Hierarchical Clustering

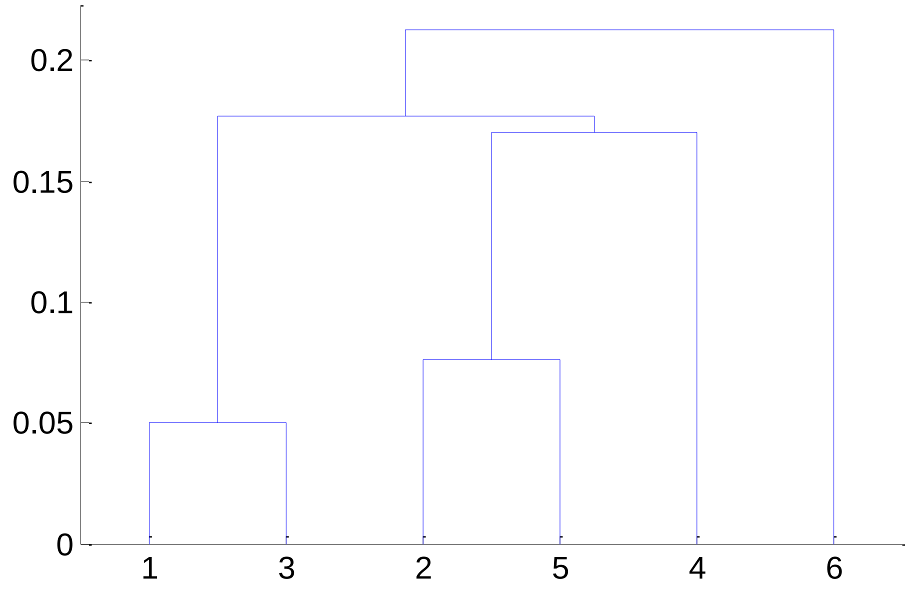
Hierarchical Clustering

- **Hierarchical clustering** is a method of cluster analysis which seeks to build a hierarchy of clusters
  - Clusters are nested
  - Each object may belong to multiple nested clusters
- **Agglomerative**:
  - Start with each point being its own cluster
  - Merge the closest pairs of clusters
- **Divisive:**
  - Start with all points belonging to one cluster
  - Split clusters until each cluster contains a single object

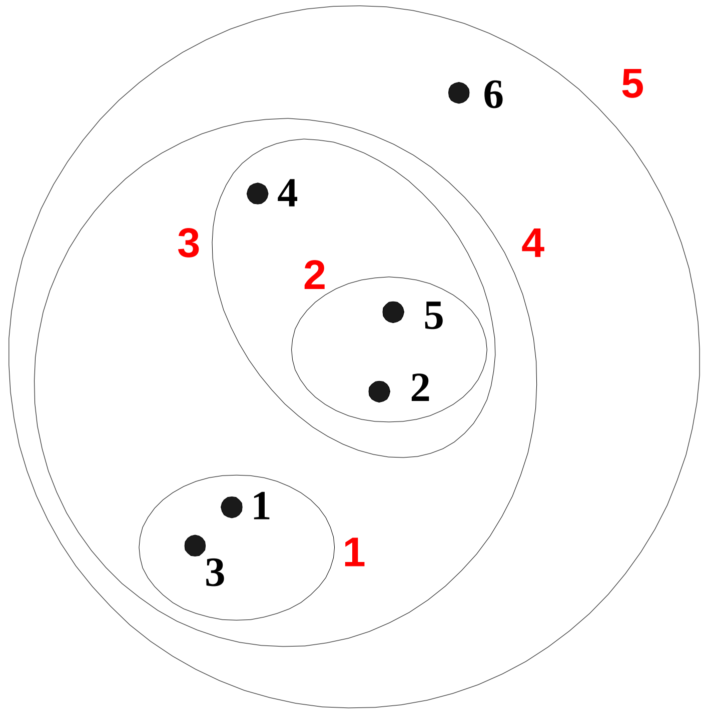
Representation

Dendrogram*:*

- shows the hierarchical relationship between objects.
- Most commonly used



Displays the objects in the order in which they were merged



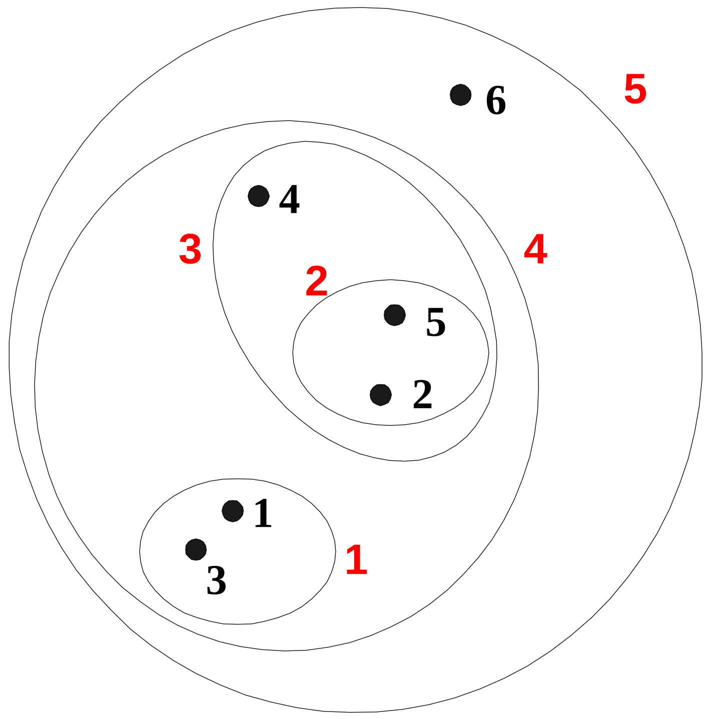
Nested cluster diagram

Agglomerative Hierarchical Clustering

```
Compute the proximity matrix, if necessary
Repeat
    Merge the closest two clusters
    Update the proximity matrix to reflect the proximity between the
    new cluster and the original clusters
Until one cluster remains
```

Need a proximity measure

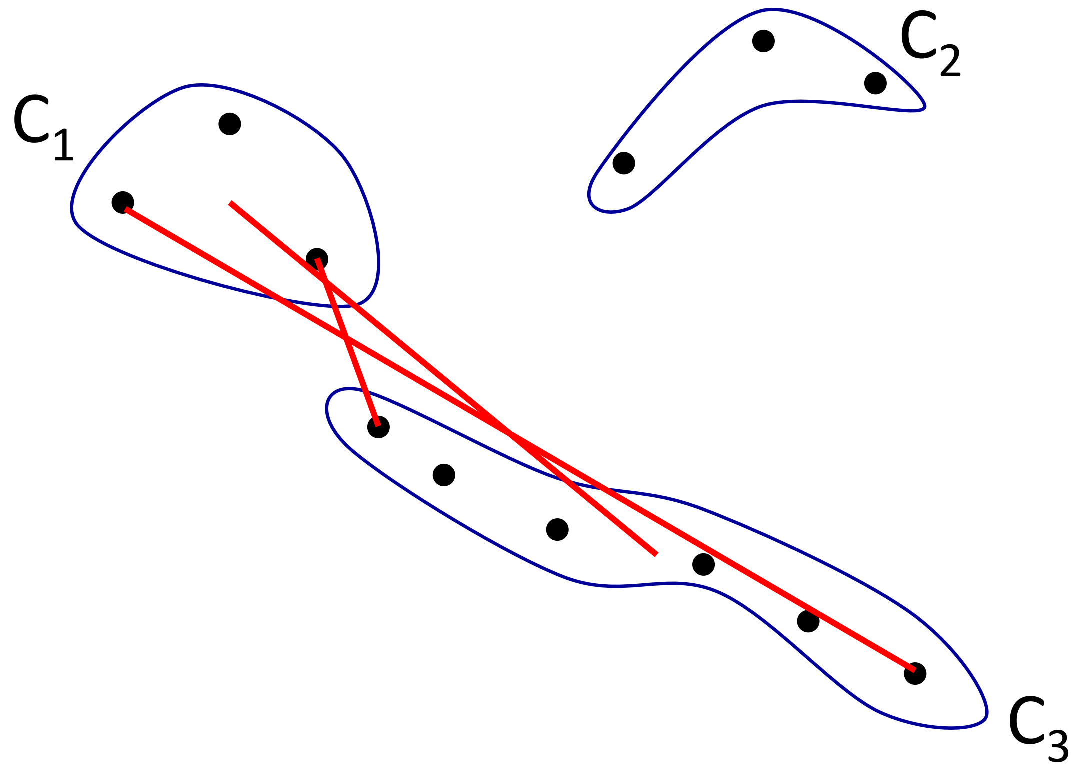
Example



Proximity between Clusters

Which two clusters are the closest?

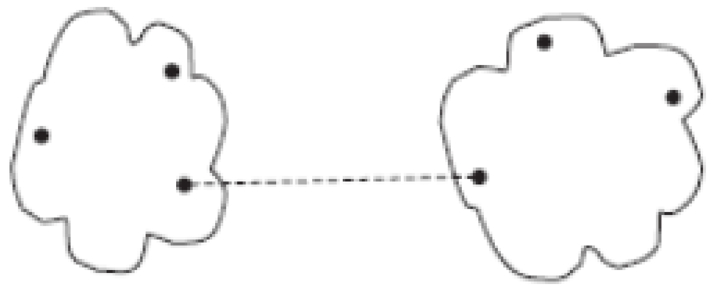
What is the distance between C_1 and C_3?



Proximity between Clusters

MIN (Single Link):

Cluster proximity is defined as the proximity between the closest two points in different clusters

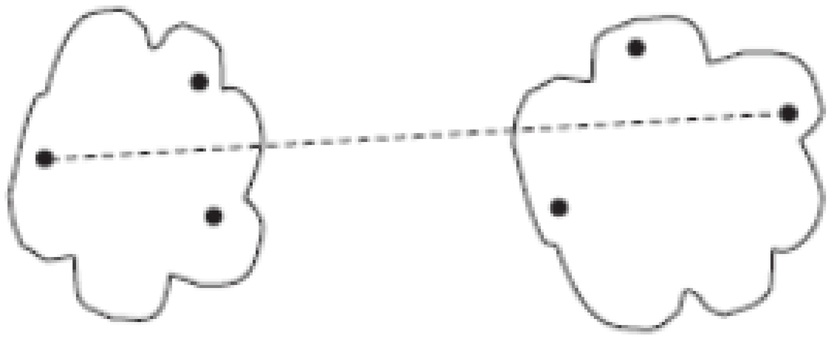


(a) MIN (single link.)

