COMS 6998-4 Fall 2017 Presenter: Daniel Hsu
Oct 18, 2017 Scribe: Ji Xu

Adaptive Hierarchical Clustering Using Ordinal Queries

1 Motivation

We have seen equivalence query, split query and merge query. In this lecture, we want to talk about ordinal query and hierarchical clustering in Zadeh and Kempe [1]. For a set of elements, we can cluster the elements that are "similar to each other". Moreover, for a same set of elements, we can have different way of clustering based on smaller concept or broader concept. For example, cat and dog are more similar to each other than zebra because cat and dog can be pets but zebra can not. Yet, {cat, dog and zebra} are more similar to each other than {tree and flowers} since the first group is animal and the second group is plant. Both group falls into category of creature. Hence, we can cluster the set {cat, dog, zebra, tree, flower} as {(cat,dog),(zebra),(tree,flower)} or {(cat,dog,zebra),(tree,flower)}. This forms a hierarchical clustering tree:

$$\Big(\big((\text{cat},\text{dog}),\text{zebra}\big),(\text{tree},\text{flower})\Big).$$

Hence, given a set of elements, we can ask users to provide information about which two elements are closer to each other comparing to others, and use the information to build a correct hierarchical clustering tree.

2 Model and Main result

2.1 Model

We have n data points $\mathcal{X} = \{x_1, x_2, x_3 \cdots, x_n\}$. We assume that there exists a hierarchical clustering over $\mathcal{X}$ and this clustering tree is a **fully rooted binary** tree with the set of leaves being $\mathcal{X}$. The goal is to learn the hierarchical clustering tree through the minimal number of ordinal queries defined as the following.

Definition 1. *Query model:*

- *Input: 3 leaves* (x_i, x_j, x_k).
- *Output:* x_i *if and only if there exists a cluster contains* x_j *and* x_k *but not* x_i.

Remark 1. *Note that the ordinal query is equivalent to output* x_i *if and only if the root-to-*x_i *path does not contain the lowest common ancestor (LCA) of* (x_j, x_k). *Also, intuitively, if the output is* x_i, *it implies that* (x_j, x_k) *are more similar comparing to* (x_i, x_k) *or* (x_i, x_j).

Remark 2. *We can show that given* $\mathcal{X}$, *two hierarchical clustering trees are equivalent if and only if the results of ordinal queries for every triplet* (x, x', x'') *are the same for the two clustering trees.*

2.2 Main results

Theorem 1. *We can learn the hierarchical clustering tree using at most $\lceil n \log_2 n \rceil$ ordinal queries.*

Theorem 2. *We have the following lower bound results:*

- *Any adaptive algorithm requires $\Omega(n \log n)$ ordinal queries to learn the hierarchical clustering over n elements.*
- *Any non-adaptive algorithm requires $\frac{1}{4}\binom{n}{3} = \Omega(n^3)$ ordinal queries to learn the hierarchical clustering over n elements.*

Remark 3. *If the clustering tree is not subject to be a binary tree, i.e, it can have more than two children for each node, the algorithm may need $O(n^2)$ number of ordinal queries to learn the correct clustering tree.*

3 Algorithm and Proof

The algorithm is basically the same algorithm in Pearl and Tarsi [3]. The main idea is incrementally build the tree from its restrictions of subsets of points. Let the input data points are in the order of $(x_1, \cdots, x_n)$. The algorithm is the following:

1. Let $\mathcal{T}_2$ be the unique (trivial) hierarchical clustering of elements x_1, x_2.
2. For $i = 3, \cdots, n$ do
   - Insert x_i in $\mathcal{T}_{i-1}$ to get $\mathcal{T}_i$
3. return $\mathcal{T}_n$

Insertion: $\mathcal{T}$ is a binary tree with l leaves L and $x_i \notin L$ is the new data point to insert.

