "_1", "_2" or "_3" to the end of an

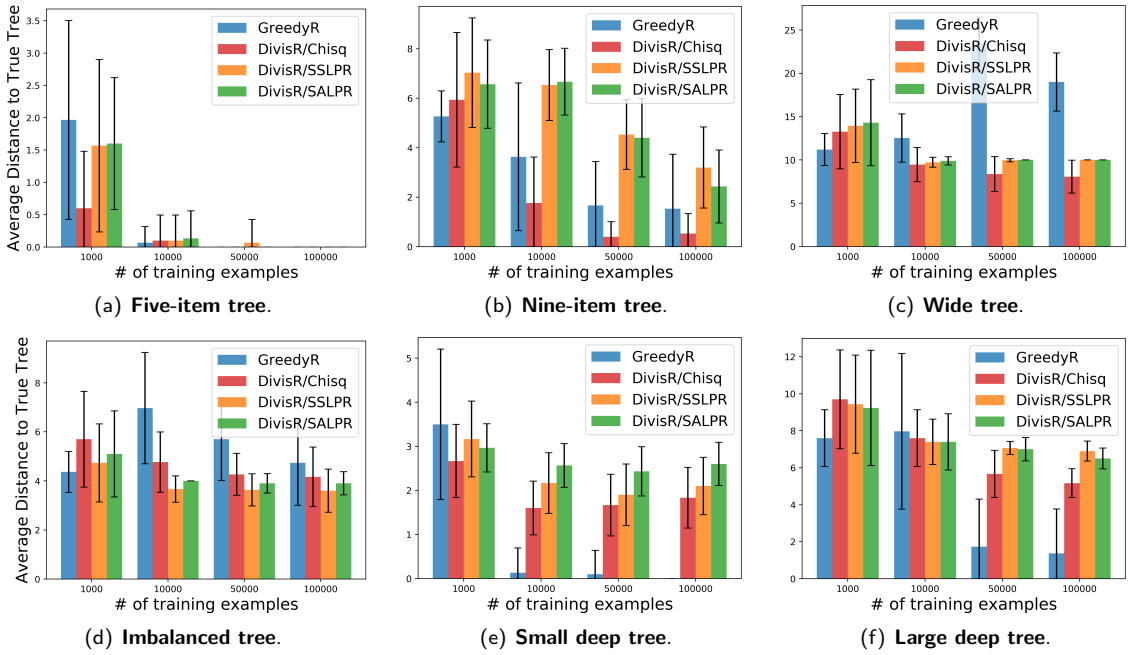


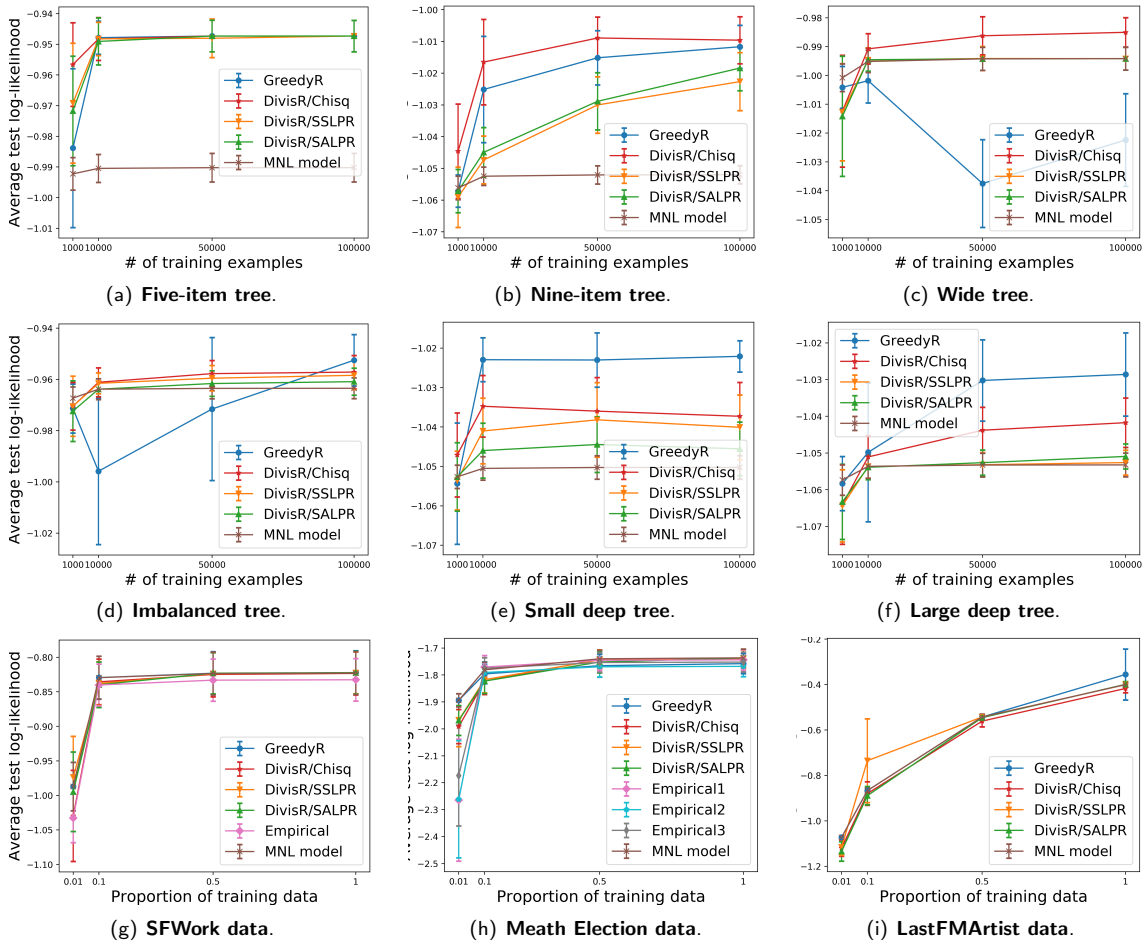
Figure 6.6: Bar plots with error bars (standard errors) for the average distance between the recovered trees and the true tree for synthetic data sets. The colored bars represent the average of the distances between the recovered trees and the true tree and the error bars show the standard deviation of the distances.

artist, indicating its position in the choice set. We keep choice sets that appear at least 10 times and items that appear in at least 10 choice sets.