Proximity between Clusters

MAX:

Cluster proximity is defined as the proximity between the farthest two points in different clusters

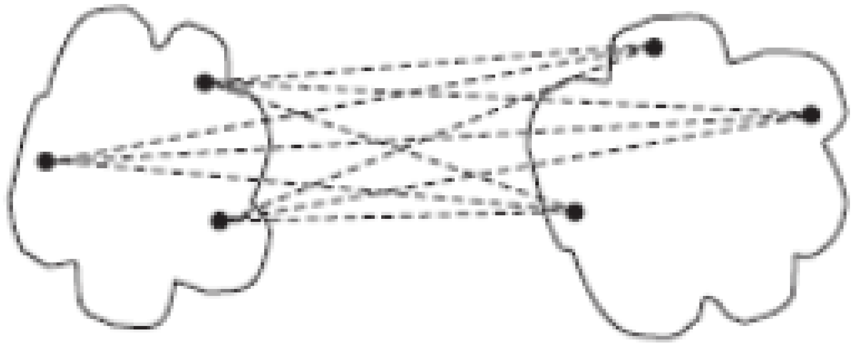


(b) MAX (complete link.)

Proximity between Clusters

Group Average:

Cluster proximity is defined as the average pairwise proximities of all pairs from different clusters

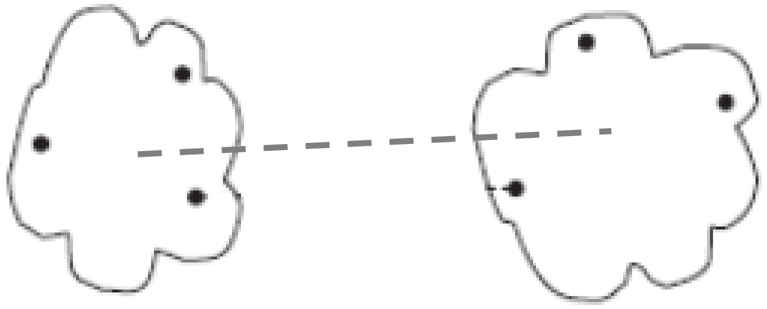


(c) Group average.

Proximity between Clusters

Centroid method:

Cluster proximity is defined as proximity between clusters centroids



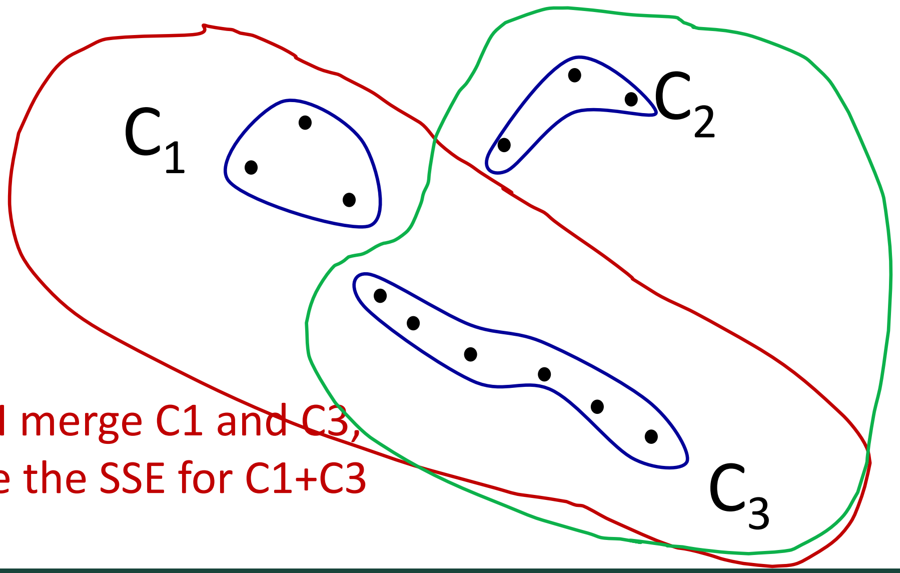
Proximity between Clusters

Ward's Method:

Represents each cluster by its centroid

Measures proximity in terms of increase in the SSE

-- merge the pair of clusters that minimizes the total within-group error (sum of squares) between each point and centroid.

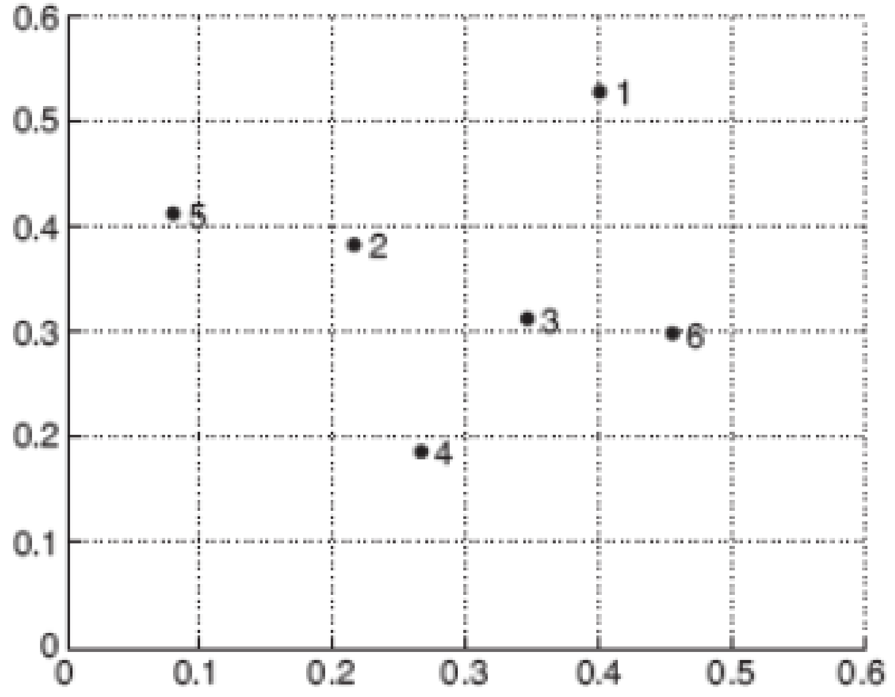


Suppose I merge C1 and C3,
I estimate the SSE for C1+C3

Alternatively, I merge C2 and C3,
I estimate the SSE for C2+C3

Ward's method will choose to merge the pair that leads to smallest SSE

Example



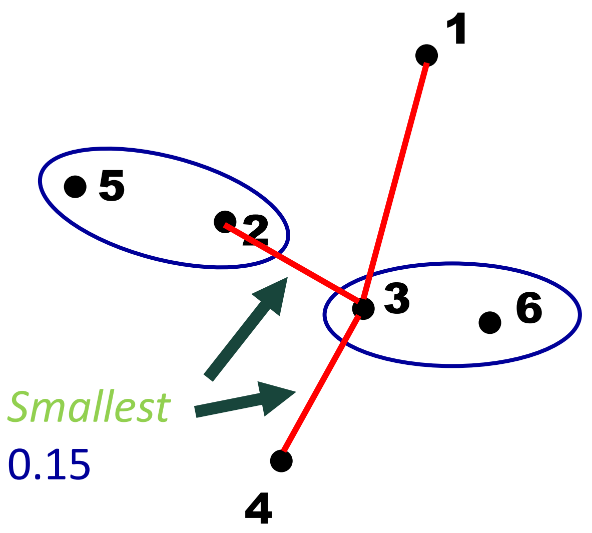
Points coordinates in 2D space

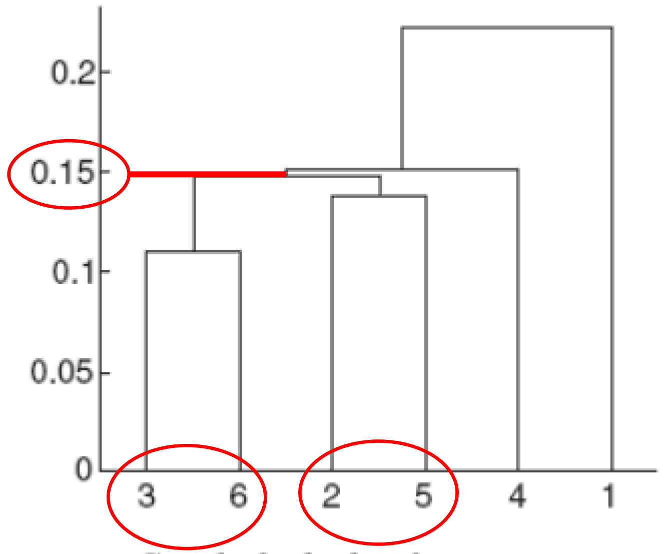
Point	x Coordinate	y Coordinate
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

Euclidean distances between each pair

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

MIN - Proximity





Single link dendrogram.

dist({3,6}, {1}) = min (dist(3, 1), dist(6, 1)) = dist(3, 1) = 0.22

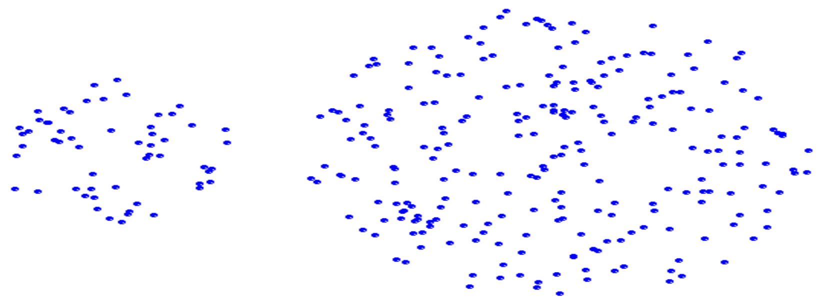
dist({3,6}, {4}) = min (dist(3, 4), dist(6, 4)) = dist(3, 4) = 0.15

dist({3,6}, {2,5}) = min (dist(2, 3), dist(2, 6), dist(5, 3), dist(5, 6))