- Case $l = 2$: let x_j, x_k are in the leaves of current tree $\mathcal{T}$. Then we just need to query (x_j, x_k, x_i) and the corresponding new tree $\mathcal{T}'$ is shown in Figure 1.
- Case $l \geq 3$: We first claim that

$$\exists v \in \mathcal{T},\ s.t.\ \frac{l}{3} \ \leq\ |\text{leaves}(v)| \ \leq\ \frac{2l}{3}. \tag{1}$$

  Suppose (1) holds and let v be the node that has the property. Let $x_j, x_k \in L$ such that $\text{LCA}(x_j, x_k) = v$. Let $\mathcal{T}_v$ be the subtree rooted at v and let $\mathcal{T}_L^v, \mathcal{T}_R^v$ be the left and right subtree of $\mathcal{T}_v$. We assume $x_j \in \mathcal{T}_L^v$ and $x_k \in \mathcal{T}_R^v$ (See Figure 2). Now, we query (x_j, x_k, x_i).

  – If the output of the query is x_i, then we insert x_i into $\mathcal{T} \backslash \mathcal{T}_v$.

– If the output of the query is x_j, then we insert x_i into $\mathcal{T}_R^v$.

– If the output of the query is x_k, then we insert x_i into $\mathcal{T}_L^v$.

Since each time we call the query, we insert x_i into a smaller subtree which total number of leaves is reduced by at least a factor of $1/3$. Hence, since there are at most i leaves at iteration i, we know the total number of queries called in iteration i is at most $\log_{3/2} i$. Hence, the total number of queries called after all n iterations is at most $n \log_{3/2} n$.

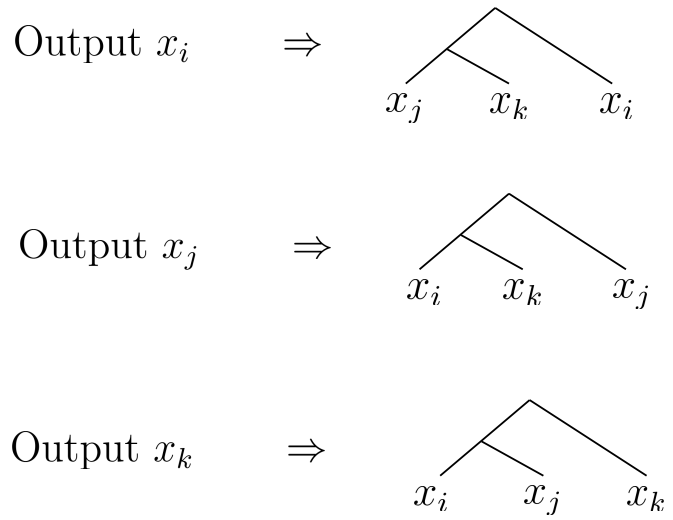

Figure 1: The corresponding 3-leaves-clustering tree based on ordinal query (x_i, x_j, x_k).

The correctness of the algorithm is straightforward by induction. We just need to show that the claim of (1) is true. We first do the following procedure to find a path from the root to a leaf:

- Start with the root v_0 of the tree $\mathcal{T}$
- For $t = 1, 2, \cdots$, let v_t is the child of v_{t-1} with higher number of leaves. i.e.,

$$|\text{leaves}(v_t)| \geq \frac{1}{2}|\text{leaves}(v_{t-1})|. \tag{2}$$

Let

$$\text{root} : v_0 \to v_1 \to v_2 \cdots \to v_k : \text{leaf}$$

is the path selected by the above procedure. Since $|\text{leaves}(v_i)|$ is a decreasing sequence and $|\text{leaves}(v_0)| = l$ and $|\text{leaves}(v_k)| = 1$, we know there exists j such that $|\text{leaves}(v_j)| \leq \frac{2}{3}l$. Let j^* is the smallest index satisfies the property, i.e,

$$|\text{leaves}(v_{j^*})| \ \leq \ \frac{2}{3}l \quad \text{and} \quad |\text{leaves}(v_{j^*-1})| \ > \ \frac{2}{3}l.$$

