Hierarchical Clustering

Zeqing Jin
zjin2017@berkeley.edu

Yifei Zhang
yifei_zhang@berkeley.edu

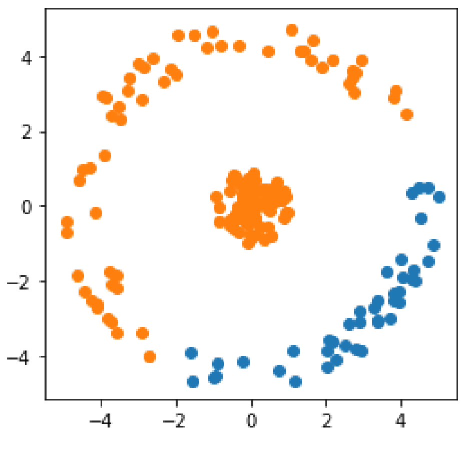
Zilan Zhang
shilan@berkeley.edu

Xianlin Shao
shayd@berkeley.edu

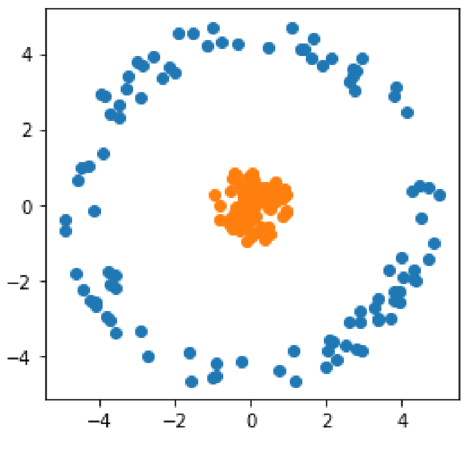
1 Introduction

In this chapter, we will introduce another clustering algorithm in machine learning, hierarchical clustering. Similar to k-means clustering, hierarchical clustering also uses similarity as a general metric. However, unlike an initially defined number of clusters in k-means clustering, the number of clusters in hierarchical clustering is always changing along the process. Besides, hierarchical clustering can perform well on certain specific cases that k-means clustering fails.

Let's look at an example. The ground truth implies two clusters. One has points concentrated in the center and the other has points surrounding the center. By setting $K = 2$ and using Lloyd's algorithm, K-means clustering ends up with results shown in Figure 1a. In comparison, Figure 1b gives the results of hierarchical clustering with single linkage.



(a)



(b)

Figure 1: (a).K-means clustering. (b) Hierarchical clustering (single-linkage).

Intuitively one should visually cluster the data the same as what the hierarchical clustering does, so is there anything wrong with the k-means algorithm? No. The algorithm did find out the global minimum of the loss function. However, as you already know in previous lectures, what k-means optimizes is the distance between the data points and centroids rather than the distance between each point. Therefore, chances are that a point is assigned to one cluster, while it is visually closer to points in the other cluster. When the underlying clusters are non-spherical or there is a prior assumption of similar number of data points in each cluster, k-means algorithm may also fail. When k-means clustering cannot give a satisfying result, we may have a try with hierarchical clustering. One thing to note is that for hierarchical clustering method, clustering results will heavily depends on the linkage function. While we choose another linkage function, results similar to Figure 1a may also occur. We will discuss this in more detail in following sections and coding assignment.

2 Dendrogram

In hierarchical clustering (HC), we construct a tree-like hierarchy which shows the relationship between objects. This is the so called dendrogram. On one end of the dendrogram, all the n objects are listed as n separate clusters. On the other end, there is only one single cluster containing all n objects [1].

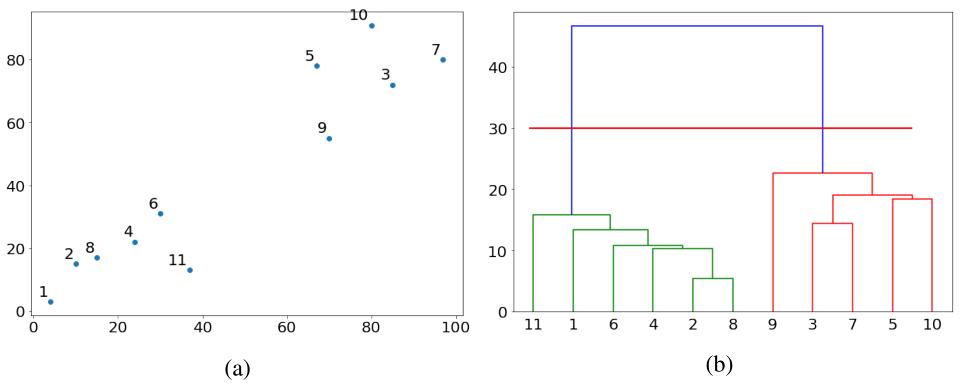


Figure 2: (a).Scatter plot of 11 sample points. (b) Corresponding dendrogram of the data.