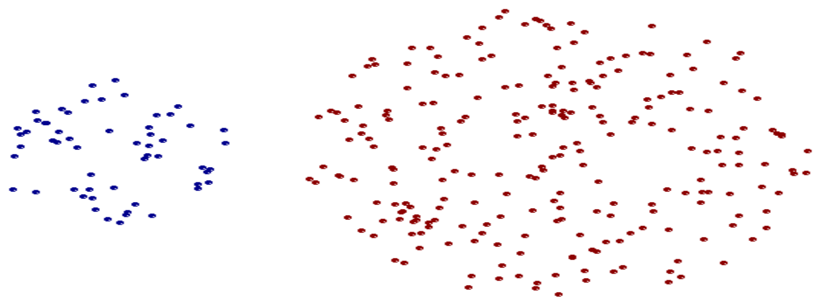
=dist(2, 3)= 0.15

Strength of MIN

Handles non-elliptical shapes



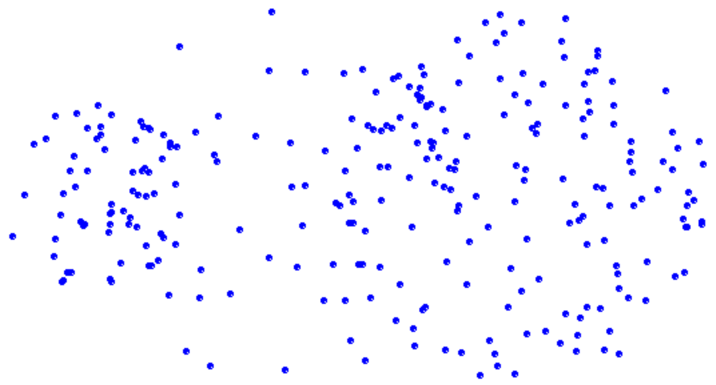
Original Points



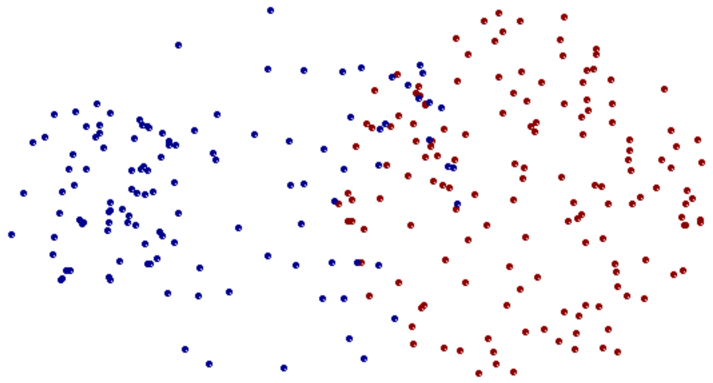
Two Clusters

Limitations of MIN

Sensitive to noise and outliers

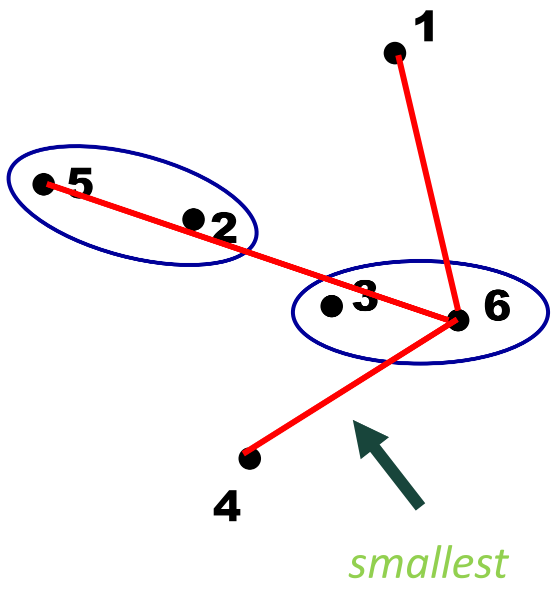


Original Points



Two Clusters

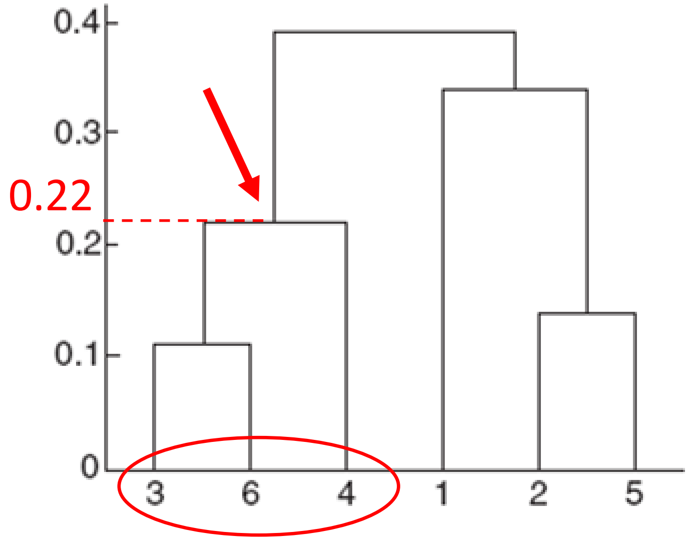
MAX - Proximity



dist({3,6}, {4}) = max (dist(3, 4), dist(6, 4))
= max (0.15, 0.22) = 0.22

dist({3,6}, {1}) = max (dist(3, 1), dist(6, 1))
= max (0.22, 0.23) = 0.23

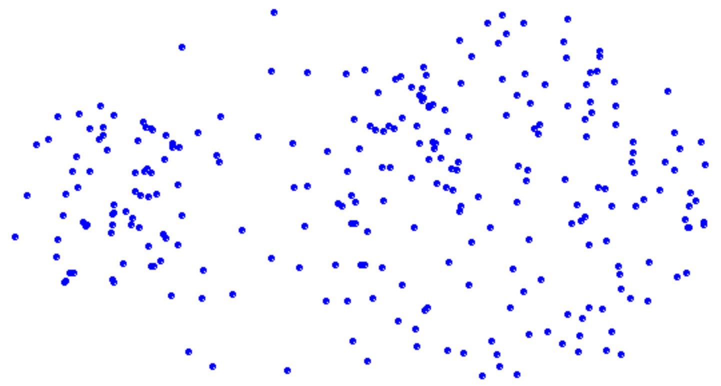
dist({3,6}, {2,5}) = max (dist(2, 3), dist(2, 6), dist(5, 3), dist(5, 6))
= max (0.15, 0.25, 0.28, 0.39)
= 0.39



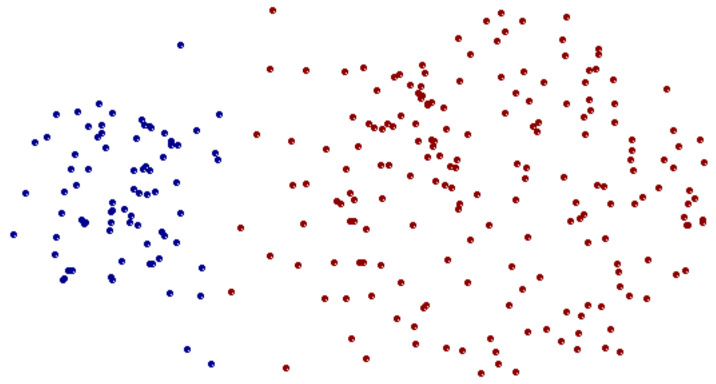
Complete link dendrogram.

