CHAPTER 2

Unsupervised Learning

(Unsupervised Learning with Python)

Content

Python

Show unsupervised learning tools via examples

- The tools are:

1. K-means clustering
2. Probabilistic clustering via EM algorithm
3. Hierarchical clustering
4. Ensemble clustering

K-MEANS CLUSTERING

Review of K-means Clustering

- The goal of K-means clustering is to minimize the following objective function

$$\min_{I_{ik},\boldsymbol{c}_k} J(I_{ik},\boldsymbol{c}_k), where\ J(I_{ik},\boldsymbol{c}_k) = \sum_{i=1}^{n}\sum_{k=1}^{c} I_{ik}\left|\left|\boldsymbol{x}_i - \boldsymbol{c}_k\right|\right|^2$$

where $I_{ik} = \{0, 1\}$ are binary variables and $\boldsymbol{c}_k$ are the cluster centers.

Review of K-means Clustering

Step 1: Updating Assignment

- Assign each sample to the closest centroid
- That is,

$$I_{ik} = 1 \text{ if } \left|\left|\boldsymbol{x}_i - \boldsymbol{c}_k\right|\right|^2 \leq \left|\left|\boldsymbol{x}_i - \boldsymbol{c}_j\right|\right|^2, \text{ for j=1,…c}$$

$$I_{ik} = 0 \quad \text{otherwise} \qquad .$$

Step 2: Updating Centroid

- Compute the centroids by the following formula

$$\boldsymbol{c}_k = \frac{\sum_{i=1}^{n} I_{ik}\boldsymbol{x}_i}{\sum_{i=1}^{n} I_{ik}}$$

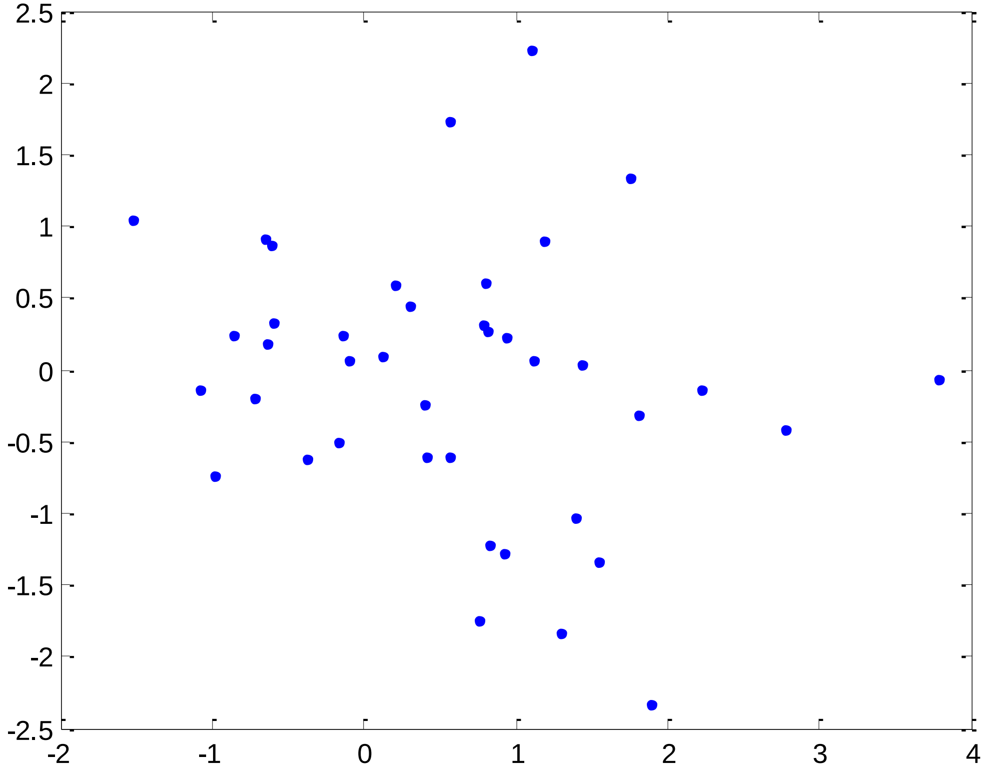
Mean of the data

Case Study

- Example 1: A data with two groups but they are closed to each other.
- Example 2: A data with two well-separated clusters.
- Example 3: A data with two clusters and outliers.
- Example 4: Manhattan distance - A data with two clusters and outliers.
- Example 5: Real-data experiment: Iris dataset.
- Example 6: Real-data experiment: prima_diabetes dataset.