A simple example is shown in Figure 2a. We have $n = 11$ points in a 2d Euclidean space, and the corresponding dendrogram is derived in Figure 2b. Each vertical line can be regarded as one cluster. The key is to focus on the position where two vertical lines merge together, which means two clusters are combined into one. If we take a horizontal cut through dendrogram, the number of intersections between the cut and the vertical lines indicates the number of clusters at that stage. Therefore, dendrogram implicitly contains all possible values of k, the number of clusters.

Another important information that the dendrogram provides is the relation between different points. The bottom end shows all the points, however, not in a normal sequential order. Instead, the points that are closest to each other are placed together, merging represented with the lowest height. For example, point 2 and 8 are merged into one cluster with the lowest height, indicating they are closest to each other among all the points. On the other hand, point 9 and 11 do not merge until they reach the top of the dendrogram, which means the they are super dissimilar. The higher the merge occurs, the more dissimilar two clusters are.

However, dendrogram fails to reveal the absolute distance relation between individual objects. If we look at point 1 and 9, since they merge at the top, they are dissimilar. As we have just mentioned above, point 9 and 11 also merge at the top. We may tell from Figure 2a that point 9 is closer to 11 than 1, while from Figure 2b, we are losing this information. Such comparison is important in the underlying HC algorithm, and the dendrogram omits it. Therefore, dendrogram can be a good outcome showing the merging sequence of HC process, but does not show how HC works.

3 Agglomerative clustering

Agglomerative clustering (HAC) works in a bottom-top manner [2]. Initially each object is considered as a single-element cluster, i.e., the bottom of dendrogram. In each step, two clusters that are the most similar are merged into one cluster. This procedure does not stop until all the objects are included in one single cluster, i.e., the top of dendrogram (Figure 3).

```
Algorithm 1: Agglomerative Hierarchical Clustering
Input: $n$ data points
Output: final clustering result over $n$ data points
1 Initialize $n$ clusters $\mathbf{c_i}, i = 1, ..., n$;
2 Initialize the dissimilarity matrix;
3 for the number of clusters $k$ decreases from $n$ to 1 do
4     Find the two clusters $\mathbf{c_i}, \mathbf{c_j}$ with the smallest dissimilarity according to dissimilarity matrix;
5     Merge $\mathbf{c_i}$ with $\mathbf{c_j}$ and update the dissimilarity matrix;
6 end for
```

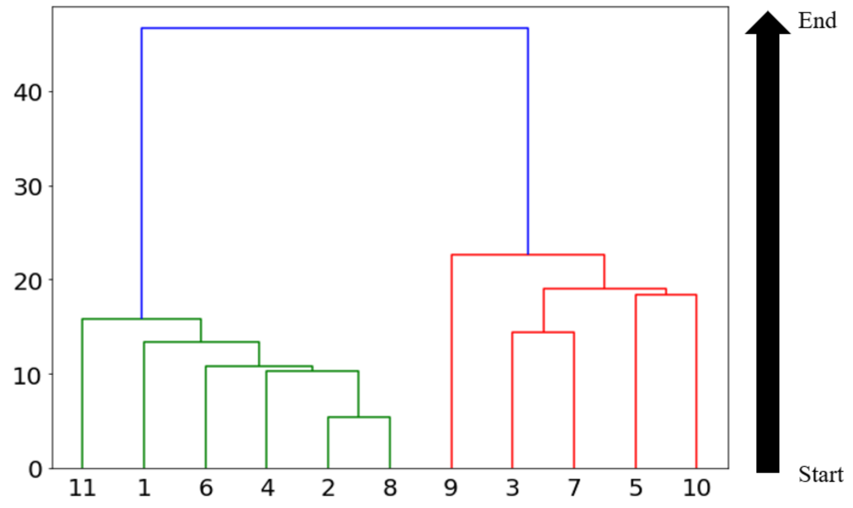


Figure 3: Agglomerative clustering algorithm direction

To get a closer understanding of how agglomerative clustering works, we first analyze a dataset with $n = 9$ points. Initially, each point forms one cluster with a different color (Figure 4a). The distance between each point is calculated using Euclidean distance

$$d(i,j) = \sqrt{\sum_{p=1}^{q}(x_{ip} - x_{jp})^2}$$

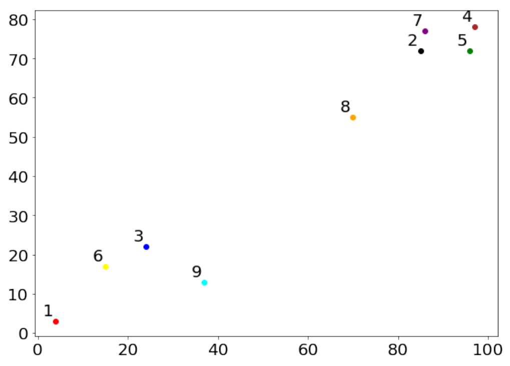
where q is the dimension of the object and equals 2 in this case. An initial dissimilarity matrix is constructed according to the distance between each points.