Strength of MAX

Less susceptible to noise and outliers



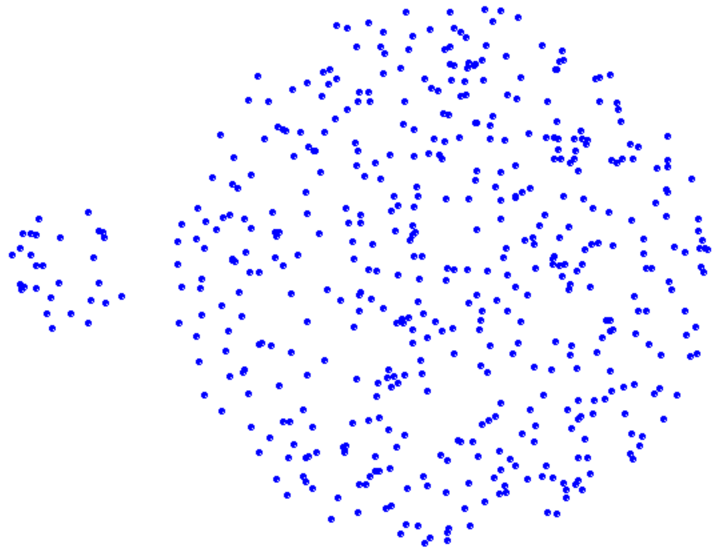
Original Points

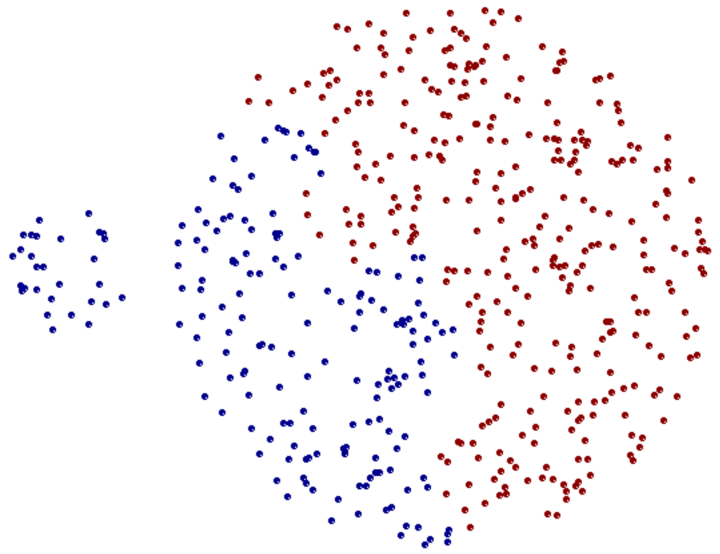


Two Clusters

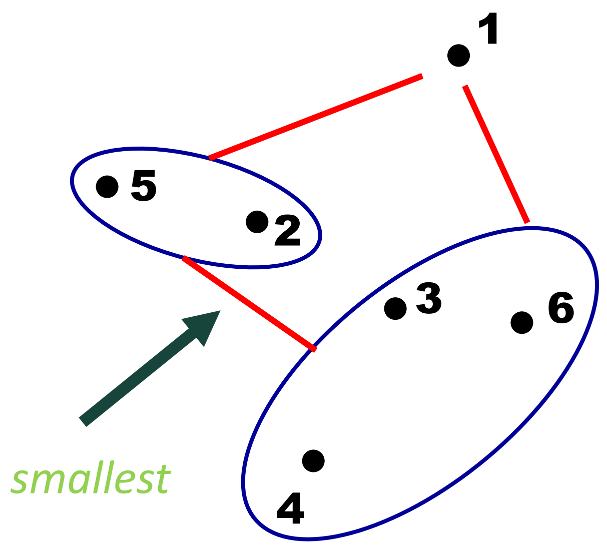
Limitations of MAX

- Tends to break large clusters
- Biased towards globular clusters


Original Points


Two Clusters

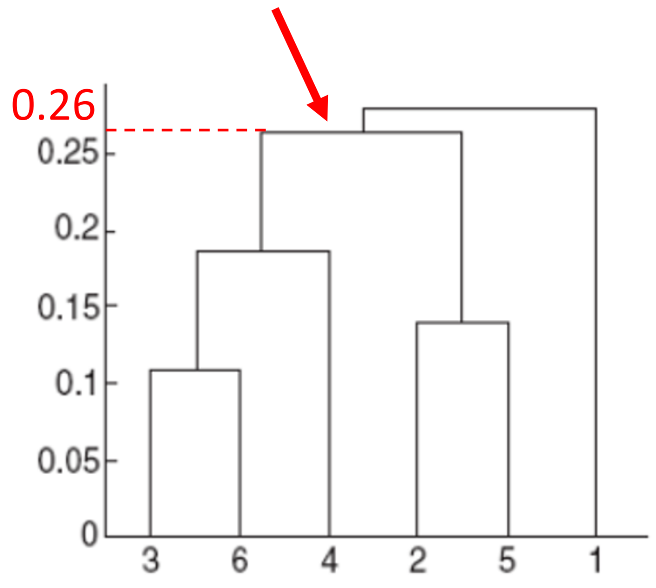
Group Average - Proximity



dist({3,6,4}, {1}) = (dist(3, 1) + dist(6, 1) + dist(4, 1)) / 3*1= 0.28

dist({2,5}, {1}) = (dist(2, 1) + dist(5, 1)) / (2*1) = 0.2889

dist({3,6,4}, {2,5}) = (dist(3, 2) + dist(3, 5) + dist(6, 2) + dist(6, 5) + dist(4, 2) + dist(4, 5)) / (3*2)= 0.26



(b) Group average dendrogram.

Group Average Characteristics

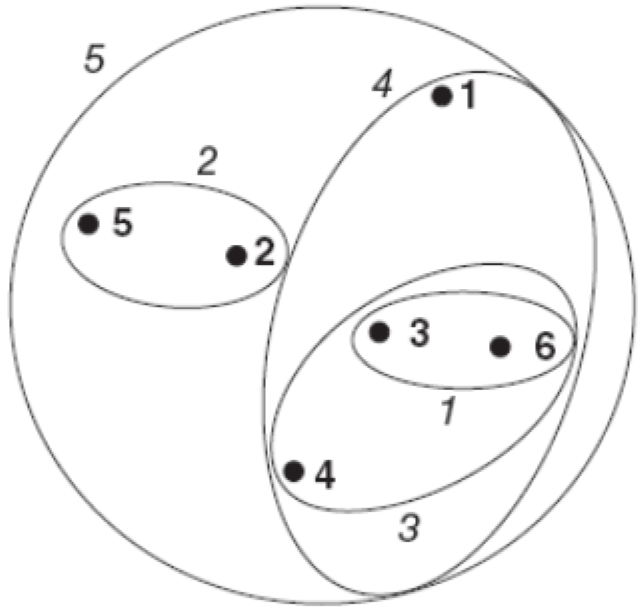
- Compromise between Single and Complete Link
- Less susceptible to noise
- Biased towards globular clusters

Ward’s Method- Proximity

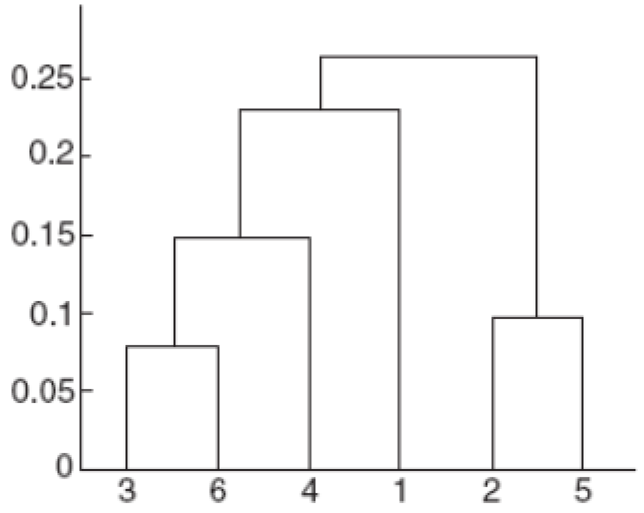
This entire cumbersome procedure makes it practically impossible to perform by hand

Centroid Methods

- Less susceptible to noise
- Biased towards globular clusters



(a) Ward’s clustering.



(b) Ward’s dendrogram.

Agglomerative Approach Summary

- Lack of global objective function, repeatedly decide locally
- Does not require a certain number of clusters
- Ability to handle different cluster sizes
- Merging decisions are final
  - When a point is assigned to a cluster, it does not get reassigned in a subsequent step
  - Problematic with noisy data
- Expensive in terms of computational time and storage requirement:
  - N^2 storage: since we maintain a proximity matrix
  - N^3 time: since we run it for N steps and each time we search N^2 matrix

Divisive Clustering

Divisive Clustering: a top-down clustering approach

1. Initially, all points in the dataset belong to one single cluster.

2. Repeat

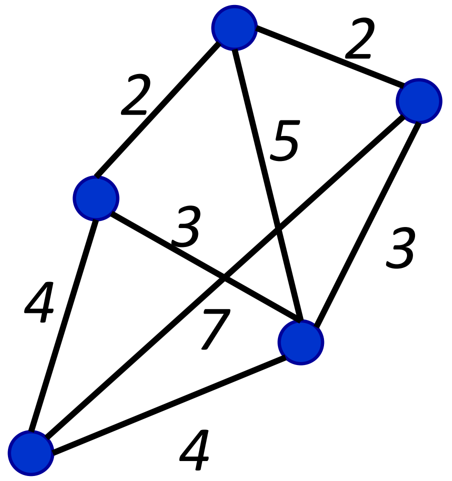
   Partition the cluster into two least similar cluster (cluster with the largest SSE value)

3. Until only singleton clusters remain, or the desired number of clusters is obtained.

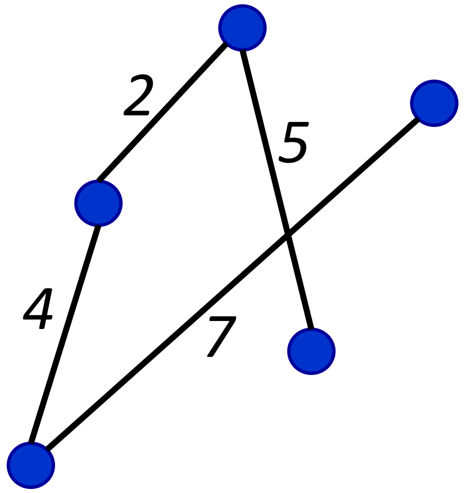
Divisive Clustering - MST

Minimum Spanning Tree (MST):

- A spanning tree is a subgraph that is a tree connecting all points of a graph
- A graph has many spanning trees
- Define: Cost = the sum of weights of its edges
- A minimum spanning tree is spanning tree with minimum total cost



Full graph

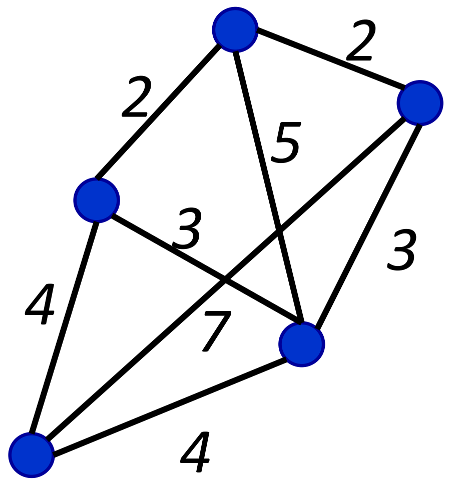


Spanning Tree
Cost = 2+4+7+5 = 18

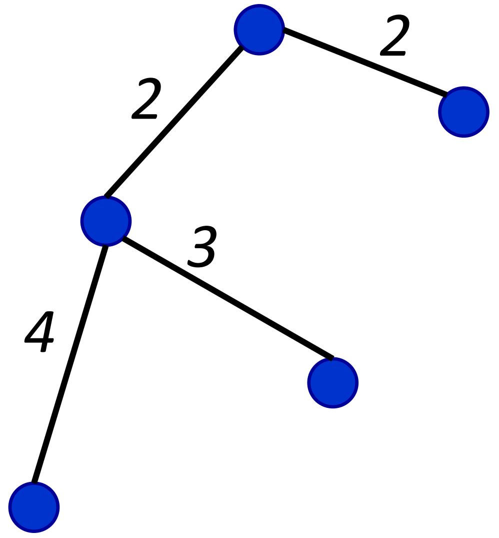
Divisive Clustering - MST

Minimum Spanning Tree (MST):

- The problem: how to find the minimum length spanning tree.
- Example question: You want to connect several computers with a network, and you want to run as little wire as possible.
  - undirected graph G with vertices for each computer
  - weights on the edges giving the distance u and v



Full graph



Spanning Tree MST
Cost = 2+2+4+3 = 11

Divisive Clustering - MST

- Input: N data points (as nodes) and the distance between them (weight of edges)
  - Distance may be an actual distance, or some abstract representation of how dissimilar two things are.

- Goal: Divide the N data points up into k groups so that the minimum distance between items in different groups is maximized.

