2.2. Hierarchical Clustering

In this section, we're going to discuss a particular clustering technique called hierarchical clustering. Instead of working with the relationships existing in the whole dataset, this approach starts with a single entity containing all elements (divisive) or N separate elements (agglomerative), and proceeds by splitting or merging the clusters according to some specific criteria, which we're going to analyze and compare.

Hierarchical strategies

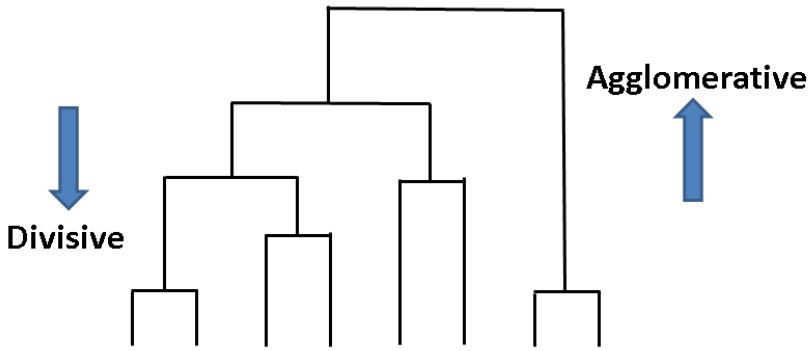
Hierarchical clustering means creating a tree of clusters by iteratively grouping or separating data points. There are two types of hierarchical clustering:

Hierarchical clustering is based on the general concept of finding a hierarchy of partial clusters, built using either a bottom-up or a top-down approach. More formally, they are called:

- **Agglomerative clustering:** The process starts from the bottom (each initial cluster

is made up of a single element) and proceeds by merging the clusters until a stop criterion is reached. In general, the target has a sufficiently small number of clusters at the end of the process.

- **Divisive clustering:** In this case, the initial state is a single cluster with all samples and the process proceeds by splitting the intermediate cluster until all elements are separated. At this point, the process continues with an aggregation criterion based on the dissimilarity between elements.



Scikit-learn implements only the agglomerative clustering. However, this is not a real limitation because the complexity of divisive clustering is higher and the performances of agglomerative clustering are quite similar to the ones achieved by the divisive approach.

Agglomerative clustering

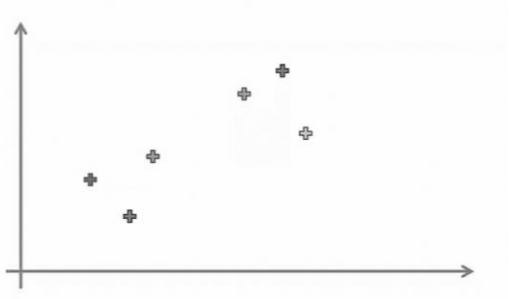
Agglomerative clustering is kind of a bottom-up approach. It starts by treating each observation as a separate cluster. Then, it repeatedly executes the following two steps:

(1) identify the two clusters that are closest together, and

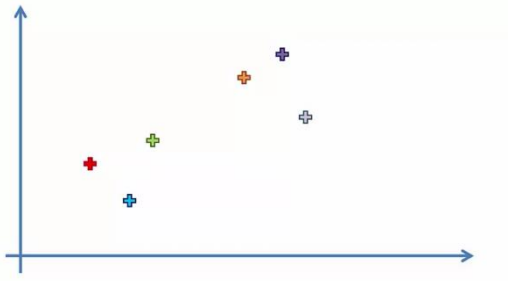
(2) merge the two most similar clusters.

This iterative process continues until all the clusters are merged together.

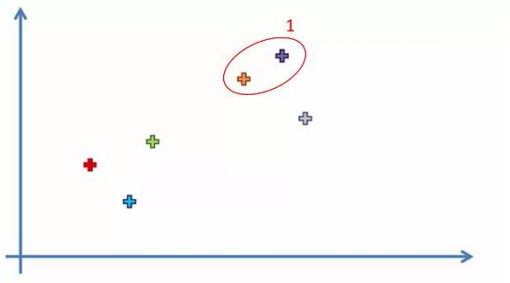
Let's go over an example to explain the concept clearly. Consider the following data points from a 2D dummy dataset (i.e., a data set with 2 features).