Example 1 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data1.csv).
- Apply K-means clustering algorithm to partition the data into two different clusters.



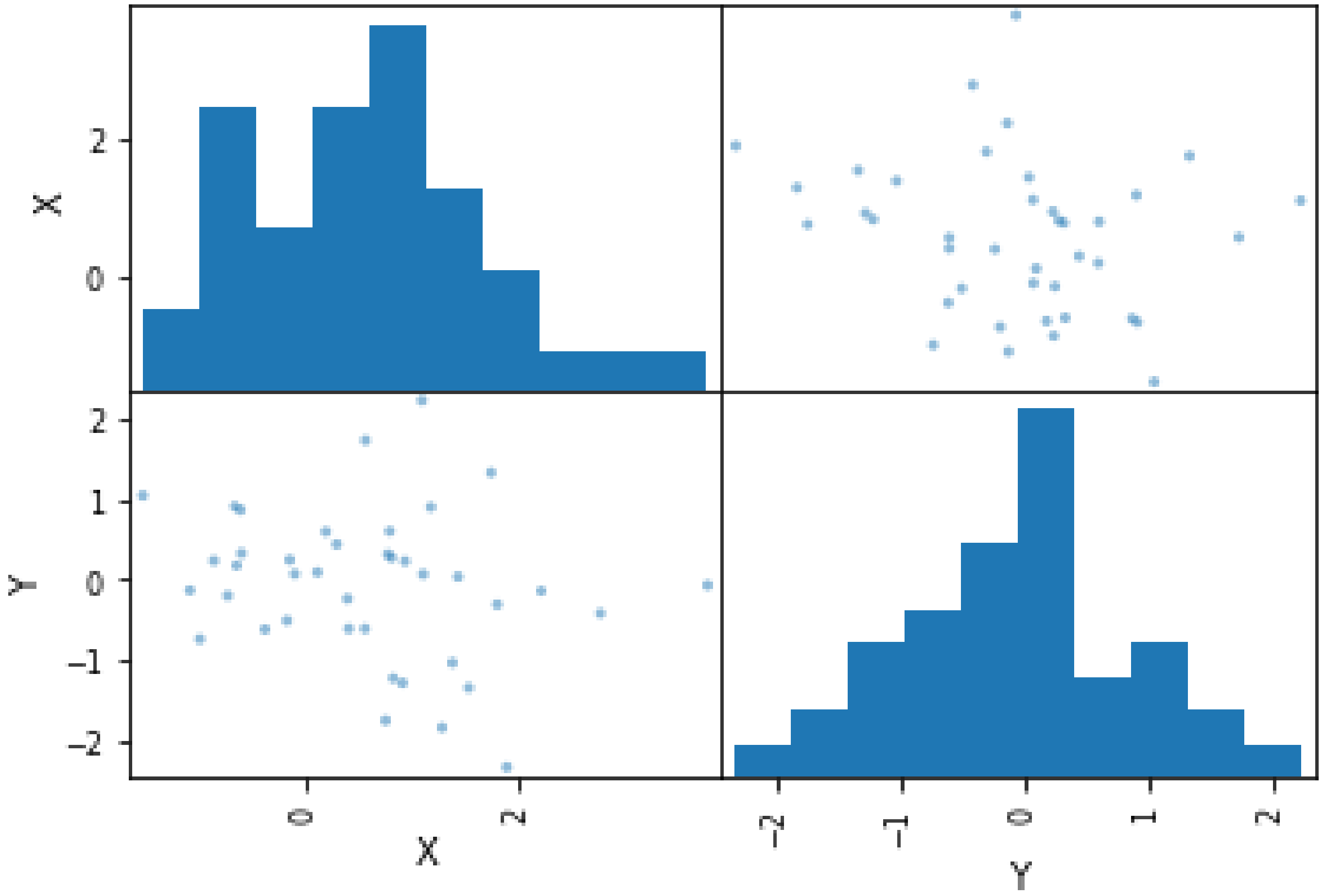
Example 1

- **Answer**: Python code for K-means clustering

```
#Example 1
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
data = pd.read_csv('Data\\kmeans_data1.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
kmeans = KMeans(n_clusters=2).fit(data)
#perform K-means clustering with number of clusters = 2
centroids = kmeans.cluster_centers_ #Extract the cluster centroids
labels = kmeans.labels_ #Extract the labels of clusters
plt.figure(); #Plot the figure
plt.plot(data.iloc[labels==0,0],data.iloc[labels==0,1],'g.');
#plot the data with label = 0 (color: g)
plt.plot(data.iloc[labels==1,0],data.iloc[labels==1,1],'.');
#plot the data with label = 1 (color: b)
plt.plot(centroids[:,0],centroids[:,1],'ro') #plot the cluster centroid
```