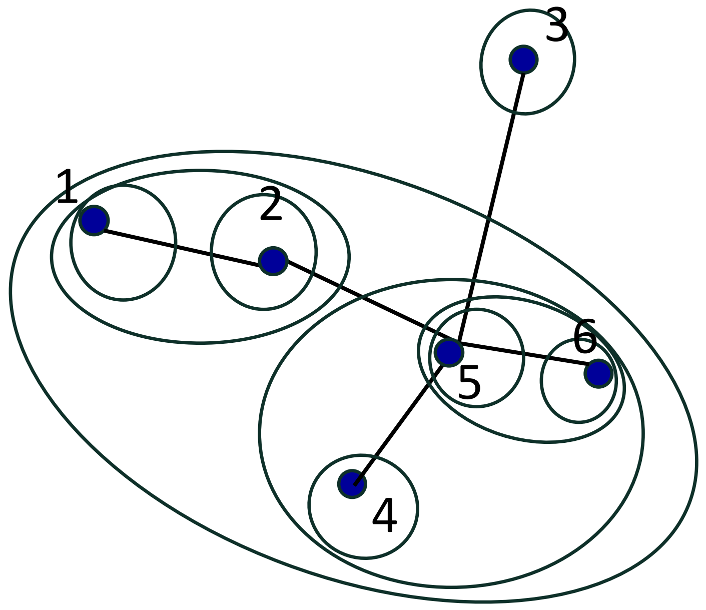
MST Clustering

1. Compute a minimum spanning tree for the dissimilarity graph

2. Repeat

    Create a new cluster by breaking the link corresponding to the largest dissimilarity

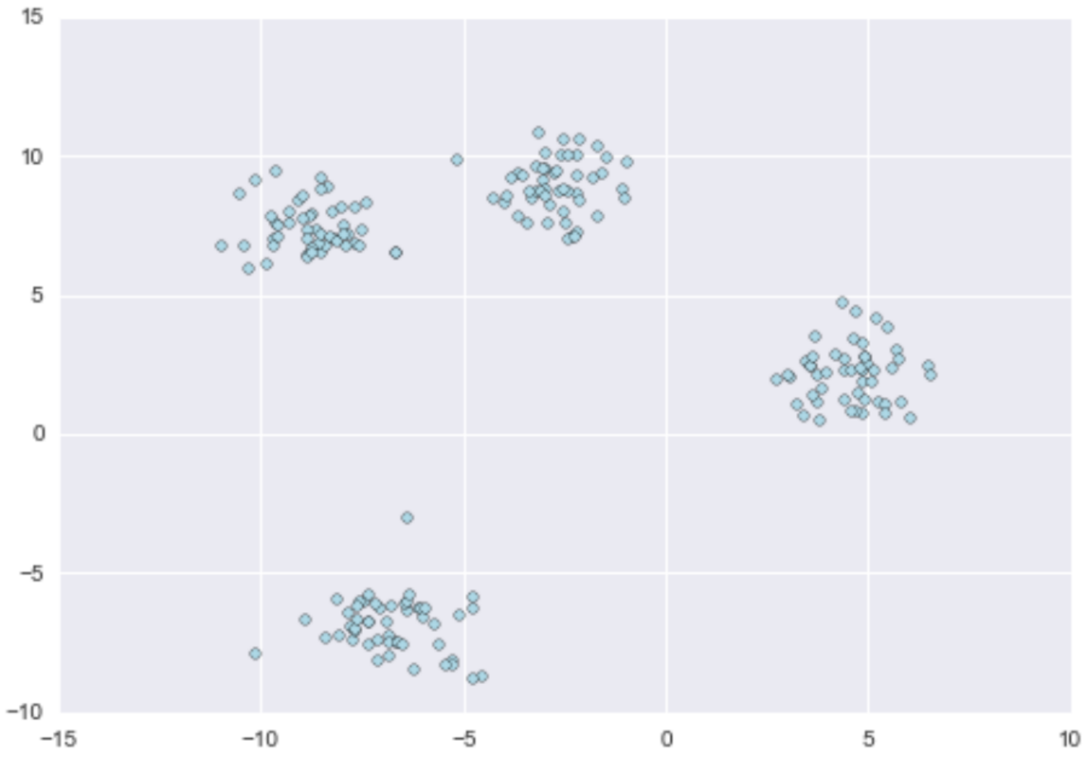
3. Until only singleton clusters remain



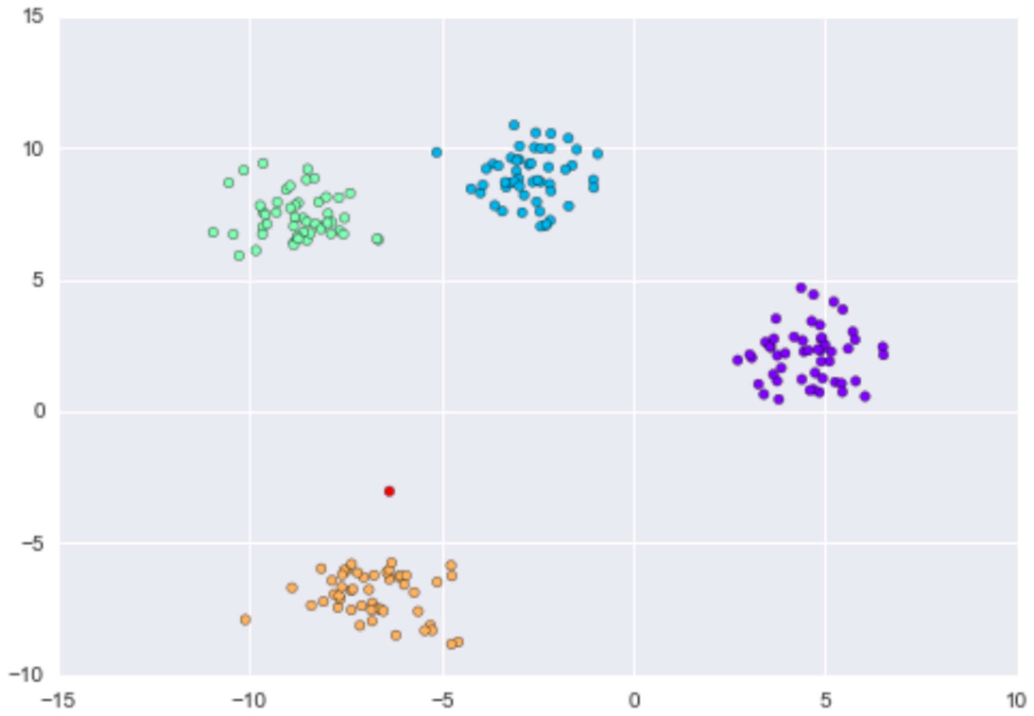
Example

Minimum Spanning Tree (MST) – Not implemented in sklearn
https://github.com/jakevdp/mst_clustering

```
In [2]: from sklearn.datasets import make_blobs
        X, y = make_blobs(200, centers=4, random_state=42)
        plt.scatter(X[:, 0], X[:, 1], c='lightblue');
```



```
In [3]: from mst_clustering import MSTClustering
        model = MSTClustering(cutoff_scale=2, approximate=False)
        labels = model.fit_predict(X)
        plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='rainbow');
```

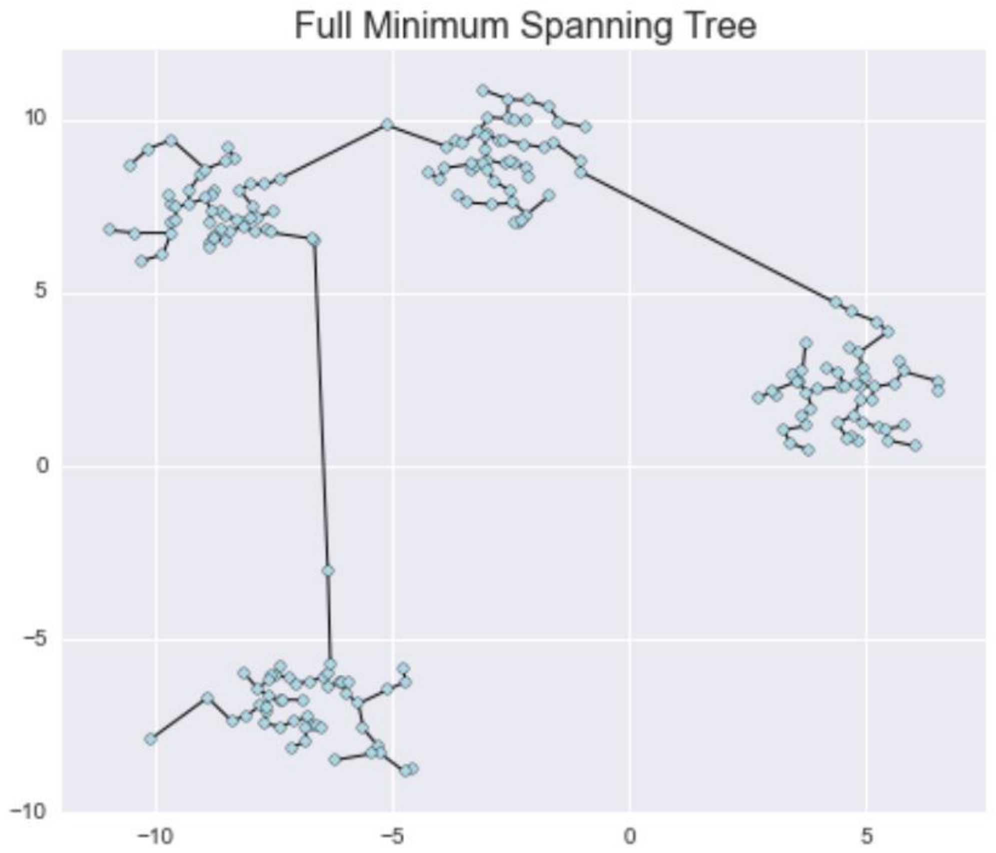


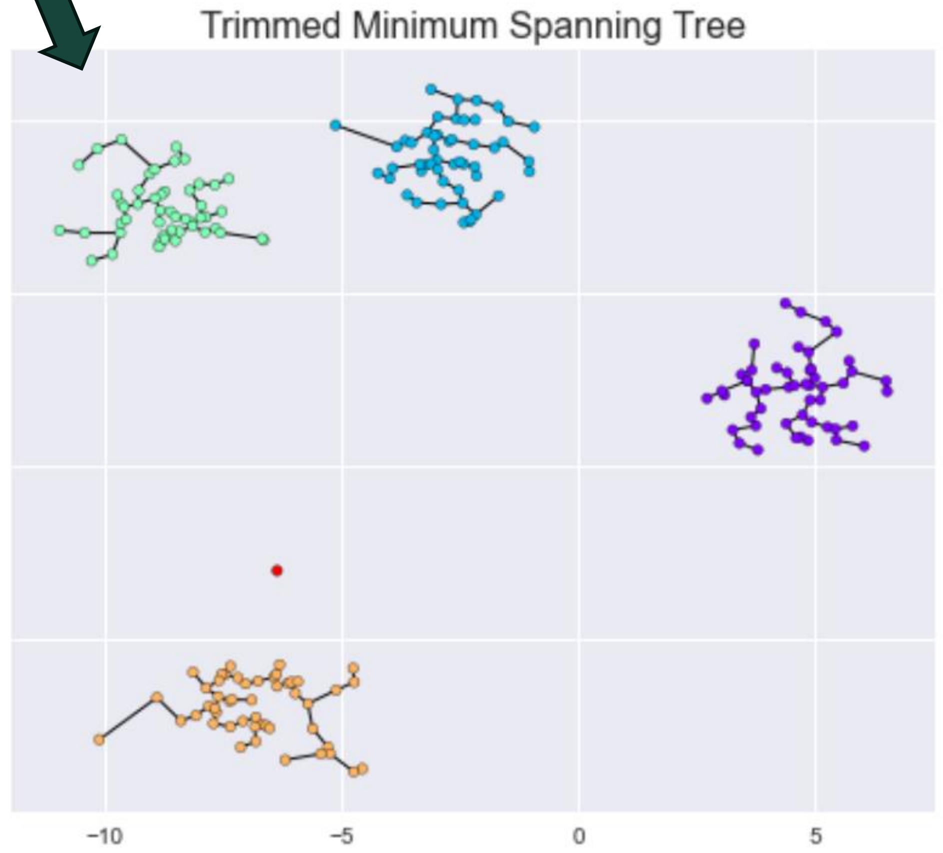
Example

Clusters derived from the minimum spanning tree, by removing all graph edges larger than the specified cutoff_scale parameter (cutoff_scale = 2).