6.4. Results

The tables for the plots below, the plots of average pairwise distance between recovered trees in different runs, and some recovered trees structures are shown in Appendix A. We compare the performance of the algorithms through the following plots: The average log-likelihoods of the recovered trees by each algorithm for each dataset, averaged over 30 experiments, are shown in Fig. 6.7; For synthetic data sets, the bar plots for the average distance between the recovered trees and the true tree for each algorithm is shown in Fig. 6.6.

For synthetic data sets, we observe that our algorithms outperform the algorithm `GreedyR` proposed by Benson et al. (2016) on Nine-item tree, Wide tree, and Imbalanced tree (only worse than `GreedyR` when using 100,000 training data), but worse than it on deep trees. On the simple Five-item tree, both algorithms can recover the trees correctly. These results indicate that our algorithm `DivisR` are good



(a) **Five-item tree**. (b) **Nine-item tree**. (c) **Wide tree**.

(d) **Imbalanced tree**. (e) **Small deep tree**. (f) **Large deep tree**.

(g) **SFWork data**. (h) **Meath Election data**. (i) **LastFMArtist data**.

Figure 6.7: Average log-likelihood of the recovered tree by each algorithm for each dataset. The error bars show the standard deviations of the log-likelihoods.

at recovering wide trees (The Nine-item tree can be also considered wide) stably (from Fig. 6.6), while `GreedyR` has better performance on deep trees. The reason might be that our algorithm is a top-down algorithm, which splits the alternatives into certain numbers clusters greedily. Hence for deep trees, it may suffer more from making a wrong decision at top several levels of the tree.

For real-world data sets, it seems that both `DivisR` and `GreedyR` produce models that have no advantage over the MNL model. The average log-likelihood for some of the trees is often only slightly higher than that of a flat tree. There can be several reasons for this phenomenon. For example, the reason that the best tree proposed in Koppelman and Bhat (2006) is no longer the best is: 1) When they learn the tree, they evaluated their tree on the same training data. Hence the tree may not generalize well. 2) They incorporated some attributes of the alternatives, such as travel time and cost in their

model, while we only assign a single constant to these choices as utilities. For the LastFM Artist data, we might need more data points to be able to recover a proper tree structure since there are a large amount of alternatives compared to its small size. And although there is research that proposed models that improved the performance on predicting choices on Meath Election, they exploit the whole ranking, while we only regard the candidate ranked first as the choice.

To summarize, `DivisR` has better performance than `GreedyR` when the underlying tree is wide, especially with insufficient training samples. The reasons might be that `GreedyR` require an accurate result from each query, which involves sensitive statistical tests, otherwise it cannot determine which two alternatives are siblings. A tiny error of estimation of the choice probabilities may make the hypotheses wrongly rejected, or not rejected, and that is also why the choice of significance level is important in `GreedyR`. While the estimation of choice probabilities becomes accurate with sufficient training data, the performance of `GreedyR` is improved accordingly. As for our algorithm, `DivisR`, it determines whether two alternatives are siblings from proxies of distance, which are sort of cumulations of the difference of relative probabilities in various choice sets. And in k-medoids clustering, the absolute values of distance do not matter that much. Instead, the relative magnitude mainly determines how we cluster the alternatives. Therefore, there is more tolerance to the errors in the estimation of choice probabilities in `DivisR`.

CHAPTER 7

Conclusion and Future Work

In this thesis, we study the problem of learning the underlying tree structures for *nested logit* models from data. We propose an algorithm `DivisR` based on divisive clustering that can recover a tree structure for a nested logit model from choice data. In the experiments, we evaluate the MNL models and the nested logit models with underlying tree structures recovered by `DivisR` and the algorithm proposed in Benson et al. (2016), `GreedyR`, on both synthetic and real-world data sets, and compare the out-of-sample likelihood. It turns out that `DivisR` outperforms `GreedyR` when the underlying tree is wide, and especially when there are insufficient training data. Our algorithms, however, do not work well when facing deep trees. As for the performance on real world choice data, the nested logit model learned by both algorithms are only marginally better or even worse than the MNL model sometimes.

Trying to improve the experiment design for both algorithms on the Meath Election data set, in Appendix B, we modified the data generation process and tested more trees proposed both empirically and by other work. And in Appendix C, we explore if other models other than nest logit models, specifically the *uniform 2-MNL models* (Chierichetti et al., 2018), better capture the choices in the Meath Election data set as compared to MNL models. Unfortunately, neither attempts produce a substantial improvement, but we believe that the study on this kind of algorithms for learning the tree structures automatically from data is useful, as nowadays nested logit models are still widely used and serve as a decent baseline of predicting users' choice behavior.

To further improve the performance of the algorithm, we may incorporate the attributes of the alternatives and the users when learning the tree structure, since it is not always true that the users are homogeneous. We could also relieve the restriction that there is no correlation among the utilities of alternatives in a nest, and hence there are more parameters to be identified in the model. From some informal trials, a general nested logit model did have better performance than the nested logit models we postulated. Also, it would be also worth it to investigate non-uniform mixture of MNL models, since they may be able to well approximate any random utility models (McFadden and Train, 2000). Any of these directions can lead to an interesting research project.