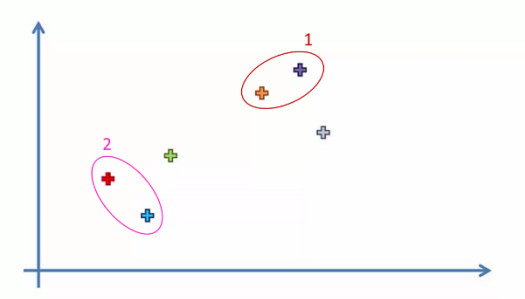
Step 1: Make each data point a cluster



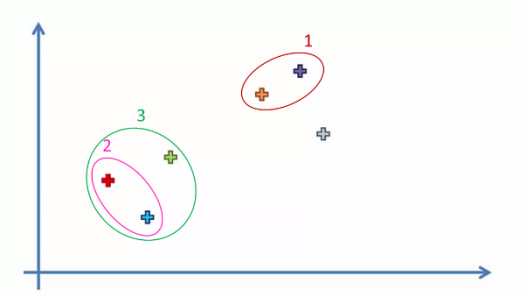
Step 2: Consider the two closest clusters and merge them into one cluster.

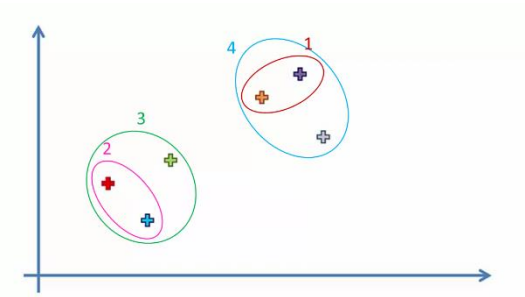


Step 3: Repeat step 2, this time considering 5 clusters.

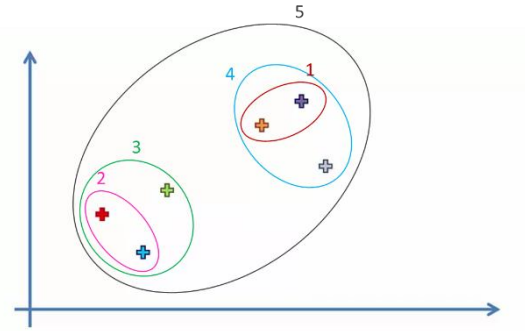


Repeat step 2 until we are left with only one cluster





Finally,



This is a very simple data set to illustrate the purpose but real life data sets are obviously more complex. ***We mention that "closest data points (or clusters)" are combined together. But how do the algorithms identify closest ones?***

In the example above, the distance between two clusters has been computed based on the length of the straight line drawn from one cluster to another. This is commonly referred to as the Euclidean distance. Many other distance metrics have been developed.

We define affinity, a metric function of two arguments with the same dimensionality m. The most common metrics (also supported by scikit-learn) are:

- Euclidean or $L2$:

$$d_{Euclidean}(\bar{x}_1, \bar{x}_2) = \sqrt{\sum_i \left(x_1^i - x_2^i\right)^2}$$

- Manhattan (also known as City Block) or $L2$:

$$d_{Manhattan}(\bar{x}_1, \bar{x}_2) = \sum_i \left|\left(x_1^i - x_2^i\right)\right|$$

- Cosine Distance

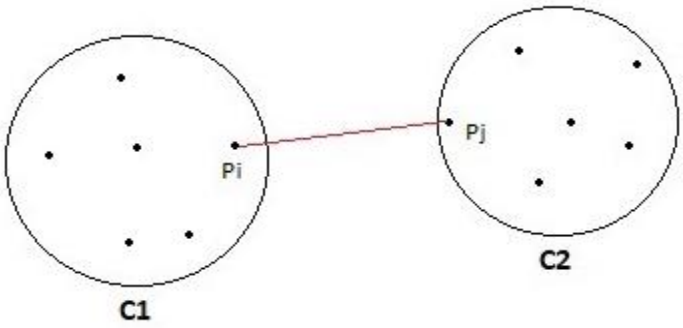
$$d_{Manhattan}(\bar{x}_1, \bar{x}_2) = 1 - \frac{\bar{x}_1 . \bar{x}_2}{\|\bar{x}_1\|_2 \ \|\bar{x}_2\|_2}$$

Once a metric has been chosen (let's simply call it $d(x, y)$), the next step is defining a strategy (called linkage) to aggregate different clusters. There are many possible methods, but scikit-learn supports the three most common ones:

Single-linkage

The distance between two clusters is defined as the shortest distance between two points in each cluster.