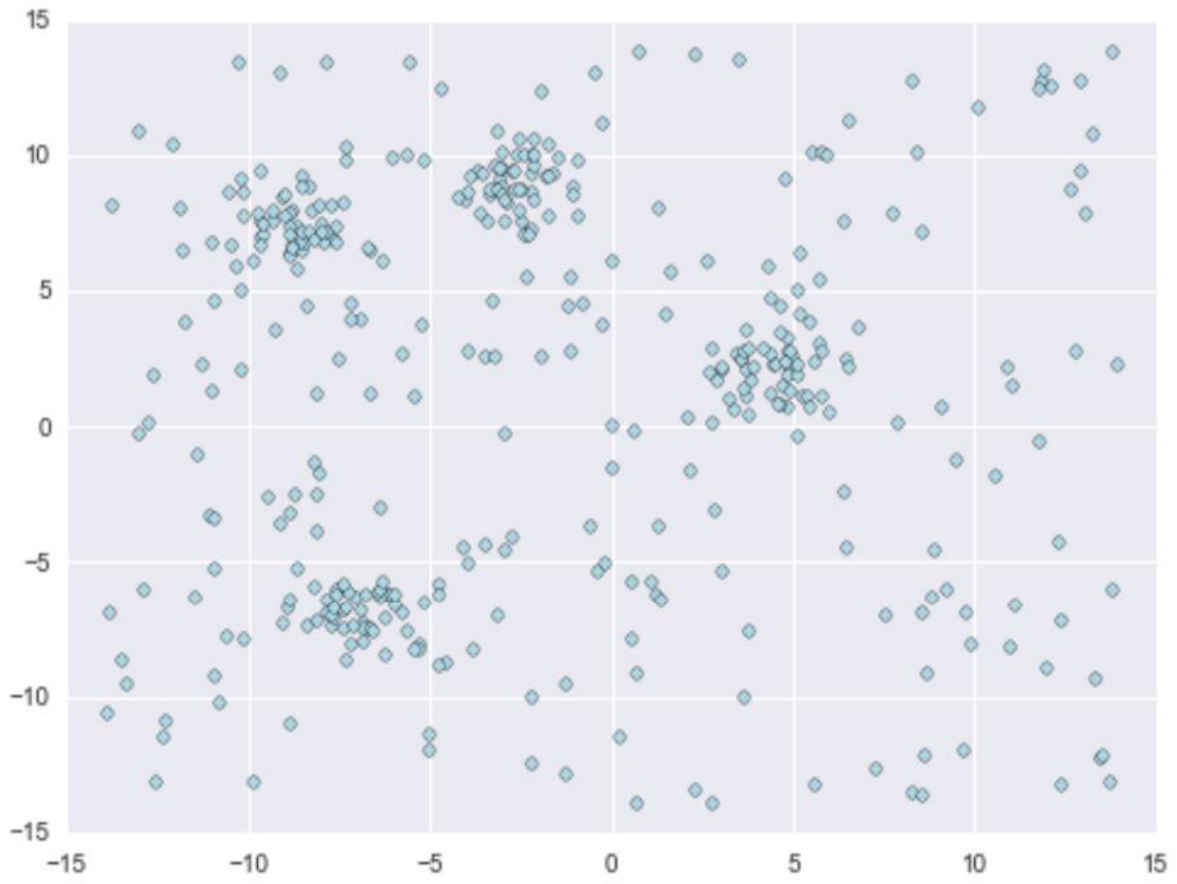
Points which remain connected after the truncation are given the same label.