APPENDIX A

Tables and Plots not Shown in Chapter 6

A.1. Tables for Average Log-likelihood

The average test log-likelihoods for all synthetic datasets are shown in Table A.1, and the average test log-likelihoods for all real-world datasets are shown in Table A.2. From the line plots, we can observe that for synthetic data, there is a significant improvement of nested logit models compared to MNL models. However, the improvement is trivial in real-world data sets, and sometimes there is no improvement at all. This fact suggests that in real-world data sets, what we posit for the nested logit models is not as proper as it is in synthetic data sets.

A.2. Frequently Recovered Trees for SFWork

Frequently recovered trees for SFWork data set when using all training data are shown in Fig. A.1. We can observe that although the nested tree proposed by Koppelman and Bhat (2006) was the best in the original paper, it is not in our setting because we do not incorporate the attributes of the alternatives. As for the nested trees recovered by `DivisR` and `GreedyR`, although the structures are quite different, they still agree on some of the choice alternatives. For example, in both trees, ShareRide(3+) is a child of the root, with learned edge probability 0.03, and PublicTransit and Walk are siblings with nearly the same edge probabilities. However, both trees fail to outperform a simple flat tree assumed by an MNL model. One possible reason is that the number of data points is not enough for a model with more parameters. It could also be due to the heterogeneity of users. If we could add more users' attributes into the model, like what Koppelman and Bhat (2006) did in their original work, the learned tree structures might have better performance.

A.3. Overlapping Bar Plots for the Average Distance between the Recovered Trees and the True Tree and the Average Pairwise Distance between the Recovered Trees in Different Runs

See Fig. A.2. These plots are about accuracy and precision of the algorithms. We can observe that for a simple tree like the Five-item tree, all algorithms produce precise and accurate results. For a complex

Table A.1: Average test log-likelihood for all synthetic data sets

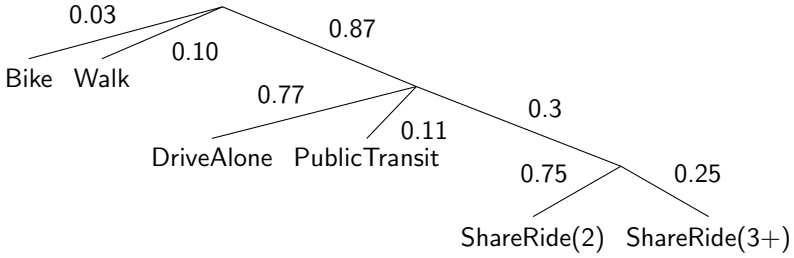
Algorithm	# of Training Examples			
	1,000	10,000	50,000	100,000
Five-item tree				
MNL model	-0.9923	-0.9905	-0.9902	-0.9902
GreedyR	-0.9839	**-0.9478**	**-0.9473**	**-0.9473**
DivisR/Chisq	**-0.9566**	-0.9483	**-0.9473**	**-0.9473**
DivisR/SALPR	-0.9717	-0.9491	**-0.9473**	**-0.9473**
DivisR/SSLPR	-0.9692	-0.9482	-0.948	**-0.9473**
Nine-item tree				
MNL model	-1.0562	-1.0525	-1.0521	-1.0520
GreedyR	-1.0571	-1.0251	-1.0151	-1.0117
DivisR/Chisq	**-1.0446**	**-1.0165**	**-1.0089**	**-1.0096**
DivisR/SALPR	-1.0572	-1.0449	-1.0289	-1.0184
DivisR/SSLPR	-1.0591	-1.0473	-1.0300	-1.0227
Wide tree				
MNL model	**-1.0008**	-0.9952	-0.9943	-0.9942
GreedyR	-1.0042	-1.0018	-1.0375	-1.0224
DivisR/Chisq	-1.0124	**-0.9908**	**-0.9862**	**-0.9850**
DivisR/SALPR	-1.0142	-0.9946	-0.9943	-0.9942
DivisR/SSLPR	-1.0128	-0.9946	-0.9941	-0.9942
Imbalanced tree				
MNL model	**-0.9673**	-0.9638	-0.9635	-0.9635
GreedyR	-0.9714	-0.9958	-0.9716	**-0.9525**
DivisR/Chisq	-0.9705	**-0.9611**	**-0.9578**	-0.9572
DivisR/SALPR	-0.9724	-0.9638	-0.9616	-0.9609
DivisR/SSLPR	-0.9705	-0.9615	-0.9596	-0.9584
Small deep tree				
MNL model	-1.0526	-1.0505	-1.0503	-1.0503
GreedyR	-1.0544	**-1.0229**	**-1.0230**	**-1.0221**
DivisR/Chisq	**-1.0471**	-1.0347	-1.0360	-1.0373
DivisR/SALPR	-1.0527	-1.0460	-1.0444	-1.0455
DivisR/SSLPR	-1.0536	-1.0410	-1.0382	-1.0401
Large deep tree				
MNL model	**-1.0573**	-1.0536	-1.0533	-1.0532
GreedyR	-1.0583	**-1.0498**	**-1.0302**	**-1.0286**
DivisR/Chisq	-1.0640	-1.0511	-1.0438	-1.0417
DivisR/SALPR	-1.0633	-1.0539	-1.0526	-1.0509
DivisR/SSLPR	-1.0644	-1.0537	-1.0532	-1.0526

tree, like Wide tree, however, `DivisR` with distance `SSLPR` or `SALPR` produces the same wrong tree consistently, which indicate that these two proxies for distance may not be as powerful as `Chisq`. We can also see that despite the fact that `GreedyR` has better performance than `DivisR` does on deep

