Fourth Industrial Summer School

Module 4: ML

Unsupervised Learning: Clustering Algorithms

Outlines

- ✓ Clustering algorithms
  - ✓ Hierarchical Based Methods
  - ✓ AC Algorithm
  - ✓ Dendrograms
  - ✓ Linkages
  - ✓ Sklearn implementation & parameters
  - ✓ Pros & Cons

Hierarchical Clustering

- It looks at the problem of clustering as an accumulative task.

- Hierarchical clustering finds successive clusters from the previously established clusters.

- There are two main scenarios

  - Divisive Style

  - Agglomerative Style

Divisive

1. It is a top down approach.
2. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.
3. In order to find the best split, it needs to explore all possibilities at each step (**expensive**).

Agglomerative

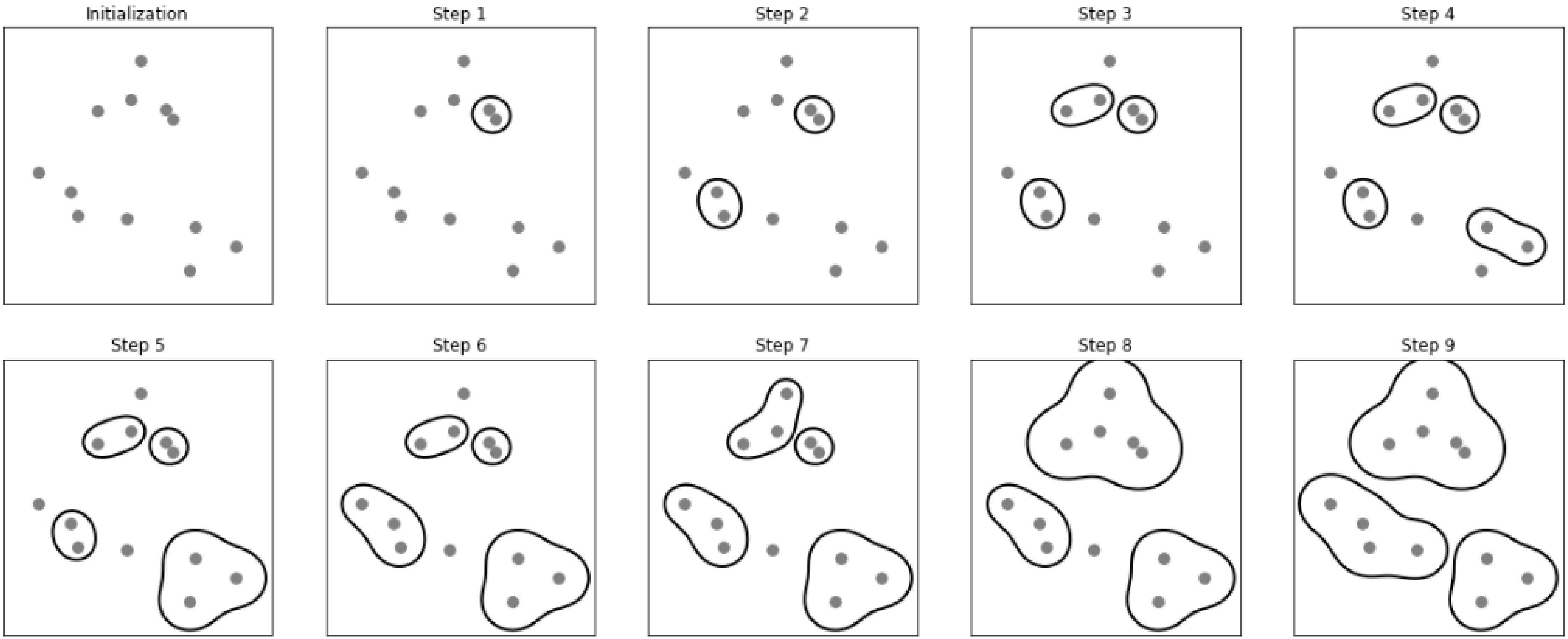
- A bottom-up approach

- It starts with every data point as a separate cluster
- Then, it merges the most similar pairs of data points/clusters until it forms one big cluster