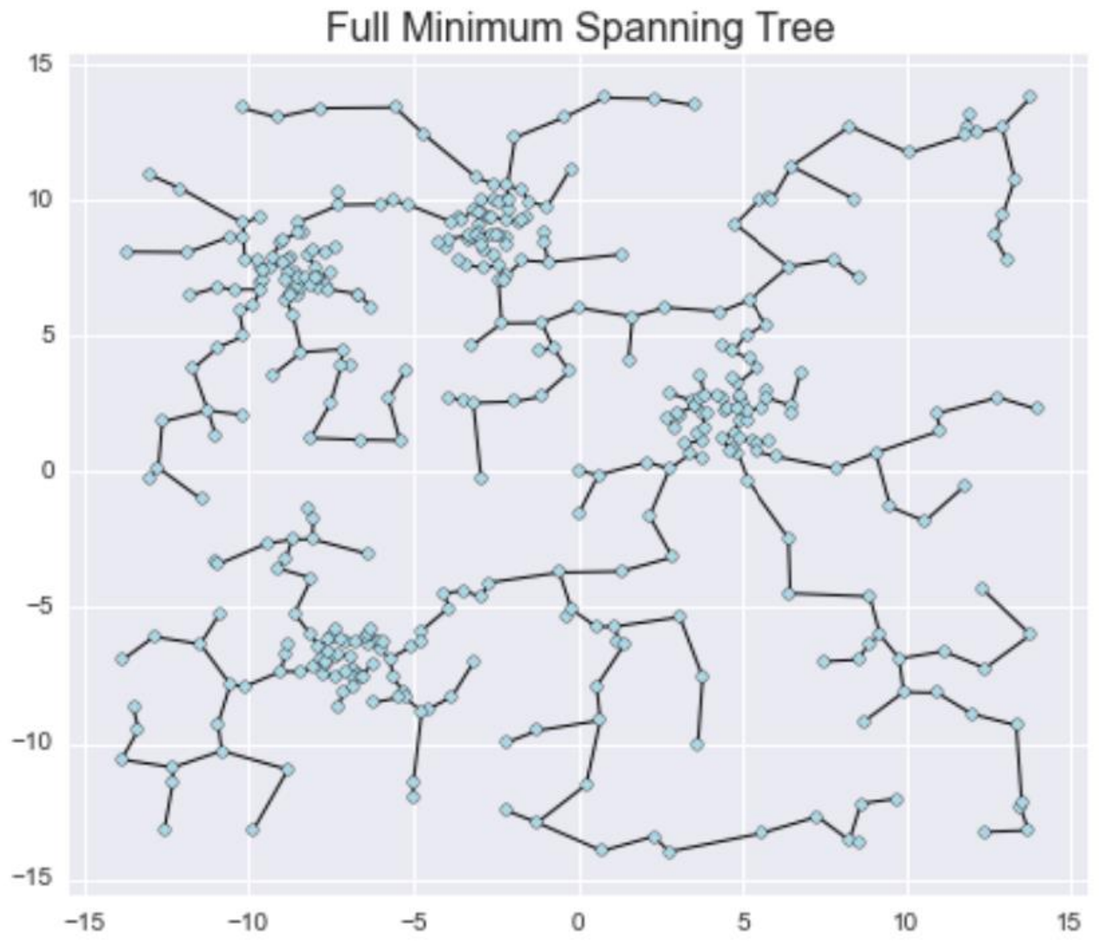
```
In [4]: plot_mst(model)
```



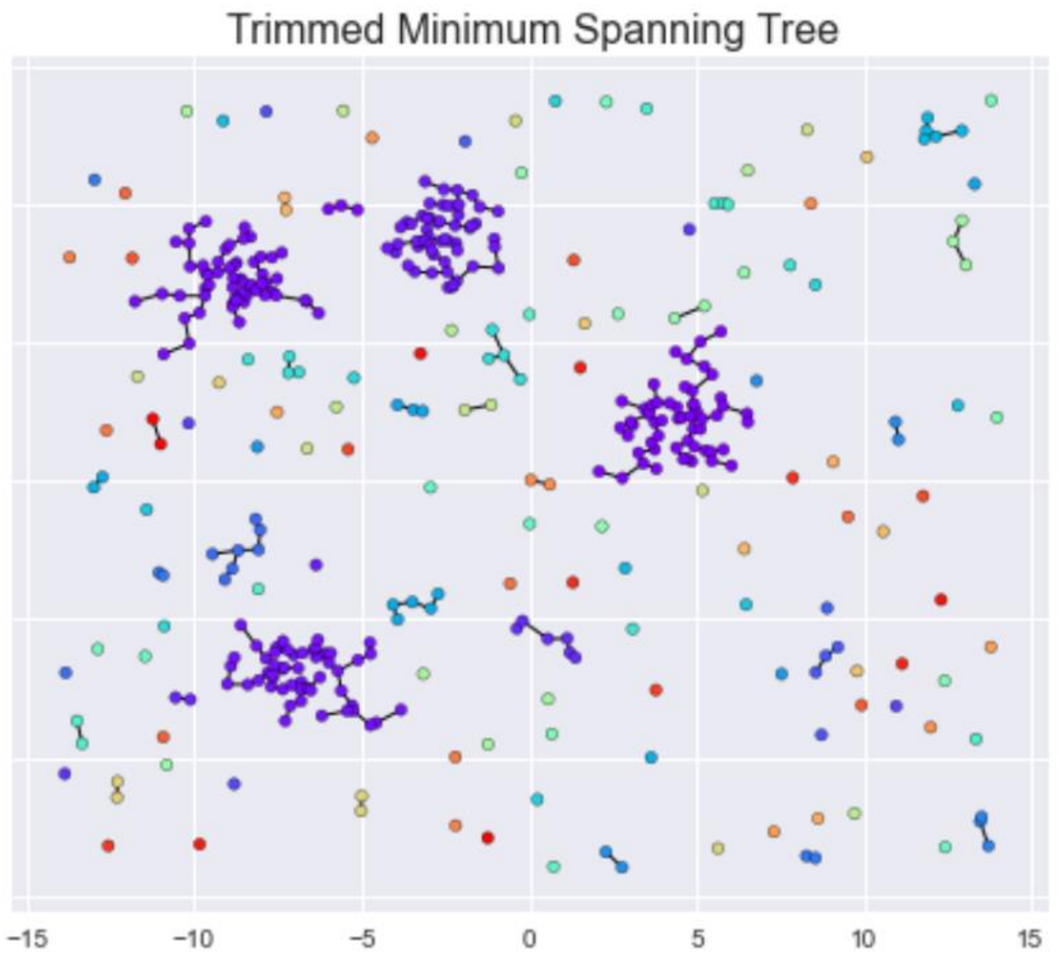


MST Clustering with **background noise**

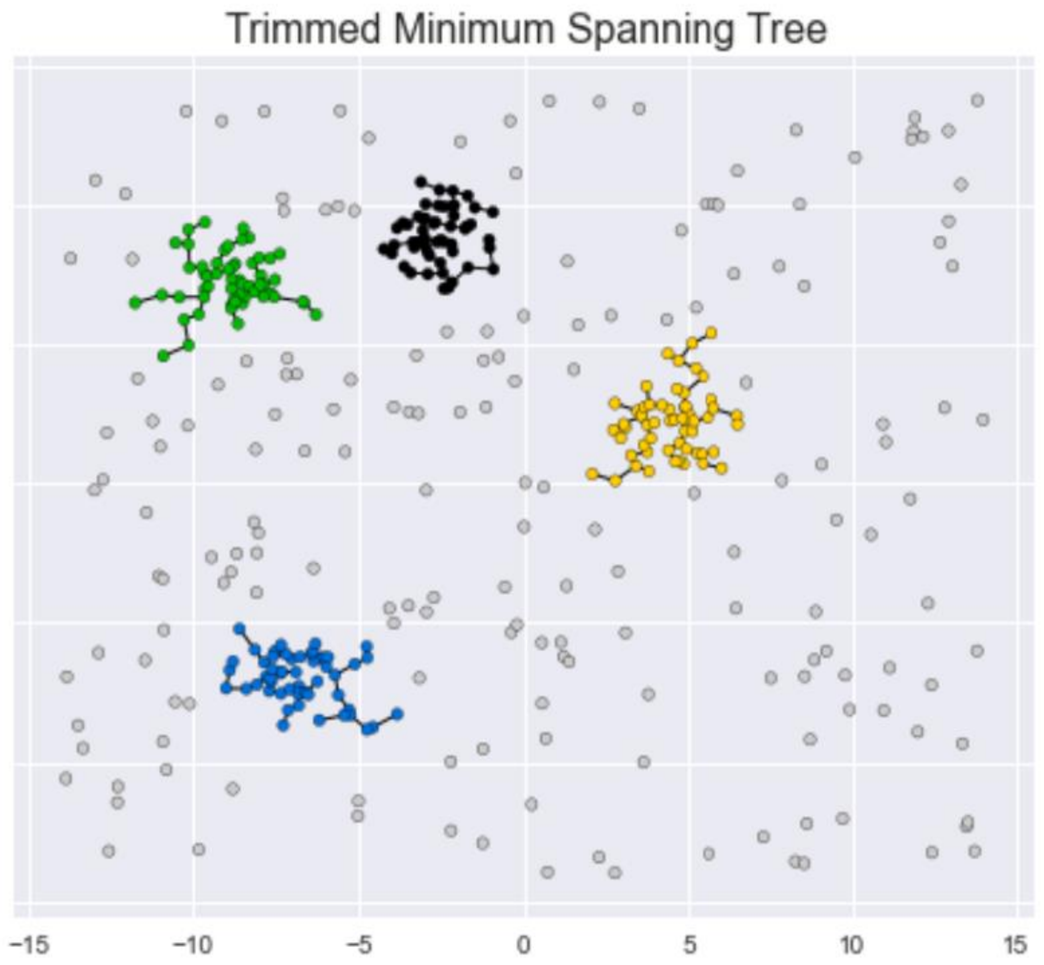




cutoff_scale=1



cutoff_scale=1, min_cluster_size=10



Agglomerative vs Divisive

- Bottom up (agglomerative):
  - Clustering decision based on local patterns
  - Does not take into account global distributions
  - Merging decisions are final
- Top down (divisive):
  - Early decisions use information based on global distribution
  - Challenge: how to partition a cluster into 2 smaller clusters?
    - 2^{n-1} -1 possibilities so heuristics are used
  - Splitting decisions are final

Example – Single Link (MIN) Agglomerative Clustering

Yearly income with different education levels and work experience

ID	Major	Degree	Salary (K)	Years
S1	CS	MS	48	2
S2	EE	MS	51	2
S3	CS	BS	38	1
S4	CS	MS	85	4
S5	EE	BS	41	3
S6	CS	MS	90	5
S7	CS	MS	48	1
S8	Cs	MS	48	3

Example

Cluster based on salary attribute

Cluster ID	Salary
C1	38
C2	41
C3	48
C4	48
C5	48
C6	51
C7	85
C8	90

Example

Compute pairwise distances

	C1	C2	C3	C4	C5	C6	C7	C8
C1	0							
C2	3	0						
C3	10	7	0					
C4	10	7	0	0				
C5	10	7	0	0	0			
C6	13	10	3	3	3	0		
C7	47	44	37	37	37	34	0	
C8	52	49	42	42	42	39	5	0

Example

Merge closest clusters (MIN) and recompute cluster distances

	C1	C2	C345	C6	C7	C8
C1	0					
C2	3	0				
C345	10	7	0			
C6	13	10	3	0		
C7	47	44	37	34	0	
C8	52	49	42	39	5	0

Example

Merge again and recompute cluster distances

	C12	C345	C6	C7	C8
C12	0				
C345	7	0			
C6	10	3	0		
C7	44	37	34	0	
C8	49	42	39	5	0

Repeat until only one cluster is obtained

Example

Results

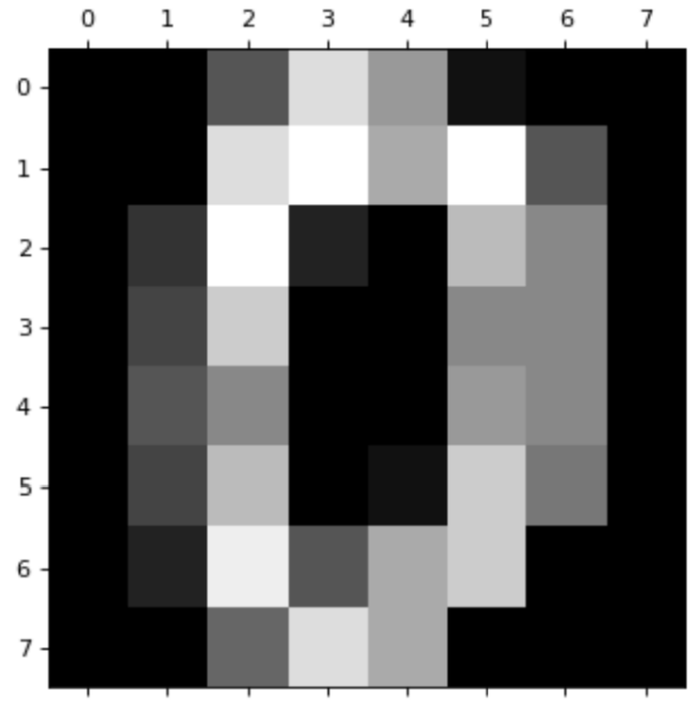
	ID	Major	Degree	Salary (K)	Years
C12	S3	CS	BS	38	1
	S5	EE	BS	41	3
C3456	S1	CS	MS	48	2
	S7	CS	MS	48	1
	S8	CS	MS	48	3
	S2	EE	MS	51	2
C78	S4	CS	MS	85	4
	S6	CS	MS	90	5

Coding

- K-Means
  - sklearn.cluster.KMeans
- Hierarchical (sklearn.cluster.AgglomerativeClustering)
  - Min Proximity (linkage: “single”)
  - MaxProximity (linkage: “complete”)
  - GroupAverage(linkage: “average”)
  - Centroid (Ward’s Method) (linkage: “ward”)
- Minimum Spanning Tree (MST) –Not implemented in sklearn
  - https://github.com/jakevdp/mst_clustering

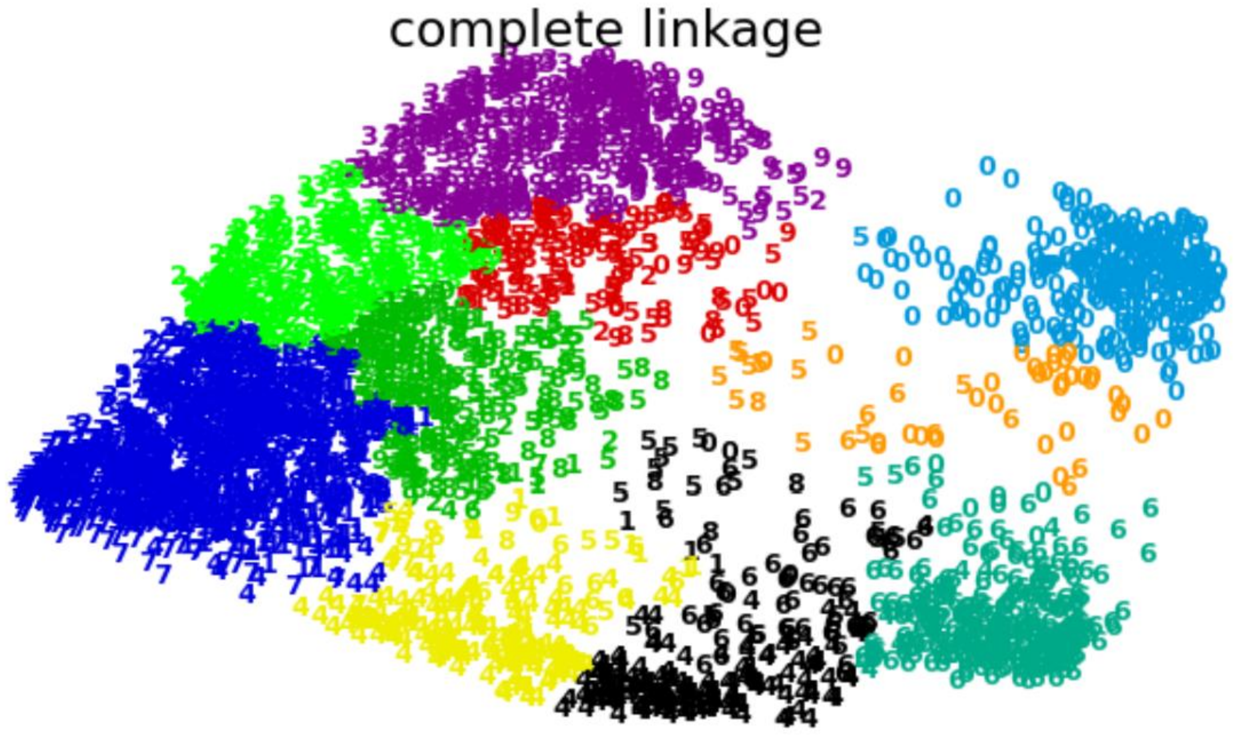
Example 1 –Digits

```
from sklearn.datasetsimport load_digits

digits = load_digits()

digits.data.shape

# Visualize 1 sample

x = digits.data[0]

axprops= dict(xticks=[], yticks=[])

barprops= dict(aspect='auto', cmap=plt.cm.binary, interpolation='nearest')

fig = plt.figure()

ax1 = fig.add_axes([0, 0, 0.5, 0.8], **axprops) #axes denoted as: [left bottom width height]

ax1.imshow(x.reshape((8,8)), **barprops)
```

