Hierarchical Clustering

Thomas Bonald
Institut Polytechnique de Paris

2021 – 2022

Consider n points $x_1, \dots, x_n \in \mathbb{R}^d$ (for instance, the spectral embedding of n objects linked by some similarity matrix). We seek to cluster these points in a hierarchical way so as to capture the complex, multi-scale nature of real datasets.

1 Divisive approach

A first approach, the *divisive* approach, consists in starting from a single cluster and to split this cluster recursively. For instance, two clusters are formed using k-means with $k = 2$, then each of these clusters is divided into two clusters separately, and so on. This approach usually provides less relevant clusterings than the *agglomerative* approach, considered in the rest of the document.

2 Agglomerative approach

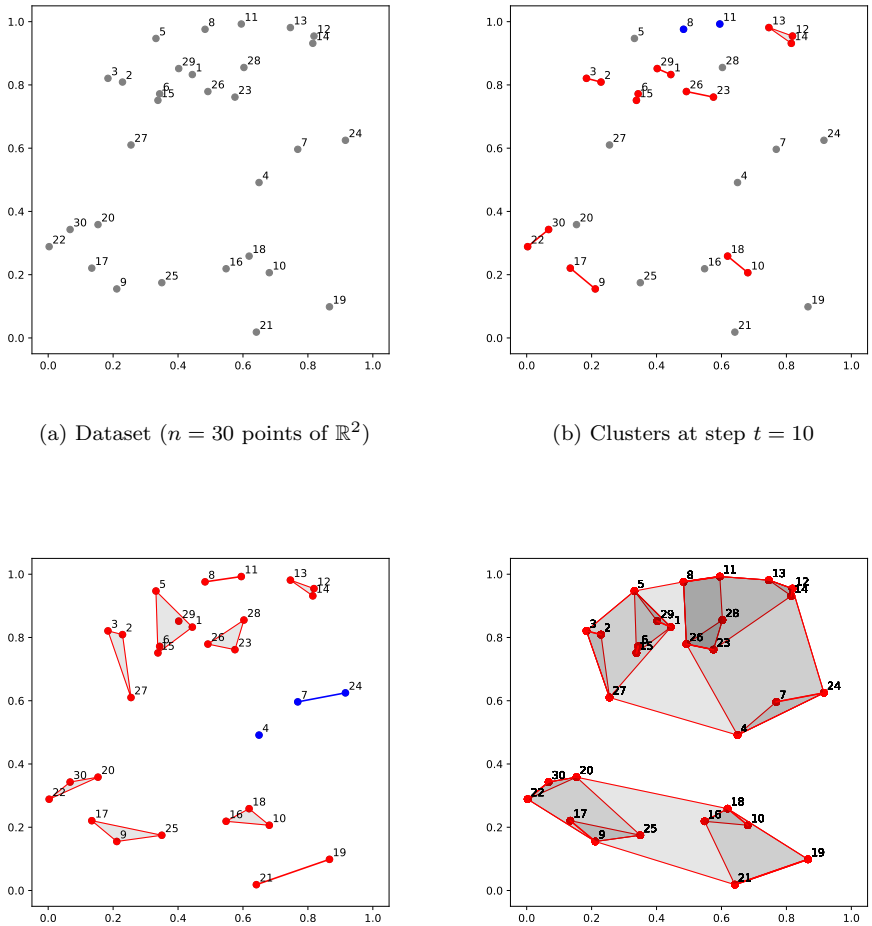
The agglomerative approach consists in starting from individual clusters (each point is in its own cluster) and to merge clusters recursively. At each step of the algorithm, the two *closest* clusters are merged. Thus the algorithm is based on some distance d between clusters. This distance d is not necessarily a metric. We only require d to be non-negative and symmetric.