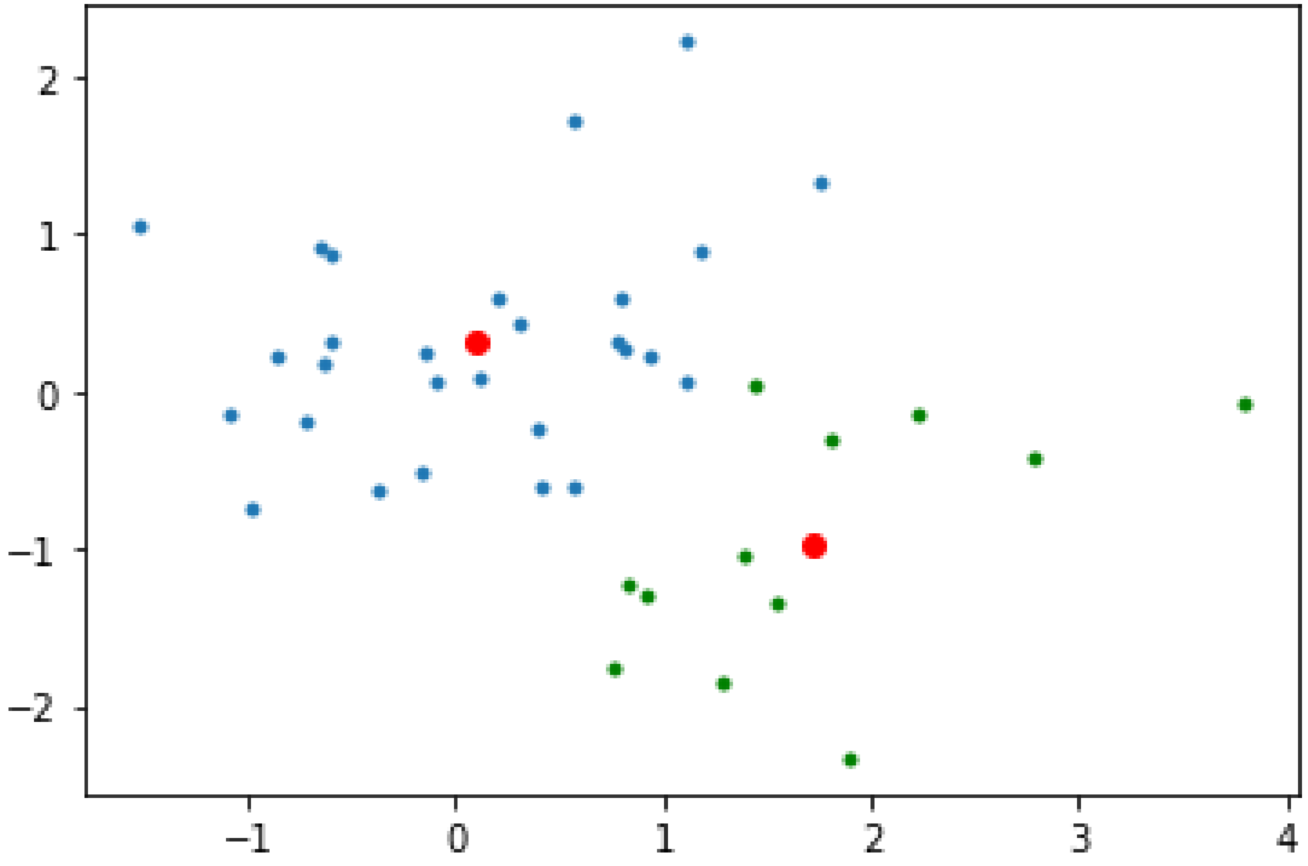
Example 1

- Data Visualization using scatter matrix



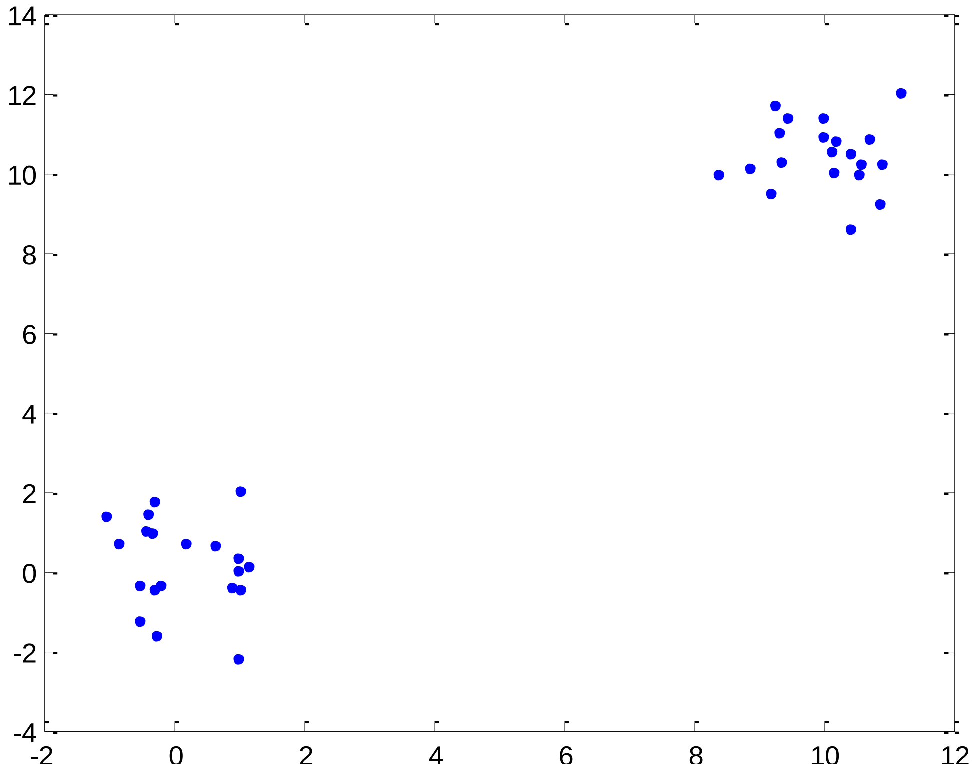
Example 1

- Clustering results



Example 2 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data2.csv).
- Apply K-means clustering algorithm to partition the data into two different clusters.