$$\mathbf{S} = \begin{bmatrix} d(1,1) & ... & d(1,n) \\ . & & . \\ . & & . \\ . & & . \\ d(n,1) & ... & d(n,n) \end{bmatrix}$$

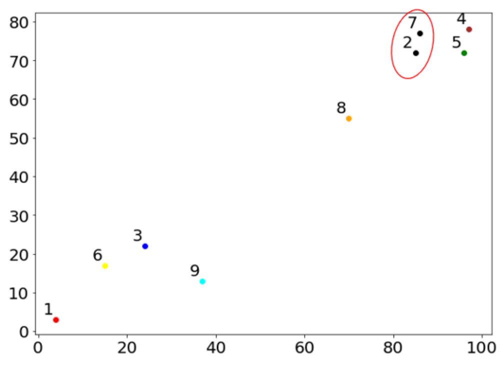
To find out the two closest points, we look for the entry with the smallest value $d_{\min}$ in the dissimilarity matrix. Actually you may notice that the matrix is symmetric and actually only $n(n-1)/2$ distances need to be compared. In this example, the distance between point 2 and 7 is the smallest, i.e., $d_{\min} = S_{27} = S_{72}$, so we merge point 2 and 7 into one cluster {2,7} (Figure 4b).

Now the updated number of clusters is 8, and the distance between these eight clusters are calculated and the dissimilarity matrix will also be updated. At this point, one practical question arises, how to define the distance between clusters that have multiple points?

For two clusters A and B with m and n objects, some common methods of calculating the distance or dissimilarity between them are listed below [1]:



(a)



(b)

Figure 4: (a).Initial separate clusters. (b).1st iteration combining point 2 and 7.

1. **Maximum or complete linkage clustering:** Calculate the largest value of dissimilarity between all the points in the two clusters.

$$d(A,B) = \max \quad d(A_i, B_j) \quad i = 1, ..., m \quad j = 1, ..., n$$

2. **Minimum or single linkage clustering:** Calculate the smallest value of dissimilarity between all the points in the two clusters.

$$d(A,B) = \min \quad d(A_i, B_j) \quad i = 1, ..., m \quad j = 1, ..., n$$

3. **Mean or average linkage clustering:** Computes all pairwise dissimilarities between the elements in cluster A and the elements in cluster B, and considers the average of these dissimilarities as the distance between the two clusters.

$$d(A,B) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} d(A_i, B_j)$$

4. **Centroid linkage clustering:** Calculate the dissimilarity between the centroids for the two clusters.

$$d(A,B) = d(C_A, C_B)$$

$$C_A = \frac{1}{m} \sum_{i=1}^{m} A_i$$

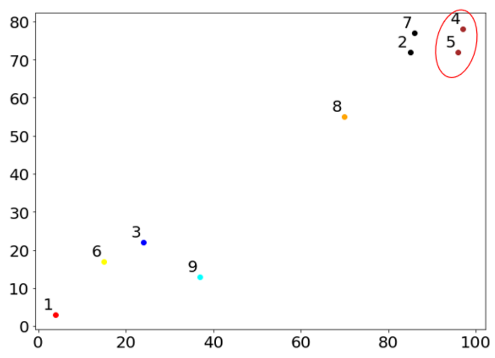
$$C_B = \frac{1}{n} \sum_{j=1}^{n} B_j$$

5. **Ward's minimum variance method:** calculate how much the sum of squares will increase when two clusters are merged [3].

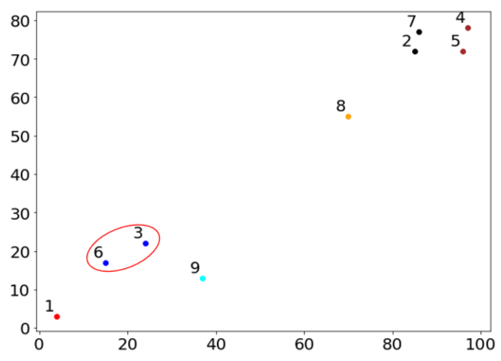
$$\begin{aligned} d^2(A,B) &= \sum_{x_i \in A \cup B} \parallel x_i - C_{A \cup B} \parallel^2 - \sum_{x_i \in A} \parallel x_i - C_A \parallel^2 - \sum_{x_i \in B} \parallel x_i - C_B \parallel^2 \\ &= \frac{mn}{m+n} d^2(C_A, C_B) \end{aligned}$$

$$C_A = \frac{1}{m} \sum_{i=1}^{m} A_i$$

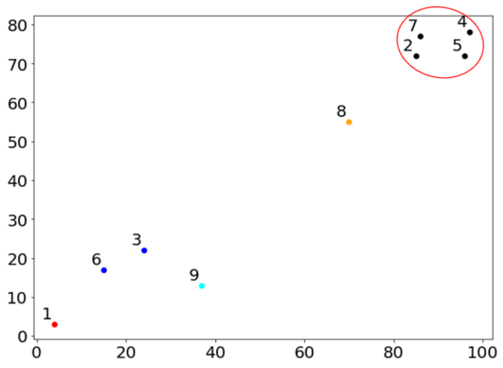
$$C_B = \frac{1}{n} \sum_{j=1}^{n} B_j$$