The algorithm is the following:

1. Initialisation: $\mathcal{C} \leftarrow \{\{1\}, \dots, \{n\}\}$, $L \leftarrow \emptyset$
2. Agglomeration: For $t = 1, \dots, n-1$,
   - $A, B \leftarrow \arg\min_{a,b \in \mathcal{C}, a \neq b} d(a, b)$
   - $C \leftarrow A \cup B$
   - $\mathcal{C} \leftarrow \mathcal{C} \setminus \{A, B\}$
   - $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$
   - $L \leftarrow L + (A, B)$
3. Return L

Observe that there are $n - t$ clusters at the end of step t. An example is shown in Figure 1 for $n = 30$ points of $\mathbb{R}^2$.

The successive clusters found by the algorithm together with the distances between the merged clusters can be represented by a dendrogram, as shown in Figure 2. The height of each merge corresponds to the distance between the two clusters before the merge, which may be viewed as the level of energy induced by the merge. In particular, high merges in the dendrogram corresponds to bad merges in the dataset (the merged clusters are far from each other and thus should stay separate).



(a) Dataset ($n = 30$ points of $\mathbb{R}^2$)

(b) Clusters at step $t = 10$

(c) Clusters at step $t = 20$

(d) Final clustering

Figure 1: Agglomerative algorithm.

3 Distance between clusters

Any agglomerative algorithm relies on some distance d between subsets of the the n data points (i.e., clusters). Typical distances are:

- **Minimum:** $d(a, b) = \min_{i \in a, j \in b} \|x_i - x_j\|$,
- **Maximum:** $d(a, b) = \max_{i \in a, j \in b} \|x_i - x_j\|$,
- **Average:** $d(a, b) = \frac{1}{|a||b|} \sum_{i \in a, j \in b} \|x_i - x_j\|$,

where $\|\cdot\|$ is an arbitrary norm on $\mathbb{R}^k$ (for instance, the Euclidian norm). Recall that d does not define a metric in general.

A key property of these three distances is that the sequence of distances formed by the successive merges is non-decreasing, as shown by the dendrogram of Figure 2 (two clusters are necessarity merged at a higher level than their own levels in the dendrogram).

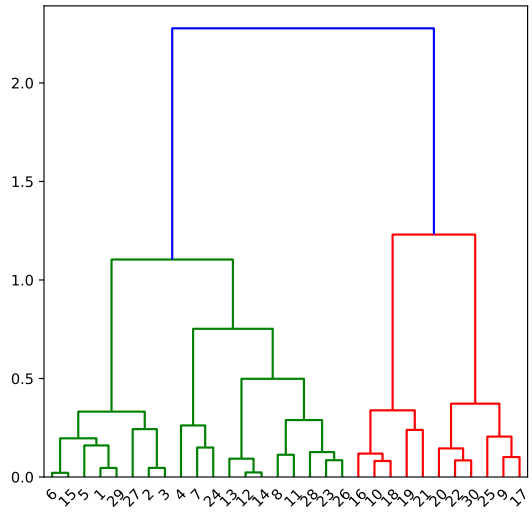


Figure 2: Dendrogram corresponding to the clustering of Figure 1.

This is a consequence of the following inequality. The corresponding distances are called *reducible*.

Proposition 1 *For any disjoint clusters a, b, c,*

$$d(a \cup b, c) \geq \min(d(a, c), d(b, c)). \tag{1}$$

Proof.

- **Minimum:**

$$\begin{aligned} d(a \cup b, c) &= \min_{i \in a \cup b, j \in c} \|x_i - x_j\|, \\ &= \min\left(\min_{i \in a, j \in c} \|x_i - x_j\|, \min_{i \in b, j \in c} \|x_i - x_j\| \right), \\ &= \min(d(a, c), d(b, c)). \end{aligned}$$

- **Maximum:**

$$\begin{aligned} d(a \cup b, c) &= \max_{i \in a \cup b, j \in c} \|x_i - x_j\|, \\ &= \max\left(\max_{i \in a, j \in c} \|x_i - x_j\|, \max_{i \in b, j \in c} \|x_i - x_j\| \right), \\ &\geq \min(d(a, c), d(b, c)). \end{aligned}$$

- **Average:**

$$d(a \cup b, c) = \frac{|a|}{|a| + |b|} d(a, c) + \frac{|b|}{|a| + |b|} d(b, c) \geq \min(d(a, c), d(b, c)).$$

□

Assume that clusters a and b are merged at some step of the algorithm. This means that $d(a, b)$ is the minimum distance between any two clusters. By Proposition 1, we have for any cluster c,

$$d(a \cup b, c) \geq \min(d(a, c), d(b, c)) \geq d(a, b),$$

showing that the sequence of distances between merged clusters is non-decreasing.