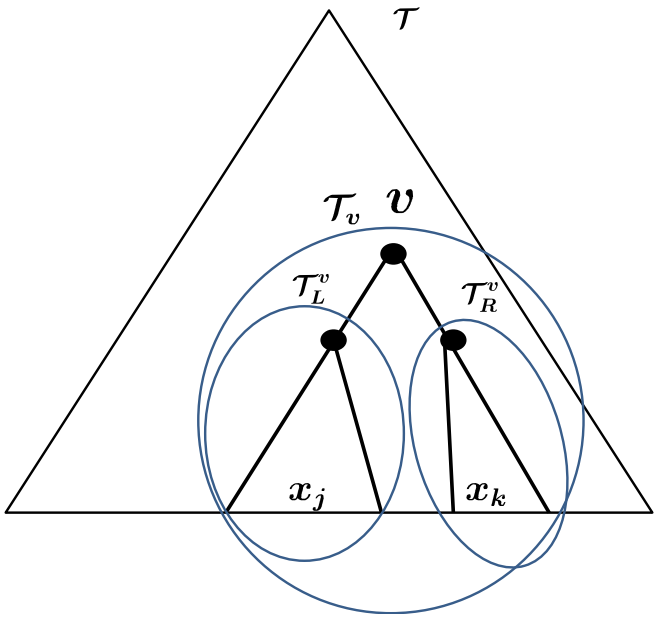

Figure 2: Structure used in case $l \geq 3$.

By (2), we have

$$|\text{leaves}(v_{j^*})| \ \geq \ \frac{1}{2}|\text{leaves}(v_{j^*-1})| \ > \ \frac{1}{3}l.$$

Hence, let $v = v_{j^*}$, we have claim (1) holds.

Remark 4. *Claim* (1) *can be improved as the following:*

$$\exists v \in \mathcal{T}, \ s.t. \ \frac{l}{2} \ < \ |leaves(v)| \ \leq \ \frac{l}{2} + 1, \tag{3}$$

which is a result proved in [2].

4 Lower Bounds

4.1 Adaptive Algorithm

Note that each ordinal query only reveals at most $\log_2(3)$ bits of information and we need at least $\log_2 \#$hierarchical clustering trees bits information to distinguish the correct clustering tree. Hence, we need at least

$$\Omega(\log_3 \#\text{hierarchical clustering trees})$$

number of queries. Now we need to show a lower bound on the number of balanced hierarchical clustering trees (See Figure 3). On one hand, the number of permutations of the leaves is $n!$. On the other hand, for each permutation of the leaves, if we switch left subtree and

right subtree at any vertex, we have exactly the same clustering tree. Hence, since the total number of vertex is at most n, the number of different clustering trees is at least $\frac{n!}{2^n}$. By Stirling's approximation, we have

$$\frac{n!}{2^n} = 2^{\Omega(n \log_2 n)}.$$

Hence, we need at least $\Omega(n \log_2 n)$ number of bits to distinguish different clustering tree, and therefore the number of queries required is at least $\Omega(n \log_3 n)$.

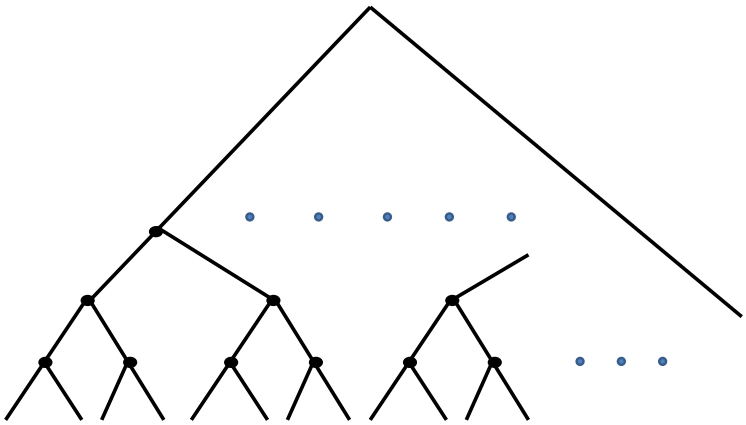

Figure 3: Balanced binary tree: the depth of the two subtrees of every node never differ by more than 1.

4.2 Non-adaptive Algorithm

Let K be the set of ordinal queries used in the non-adaptive algorithm with $|K| = k$. Since the algorithm is non-adaptive, the k ordinal queries should be able to determine the structure of any hierarchical clustering tree. We uniformly at random pick a permutation of n data points and let them be the leaves of a balanced hierarchical clustering tree T (See Figure 3) in the corresponding order of the permutation.