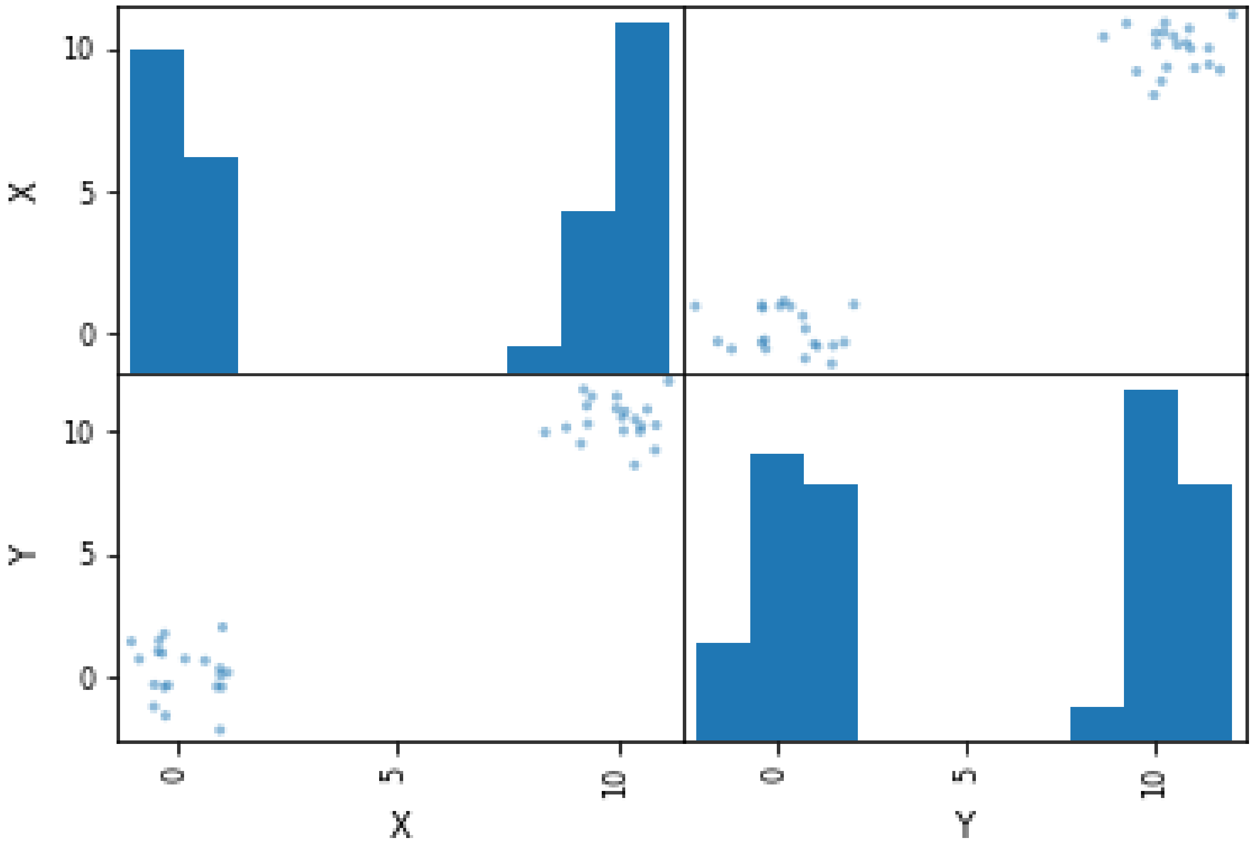
Example 2

- **Answer**: Python code for K-means clustering

```
#Example 2
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
data = pd.read_csv('Data\\kmeans_data2.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
kmeans = KMeans(n_clusters=2).fit(data)
#perform K-means clustering with number of clusters = 2
centroids = kmeans.cluster_centers_ #Extract the cluster centroids
labels = kmeans.labels_ #Extract the labels of clusters
plt.figure(); #Plot the figure
plt.plot(data.iloc[labels==0,0],data.iloc[labels==0,1],'g.');
#plot the data with label = 0 (color: g)
plt.plot(data.iloc[labels==1,0],data.iloc[labels==1,1],'.');
#plot the data with label = 1 (color: b)
plt.plot(centroids[:,0],centroids[:,1],'ro') #plot the cluster centroid
```