4 Ward's method

The most popular distance is known as Ward's distance [Ward, 1963]. It is based on the sum of square errors, as in k-means. For any cluster c, let $g(c)$ be its centroid,

$$g(c) = \frac{1}{|c|} \sum_{i \in c} x_i.$$

and S be the total square Euclidian distance of the points in c to their centroid,

$$S(c) = \sum_{i \in c} \|x_i - g(c)\|^2 = \sum_{i \in c} \|x_i\|^2 - |c| \ \|g(c)\|^2.$$

The *in-cluster variance* of any clustering $\mathcal{C}$ (partition of $\{1, \ldots, n\}$) is

$$V(\mathcal{C}) = \frac{1}{n} \sum_{c \in \mathcal{C}} S(c).$$

For any disjoint clusters a, b, define

$$d(a, b) = S(a \cup b) - S(a) - S(b).$$

Using the fact that

$$g(a \cup b) = \frac{|a| g(a) + |b| \ g(b)}{|a| + |b|},$$

we get

$$\begin{aligned} d(a, b) &= |a| \ \|g(a)\|^2 + |b| \ \|g(b)\|^2 - (|a| + |b|)\|g(a \cup b)\|^2, \\ &= |a| \ \|g(a)\|^2 + |b| \ \|g(b)\|^2 - \frac{1}{|a| + |b|}(|a|^2 \ \|g(a)\|^2 + |b|^2 \ \|g(b)\|^2 + 2|a| \ |b| g(a).g(b)), \\ &= \frac{|a| \ |b|}{|a| + |b|} \|g(a) - g(b)\|^2. \end{aligned}$$

In particular, $d(a, b) \geq 0$. This is the in-cluster variance increase induced by merging a and b. Applying this distance to the agglomerative algorithm minimizes the in-cluster variance at each step of the algorithm (but not necessariy globally). This is known as Ward's method.

Observe that, while Ward's method minimizes the same metric as k-means, the algorithms are very different (see Table 1).

	Ward's algorithm	k-means
Clustering	Hierarchical	Flat
Parameter	–	k
Output	Deterministic	Random

Table 1: Comparison of Ward's algorithm and k-means.

5 Update formula

When two clusters a and b are merged, the distance from $a \cup b$ to any other cluster c needs to be updated. For the considered distances, this follows from $d(a,c), d(b,c)$ and $d(a,b)$:

- **Minimum** (single linkage):
$$d(a \cup b, c) = \min(d(a,c), d(b,c)).$$
- **Maximum** (complete linkage):
$$d(a \cup b, c) = \max(d(a,c), d(b,c)).$$
- **Average** (average linkage):
$$d(a \cup b, c) = \frac{|a|}{|a|+|b|} d(a,c) + \frac{|b|}{|a|+|b|} d(b,c).$$
- **Ward** (minimum variance):
$$d(a \cup b, c) = \frac{(|a|+|b|)|c|}{|a|+|b|+|c|} \|g(a \cup b) - g(c)\|^2.$$

  Since

$$\begin{aligned}
\|g(a \cup b) - g(c)\|^2 &= \left\| \frac{|a| g(a) + |b|\ g(b)}{|a|+|b|} - g(c) \right\|^2, \\
&= \frac{|a|^2}{(|a|+|b|)^2} \|g(a)\|^2 + \frac{|b|^2}{(|a|+|b|)^2} \|g(b)\|^2 + \|g(c)\|^2 \\
&+ 2\frac{|a||b|}{(|a|+|b|)^2} g(a).g(b) - 2\frac{|a|}{|a|+|b|} g(a).g(c) - 2\frac{|b|}{|a|+|b|} g(b).g(c), \\
&= \frac{|a|}{|a|+|b|} \|g(a) - g(c)\|^2 + \frac{|b|}{|a|+|b|} \|g(b) - g(c)\|^2 - \frac{|a||b|}{(|a|+|b|)^2} \|g(a) - g(b)\|^2,
\end{aligned}$$

