Hierarchical Clustering

Hierarchical clustering is a type of unsupervised machine learning algorithm used to cluster unlabeled data points. Like K-means clustering, hierarchical clustering also groups together the data points with similar characteristics. In some cases the result of hierarchical and K-Means clustering can be similar. Before implementing hierarchical clustering using Scikit-Learn, let's first understand the theory behind hierarchical clustering.

Theory of Hierarchical Clustering

There are two types of hierarchical clustering: Agglomerative and Divisive. In the former, data points are clustered using a bottom-up approach starting with individual data points, while in the latter top-down approach is followed where all the data points are treated as one big cluster and the clustering process involves dividing the one big cluster into several small clusters.

Here we will focus on agglomerative clustering that involves the bottom-up approach.

Steps to Perform Hierarchical Clustering

The steps involved in agglomerative clustering are:

1. At the start, treat each data point as one cluster. Therefore, the number of clusters at the start will be K, where K is the number of data points.
2. Form a cluster by joining the two closest data points resulting in K-1 clusters.
3. Form more clusters by joining the two closest clusters resulting in K-2 clusters.
4. Repeat the above three steps until one big cluster is formed.
5. Once a single cluster is formed, use dendrograms to divide it into multiple clusters.

There are different ways to measure the distance between two clusters. The distance itself can be Euclidean or Manhattan distance. Some of the options to measure the distance between two clusters are:

- Measure the distance between the closes points of two clusters.
- Measure the distance between the farthest points of two clusters.
- Measure the distance between the centroids of two clusters.
- Measure the distance between all possible combination of points between the two clusters and take the mean.

Role of Dendrograms for Hierarchical Clustering

In the last section, we said that once one large cluster is formed by the combination of small clusters, dendrograms of the cluster are used to actually split the cluster into multiple clusters of related data points.

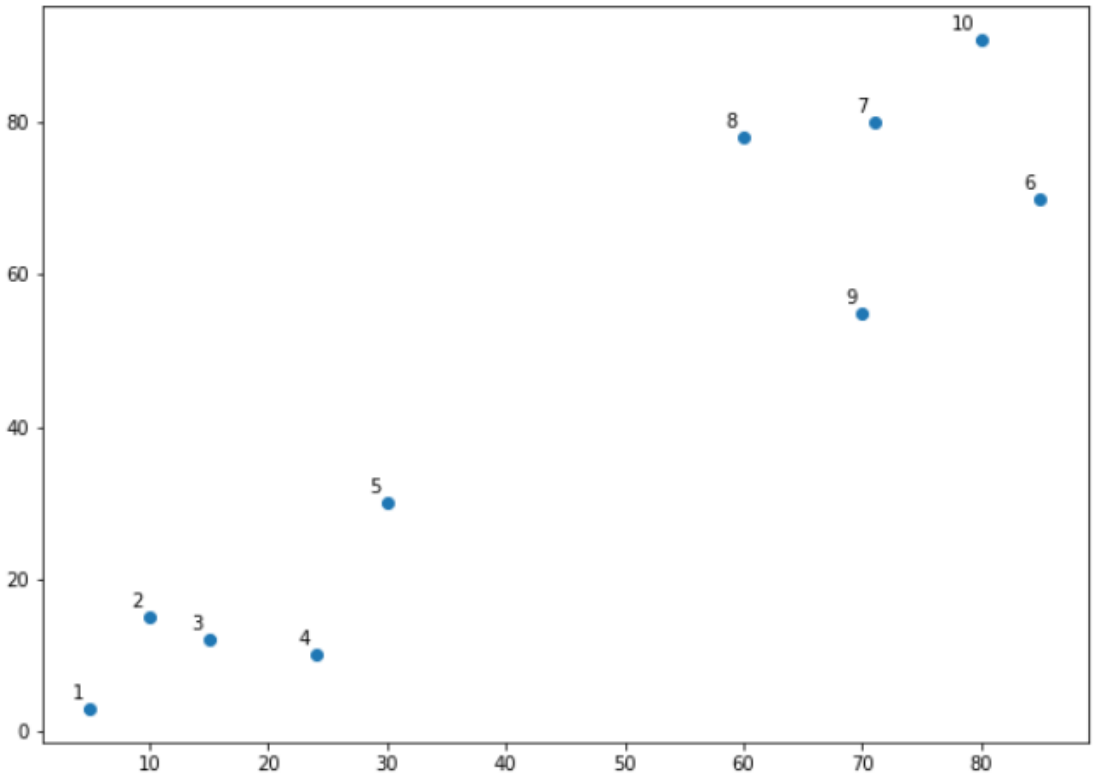
Suppose we have a collection X of data points represented by a numpy array:

```
import numpy as np
X = np.array([[5,3],
        [10,15],
        [15,12],
        [24,10],
        [30,30],
        [85,70],
        [71,80],
        [60,78],
        [70,55],
        [80,91] ])
```

Let's plot the above data points.

```
import matplotlib.pyplot as plt
labels = range(1, 11)
plt.figure(figsize=(10, 7))
plt.subplots_adjust(bottom=0.1)
plt.scatter(X[:,0],X[:,1], label='True Position')
for label, x, y in zip(labels, X[:, 0], X[:, 1]):
    plt.annotate(label,xy=(x, y),xytext=(-3, 3),textcoords='offset points', ha='right',va='bottom')
plt.show()
```

The script above draws the data points in the X numpy array and label data points from 1 to 10. In the image below you'll see the plot generated from this code. Let's name the plot Graph1.