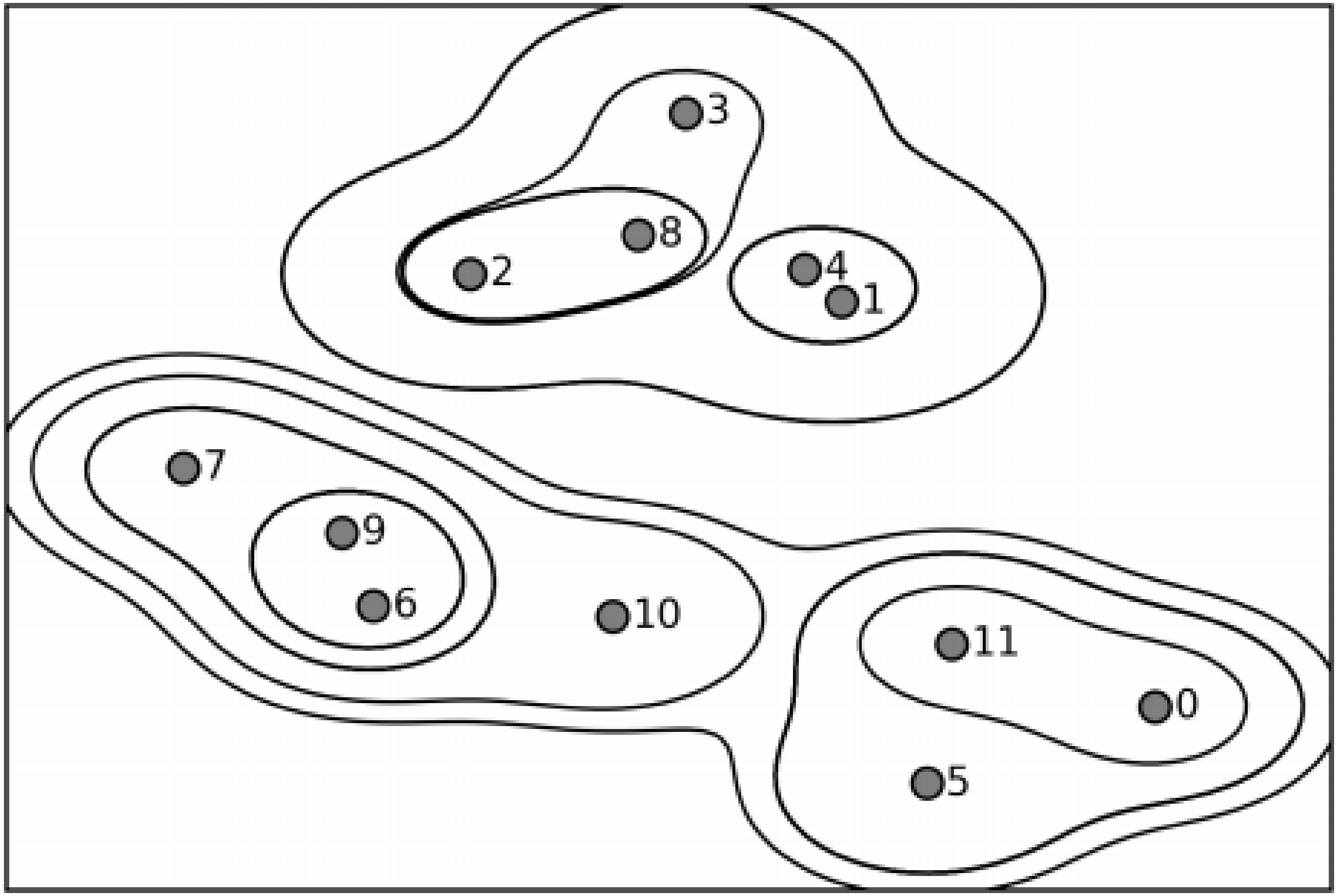
- In Kaufmann et. al, 1990, he described an algorithm called Agglomerative nesting (AGNES).
  - Use a single-linkage (nearest neighbor) using Euclidean distance
  - It merges similar points to form larger clusters at each level
  - Eventually all points gathered into one cluster