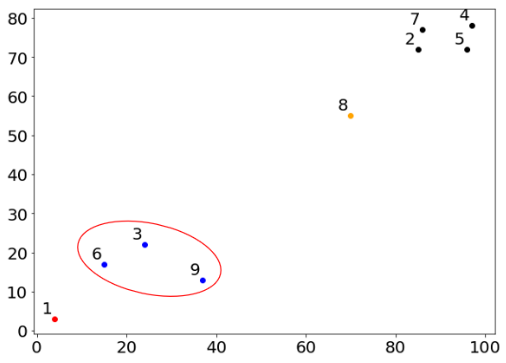
(a)



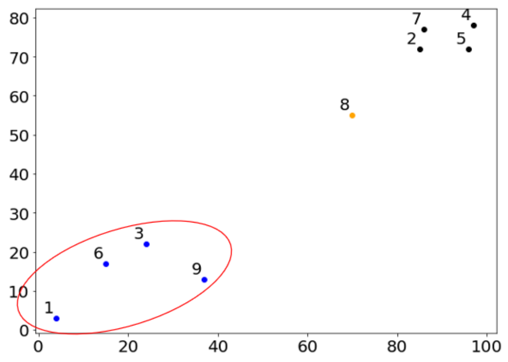
(b)



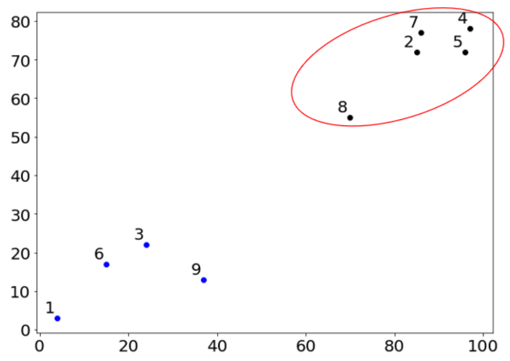
(c)



(d)



(e)



(f)

Figure 5: Iteration procedure showing the order of merge of points.

Later we will discuss the properties of different methods. Now we will continue with the example and use single linkage clustering as the method. The next iteration merges point 4 and 5 which give the smallest distance, into $\{4,5\}$ (Figure 5a).

We may also have a look at how the dissimilarity matrix is updated. In the first iteration, point 2 and 7 are merged, so we may move the 2nd and 7th rows and columns of **S** and replace them with a new row and column and the matrix becomes 8 by 8. Actually, the remaining entries do not change, since the distances between unmoved points do not change. We only need to update the entries in the new formed row and column, i.e., the distances between the new merged cluster $\{2,7\}$ and all the other points, which are calculated from single linkage. Again the minimum value is chosen in the new dissimilarity matrix and we merge the corresponding clusters representing the row and column of that entry.

The final iteration is reached when there are only two clusters $\{1,3,6,9\}$ and $\{2,4,5,7,8\}$ as is shown in Figure 5f. If we go on with the final merge, all the objects forms one single cluster (Figure 6a), and that ends the agglomerative clustering algorithm.

Figure 6b is the dendrogram of the previous example. As we have already mentioned, in a dendrogram, the higher the merge occurs, the more dissimilar two clusters are. Therefore, the values on the y axis are the magnitudes of dissimilarity between different clusters. Also, during the whole procedure of agglomerative clustering, the minimum dissimilarity in each iteration form a nondecreasing sequence. This actually matches the direction of agglomerative clustering algorithm shown in Figure 3.