Graph 1

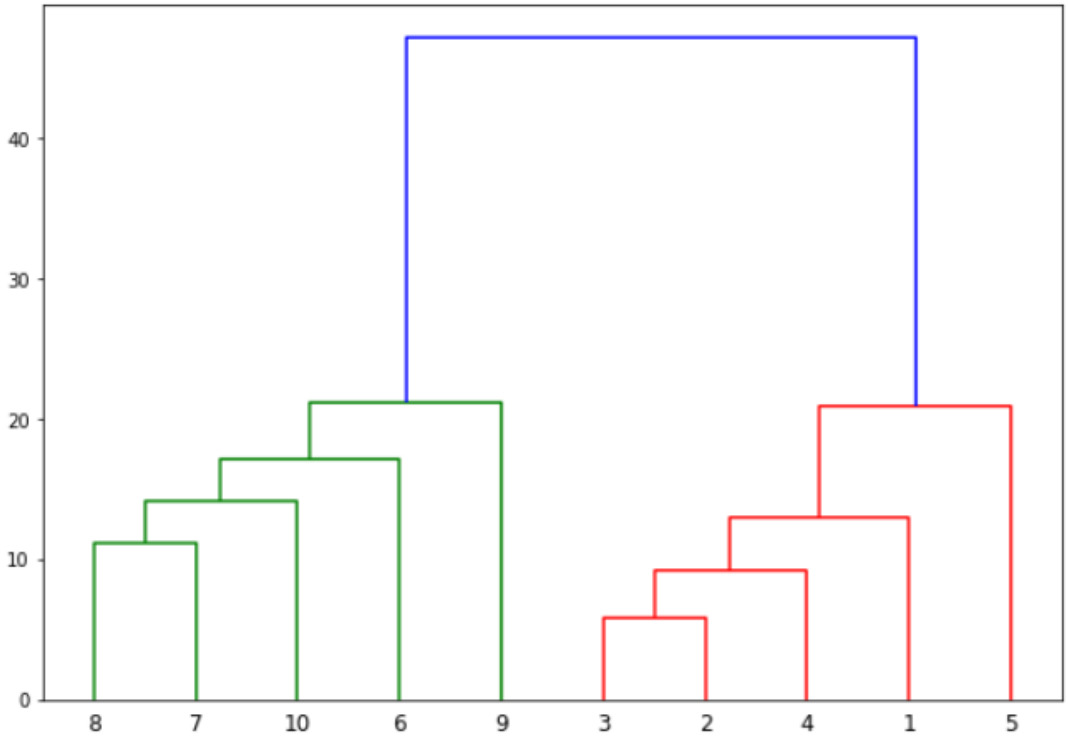
It can be seen that the data points form two clusters: one at the bottom left consisting of points 1-5 and the other at the top right consisting of points 6-10.

In the real world we may have thousands of data points in many more than 2 dimensions. In that case it would not be possible to spot clusters with the naked eye. This is why clustering algorithms have been developed.

Coming back to use of dendrograms in hierarchical clustering, let's draw the dendrograms for our data points. We will use the scipy library for that purpose.

```
from scipy.cluster.hierarchy import dendrogram, linkage
from matplotlib import pyplot as plt
linked = linkage(X, 'single')
labelList = range(1, 11)
plt.figure(figsize=(10, 7))
dendrogram(linked,
   orientation='top',
   labels=labelList,
   distance_sort='descending',
   show_leaf_counts=True)
plt.show()
```

The output graph looks like the one below. Let's name this plot Graph2.