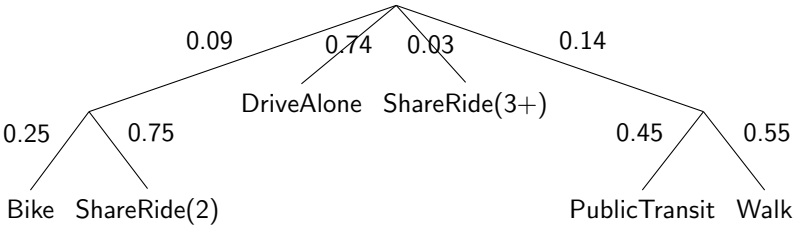
Table A.2: Average test log-likelihood for all real-world data sets

Algorithm	Proportion of Training Examples			
	0.01	0.1	0.5	1
SFWork data				
MNL model	-0.9872	-0.8295	-0.8235	-0.8229
GreedyR	-0.9872	**-0.8294**	-0.8231	**-0.8222**
DivisR/Chisq	-1.0296	-0.8356	-0.8249	-0.8234
DivisR/SALPR	-0.9948	-0.8398	**-0.8230**	-0.8223
DivisR/SSLPR	**-0.9736**	-0.8379	-0.8238	-0.8225
Emprical tree	-1.0328	-0.8401	-0.8332	-0.8326
Meath Election data				
MNL model	**-1.8940**	-1.7809	**-1.7400**	**-1.7365**
GreedyR	**-1.8940**	-1.7961	-1.7655	-1.7579
DivisR/Chisq	-1.9914	-1.8231	-1.7416	-1.7399
DivisR/SALPR	-1.9687	-1.8227	-1.7518	-1.7369
DivisR/SSLPR	-1.9675	-1.8172	-1.7443	-1.7390
Empirical1	-2.2644	**-1.7706**	-1.7455	-1.7438
Empirical2	-2.2612	-1.7909	-1.7696	-1.7681
Empirical3	-2.1732	-1.7749	-1.7537	-1.7521
LastFMArtist data				
MNL model	**-1.0729**	-0.8691	**-0.5411**	-0.3984
GreedyR	**-1.0729**	-0.8691	**-0.5411**	-0.4053
DivisR/Chisq	-1.1120	-0.8828	-0.5555	-0.4186
DivisR/SALPR	-1.1449	-0.9011	-0.5440	**-0.3963**
DivisR/SSLPR	-1.1051	**-0.7000**	**-0.5411**	-0.3984

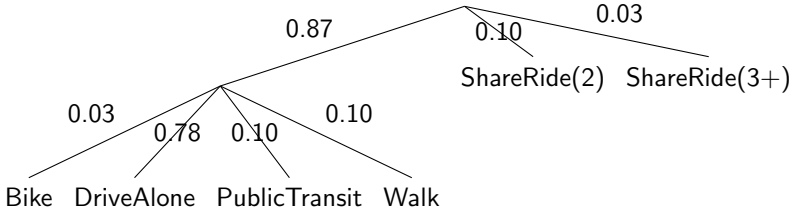
trees, it is less stable than `DivisR` with `Chisq` distance proxy when facing other tree structures, and produces less correct results.



(a) Nested tree by Koppelman, F. S., & Bhat, C. (2006) - the best of 15 tested trees

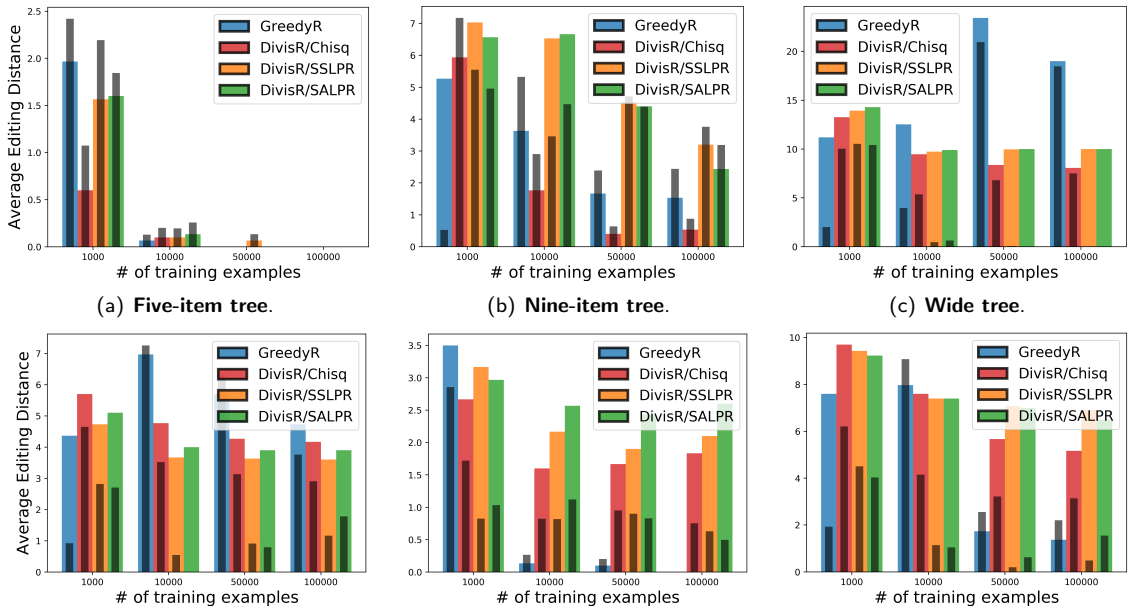


(b) Nested tree by Benson, et al. (2016) - the most frequently recovered tree



(c) Nested tree by DivisR/SALPR - the second most frequently recovered tree (the most frequent one is a flat tree)

Figure A.1: Recovered trees for SFWork dataset.



