COMS 6998-4 Fall 2017 Presenter: Daniel Hsu
October 18, 2017 Scribe: Xuefeng Hu

Adaptive hierarchical clustering

1 Introduction

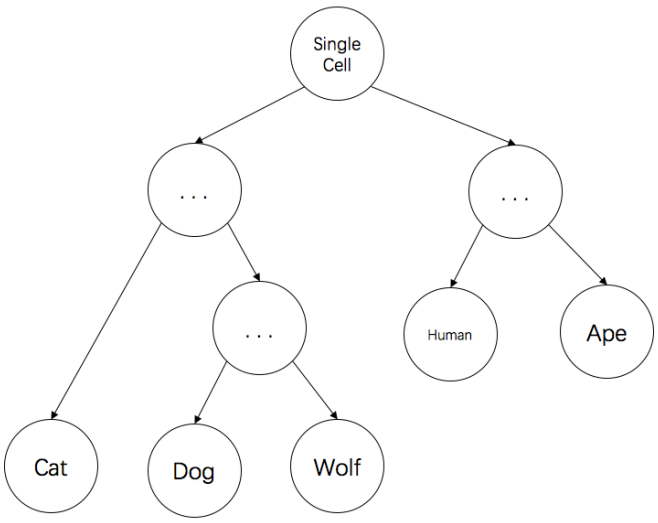

Figure 1: Phylogenetic Tree

Instead of just looking at the flat clustering of the data, the hierarchical structure between different clusters may be more interesting in some scenarios such as the clustering of animal species. Figure 1 shows a hierarchical clustering structure in animal species with the phylogenetic tree. If we want to cluster animals by their common ancestors, it will be nearly impossible to do so with just a flat clustering.

A key question in clustering is how to measure similarities between samples. Instead of using any pre-determined metrics to estimate the similarities on our own, we are more interested in the model where we can get this similarity measure from the feedback of human experts (or ideally, an Oracle).

In particular, in this lecture, we will show that if we have the ability to make Ordinal Queries, which returns a nearer pair between three input samples, it will be possible to have an adaptive algorithm that makes $O(n \log n)$ queries to reconstruct the hierarchical clustering [1]. Furthermore, we will also show that this bound is tight for adaptive algorithms, while any non-adaptive algorithms will suffer an even higher query complexity, $O(n^3)$ [1].

2 Notations and Preliminaries

We have n points $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ as input data.

Definition 2.1. *Hierarchical Clustering*
A hierarchical clustering $\mathcal{H}$ on data $\mathcal{X}$ is a rooted complete binary tree with all elements in $\mathcal{X}$ as leaves.

Definition 2.2. *Ordinal Queries*
An Ordinal Query $\mathcal{Q}$ will be the process to take following inputs to return the following outputs.

- **Input**: 3 leaves $x_i, x_j, x_k \in \mathcal{X}$.
- **Output**: one of 3 leaves $x_o \in \{x_i, x_j, x_k\}$ if and only if there exists an cluster contains $\{x_i, x_j, x_k\} \backslash \{x_o\}$ without x_o.

Remark 1. *It is intuitive to understand the Ordinal Query as a model that returns the nearer pair among three points. Therefore we may be able to consider an equivalent definition of Ordinal Query as follows: the Ordinal Query returns x_i upon taking $\{x_i, x_j, x_k\}$ as inputs is equivalent to x_i is the point and the only point such that the root-to-x_i path does not contain* $\mathrm{LCA}(x_j, x_k)$*, where* $\mathrm{LCA}(\cdot)$ *denotes the least common ancester in $\mathcal{H}$.*

Remark 2. *Any Hierarchical Clustering $\mathcal{H}$ as defined above should be uniquely identified by the results of Ordinal Queries to all triplets $\{x_i, x_j, x_k\} \in \mathcal{H}$.*

Definition 2.3. *Adaptive and Non-Adaptive algorithms*
By saying an algorithm $\mathcal{A}$ is adaptive, we mean that $\mathcal{A}$ can make queries based on previous queries results. In contrast, a non-adaptive algorithm should make all of the queries before the program starts and therefore there is no dependency between different queries. Denote the set of algorithms that adaptive learn hierarchical clustering $\mathcal{H}$ by $\mathcal{P}_A$, and the set of algorithms that learn hierarchical clustering $\mathcal{H}$ in a non-adaptive way by $\mathcal{P}_N$.