Illustration

- Accumulative task of clustering

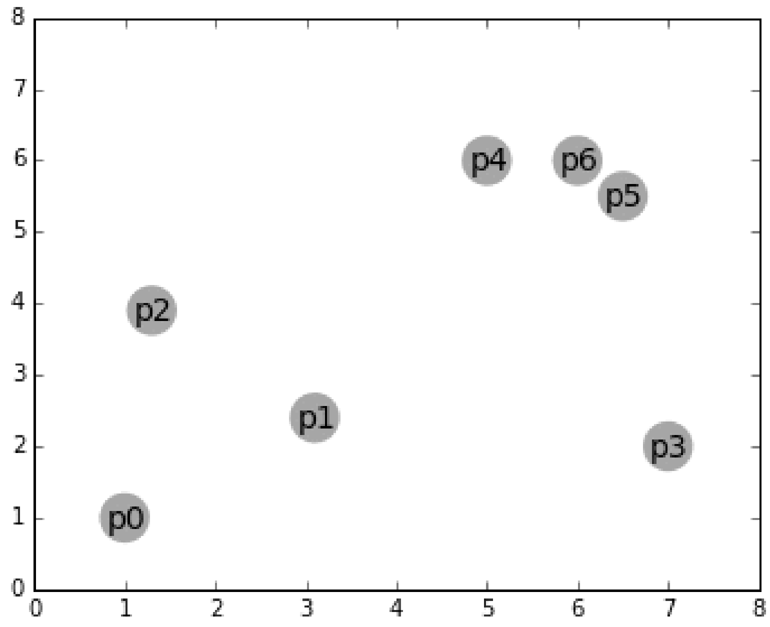


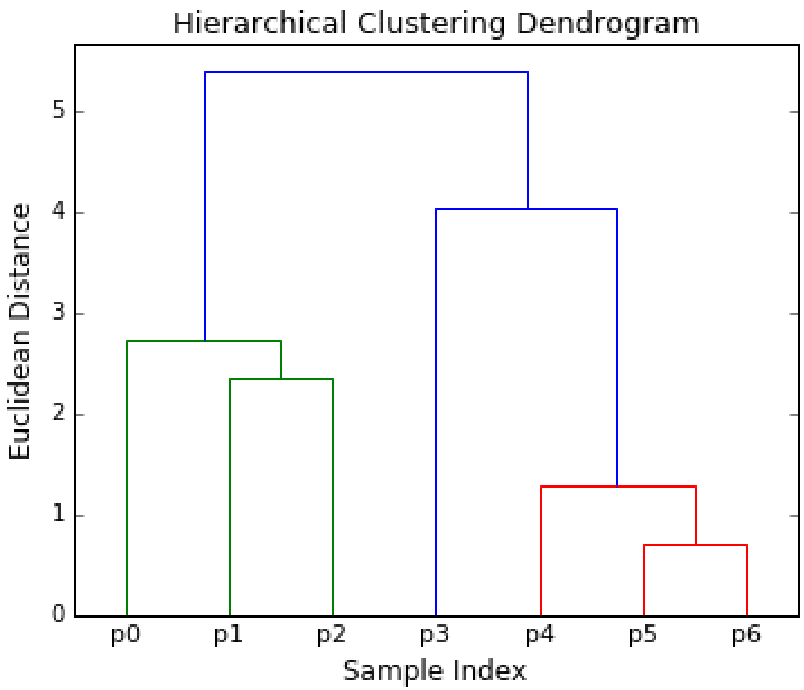
2d Plot of clustering



Dendrogram structure

- The AC algorithm builds a tree-based structure to represent all possible solutions
- This tree-structure is called a **dendrogram**