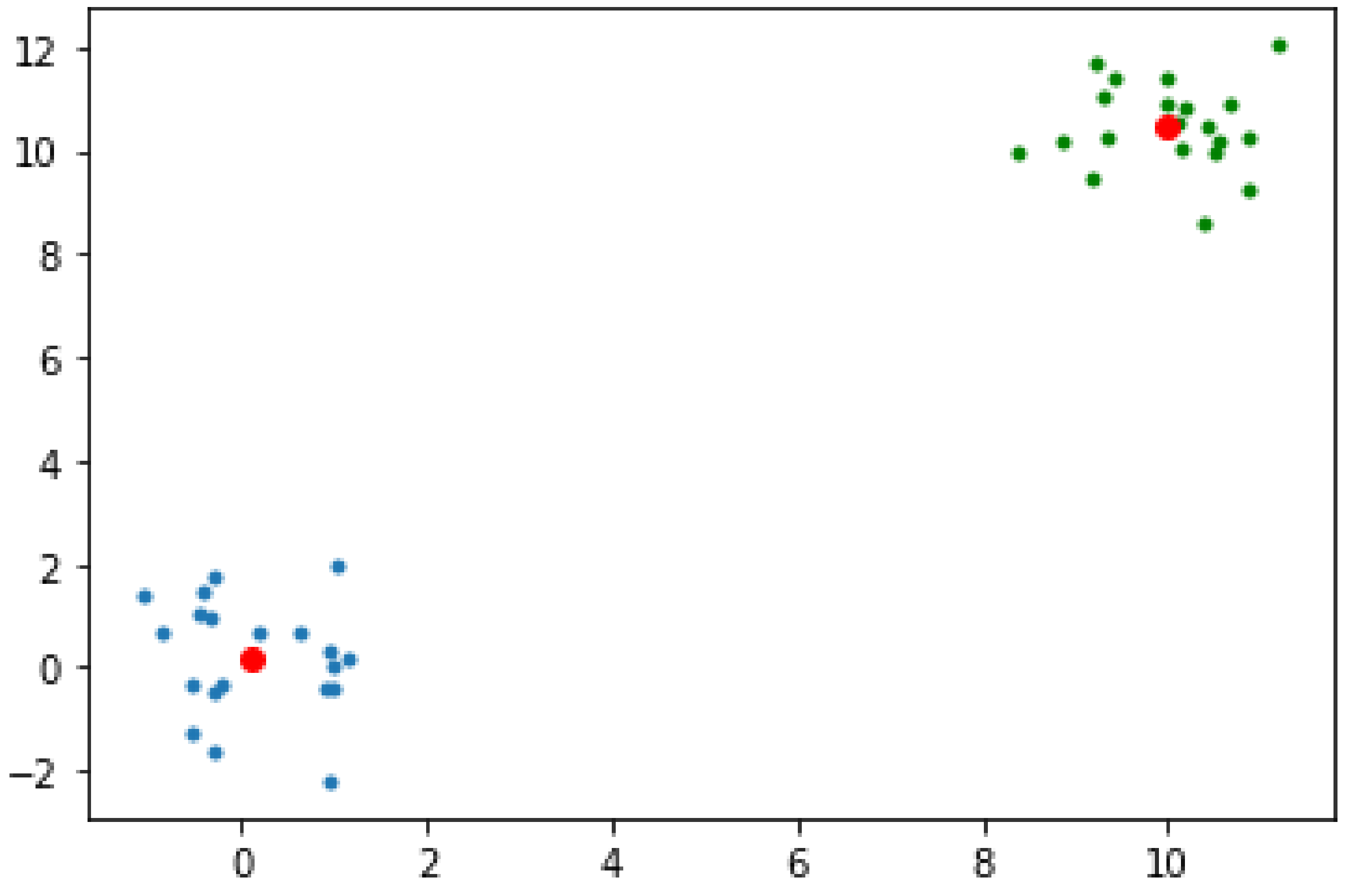
Example 2

- Data visualization using scatter matrix



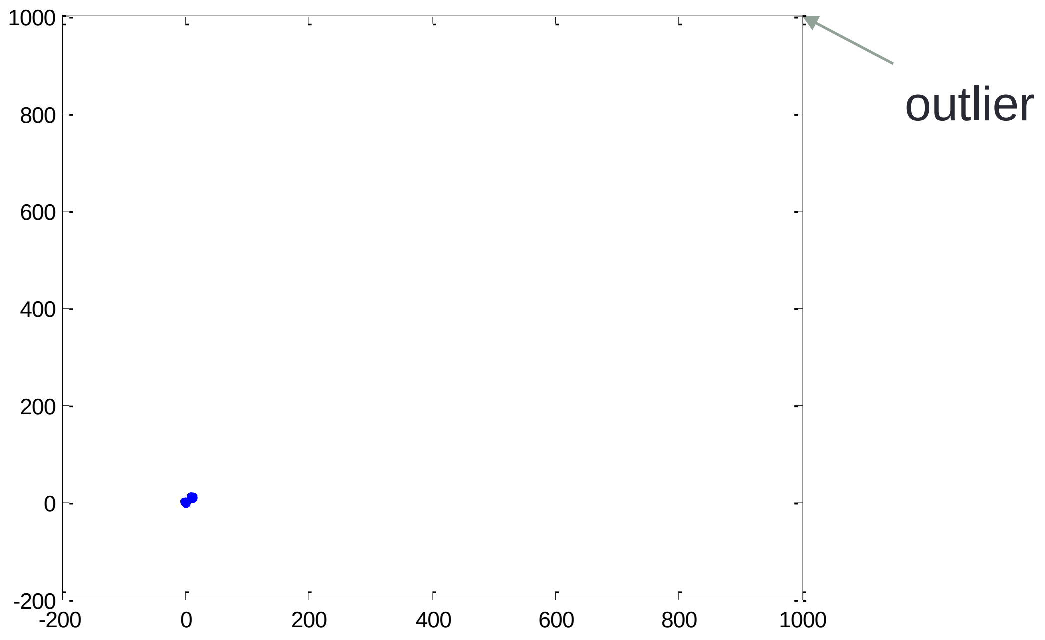
Example 2

- Clustering results



Example 3 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data3.csv).
- 20 samples for group 1 while 20 samples for group 2. 10 samples are outliers.
- Apply K-means clustering algorithm to partition the data into two different clusters.