(a) **Five-item tree**. (b) **Nine-item tree**. (c) **Wide tree**.

(d) **Imbalanced tree**. (e) **Small deep tree**. (f) **Large deep tree**.

Figure A.2: Overlapping bar plots for the average distance to the true tree and the average pairwise distance between the recovered trees. The colored bars show the average distance between the recovered trees and the true tree, while the dark thinner bars inside represent the average of distances between all pairs of recovered trees.

APPENDIX B

Further Exploration on Meath Election Data Set

Here we present some improvement on the experiment design for `DivisR` and `GreedyR` on the **Meath Election** data set.

B.1. Modification to the Data Generation

On the one hand, since in the original set of experiments we performed, the choice sets are generated randomly from all 14 candidates, for a certain choice set, the number of times it appears in the generated data can be so small that our estimation on the choice probabilities can be inaccurate. Hence, the new data set is generated in the following manner:

1. Generate N choice sets of size 7, in which the 7 candidates are selected uniformly at random.
2. Check whether any two candidates are connected in these N choice sets if they are put in a graph (Typically yes if $N \geq 20$).
3. Since each voter ranks top-k ($1 \leq k \leq 14$) candidates in the raw data. For each ranking, from all N choice sets we generated, we randomly selected one choice set that contains at least one of these top-k candidates and make candidate with the highest ranking the choice.
4. The generated data have 64,081 observations.
5. $N = 20$ and $N = 40$ are used for the experiments.
6. Cap the log-likelihood at -5.298 ($= \log 0.005$), which means the smallest possible predicted probability is 0.005. The idea is that since predicting for a choice an overly small probability can incur a significant cost on the model, during the MLE process, the learner may focus too much on improve these tiny probability values and ignore to improve the prediction of other choices. Hence capping the prediction may help to improve the overall performance of any model learned with MLE.

B.2. Empirical Tree Structures

Political Affiliation of Candidates. In Table B.1, the political affiliation of each candidate is given as well as an abbreviation of their surname and party. Independent candidates are not affiliated to any party. Two of the empirical trees are generated based on this table. Hence, we have the following 3 empirical

Table B.1: Political affiliation of 14 candidates (Gormley and Murphy, 2006)

Candidate	**Party**	**Elected**
Bardy, J. (By)	Fianna F'ail (FF)	✓
Bruton, J. (Bt)	Fine Gael (FG)	✓
Colwell, J. (Cl)	Independent (Ind)	
Dempsey, N. (Dp)	Fianna F'ail (FF)	✓
English, D. (Eg)	Fine Gael (FG)	✓
Farrelly, J. (Fr)	Fine Gael (FG)	
Fitzgerald, B. (Fz)	Independent (Ind)	
Kelly, T. (Kl)	Independent (Ind)	
O'Brien, P. (Ob)	Independent (Ind)	
O'Byrne, F. (Oby)	Green Party (GP)	
Redmond, M. (Rd)	Christian Solidarity (CSP)	
Reilly, J. (Rl)	Sinn F'ein (SF)	
Wallace, M. (Wl)	Fianna F'ail (FF)	✓
Ward, P. (Wd)	Labour (Lab)	

tree structures, shown in Fig. B.1 - B.3:

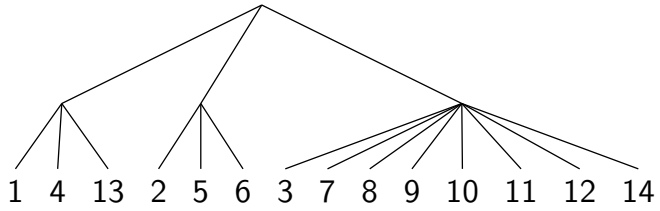


Figure B.1: Empirical tree 1

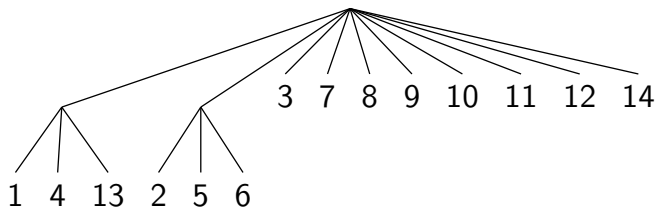


Figure B.3: Empirical tree 3

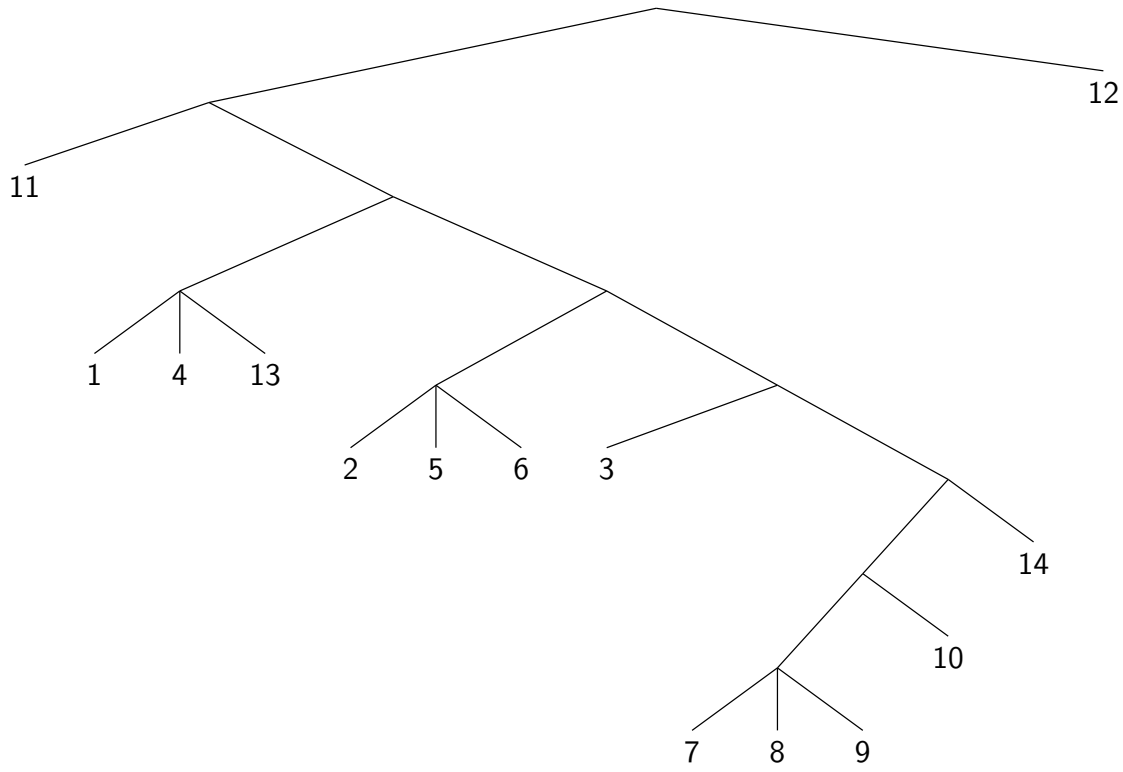


Figure B.2: Empirical tree 2 (Huang and Guestrin, 2012)

B.3. Results

The results are shown in Section C.4 in Appendix C.

APPENDIX C

Uniform 2-MNL models

C.1. Introduction

The uniform 2-MNL model is proposed by Chierichetti et al. (2018), denoted by $\mathcal{A}(a, b, 1/2)$, where a and b are 2 weight functions, each mapping an alternative i in the universe $\mathcal{U}$ to a positive real value. Let $n = |\mathcal{U}|$, and let the n alternatives in $\mathcal{U}$ be $1, 2, \cdots, n$. Then, the weight functions are $a, b : [n] \to \mathbb{R}^+$. Then, given a choice set S, the probability that alternative $i \in S$ is chosen is

$$P_S^{\mathcal{A}}(i) = \frac{1}{2} \cdot \frac{a_i}{\sum_{j \in S} a_j} + \frac{1}{2} \cdot \frac{b_i}{\sum_{j \in S} b_j},$$

where $a_i = a(i)$, $\sum_{j \in \mathcal{U}} a_j = 1$ and $\sum_{j \in \mathcal{U}} b_j = 1$. The paper proved that there is a linear time algorithm in terms of the size of $\mathcal{U}$, able to infer a unique $\mathcal{A}$ (up to switching a and b) using an **oracle** that returns the true choice probabilities for every alternative in every choice set of size 2 and 3.

Without experiments, it is unclear whether a uniform 2-MNL model is better than an MNL model or a nested logit model. On the one hand, as the uniform 2-MNL model is a more general version of MNL (when the 2 weight functions $a = b$, it is reduced to MNL), we should expect that its performance is non-worse than MNL models. On the other hand, the original algorithm is based on oracles revealing exact choice probabilities, which is not true for real-world data sets - we can only estimate the probabilities. Hence the learned uniform 2-MNL model may not work well. The assumption that users' selection in these data sets can be modeled by uniform 2-MNL models may be also inaccurate. Hence, we reproduced the algorithm for learning uniform 2-MNL models for further exploration.

C.2. Experiment on Synthetic Data

To check the reproduction of the algorithm, first, we performed some experiments on synthetic datasets exclusively generated for uniform 2-MNL models in the following manner:

1. Find 2 weight functions a and b arbitrarily.

2. Calculate the choice probabilities for every item in every choice set of size 2 and 3

3. Using these probabilities as the returned values of the oracle, estimate 2 weight functions $\hat{a}$ and $\hat{b}$.

4. Given a precision $\varepsilon > 0$, if $\forall i \in \mathcal{U}$, $|a_i - \hat{a}_i| < \varepsilon$ and $|b_i - \hat{b}_i| < \varepsilon$, or $|a_i - \hat{b}_i| < \varepsilon$ and $|b_i - \hat{a}_i| < \varepsilon$, we say the algorithm successfully recover the uniform 2-MNL.

Results. For a universe of size 20, when $\varepsilon = 1 \times 10^{-4}$, the algorithm succeeds more than half of the time; when $\varepsilon = 1 \times 10^{-2}$, the algorithm succeeds all the time.

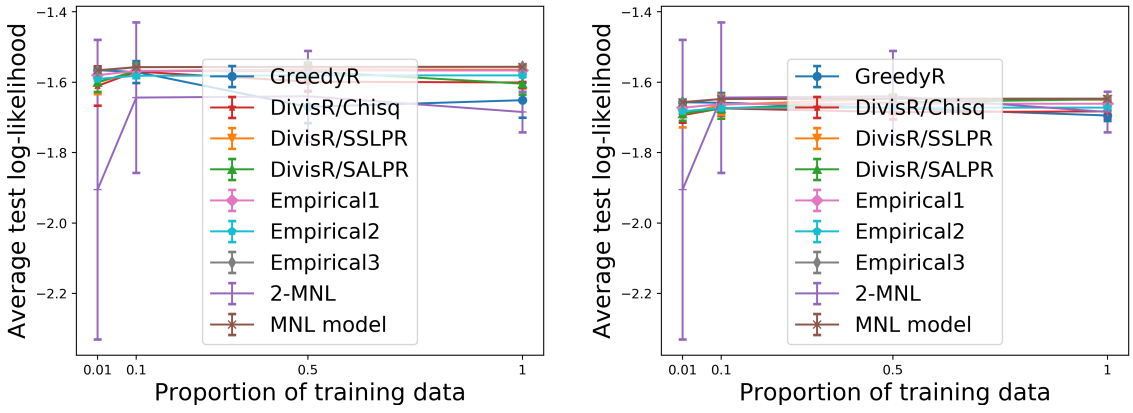
C.3. Experiment on Ranking Data - Meath Election