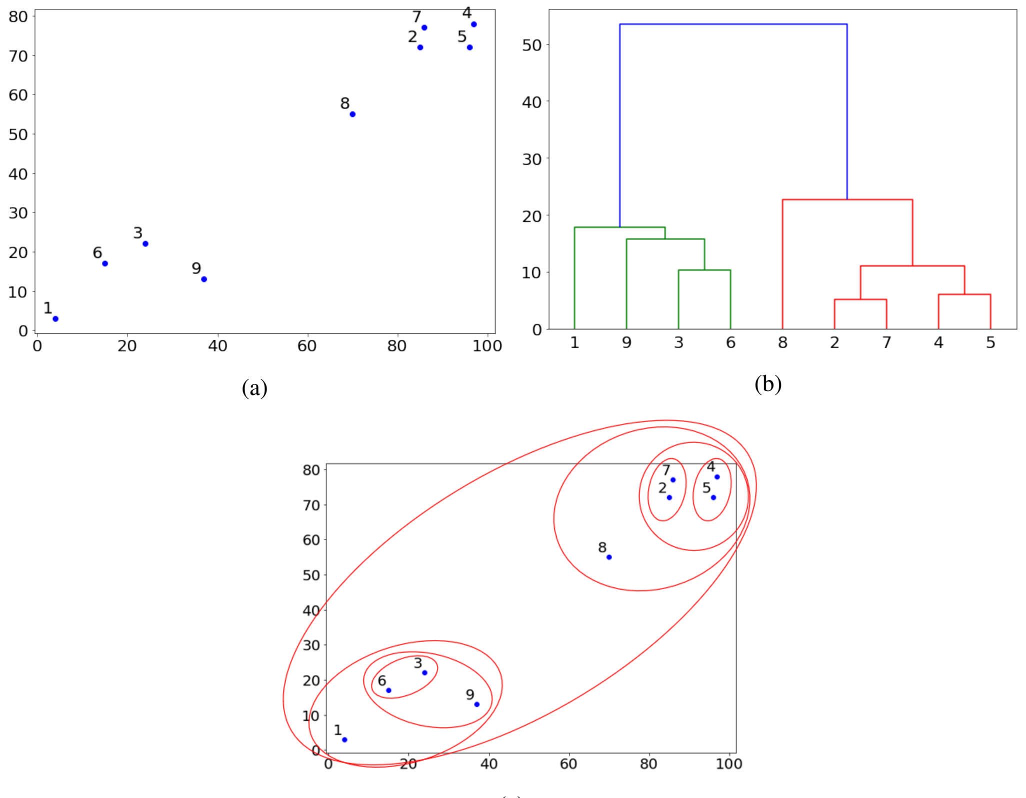


Figure 6: (a).Completed clustering result of HAC. (b).Dendrogram of HAC.(c).Alternative representation of HAC (also called agglomerative nesting, AGNES) algorithm.

4 Divisive hierarchical clustering

The other common method in hierarchical clustering is known as divisive hierarchical clustering, among which the most traditional approach is called divisive analysis clustering (DIANA) [2]. The major difference between DIANA and agglomerative clustering is that it proceeds in an inverse order (Figure 7). DIANA starts from a single cluster containing all the objects and finally ends up with clusters containing only one single element [1].

Intuitively, the 1st step of DIANA algorithm can be complicated, since all possible divisions should be considered before splitting up the whole data set. That means if we have a data set of n objects, the number of possibilities of binary division of data set is $2^{n-1} - 1$. For a large n, this is non practical because of the exponential time complexity [1].

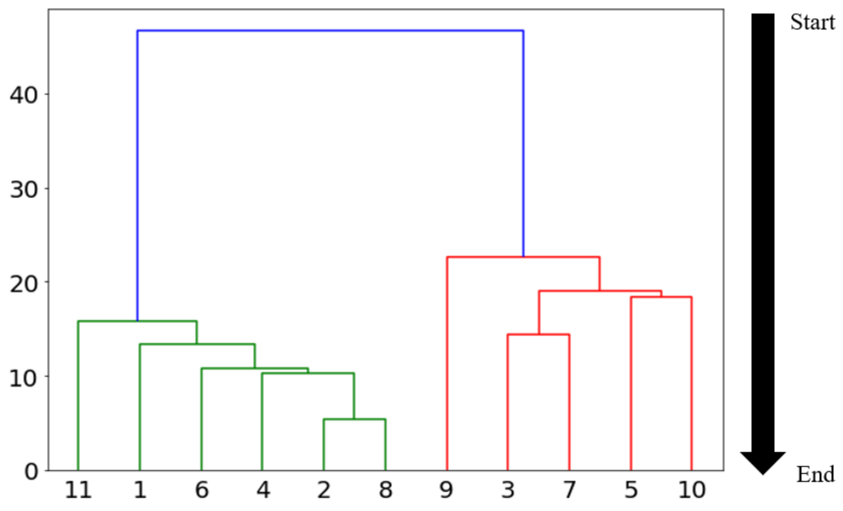


Figure 7: DIANA algorithm direction.