Pick the two closest points such that one point lies in cluster one and the other point lies in cluster 2 and takes their similarity and declares it as the similarity between two clusters.



Single-linkage algorithm also known as MIN can be defined as the similarity of two clusters $C1$ and $C2$ is equal to the minimum of the similarity between points Pi and Pj such that Pi belongs to $C1$ and Pj belongs to $C2$.

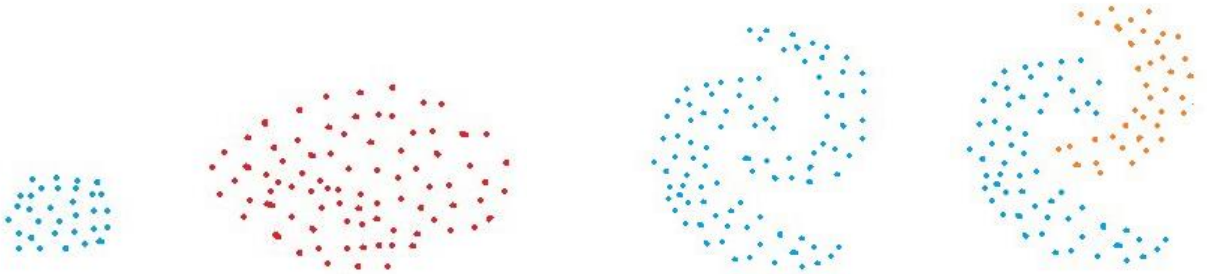
Mathematically this can be written as,

$$Sim(C1, C2) = Min.Sim(Pi, Pj) \quad such\ that \quad Pi \in C1 \quad \& \quad Pj \in C2$$

This linkage may be used to detect high values in your dataset which may be outliers as they will be merged at the end.

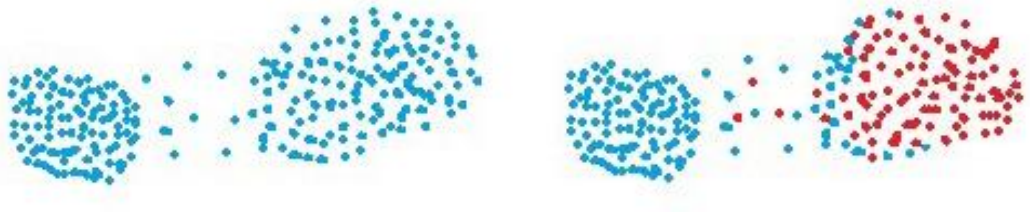
Advantages:

- This approach can separate non-elliptical shapes as long as the gap between the two clusters is not small.



Disadvantages:

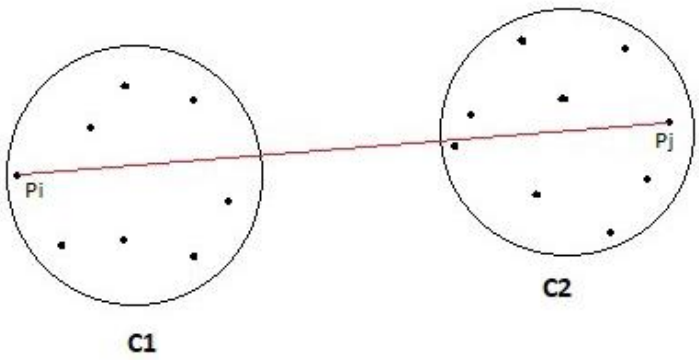
MIN approach cannot separate clusters properly if there is noise between clusters.



Complete-Linkage

The distance between two clusters is defined as the longest distance between two points in each cluster.