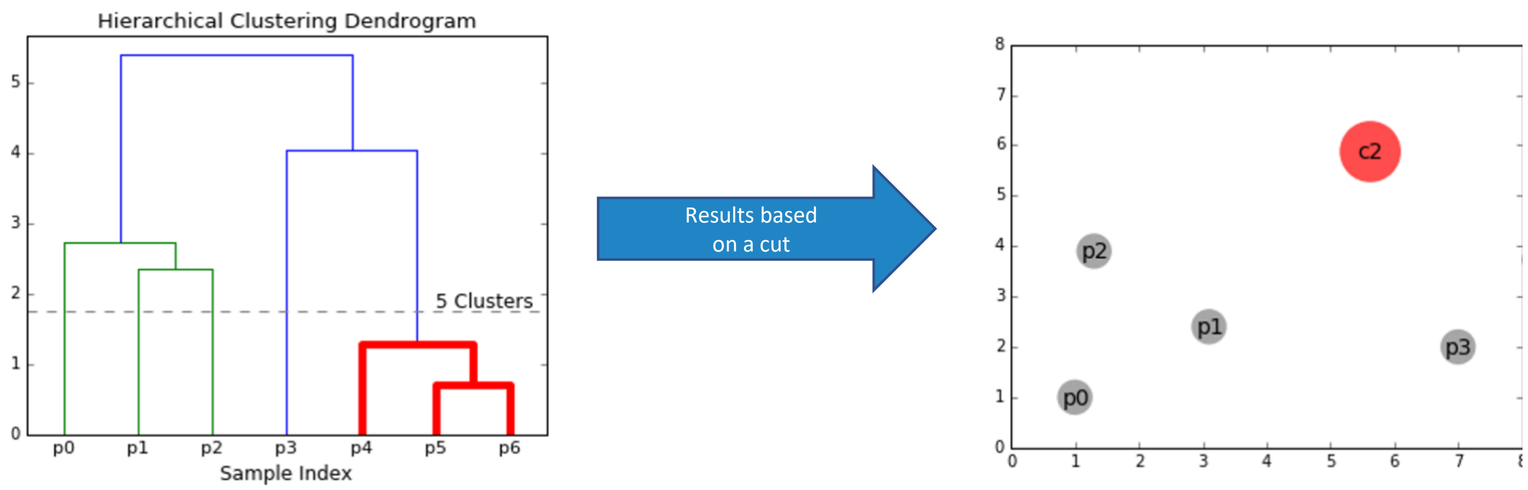




Source: https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

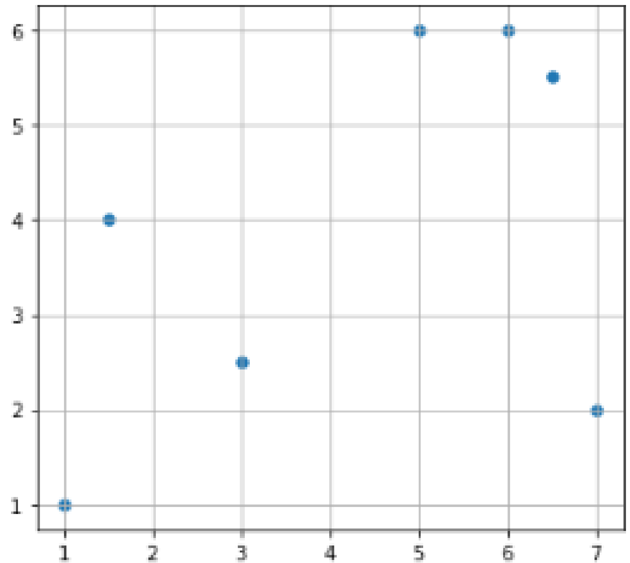
Dendrogram

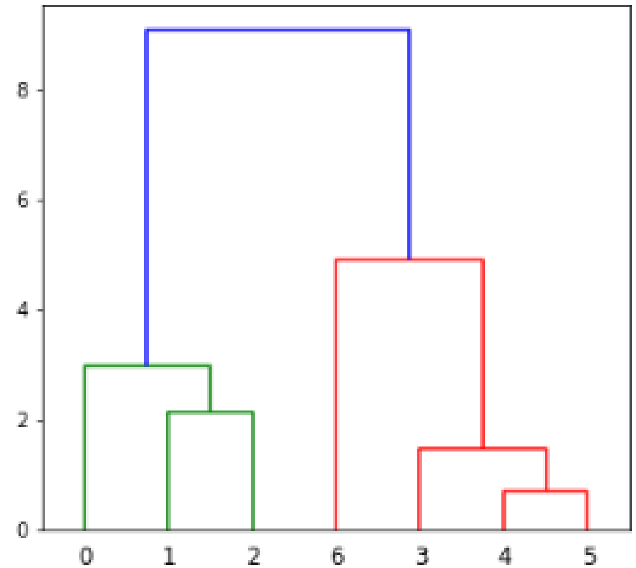
- All possible solutions are nested inside the dendrogram
  - At start each datapoint is a separate cluster
  - Then, elements are merged based on a metric used
- Dendrogram can provide useful information:
  - Data is easily summarized/organized into a hierarchy
  - The vertical distance of a dendrogram indicates the level of similar among datapoints



Python Code

```python
# ploting dendrograms
from scipy.cluster.hierarchy import dendrogram, ward, linkage
example = np.array([[1,1], [1.5,4], [3,2.5], [5,6], [6,6], [6.5, 5.5], [7,2]])
linkage_array = linkage(example, 'ward')
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.scatter(example[:,0], example[:,1], s=30)
plt.grid()
plt.subplot(1,2,2)
dendrogram(linkage_array)
plt.show()
```





Copy

```python
# from scipy.cluster.heirarchy import dendrogram, ward, linkage
from scipy.cluster.hierarchy import dendrogram, ward, linkage
example  = np.array([[1,1], [1.5, 4], [3, 2.5], [5, 6], [6.5, 5.5], [7,2]])
# compute the dissimilarity matrix
linkage_matrix = linkage(example, 'ward')
fig = plt.figure(figsize = (12,6))
plt.subplot(1,2,1)
plt.scatter(example[:,0], example[:,1], s= 30)
plt.grid()
plt.subplot(1,2,2)
dendrogram(linkage_matrix)
plt.show()
```

Linkage

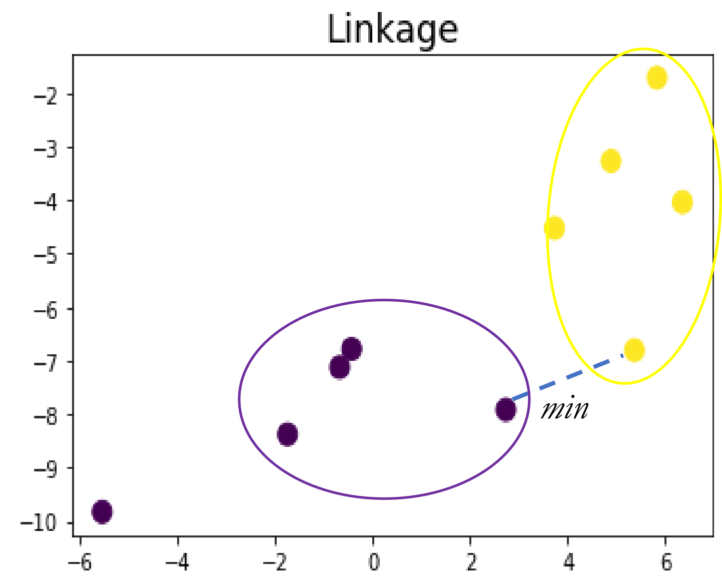
- AGNES (the original version of AC) is based on **single link**

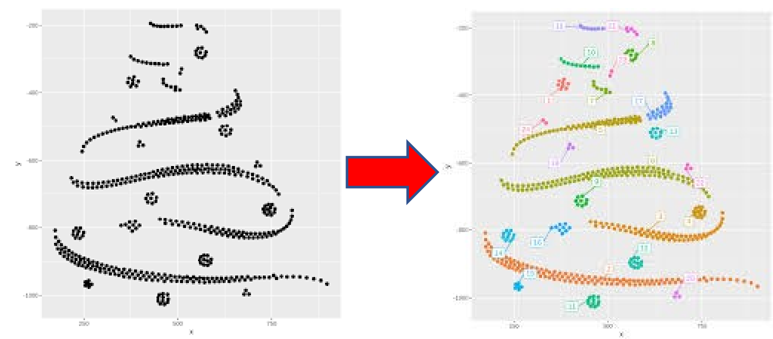
- There are several other linkages developed by researchers as

  - Complete link (Diameter)
  - Average link (Group average)
  - Centroid link (Centroid similarity)
  - Ward Link (WSS control)