Definition 2.4. *Leaves in subtree*
For simplicity in the proof, we define function

$$\text{Leaves}(a) := \text{number of leaves in the subtree rooted by node a.}$$

3 Main Results

Here are the three most import results in this lecture:

Theorem 1. *$\exists \mathcal{A} \in \mathcal{P}_A$, such that $\mathcal{A}$ constructs $\mathcal{H}$ with $O(n \log n)$ Ordinal Queries.*

Theorem 2. *For every adaptive algorithm $\mathcal{A} \in \mathcal{P}_A$, there exist tree $\mathcal{H}$ for which $\mathcal{A}$ requires at least $O(n \log n)$ Ordinal Queries to reconstruct $\mathcal{H}$.*

Theorem 3. *For every non-adaptive algorithm $\mathcal{A} \in \mathcal{P}_N$, there exist tree $\mathcal{H}$ for which $\mathcal{A}$ requires at least $O(n^3)$ Ordinal Queries to reconstruct $\mathcal{H}$.*

4 Adaptive Algorithm

Very much like the idea in Pearls work [3], we are going to introduce an algorithm that incrementally build the tree from its "restriction" of subsets of points.

Algorithm 1 Insertion Based Clustering ($\mathcal{X} = \{a_1, a_2, ..., a_n\}$)

Order the tree arbitrarily, $\mathcal{X} \leftarrow \{x_1, x_2, ..., x_n\}$, where $\{x_1, x_2, ..., x_n\}$ is an random permutation of $\{a_1, a_2, ..., a_n\}$
Let T_2 be the trivial hierarchical clustering of x_1, x_2
for each i=3,4,...,n **do**
    $T_i \leftarrow$**Insert**(x_i, T_{i-1})
end for
Return T_n

Algorithm 2 Insert(x, T)

Input: T : tree restrict to leaves $\mathcal{L} \in \mathcal{X}$; $l = |\mathcal{L}|$; $x \notin \mathcal{L}$

if l=2 **then**
    Denote the two leaves of T by x_l, x_r.
    Make Query $\mathcal{Q}(x, x_l, x_r)$ reveals true insertion way in three possible ways.
else
    Pick $v \in T$ such that
$$\frac{1}{3}l \leq \text{Leaves}(v) \leq \frac{2}{3}l.$$
    and denote the substree rooted by v by V
    Pick x_l, x_r such that LCA$(x_l, x_r) = v$.
    Make Query $\mathcal{Q}(x, x_l, x_r)$.
    if $x = \mathcal{Q}(x, x_l, x_r)$ **then**
        Insert$(x, T \backslash V)$
    else
        Let $V' = V_l$ if $x_l = \mathcal{Q}(x, x_l, x_r)$, $V' = V_r$ if $x_r = \mathcal{Q}(x, x_l, x_r)$, where V_l and V_r are subtrees under v that contains v_r, v_l as show in Figure 3.
        Insert(x, V')
    end if
end if

Correctness Analysis

As Algorithm 1,2 show, we construct the hierarchical clustering by recursively insert new points into the existing clustering using Ordinal Queries. We will prove the algorithm works in an inductive way:

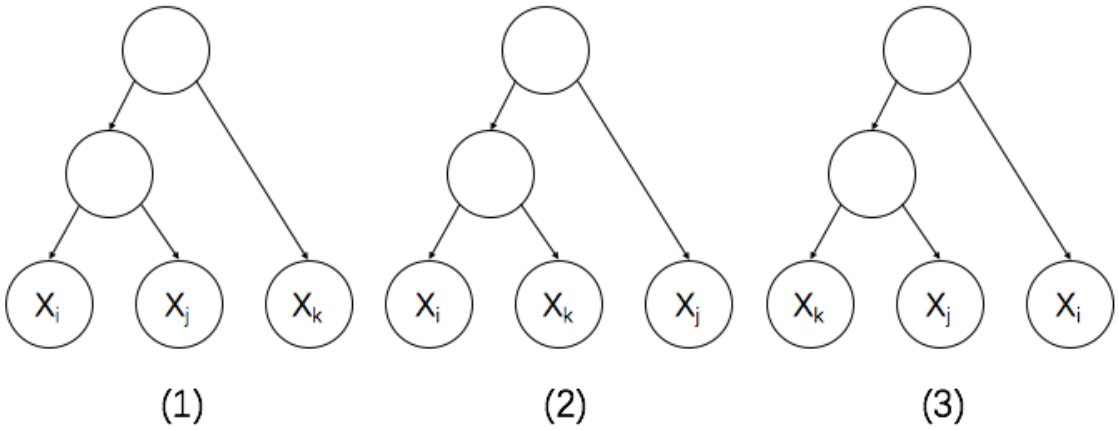

Figure 2: A demonstration of base case of the insert algorithm

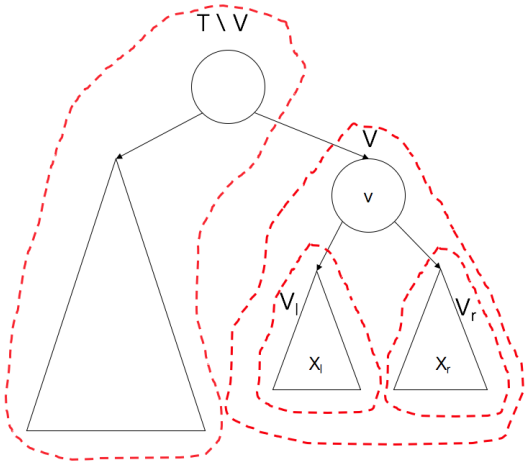

Figure 3: A demonstration of the insert algorithm

- Base case: We can construct the trivial clustering structure with a single query $\mathcal{Q}$ in the base case where $l = 2$. As shown in Figure 2, there is only 3 possible way to insert x_i into an the trivial clustering of x_j, x_k. According to the definition of $\mathcal{Q}(x_i, x_j, x_k)$, we should insert x_i as way (1) if $x_k = \mathcal{Q}(x_i, x_j, x_k)$, insert x_i as way (2) if $x_j = \mathcal{Q}(x_i, x_j, x_k)$ and insert x_i as way (3) if $x_i = \mathcal{Q}(x_i, x_j, x_k)$. Therefore, the base case works.

- Recursive Step: the major concern of the recursive step is the existence of v such that

$$\frac{1}{3}l \leq \text{Leaves}(v) \leq \frac{2}{3}l.$$

To prove this, consider an arbitrary complete binary tree T rooted by a_0, and a nodes sequence: $\{a_0, a_1, a_2, ..., a_t\}$, where each a_i is the child of a_{i-1} with a larger subtree, and

a_t achieves the leaf since T is finite. Therefore by definition, we have

$$\text{Leaves}(a_0) = |T|,$$

$$\text{Leaves}(a_t) = 1,$$

$$\text{Leaves}(a_i) \leq \frac{1}{2}\,\text{Leaves}(a_{i-1}), \forall i \in \{2, 3, 4..., t\}.$$

Thus, we must have a j such that

$$\frac{1}{3}|T| \leq \text{Leaves}(a_j) \leq \frac{2}{3}|T|.$$

We can not jump through this gap with step large than $\frac{1}{2}$, thus we proved the existence of v in the recursive step. Therefore, as shown in Figure 3, by query $\mathcal{Q}(x, x_l, x_k)$, we can uniquely find the subtree that contains v among $T\backslash V$, V_l and V_r, and continue on the recursive step with at most $\frac{2}{3}$ of the problem size.

Query Complexity analysis

Denote the time complexity of the Algorithm **Insert** by $T(l)$ where l is the size of the current hierarchical clustering tree. With the result from the last section we have

$$T(l) \leq T\left(\frac{2}{3}l\right) + 1.$$

By master theorem we have $T(l) = \log_{3/2}(l)$. Thus the total complexity of Algorithm **Insertion Based Clustering** is

$$\begin{aligned}\sum_{i=1}^{n} \log_{3/2}(i) &\leq \sum_{i=1}^{n} \log(n) \\ &= O(n \log n).\end{aligned}$$

Therefore **Theorem 1** is proved: Algorithm **Insertion Based Clustering** can construct the hierarchical clustering with only $O(n \log n)$ Ordinal Queries $\mathcal{Q}$.

Supplement Discussions

As mentioned in the lecture, for any complete binary tree T, Jordan [2] has proved that there actually exists a node $v \in T$, such that

$$\frac{1}{2}l \leq \text{Leaves}(v) \leq \frac{1}{2}l + 1,$$

where l is the number of leaves in T. This result will induce a constant factor speedup to the result proved in the last section.

Another note is if the hierarchical clustering structure is not defined as a binary tree but a tree with maximum degree k, the query complexity of Algorithm **Insertion Based Clustering** will be

$$\begin{aligned}\sum_{i=1}^{n} \log_{1+1/k}(i) &\leq \sum_{i=1}^{n} k \log(n) \\ &= O(kn \log n),\end{aligned}$$

which will suffer from the high dimension cost.

5 Lower Bounds

Adaptive Algorithm

We will prove **Theorem 2** in an information theoretical way.

Proof. Consider the number of all possible trees of n points: there will be

$$n! = O(n \log n)$$

ways to construct hierarchical clustering on n data points.
However, for each query $\mathcal{Q}$, we can only get

$$\log_2 3$$

bits of information.
Therefore, we need at least

$$\frac{O(n \log n)}{\log_2 3} = O(n \log n)$$

queries to uniquely identify a hierarchical clustering on n points. □

Non-adaptive Algorithm

In this section we will prove **Theorem 3**, that there exist a hierarchical clustering $\mathcal{H}$, such that any non-adaptive algorithm will have to take at least $O(n^3)$ Ordinal Queries. More specifically, we will show that this lower bound is