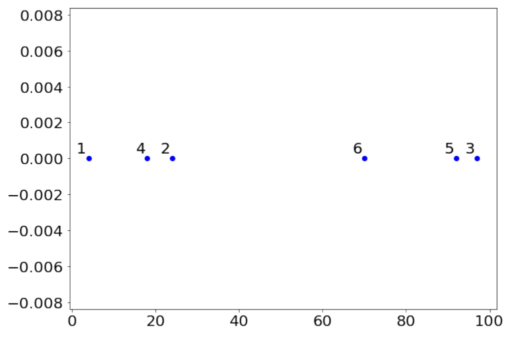
However, in DIANA, instead of considering all division possibilities, we are more likely to use an iterative procedure, summarized from Kaufman's book [1]:

Algorithm 2: Divisive Analysis Clustering (DIANA)

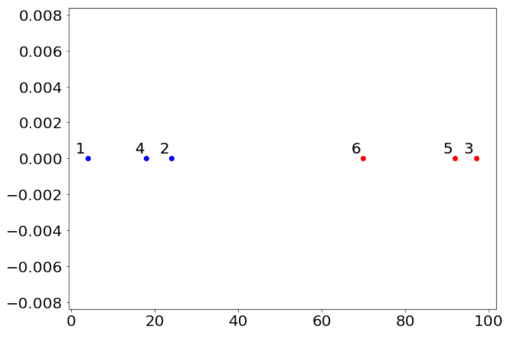
```
Input: $n$ data points
Output: final clustering result
1  Initialize one cluster with all objects $\mathbf{c}_1$;
2  for the number of clusters k increases from 1 to n do
3      Choose the cluster $\mathbf{C}_i$ with the largest diameter value;
4      Within $\mathbf{C}_i$, choose the object that has the maximum distance with the other objects as one
         cluster and split this object as a splinter cluster;
5      Update $\mathbf{C}_i$;
6      while True do
7          for each data point j in $\mathbf{C}_i$ do
8              Calculate the distance $d_1$ between the data $j$ and the other objects in $\mathbf{C}_i$ as one
                 cluster;
9              Calculate the distance $d_2$ between the data $j$ and the splinter cluster;
10             Calculate the difference $\delta d_j = d_1 - d_2$;
11         end for
12         if max $\delta d_j$ is positive then
13             Move the data $j$ with positive $\delta d_j$ to the splinter cluster and update $\mathbf{C}_i$;
14         else
15             break;
16         end if
17     end while
18 end for
```

The algorithm looks more complex than agglomerative clustering. Again, we will look at a 1D example with 6 points (Figure 8a). Initially all the points are in one cluster, followed by the first step of dividing it in to two most dissimilar clusters. We may first come up with the dissimilarity matrix:

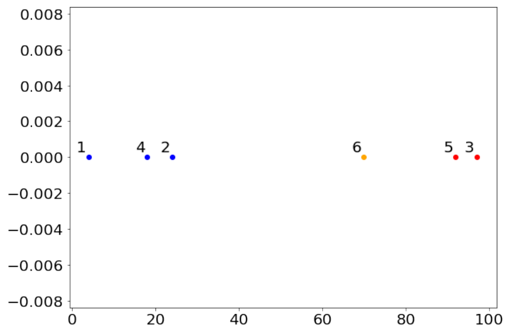
$$\mathbf{S} = \begin{bmatrix} 0 & 20 & 93 & 14 & 88 & 66 \\ 20 & 0 & 73 & 6 & 68 & 46 \\ 93 & 73 & 0 & 79 & 5 & 27 \\ 14 & 6 & 79 & 0 & 74 & 52 \\ 88 & 68 & 5 & 74 & 0 & 22 \\ 66 & 46 & 27 & 52 & 22 & 0 \end{bmatrix}$$



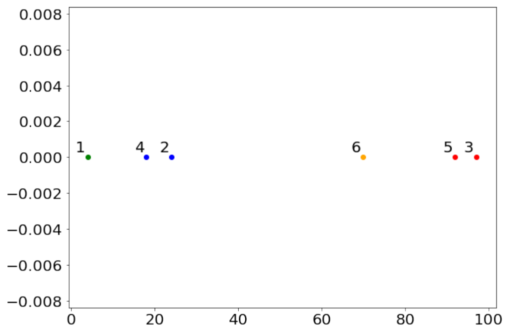
(a)



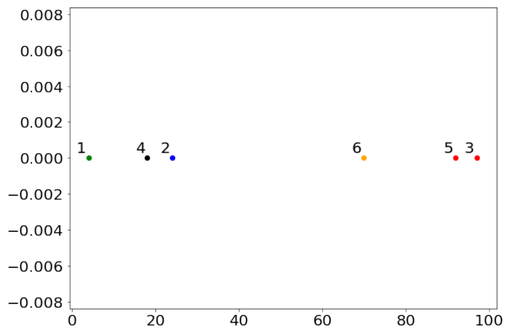
(b)



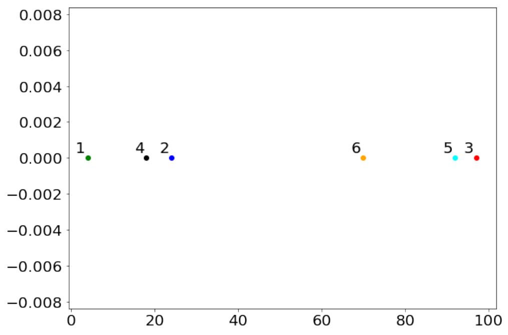
(c)



(d)



(e)



(f)

Figure 8: DIANA example, from one cluster to n clusters.