Pick the two farthest points such that one point lies in cluster one and the other point lies in cluster 2 and takes their similarity and declares it as the similarity between two clusters.



The Complete Linkage algorithm also known as MAX, is exactly opposite to the MIN approach. The similarity of two clusters $C1$ and $C2$ is equal to the maximum of the similarity between points Pi and Pj such that Pi belongs to $C1$ and Pj belongs to $C2$.

Mathematically this can be written as,

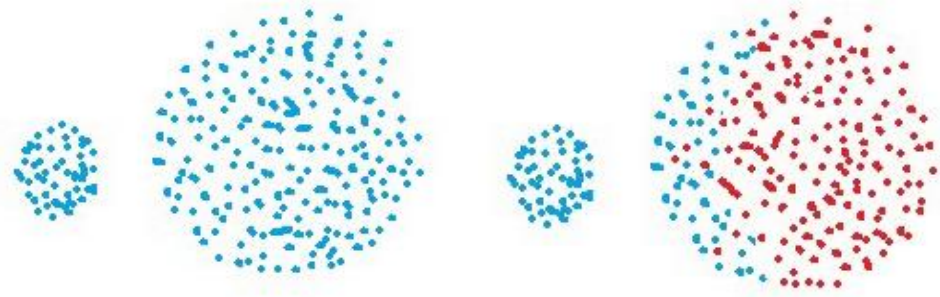
$$Sim(C1, C2) = Max\ Sim(Pi, Pj) \quad such\ that \quad Pi \in C1 \quad \& \quad Pj \in C2$$

Advantage:

MAX approach does well in separating clusters if there is noise between clusters.

Disadvantages:

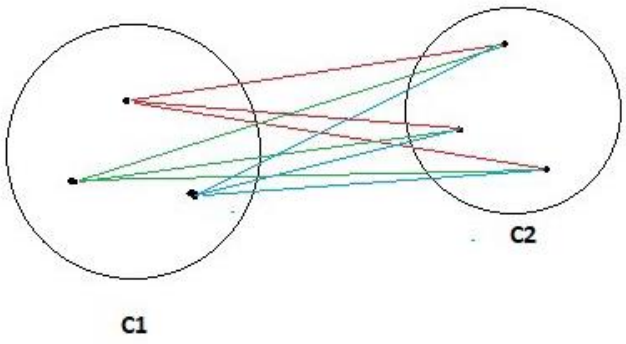
Max approach is biased towards globular clusters. Also, Max approach tends to break large clusters.



Average-Linkage

The distance between two clusters is defined as the average distance between each point in one cluster to every point in the other cluster.

Take all the pairs of points and compute their similarities and calculate the average of the similarities.



Mathematically this can be written as,

$$\frac{sim(C1, C2) = \sum sim(Pi, Pj)}{|C1| * |C2|} \qquad where, Pi \in C1 \,\&\, Pj \in C2$$

Advantage: The group Average approach does well in separating clusters if there is noise between clusters.

Disadvantage: The group Average approach is biased towards globular clusters.

Ward's Method

This approach of calculating the similarity between two clusters is exactly the same as Group Average except that Ward's method calculates the sum of the square of the distances Pi and PJ.

Mathematically this can be written as,

$$\frac{sim(C1, C2) = \sum \left(dist(Pi, Pj)\right)^2}{|C1| * |C2|} \qquad where, Pi \in C1 \,\&\, Pj \in C2$$

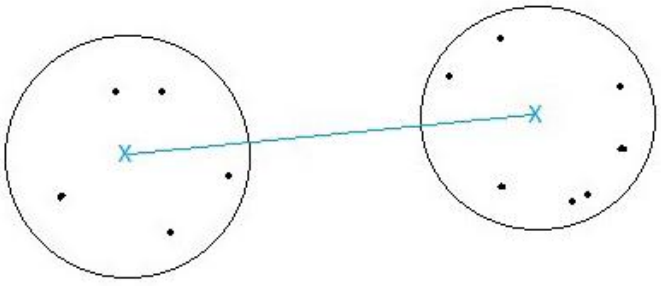
Advantage: Ward's method approach also does well in separating clusters if there is noise between clusters.