Implementing the Oracle. After verifying the reproduction of the algorithm, we then perform experiments on ranking data sets. The first thing to do is to implement the oracle. The biggest advantage of ranking data sets over choice data sets is that we can obtain most of the choice probabilities required for the 2-MNL algorithm. To be explicit, for example, for a choice set $S = \{i, j, k\}$, where i, j and k are 3 items from $\mathcal{U}$, the probability that i is chosen given S is the number of times that i is ranked higher than j and k divided by the total number of rankings that contain all items in S.

A numeric example. Let $i = 1$, $j = 3$, $k = 4$, and $\mathcal{U} = [5]$. Then $S = \{1, 3, 4\}$. Suppose we have 5 observations of ranking (from the highest rank to the lowest): $\{\{1, 2, 5, 4, 3\}$, $\{2, 5, 4\}$, $\{4, 1, 3, 2\}$, $\{1, 4, 5, 3\}$, $\{4, 5, 3\}\}$. Then, only $\{1, 2, 5, 4, 3\}$, $\{4, 1, 3, 2\}$ and $\{1, 4, 5, 3\}$ contain all items in S. Thus, the choice probabilities for items in S are $P_{\{1,3,4\}}(1) = 2/3$, $P_{\{1,3,4\}}(3) = 0$ and $P_{\{1,3,4\}}(4) = 1/3$. Such calculation may be unrealistic in choice data sets because if the actual choice is not an item in S, even though the choice set may contain all items in S, we would not know which item in S is preferred.

C.4. Results

The modification to the data generation seems to have little effect on the improving the learned nested tree structures. The best of our proposed algorithm is still slightly worse than the MNL model. The reason that the empirical tree structure we obtained from Huang and Guestrin (2012) has worse performance than the MNL tree does is, in the original paper, the best model was compared to an optimized 1-chain tree structure, rather then the MNL tree. Hence the MNL model still has the best performance overall. As for the uniform 2-MNL models, we can see that in most of the cases, especially when training data are sufficient, uniform 2-MNL models do not have an advantage over nested logit models and MNL models. They can be much worse than the nested logit models. Only when using 10% or 50% training data with



(a) $N = 20$ (b) $N = 40$

Figure C.1: Line plots of log-likelihood on the new generated choice data for all algorithms and empirical trees with error bars. N is the initial number of generated choice sets of size 7 in Section B.1.

Table C.1: Average test log-likelihood for the new Meath Election data sets

Algorithm	Proportion of Training Examples			
	0.01	0.1	0.5	1
New Meath Election data, $N = 20$				
MNL model	**-1.5663**	**-1.5573**	**-1.5566**	**-1.5565**
GreedyR	**-1.5663**	-1.5711	-1.6701	-1.6513
DivisR/Chisq	-1.6106	-1.5700	-1.5985	-1.6010
DivisR/SALPR	-1.6004	-1.5679	-1.5694	-1.6044
DivisR/SSLPR	-1.6011	-1.5682	-1.5640	-1.5647
Empirical1	-1.5802	-1.5678	-1.5671	-1.5669
Empirical2	-1.5908	-1.5819	-1.5810	-1.5808
Empirical3	-1.5679	-1.5575	-1.5567	-1.5566
2-MNL	-1.9050	-1.6438	-1.6402	-1.6847
New Meath Election data, $N = 40$				
MNL model	-1.6572	-1.6477	-1.6470	**-1.6469**
GreedyR	-1.6572	-1.6585	-1.6736	-1.6947
DivisR/Chisq	-1.6940	-1.6744	-1.6852	-1.6831
DivisR/SALPR	-1.6896	-1.6745	-1.6573	-1.6487
DivisR/SSLPR	-1.6922	-1.6641	-1.6470	-1.6501
Empirical1	-1.6726	-1.6622	-1.6614	-1.6613
Empirical2	-1.6828	-1.6732	-1.6724	-1.6723
Empirical3	**-1.6565**	-1.6479	-1.6471	-1.6470
2-MNL	-1.9050	**-1.6438**	**-1.6402**	-1.6847

$N = 40$, the uniform 2-MNL slightly outperforms other models. The poor performance with sufficient training data can be attributed to the aforementioned "inaccurate" assumption, while with insufficient training data, as uniform 2-MNL models exploit the rankings in the training data, it is reasonable that the learned models can be better than other models. It is worth to mention, however, that learning uniform

2-MNL models is very efficient, which only takes several seconds, while other algorithms may take up to tens of thousand seconds because of the MLE, which does not exist in the learning process of uniform 2-MNL models.

BIBLIOGRAPHY

Agarwala, R, Bafna, V, Farach, M, Paterson, M, and Thorup, M (1998). On the approximability of numerical taxonomy (fitting distances by tree metrics). *SIAM Journal on Computing* 28.3, 1073–1085.

Aggarwal, CC (2013). *An Introduction to Cluster Analysis*.

Agresti, A (2018). *An introduction to categorical data analysis*. Wiley.

Aho, AV, Sagiv, Y, Szymanski, TG, and Ullman, JD (1981). Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing* 10.3, 405–421.

Bandelt, H-J and Dress, A (1986). Reconstructing the shape of a tree from observed dissimilarity data. *Advances in applied mathematics* 7.3, 309–343.