Single-Linkage

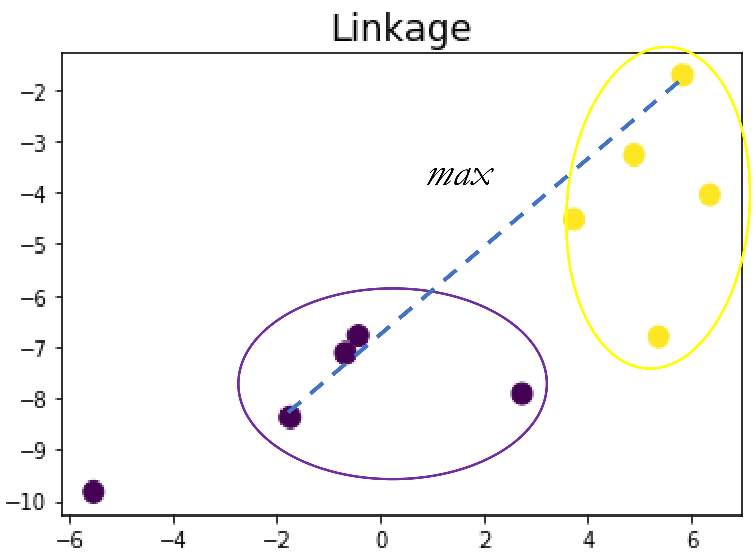
- The similarity between two clusters is the similarity between their most similar elements (**nearest neighbor**)
- It is emphasizing more on close regions, ignoring the overall structure of the cluster (could be a good choice in some situations)
- Sensitive to noise and outliers

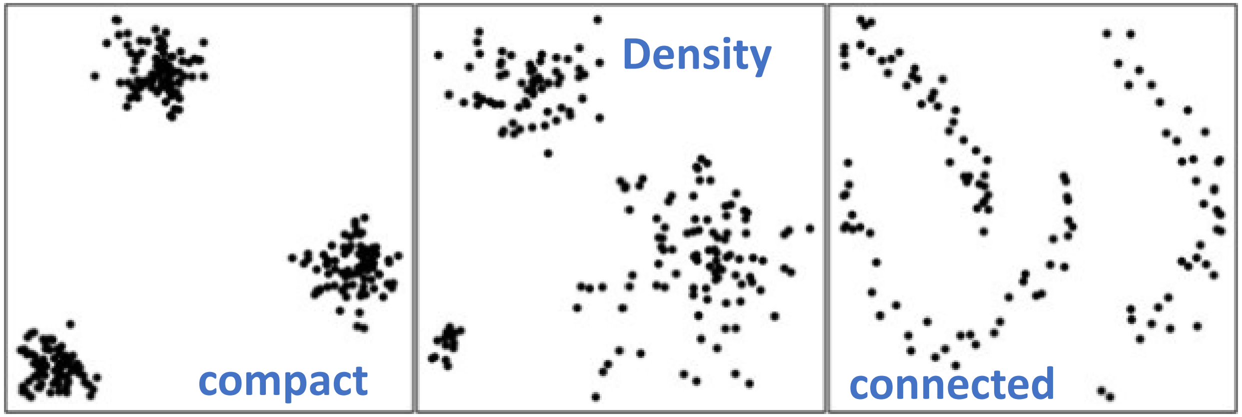




Complete-Linkage

- The similarity between tw clusters is the similarity betwee their most dissimilar members
- Merge two clusters to form on with **the smallest diameter**
- Nonlocal in behavior, obtainin compact shaped clusters
- Sensitive to outliers

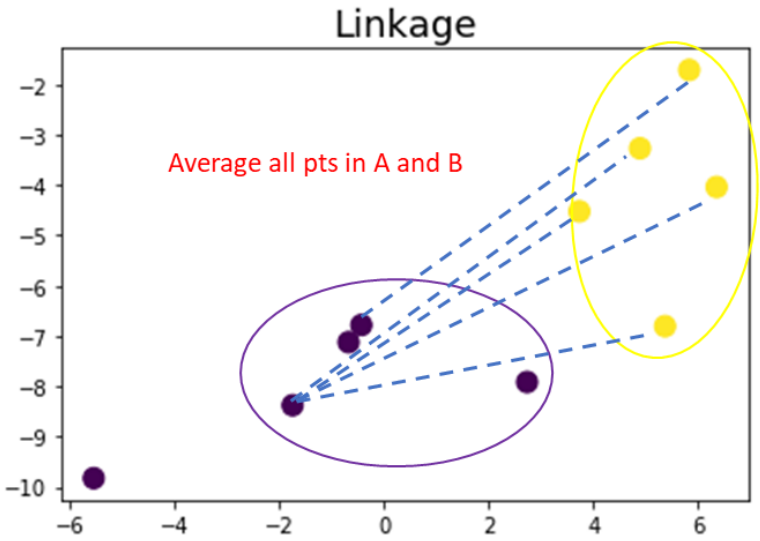




Source: https://www.researchgate.net/publication/326222960_Model-based_Clustering/figures?lo=1

Average-Linkage

- It is expensive to compute
- The average distance between all elements in one cluster to all elements in the other (i.e., all pairs in two clusters)
- It tries to find balanced clusters
- Could be used when mixed cluster shapes exist in the dataset.



Centroid-Linkage

- In centroid link only the distance between centroids of two clusters is considered.

- This means if two clusters are merged and one of them has larger number of elements, then the new centroid will be towards that previous larger cluster.