Disadvantage: Ward's method approach is also biased towards globular clusters.

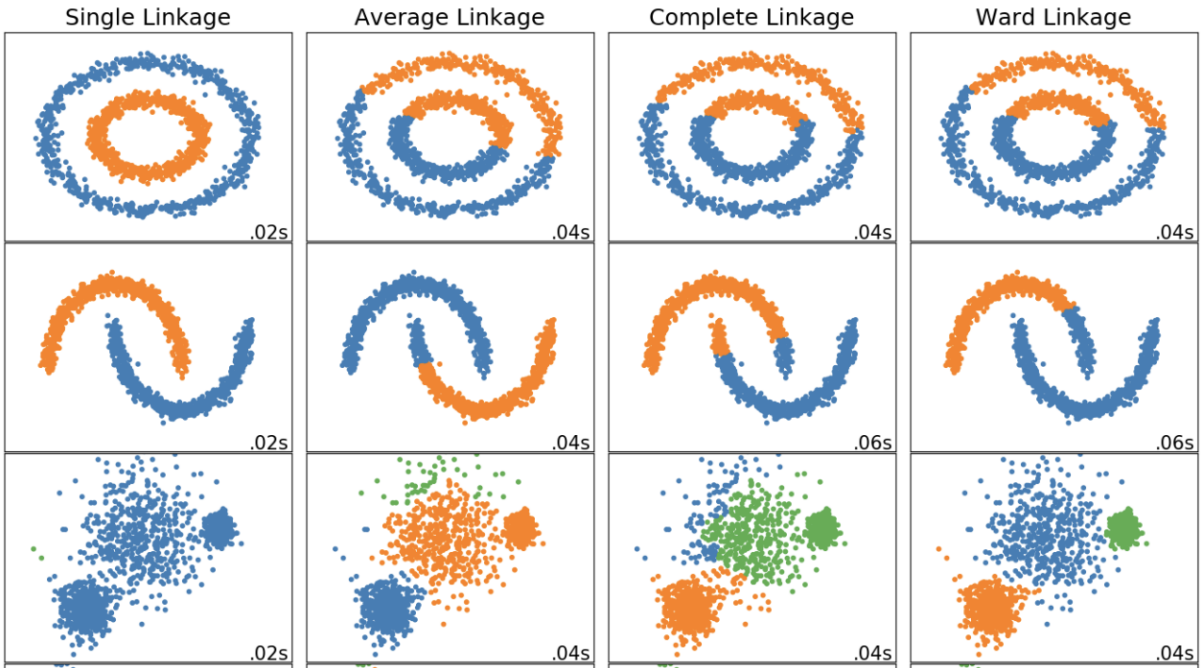
Centroid-Linkage

Finds the centroid of cluster 1 and centroid of cluster 2, and then calculates the distance between the two before merging.

Compute the centroids of two clusters C1 & C2 and take the similarity between the two centroids as the similarity between two clusters. This is a less popular technique in the real world.



The choice of linkage method entirely depends on you and there is no hard and fast method that will always give you good results. Different linkage methods lead to different clusters.

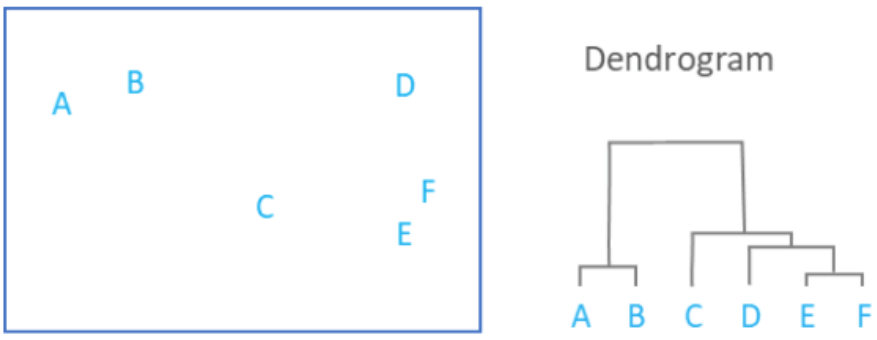


Dendrogram

To better understand the agglomeration process, it's useful to introduce a graphical method called a dendrogram, which shows in a static way how the aggregations are performed, starting from the bottom (where all samples are separated) till the top (where the linkage is complete).