Benson, AR, Kumar, R, and Tomkins, A (2016). "On the relevance of irrelevant alternatives". In: International World Wide Web Conferences Steering Committee, 963–973.

Berry, ST (1994). Estimating discrete-choice models of product differentiation. *The RAND Journal of Economics*, 242–262.

Bierlaire, M (1998). Discrete choice models. In: *Operations research and decision aid methodologies in traffic and transportation management*. Springer, 203–227.

Boyd, S and Vandenberghe, L (2004). *Convex optimization*. Cambridge university press.

Brodal, GS, Fagerberg, R, Pedersen, CN, and Östlin, A (2001). "The complexity of constructing evolutionary trees using experiments". In: *International Colloquium on Automata, Languages, and Programming*. Springer, 140–151.

Buneman, P (1971). The recovery of trees from measures of dissimilarity. *Mathematics in the archaeological and historical sciences*.

Celma Herrada, Ò (2009). Music recommendation and discovery in the long tail.

Chierichetti, F, Kumar, R, and Tomkins, A (2018). "Learning a Mixture of Two Multinomial Logits". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J Dy and A Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmssan, Stockholm Sweden: PMLR, 961–969. URL: http://proceedings.mlr.press/v80/chierichetti18a.html.

Colonius, H and Schulze, HH (1981). Tree structures for proximity data. *British Journal of Mathematical and Statistical Psychology* 34.2, 167–180.

Das, SS, Ghosh, S, Maitra, B, and Boltze, M (2012). Search Strategy for Nested Logit Tree Structure: A Case Study of Rural Feeder Service to Bus Stop. *International Journal for Traffic and Transport Engineering* 2.4, 555–560.

Elshiewy, O, Guhl, D, and Boztug, Y (2017). Multinomial Logit Models in Marketing-From Fundamentals to State-of-the-Art. *Marketing ZFP* 39.3, 32–49.

Emamjomeh-Zadeh, E and Kempe, D (2018). "Adaptive hierarchical clustering using ordinal queries". In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 415–429.

Fitch, WM and Margoliash, E (1967). Construction of phylogenetic trees. *Science* 155.3760, 279–284.

Gambette, P, Berry, V, and Paul, C (2012). Quartets and unrooted phylogenetic networks. *Journal of bioinformatics and computational biology* 10.04, 1250004.

Gormley, IC and Murphy, TB (2006). "A latent space model for rank data". In: *ICML Workshop on Statistical Network Analysis*. Springer, 90–102.

Hausman, J and McFadden, D (1984). Specification tests for the multinomial logit model. *Econometrica: Journal of the Econometric Society*, 1219–1240.

Henderson, T (2018). *Tree edit distance using the Zhang Shasha algorithm*. `https://github.com/timtadh/zhang-shasha`.

Hensher, DA and Ton, TT (2000). A comparison of the predictive potential of artificial neural networks and nested logit models for commuter mode choice. *Transportation Research Part E: Logistics and Transportation Review* 36.3, 155–172.

Huang, J, Guestrin, C, et al. (2012). Uncovering the riffled independence structure of ranked data. *Electronic Journal of Statistics* 6, 199–230.

Kannan, SK, Lawler, EL, and Warnow, TJ (1996). Determining the evolutionary tree using experiments. *Journal of Algorithms* 21.1, 26–50.

Koppelman, FS and Bhat, C (2006). A self instructing course in mode choice modeling: multinomial and nested logit models.

Lockshin, L, Jarvis, W, d?Hauteville, F, and Perrouty, J-P (2006). Using simulations from discrete choice experiments to measure consumer sensitivity to brand, region, price, and awards in wine choice. *Food quality and preference* 17.3-4, 166–178.

MacQueen, J et al. (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA, 281–297.

Manski, CF (1975). Maximum score estimation of the stochastic utility model of choice. *Journal of Econometrics* 3.3, 205–228.

Mattei, N and Walsh, T (2013). "Preflib: A library for preferences http://www. preflib. org". In: *International Conference on Algorithmic DecisionTheory*. Springer, 259–270.

McFadden, D (1981). Econometric models of probabilistic choice. *Structural analysis of discrete data with econometric applications* 198272.

McFadden, D and Train, K (2000). Mixed MNL models for discrete response. *Journal of applied Econometrics* 15.5, 447–470.

McFadden, D et al. (1973). Conditional logit analysis of qualitative choice behavior.

McFadden, D, Talvitie, A, Cosslett, S, Hasan, I, Johnson, M, Reid, F, and Train, K (1977). Demand model estimation and validation. *Urban Travel Demand Forecasting Project, Phase* 1.

Mojaverian, S, Rasouli, F, and Hosseini-Yekani, S (2014). Citrus marketing channel strategy and its determinants in Mazandaran province of Iran: An application of nested logit model. *Journal of Agricultural Science and Technology* 16, 1469–1479.

Reynolds, AP, Richards, G, and Rayward-Smith, VJ (2004). "The application of k-medoids and pam to the clustering of rules". In: *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 173–178.

Sattath, S and Tversky, A (1977). Additive similarity trees. *Psychometrika* 42.3, 319–345.

Train, K (1978). A validation test of a disaggregate mode choice model. *Transportation Research* 12.3, 167–174.

Wu, G, Kao, M-Y, Lin, G, and You, J-H (2008). Reconstructing phylogenies from noisy quartets in polynomial time with a high success probability. *Algorithms for Molecular Biology* 3.1, 1.

Yang, S-H, Long, B, Smola, AJ, Zha, H, and Zheng, Z (2011). "Collaborative competitive filtering: learning recommender using context of user choice". In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 295–304.

Zhang, K and Shasha, D (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing* 18.6, 1245–1262.