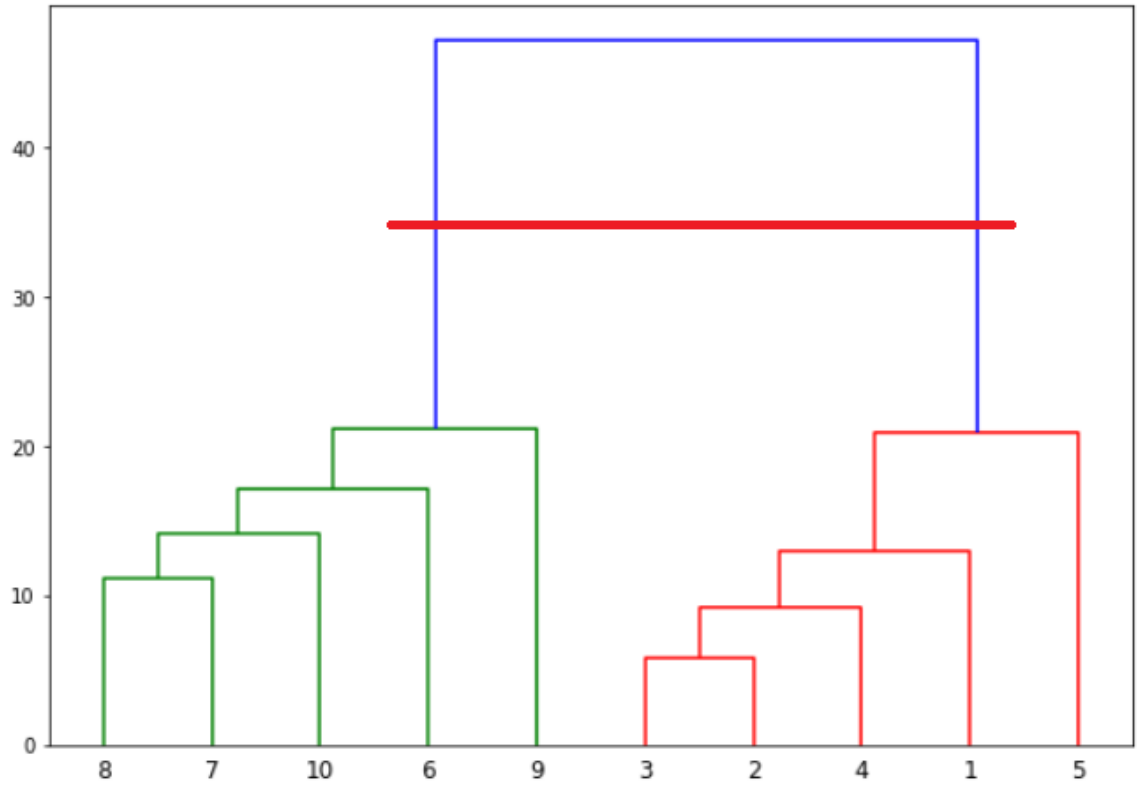
Graph 2

The algorithm starts by finding the two points that are closest to each other on the basis of Euclidean distance. If we look back at Graph1, we can see that points 2 and 3 are closest to each other while points 7 and 8 are closes to each other. Therefore a cluster will be formed between these two points first. In Graph2, you can see that the dendograms

have been created joining points 2 with 3, and 8 with 7. The vertical height of the dendogram shows the Euclidean distances between points. From Graph2, it can be seen that Euclidean distance between points 8 and 7 is greater than the distance between point 2 and 3.

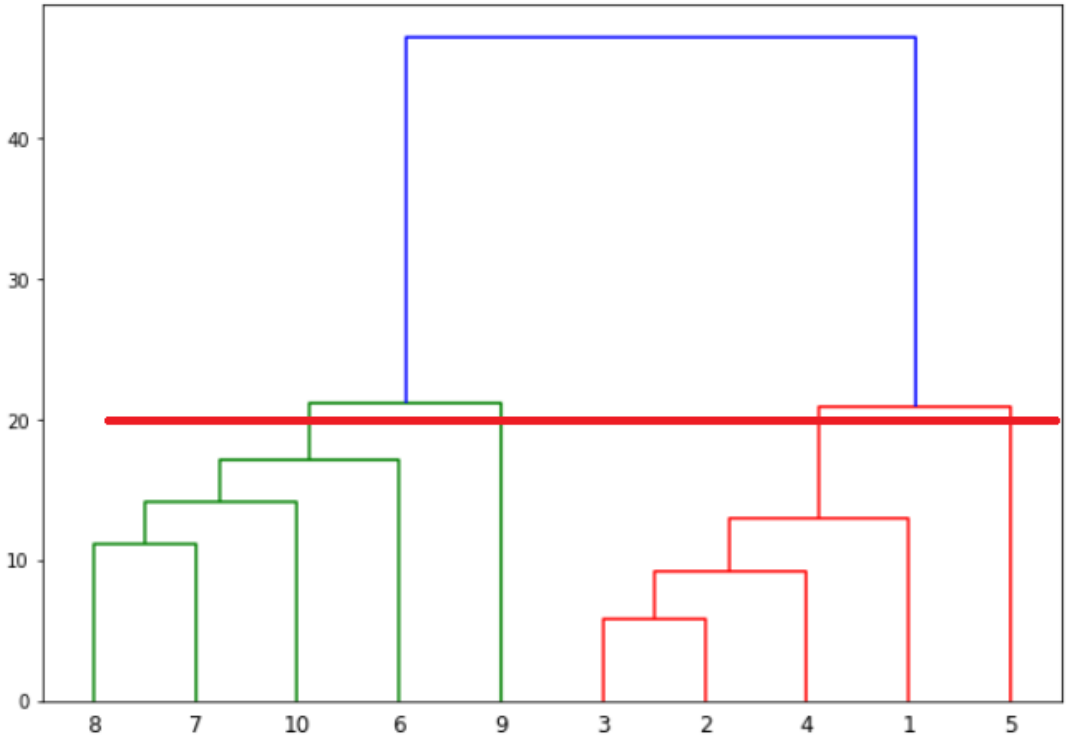
The next step is to join the cluster formed by joining two points to the next nearest cluster or point which in turn results in another cluster. If you look at Graph1, point 4 is closest to cluster of point 2 and 3, therefore in Graph2 a dendrogram is generated by joining point 4 with dendrogram of point 2 and 3. This process continues until all the points are joined together to form one big cluster.