$$\frac{1}{4}\begin{pmatrix} n \\ 3 \end{pmatrix}.$$

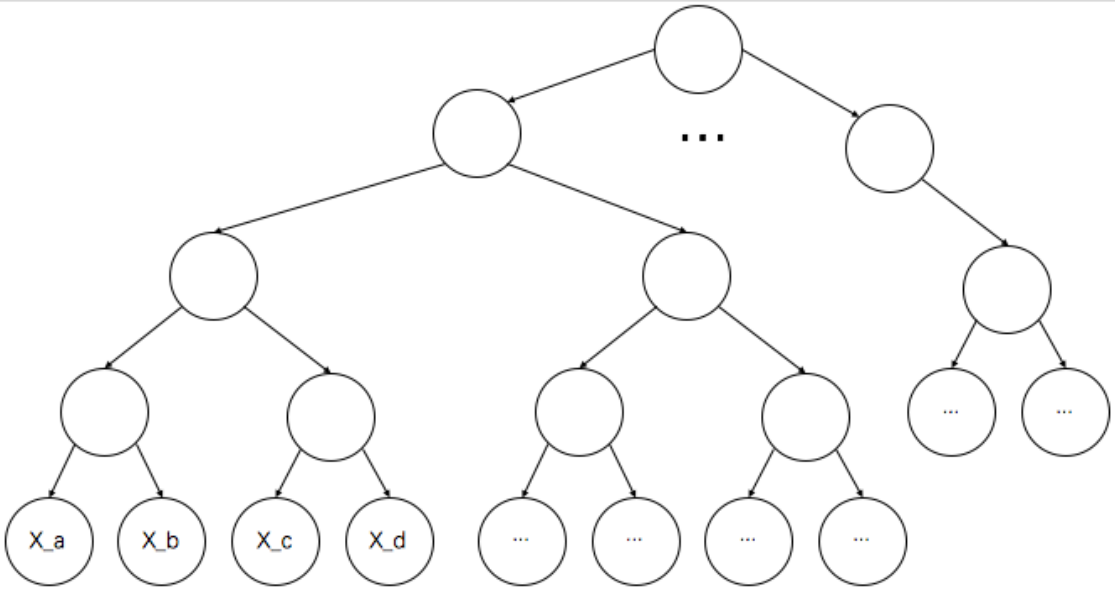

Figure 4: 4-partition demonstration

Proof. Without loss of generality, assume $n = 2^k$ for some $k \in \mathbb{N}^*$. Tree $\mathcal{H}$ gives an order of the n points $\{x_1, x_2, ..., x_n\}$, and as shown in Figure 4, it is possible for us partition the leaves into group of 4 from left to right. We denote each 4-leaves group by a "quarter", and there will be $n/4$ such quarters.

Consider the first quarter as a representative. A query $\mathcal{Q}$ is "useful" in figuring out the sub-structure between $\{x_a, x_b, x_c, x_d\}$ if and only if the query is made within points from $\{x_a, x_b, x_c, x_d\}$.

For a fixed $\mathcal{Q}_i$,

$$\begin{aligned} &\Pr(\mathcal{Q}_i \text{ is useful for the first quarter of a random tree}) \\ =&\Pr(\text{Uniformly random } \mathcal{Q} \text{ is useful for first quarter of some fixed tree}) \\ =&\frac{4}{\binom{n}{3}}. \end{aligned}$$

Therefore, Consider we make k queries, the expectation of useful queries with respect to

random hierarchical clustering of n points, we have

$$\begin{aligned}
&\mathbb{E}[\#\text{useful queries}] \\
=&\sum_{\text{queries}} \sum_{\text{quarters}} \Pr(\text{Uniformly random } \mathcal{Q} \text{ is useful for first quarter of some fixed tree}) \\
=&k \cdot \frac{n}{4} \cdot \frac{4}{\binom{n}{3}}.
\end{aligned}$$

Therefore, there exist a random hierarchical clustering of n points, that only has less than

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

useful queries in k queries, and therefore to determine the sub-structure of all $\frac{n}{4}$ quarters, we need

$$\begin{aligned}
k \cdot \frac{n}{4} \cdot \frac{4}{\binom{n}{3}} &> \frac{n}{4} \\
\Rightarrow k &\geq \frac{1}{4}\binom{n}{3}
\end{aligned}$$

queries.
Therefore we need $O(n^3)$ queries to identify this hierarchical clustering. □

References

[1] Ehsan Emamjomeh-Zadeh and David Kempe. Adaptive hierarchical clustering using ordinal queries. *arXiv preprint arXiv:1708.00149*, 2017.

[2] Camille Jordan. Sur les assemblages de lignes. *J. Reine Angew. Math*, 70(185):81, 1869.

[3] Judea Pearl and Michael Tarsi. Structuring causal trees. *Journal of Complexity*, 2(1):60–77, 1986.