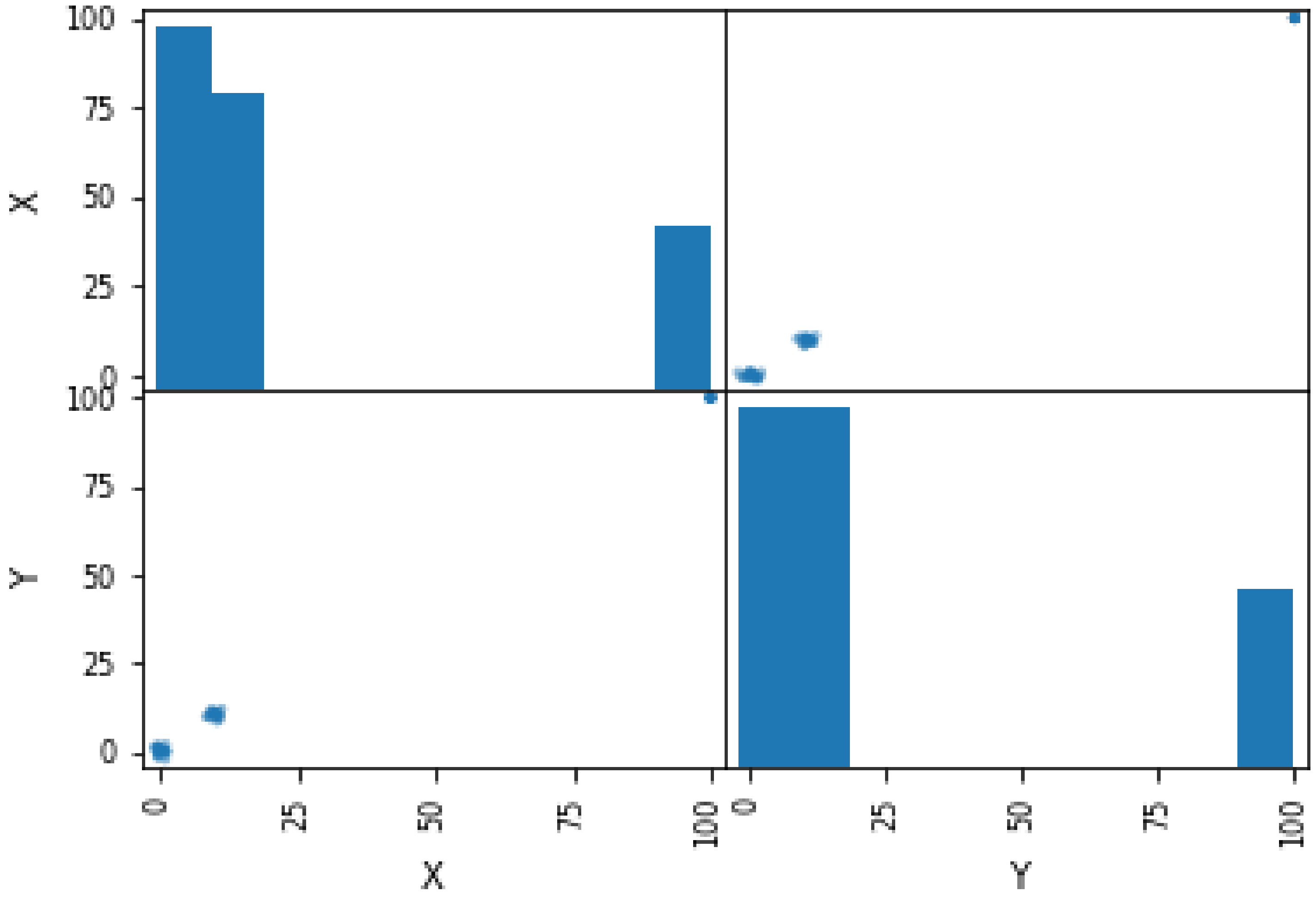
Example 3

- **Answer**: Python code for K-means clustering

```
#Example 3
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
data = pd.read_csv('Data\\kmeans_data3.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
kmeans = KMeans(n_clusters=2).fit(data)
#perform K-means clustering with number of clusters = 2
centroids = kmeans.cluster_centers_ #Extract the cluster centroids
labels = kmeans.labels_ #Extract the labels of clusters