  we obtain

$$d(a \cup b, c) = \frac{|a|+|c|}{|a|+|b|+|c|} d(a,c) + \frac{|b|+|c|}{|a|+|b|+|c|} d(b,c) - \frac{|c|}{|a|+|b|+|c|} d(a,b). \tag{2}$$

6 Nearest-neighbor chain

Finding the two closest clusters at each step of the agglomerative algorithm requires $O(n^2)$ operations for n clusters, hence an overall complexity in $O(n^3)$. When the distance is reducible, it is possible to reduce the complexity of the algorithm to $O(n^2)$ on observing that any clusters a, b that are nearest from each other, in the sense that

$$d(a,b) = \min_c d(a,c) = \min_c d(b,c), \tag{3}$$

can be merged. In particular, it is not necessary to merge those clusters a, b that attain the global minimum of $d(a,b)$; a *local* minimum is sufficient. Indeed, any clusters a, b that satisfy (3) can be merged at any step of the algorithm, because, in view of (1), any other merge will not change the fact that a, b are nearest from each other.

The Ward distance satisfies a weak form of reducibility in the sense that

$$d(a,b) \leq \min(d(a,c), d(b,c)) \quad \Longrightarrow \quad d(a \cup b, c) \geq \min(d(a,c), d(b,c)). \tag{4}$$

This property is sufficient as two clusters a, b can be merged only if $d(a,b) \leq \min(d(a,c), d(b,c))$ for any other cluster c. The proof of (4) follows from the update formula (2).

Given some clustering $\mathcal{C}$ with at least two clusters, an efficient way to find two clusters that are nearest from each other is to build the *nearest-neighbor chain* from an arbitrary cluster $c \in \mathcal{C}$. The first element of the chain is the nearest neighbor a_1 of c in $\mathcal{C} \setminus \{c\}$; the second element of the chain is the nearest neighbor a_2 of a_1 in $\mathcal{C} \setminus \{a_1\}$; if $a_2 = c$, the chain is complete and a_1, c are nearest neighbors from each other; otherwise, a third element is added to the chain, and so on until two clusters are nearest neighbors. The algorithm stops provided ties are broken with a fixed, pre-defined rule; for instance, if $d(a, b) = d(a, c)$ then decide that b is nearest from a than c if and only if $\min b < \min c$ (recall that b and c are subsets of $\{1, \ldots, n\}$). Then the chain cannot form any triangle, which guarantees that the algorithm stops in finite time and outputs two nearest neighbors.

The algorithm consists in merging recursively clusters appearing at the end of the chain. A pseudo-code of the algorithm is the following:

Algorithm 1: Nearest-neighbor chain

```
Input: c (initial cluster)
Output: Sequence of cluster merges
S ← empty stack
S.push(c)
while S is not empty do
    a ← S.pop
    b ← nearest neighbor of a
    if b is in S then
        b ← S.pop
        merge a,b
    else
        S.push(a)
        S.push(b)
```

When the chain is exhausted, a new chain is started whenever there are at least two clusters left. After each merge, the distances are updated by the formulas of Section 5.

Further reading

- Hiearchical clustering [Murtagh and Contreras, 2012]
- Divisive algorithms [Newman and Girvan, 2004]
- Link with random walks [Lambiotte et al., 2014]

References

[Lambiotte et al., 2014] Lambiotte, R., Delvenne, J.-C., and Barahona, M. (2014). Random walks, markov processes and the multiscale modular organization of complex networks. *IEEE Transactions on Network Science and Engineering*.

[Murtagh and Contreras, 2012] Murtagh, F. and Contreras, P. (2012). Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*.

[Newman and Girvan, 2004] Newman, M. E. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical review E*.

[Ward, 1963] Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*.