- It is faster than calculating all pairwise distances between data points in the clusters, Use Centroid Linkage:

  - Large Datasets

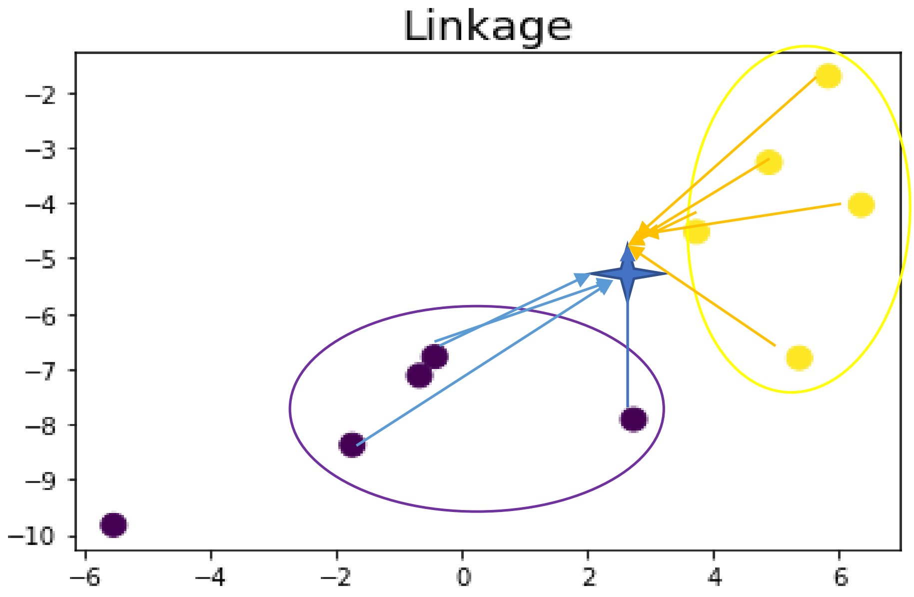
  - Spherical Clusters

- The inertia maybe increased due to merging high variance clusters and not accounting for the inertia .

Ward Linkage

- Ward's linkage minimizing inertia (WSS)

- Ward's linkage weighs the increase by distance between centroids
- For each cluster, the inertia is calculated. The two clusters with the smallest increase in the inertia are combined.



How are clusters combined ?

- The algorithm starts by computing a distance between every pair of points in the data. This creates a distance matrix
- To start merging clusters, the algorithm finds the two points that have shortest distance. (found to be between **3 and 5**)

	1	2	3	4	5
1	0				
2	9	0			
3	3	7	0		
4	6	5	9	0	
5	11	10	$\boxed{2}$	8	0

Contd.

- The points with the shortest distance, are combined into **35 cluster**, the distance table is updated as shown

	1	2	3	4	5
1	0				
2	9	0			
3	3	7	0		
4	6	5	9	0	
5	11	10	$\boxed{2}$	8	0

	35	1	2	4
35	0			
1	11	0		
2	10	9	0	
4	9	6	5	0

Complete linkage is used in this example!

Agglomerative clustering

Loading the package

```
# Load the package from Scikit learn library
from sklearn.cluster import AgglomerativeClustering
```

Make an instance object

```
# Create an instance of Agglomerative
AggloCluster = AgglomerativeClustering(n_clusters =2, # default is 2
                                       affinity  ='euclidean', # manhattan or cosine
                                       linkage ='ward'
                                      )

```

Perform the clustering

```
# Do clustering
AggloCluster.fit(X)

# Do clustering and return data predicted labels
labels  =AggloCluster.fit_predict(X)
```

Contd.

- Using **.fit( )** is performing the clustering. The labels can be found in the attribute **labels_**

```python
# Compute clustering
AggloCluster = AgglomerativeClustering(n_clusters=5, linkage='ward').fit(Xb)
labels = AggloCluster.labels_
lbl = AggloCluster.fit_predict(Xb)
print("Number of points: %i" % labels.size)
print('Clustering Labels:\n', lbl)
```