For each point, we calculate its distance to the other objects, using complete linkage:

Point	Maximum distance to other objects
1	93
2	73
3	93
4	79
5	88
6	66

We will pick either point 1 or 3 as a splinter cluster, say point 1. Next, for each of the remaining points, we calculate the distance to other points and to the splinter cluster:

Point	Distance to other objects	Distance to the splinter cluster	Difference
2	73	20	53
3	79	93	-14
4	79	14	65
5	74	88	-14
6	52	66	-14

The maximum positive difference occurs at point 4, so we move point 4 to the splinter cluster, and repeat the previous step, with four remaining points. Again, point 2 is moved to the splinter cluster.

Point	Distance to other objects	Distance to the splinter cluster	Difference
2	73	20	53
3	73	93	-20
5	68	88	-20
6	46	66	-20

If we further analyze the remaining point 3,5 and 6, they provide negative difference, which means compared to the splinter cluster, they are closer to each other. We may stop here and end the first division $\{1,2,4\}$ and $\{3,5,6\}$. Actually from Figure 8b, it is reasonable that the six points are first divided into three points on the left and three on the right.

To determine which group is split next, the diameter of each cluster $\mathbf{C}_i$ is calculated:

$$r_i = \max \quad d(x_j, x_k), \quad x_j, x_k \in \mathbf{C}_i$$

Here the cluster $\{3,5,6\}$ has larger diameter and will be divided in the next step. Note that this is not determined by number of objects in each cluster. Following the same procedure as before, it is divided into two clusters $\{3,5\}$ and $\{6\}$. Now, point 6 forms one single cluster and will not be considered any more.

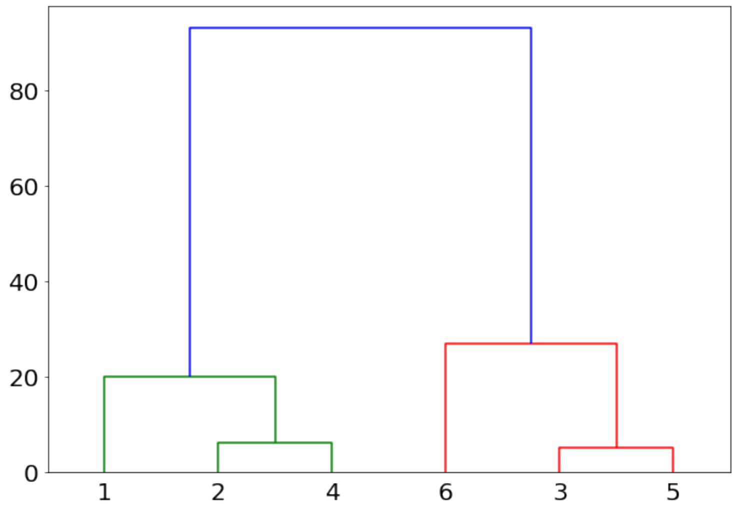


Figure 9: DIANA dendrogram.

Similar to agglomerative clustering, by plotting the dendrogram, the height of each division reveals the order of iteration. In divisive hierarchical clustering, the earlier the division occurs, the more dissimilar two clusters are.

5 Determining the number of clusters

In the previous section, we have gone through the whole process of HAC, starting from n clusters and ending with one cluster. However, for a clustering problem, we always need to figure out how many clusters to end up with [2].

For partition clustering algorithms like k-means, we will run the algorithm with different k for several times and then tipically use elbow method to determine the best k.

In comparison, as for hierarchical clustering, the existence of dendrogram gives us two more ways to determine the number of clusters. Besides, in this case, you no longer need to run the algorithm for more than two times.

5.1 Cut at certain dissimilarity level

One may roughly determine the number of clusters by introducing cuts on dendrogram. Figure 10 shows various cuts on the tree diagram, resulting in different number of clusters of the example in Section 3. If you think we should no longer continue merging clusters with dissimilarity larger than 20, we can cut at line 2 and get the resulting 3 clusters.

5.2 Cut at the largest dissimilarity gap

Traditionally, the optimal number of k is obtained by cutting the dendrogram at the largest dissimilarity gap. As for Figure 10, the largest dissimilarity gap is the one between [22, 54]. The resulting best k value is 2.