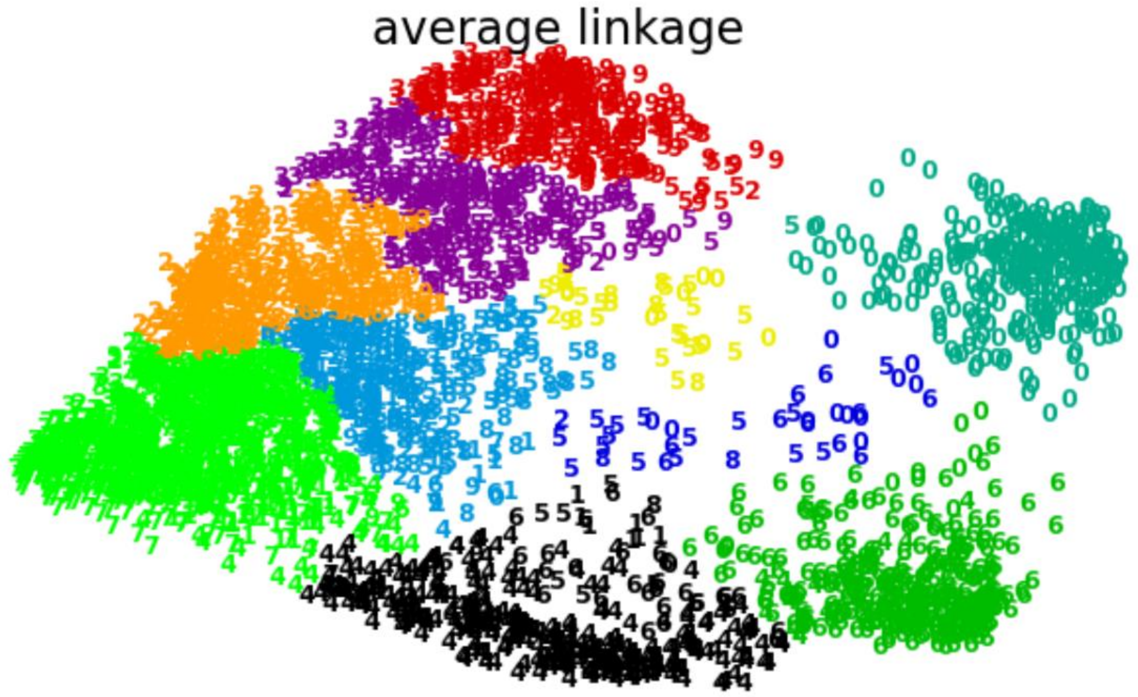


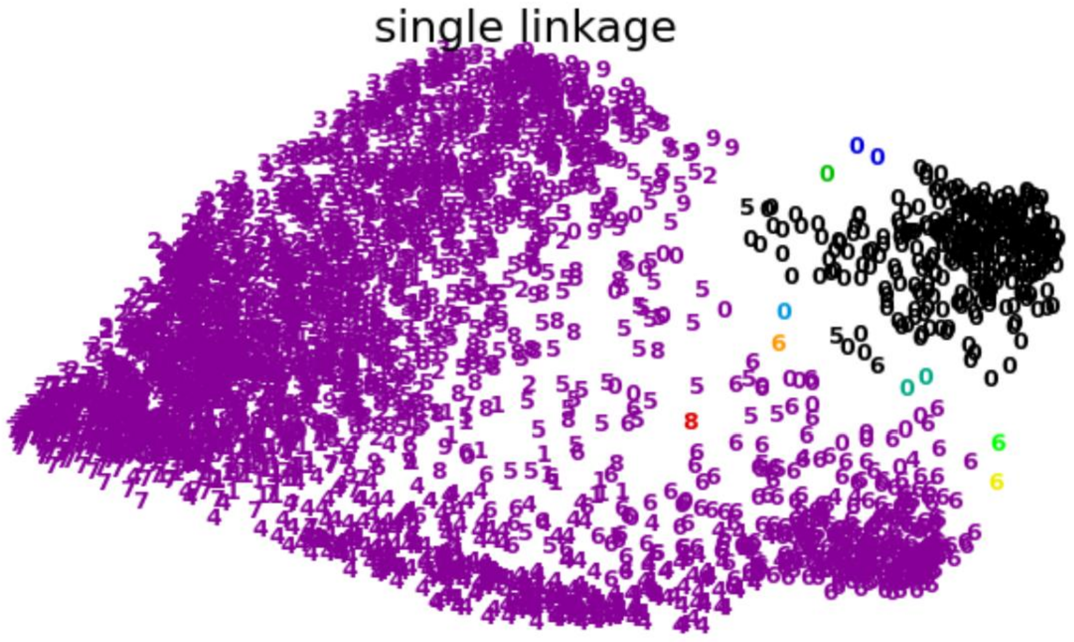
Example 1 –Digits -Hierarchical

```
from sklearn.clusterimport AgglomerativeClustering

cls= AgglomerativeClustering(linkage='complete', n_clusters=10)

cls.fit(digits.data)

y_cls= cls.labels_

plt.scatter(xt[:,0],xt[:,1],c=y_cls)

plt.show()
```

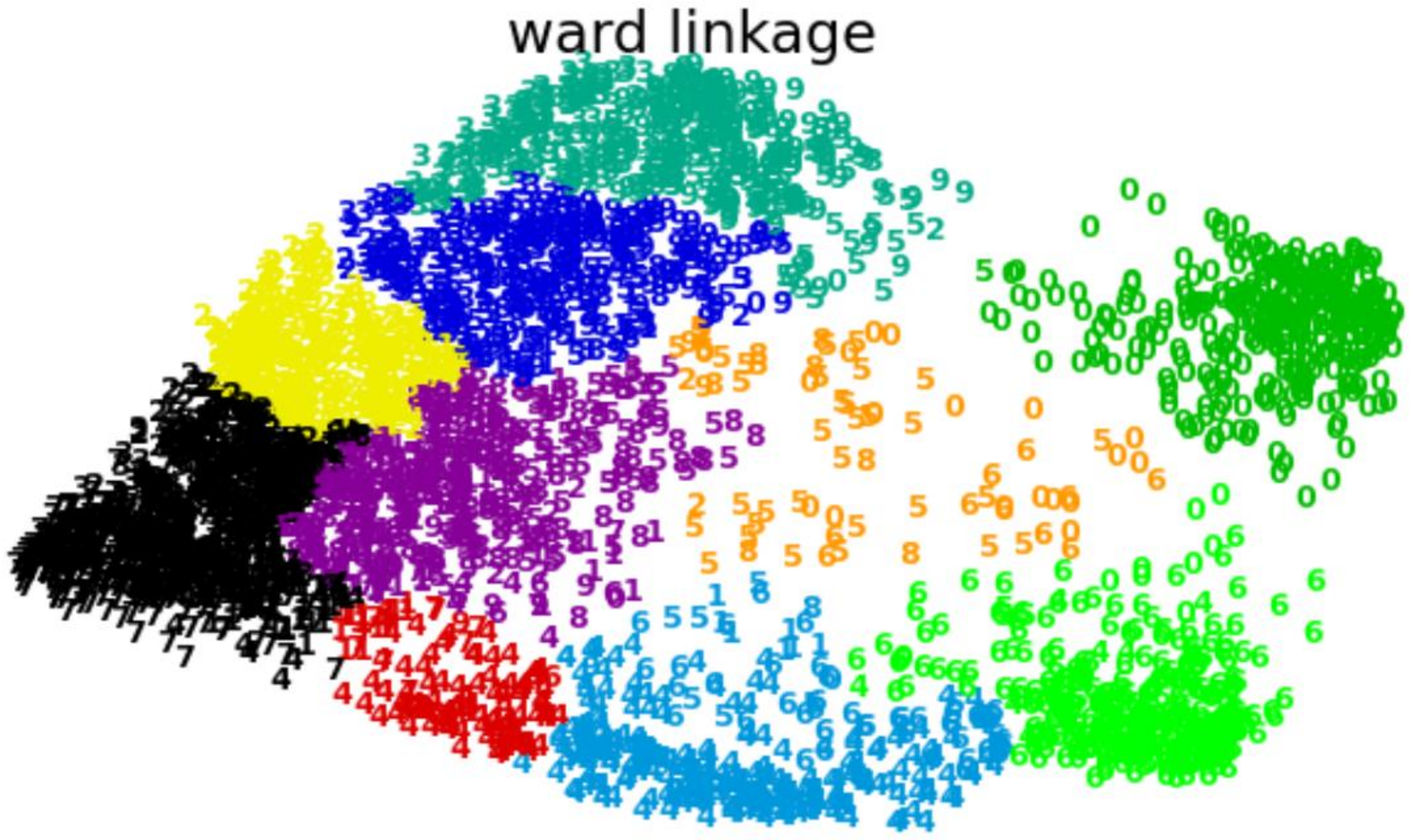


Example 1 –Digits -Hierarchical





Example 1 –Digits -Hierarchical



https://scikit-learn.org/0.24/auto_examples/cluster/plot_digits_linkage.html

Example 2 -Moons

```
from sklearn.datasetsimport make_moons

Xm, y = make_moons(200, noise=.05, random_state=0)

plt.scatter(Xm[:, 0], Xm[:, 1], s=50, cmap='viridis');
```

Example 2 –Moons -Clustering

```
# Improve by specifying a sample is only connected to its 10 nearest neighbors
from sklearn.neighborsimport kneighbors_graph
connectivity = kneighbors_graph(Xm, n_neighbors=10, include_self=False)
connectivity = 0.5 * (connectivity + connectivity.T)
cls= AgglomerativeClustering(linkage='ward', n_clusters=2, connectivity=connectivity)
cls.fit(Xm)
y_cls= cls.labels_
plt.scatter(Xm[:,0],Xm[:,1],c=y_cls)
plt.show()
```