Once such a cluster is formed, the longest vertical distance without any horizontal line passing through it is selected and a horizontal line is drawn through it. The number of vertical lines this newly created horizontal line crosses is equal to number of clusters.



We can see that the largest vertical distance without any horizontal line passing through it is represented by blue line. So we draw a new horizontal red line that passes through the blue line. Since it crosses the blue line at two points, the number of clusters is 2.

Basically the horizontal line is a threshold, which defines the minimum distance required to be a separate cluster. If we draw a line further down, the threshold required to be a new cluster will decrease and more clusters will be formed:



In the above plot, the horizontal line passes through four vertical lines resulting in four clusters: cluster of points 6,7,8 and 10, cluster of points 3,2,4 and points 9 and 5 will be treated as single point clusters.

Hierarchical Clustering via Scikit-Learn

Now let's implement hierarchical clustering using Python's Scikit-Learn library. In our example we will cluster the X numpy array of data points that we created in the previous section.

The process of clustering is similar to any other unsupervised machine learning algorithm. We start by importing the required libraries.

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

The next step is to import or create the dataset. In this example, we'll use the same example data:

```
X = np.array([[5,3],
      [10,15],
      [15,12],
      [24,10],
      [30,30],
      [85,70],
      [71,80],
      [60,78],
      [70,55],
      [80,91] ])
```

The next step is to import the library for clustering and call its fit_predict method to predict the clusters that each data point belongs to.

```
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='ward')
cluster.fit_predict(X)
```

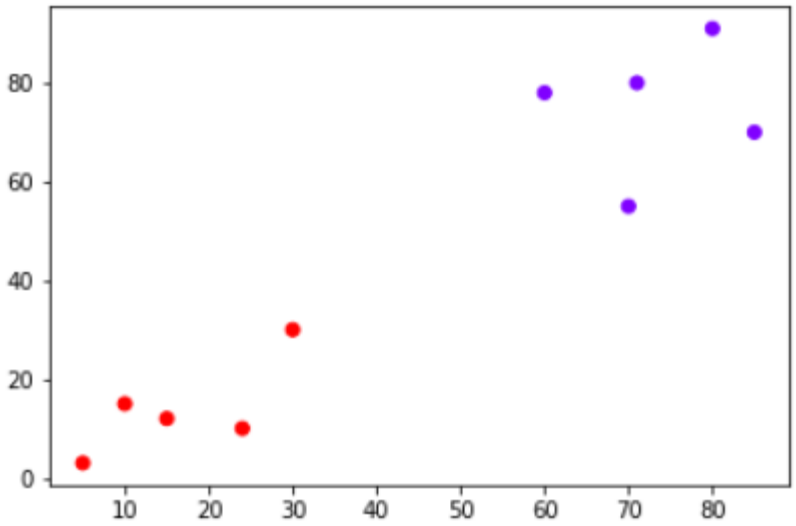
In the code above we import the AgglomerativeClustering class from the sklearn.cluster library. The number of parameters is set to 2 using the n_clusters parameter while the affinity is set to "euclidean" (distance between the datapoints). Finally linkage parameter is set to "ward", which minimizes the variant between the clusters.

Next we call the fit_predict method from the AgglomerativeClustering class variable cluster. This method returns the names of the clusters that each data point belongs to. Execute the following script to see how the data points have been clustered.

Finally, let's plot our clusters. To do so, execute the following code:

```
plt.scatter(X[:,0],X[:,1], c=cluster.labels_, cmap='rainbow')
plt.show()
```

You can see points in two clusters where the first five points clustered together and the last five points clustered together.



Conclusion

The clustering technique can be very handy when it comes to unlabeled data. Since most of the data in the real-world is unlabeled and annotating the data has higher costs, clustering techniques can be used to label unlabeled data.