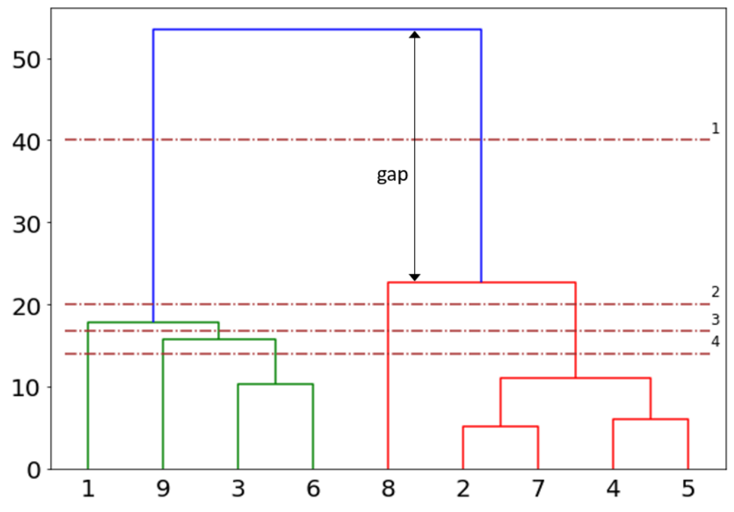


Figure 10: Cuts on dendrogram.

6 Specific hierarchical algorithms (Optional)

Here we will introduce some useful specific hierarchical algorithms.

CURE (Clustering Using REpresentatives)

CURE is a hierarchical based clustering technique, which has several advantages over other clustering algorithms. When data is given to CURE, it first draws the random sample and use agglomerative clustering to partially cluster the points. In each cluster, several (not all) points are chosen as the representative points, which is achieved by centroid or single linkage approaches. A shrinking factor would slight move the representative points towards the centroid of each cluster, eliminating outliers. Finally, all the other data points are clustered to the nearest group formed by the representative points. The whole process can be represented by the flow chart in Figure 11 [4].

CURE is able to recognize arbitrarily shaped clusters such as ellipsoidal and effectively eliminate the effect of outliers. In terms of complexity, CURE is similar to agglomerative clustering and DIANA, between $\mathcal{O}(n^2 \log n)$ and $\mathcal{O}(n^2)$. Also, it can handle large data sets. [4]

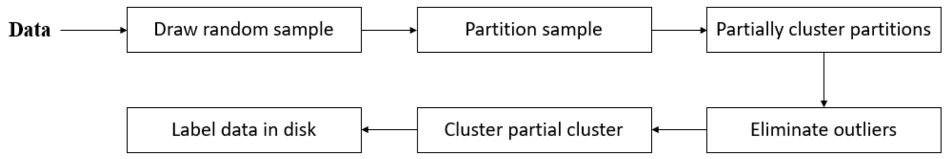


Figure 11: CURE Process

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) (Optional)

BIRCH is an agglomerative hierarchical clustering algorithm which works especially well for large data sets. It summarizes large data sets into smaller regions known as Clustering Feature (CF) entries, which is defined by an ordered triple, (N, LS, SS), number of points, linear sum of the points and squared sum of the points. Then BIRCH constructs a clustering feature tree (CF Tree) incrementally, whose leaf node contains a sub-cluster. Every entry of CF tree points to a child node and is made up of the sum of entries in the child nodes. [5]

As the sub-clusters are being constructed, the nodes in CF Tree keep computed and updated, resulting an ”in-memory” summary of the data [4]. The greatest advantage of BIRCH is its low complexity of $\mathcal{O}(n)$, since it was originally designed to minimize the number of I/O operations . Nevertheless, BIRCH can only be applied to numeric data and is sensitive to the data recording order [4].

Linkage algorithms

As we have seen in Section 3, when determining the distance between two clusters, there are multiple metrics. Linkage algorithms are commonly used in HAC. Let's revisit the three important algorithms, single linkage, complete linkage and average linkage.

Single linkage gives the shortest distance between two clusters, which is also known as the nearest neighbor method. The simplest implementation without using priority queue or union find has time complexity of $\mathcal{O}(n^3)$ and space complexity $\mathcal{O}(n^2)$. Union find can reduce the time complexity of single linkage to $\mathcal{O}(n^2)$. Single linkage can be sensitive to outliers. Also it is unsuitable when dealing with severe differences in the density of clusters [4].

Complete linkage is the opposite of single linkage. It is also referred to as the furthest neighbor method. However, it may cluster similar objects into different clusters, which is said to be *space dilating* [1].

The compromise between single and complete linkage leads to average linkage [1]. However, average linkage is sensitive to the shape and size of the clusters. For complicated shaped clusters, average linkage can easily fail [4].

References

[1] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Vol. 344. John Wiley & Sons, 2009.

[2] Bradley Boehmke Brandon Greenwell. *Hands-On Machine Learning with R*. Feb. 2020. URL: `https://bradleyboehmke.github.io/HOML/hierarchical.html#fig:dendrogram2`.

[3] Cosma Shalizi. "Distances between clustering, hierarchical clustering". In: *Lectures notes* (2009).

[4] M Kuchaki Rafsanjani, Z Asghari Varzaneh, and N Emami Chukanlo. "A survey of hierarchical clustering algorithms". In: *The Journal of Mathematics and Computer Science* 5.3 (2012), pp. 229–240.

[5] *ML: BIRCH Clustering*. July 2020. URL: `https://www.geeksforgeeks.org/ml-birch-clustering/`.