Without loss of generality, we assume the leaves of the tree are in this order $\{x_1, x_2, \cdots, x_n\}$, We first look at any quartet $T_i = (x_i, x_{i+1}, x_{i+2}, x_{i+3})$, i.e, a subtree only contains 4 leaves (See Figure 4). We know that only the following 4 ordinal queries

$$S_i = \{(x_i, x_{i+1}, x_{i+2}), (x_i, x_{i+1}, x_{i+3}), (x_i, x_{i+2}, x_{i+3}), (x_{i+1}, x_{i+2}, x_{i+3})\}$$

can provide information of the structure of T_i (the other two wrong structure of T_i will have same output as T_i for all other ordinal queries). Note that the number of such quartet T_i is $\frac{n}{4}$. Hence, we need at least $\frac{n}{4}$ ordinal queries in $\bigcup S_i$ to determine the structure of each quartet T_i. On the other hand, since we pick the permutation uniformly at random, we have

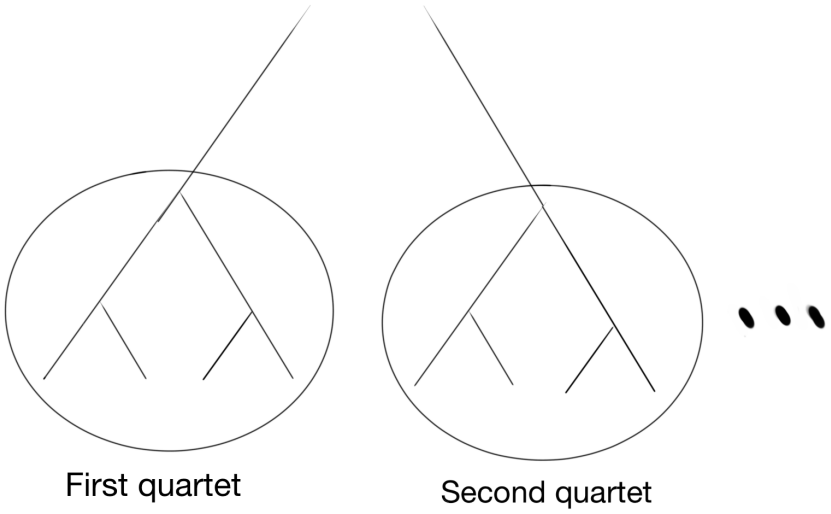

Figure 4: First quartet and second quartet

probability of $\frac{4}{\binom{n}{3}}$ for the event that any fixed ordinal query used by the algorithm is one of the ordinal queries in S_i for a fixed i. (This is true because the target probability is just the probability that one uniformly chosen triplet is equal to one specific triplet). Hence, the expected number of queries in $K \bigcap (\bigcup S_i)$ is

$$\mathbb{E} \sum_k \sum_j \text{Indicator(the kth ordinal query in } K \text{ is in } S_j) \;=\; k \cdot \frac{n}{4} \cdot \frac{4}{\binom{n}{3}}.$$

Hence, we require

$$k \cdot \frac{n}{4} \cdot \frac{4}{\binom{n}{3}} \geq \frac{n}{4},$$

otherwise, there will exist a tree T such that the algorithm can not cover at least $\frac{n}{4}$ ordinal queries in $\bigcup S_i$. Hence, we have

$$k \geq \frac{4}{\binom{n}{3}}.$$

Remark 5. *One simpler way to see the argument is we only focus on the first quartet. Then, for any query $q_i \in K$, the probability of $\{q_i \in S_1\}$ is $4/\binom{n}{3}$. Hence, the expected number*

of queries in $K \bigcup S_1$ *is* $4k/\binom{n}{3}$. *Since we need at least one query in* S_1 *to determine the structure, we require*

$$4k \Big/ \binom{n}{3} \geq 1,$$

which concludes the same result.

References

[1] E. E.-Zadeh and D. Kempe. Adaptive Hierarchical Clustering Using Ordinal Queries. *Arxiv preprint.*

[2] C. Jordan. Sur les assemblages de lignes. *Journal für die reine und angewandte Mathematik*, 70:185-190, 1869.

[3] J. Pearl and M. Tarsi. Structuring causal trees. *ournal of Complexity*, 2.1 (1986): 60-77.