A Dendrogram is a type of tree diagram showing hierarchical relationships between different sets of data. Dendrogram contains the memory of hierarchical clustering algorithm, so just by looking at the Dendrogram you can tell how the cluster is formed.

Dendrogram is often miswritten as Dendogram.



How to read a dendrogram?

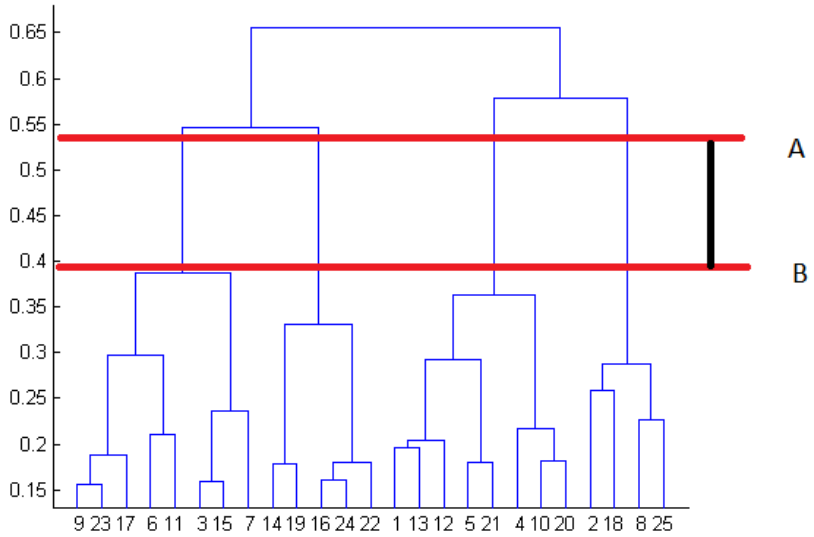
The key to interpreting a dendrogram is to focus on the height at which any two objects are joined together. In the example above, we can see that E and F are most similar, as the height of the link that joins them together is the smallest. The next two most similar objects are A and B.

One question that might have intrigued you by now is how do you decide the number of clusters in hierarchical clustering?

We can use a dendrogram to visualize the history of groupings and *figure out the optimal number of clusters* by applying the following steps:

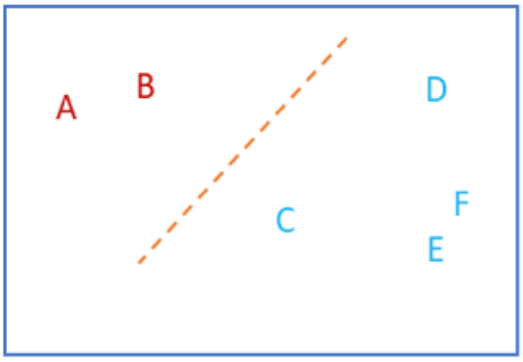
1. Determine the largest vertical distance that doesn't intersect any of the other clusters
2. Draw a horizontal line at both extremities
3. The optimal number of clusters is equal to the number of vertical lines going through the horizontal line

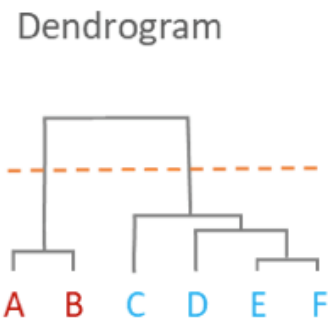
For eg., in the below case, best choice for no. of clusters will be 4.



- ✓ Distance between data points represents dissimilarities.
- ✓ Height of the blocks represents the distance between clusters.

The dendrogram below shows the hierarchical clustering of six observations shown on the scatterplot to the left. Observations are allocated to clusters by drawing a horizontal line through the dendrogram. Observations that are joined together below the line are in clusters.





Unfortunately, scikit-learn doesn't support them. However, SciPy (which is a mandatory requirement for it) provides some useful built-in functions.