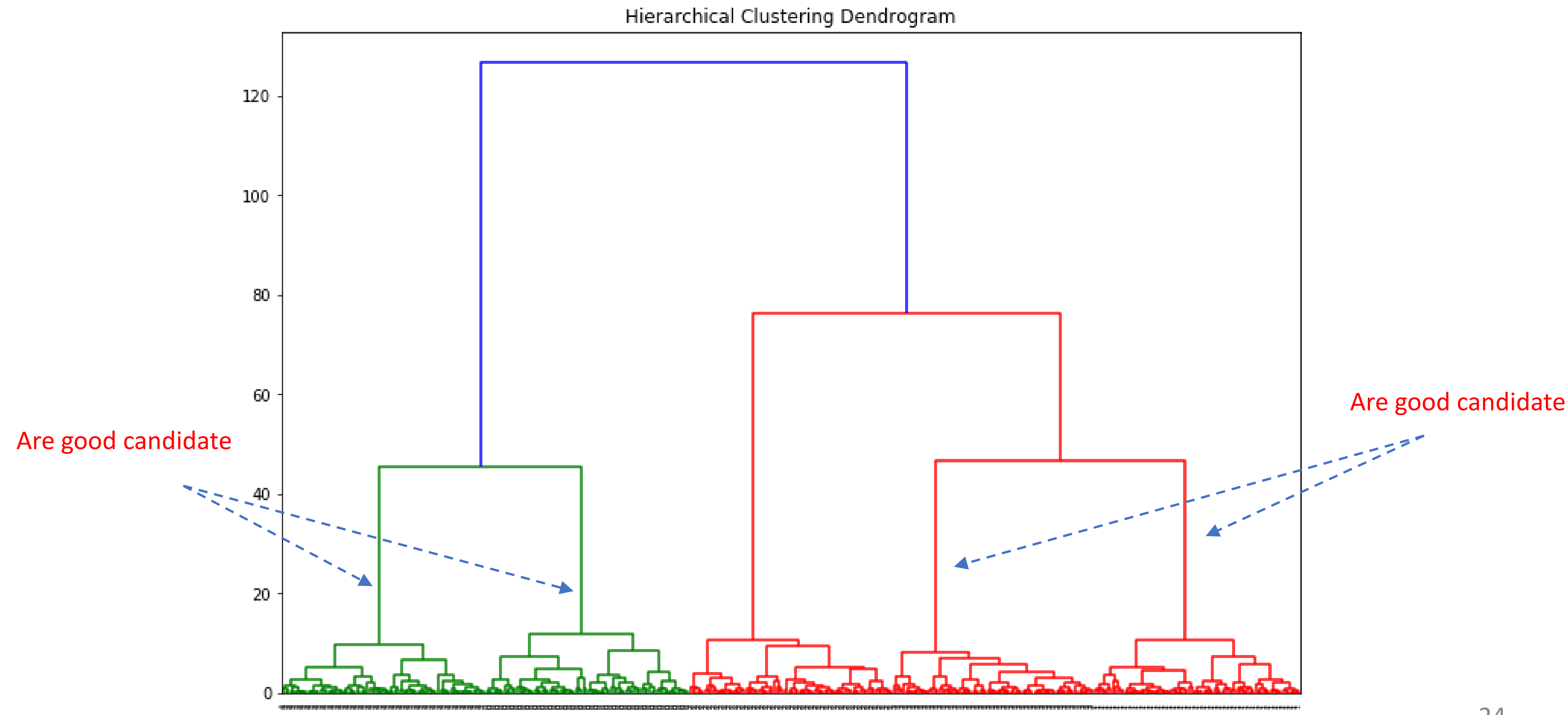
```
Number of points: 400
Clustering Labels:
 [4 4 3 2 0 1 4 3 2 3 1 4 4 4 2 1 0 3 2 4 0 0 1 2 0 4 4 4 3 4 2 3 3 2 1 2 1
 2 2 1 2 4 0 0 0 4 1 2 3 4 1 1 4 2 4 0 1 4 3 4 1 2 2 2 3 4 3 2 0 2 3 0 4 3
 3 0 4 0 4 3 1 2 2 4 3 3 0 3 3 0 1 1 4 1 3 1 2 1 3 4 1 3 3 4 2 3 0 1 1 3 1
 3 0 2 3 3 0 3 0 3 0 4 0 4 4 1 4 0 2 1 1 2 2 1 3 0 0 2 0 4 2 2 1 0 2 4 2 1
 1 1 1 3 1 4 3 3 1 0 4 2 3 3 3 3 1 3 2 1 3 4 4 4 2 0 3 1 3 0 2 0 4 0 3 1 0
 2 3 1 4 1 1 0 0 0 0 3 2 0 4 1 3 2 3 1 1 3 4 4 2 0 1 4 2 2 3 3 4 1 4 4 0 3
 4 4 0 1 1 1 2 2 0 2 0 0 1 0 0 0 4 2 2 4 4 0 2 0 0 4 1 3 1 0 0 4 3 0 1 4 1
 0 0 4 1 2 4 1 3 2 2 2 4 4 4 1 1 2 1 1 1 3 3 1 3 4 3 4 1 2 4 3 2 3 4 4 3 3
 1 3 3 2 0 1 4 4 4 0 1 3 1 2 2 3 3 2 2 0 0 2 2 3 0 2 3 0 1 3 2 0 1 4 4 2 2
 3 2 0 0 0 1 0 1 4 3 0 1 2 0 4 2 4 1 0 4 1 0 4 4 1 3 0 0 2 2 2 4 0 1 2 0 2
 2 2 3 4 3 3 1 1 0 3 2 2 4 3 0 1 0 0 2 1 1 1 0 4 4 2 0 0 3 2]
```

