Hierarchical Clustering

This clustering algorithm will try to continuously split out and merge new clusters successively until it reaches a level of convergence.

This algorithm identifies the first pair of points, which has the minimal distance, and it turns it into the first cluster, then, the second pair of points with the second minimal distance will form the second cluster, and so on. As the algorithm continues doing this with all the pairs of closest points, it can turn all points into just one cluster, which is why HAC also needs a stopping criterion.

There are a few linkage types or methods to measure the distance between clusters. These are the most common:

Single linkage: minimum pairwise distance between clusters.

It takes the distance between specific points and declares that as the distance between 2 clusters. Then, it tries to find these distances for all the pairwise linkages. It will take the minimum distances and then it will combine those together as we move up to a higher hierarchy.

Pros: It helps ensuring a clear separation between clusters.

Cons: It won't be able to separate out cleanly if there is some noise between 2 different clusters.

Complete linkage: maximum pairwise distance between clusters.

Instead of taking the minimum distance given the points within each cluster, it will take the maximum value. Then, from those maximum distances it decides which one is the smallest and then we can move up that hierarchy.

Pro: It would do a much better job of separating out the clusters if there's a bit of noise or overlapping points of two different clusters.

Cons: Tends to break apart a larger existing cluster depending on where that maximum distance of those different points may end up lying.

Average linkage: Average pairwise distance between clusters.

Takes the average of all the points for a given cluster and uses those averages or clusters centroids to determine the distance between the different clusters.

Pros: The same as the single and complete linkage.

Cons: It also tends to break apart a larger existing cluster.

Ward linkage: Cluster merge is based on inertia.

Computes the inertia for all pairs of points and picks the pair that will ultimately minimizes the value of inertia.

The pros and cons are the same as the average linkage.

Syntax for Agglomerative Clusters

First, import AgglomarativeClustering

From sklearn.cluster import AgglomerativeClustering

then, create an instance of class,

agg = AgglomerativeClustering (n_clusters=3, affinity='euclidean', linkage='ward')

and finally, fit the instance on the data and then predict clusters for the new data

```
agg=agg.fit(X1)

y_predict=agg.predict(X2)
```