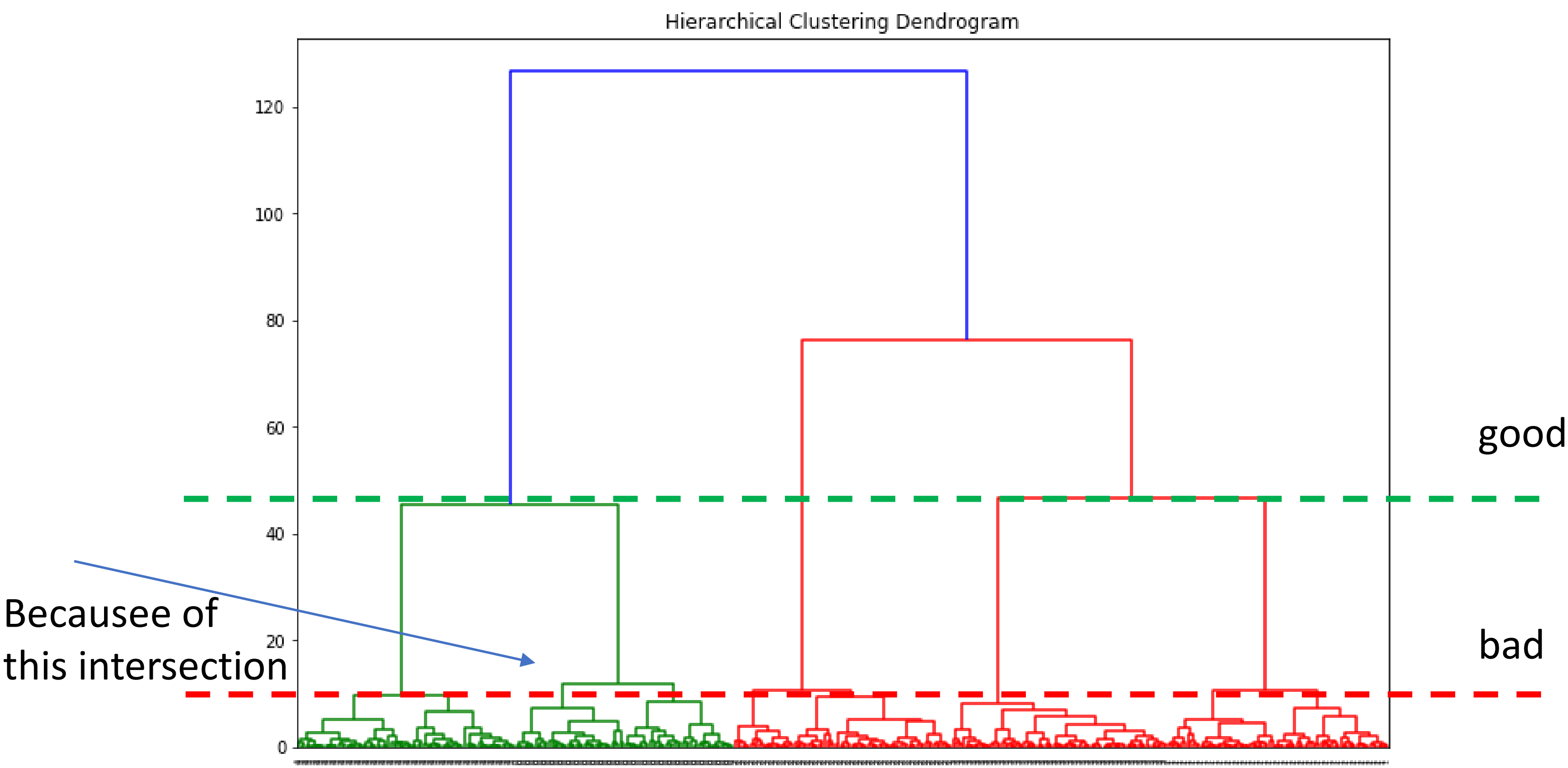
Contd.

- The default number of clusters used by agglomerative clustering is **2**.
- The linkage is very important parameter, and it depends on the domain.
- Same issue of determining the number of clusters also exists here!
- but with better structure to understand your data!

Determine number of clusters

- Dendrogram might help to determine the number of clusters.
- Determine the vertical line/s that crossed by no other horizontal line/s




Hierarchical Clustering Dendrogram
120
100
80
60
40
20
0
good
bad
Becausee of
this intersection

Plot the dendrogram of the clustering

Load the two methods { **dendrogram**, **linkage**)

```
# Get the dendrogram and linkage packages from scipy
from scipy.cluster.hierarchy import dendrogram, linkage
```

Compute the linkage matrix using **linkage**

```
# Compute the linkage matrix
linkage_matrix = linkage(Xb, 'ward')
```

Plot the tree structure using **dendrogram**

```
plt.figure(figsize=[12,8])
plt.title('Hierarchical Clustering Dendrogram')
dendrogram( linkage_matrix)
plt.show()
```

Python Linkage matrix

- The matrix format is (**Cluster i, Cluster j, Distance, Count of clusters joined**)

```
example = np.array([[1,1], [1.5,4], [3,2.5], [5,6], [6,6], [6.5, 5.5], [7,2]])
linkage_array = linkage(example, 'single')
linkage_array
```

```
array([[ 4.        ,  5.        ,  0.70710678,  2.        ],
       [ 3.        ,  7.        ,  1.        ,  3.        ],
       [ 1.        ,  2.        ,  2.12132034,  2.        ],
       [ 0.        ,  9.        ,  2.5       ,  3.        ],
       [ 6.        ,  8.        ,  3.53553391,  4.        ],
       [10.        , 11.        ,  4.03112887,  7.        ]])
```

Pros and Cons

- ▷ Hierarchical clustering outputs a structure that is more informative than flat clusters
- ▷ It is easier to decide on the number of clusters by looking at the dendrogram with small datasets

- ▷ High Computational / time Complexity
- ▷ Sensitive to noise
- ▷ Dendrogram may become complicated


® Creative Commons licenses