plt.figure(); #Plot the figure
plt.plot(data.iloc[labels==0,0],data.iloc[labels==0,1],'g.');
#plot the data with label = 0 (color: g)
plt.plot(data.iloc[labels==1,0],data.iloc[labels==1,1],'.');
#plot the data with label = 1 (color: b)
plt.plot(centroids[:,0],centroids[:,1],'ro') #plot the cluster centroid
```

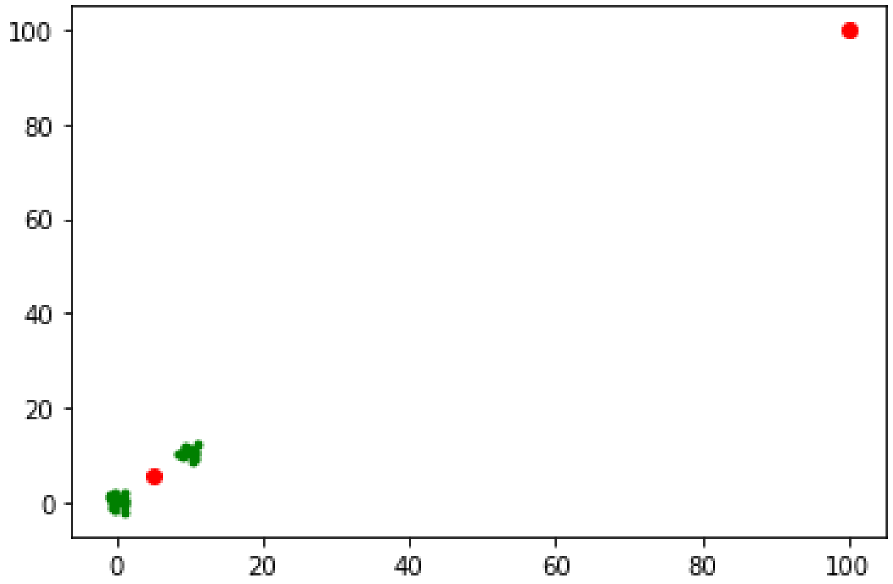
Example 3

- Data visualization using scatter matrix



Example 3

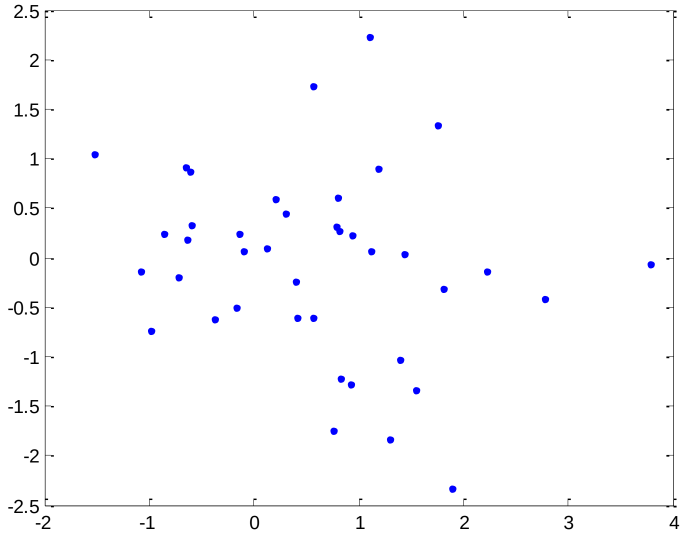
- Results shown in figure

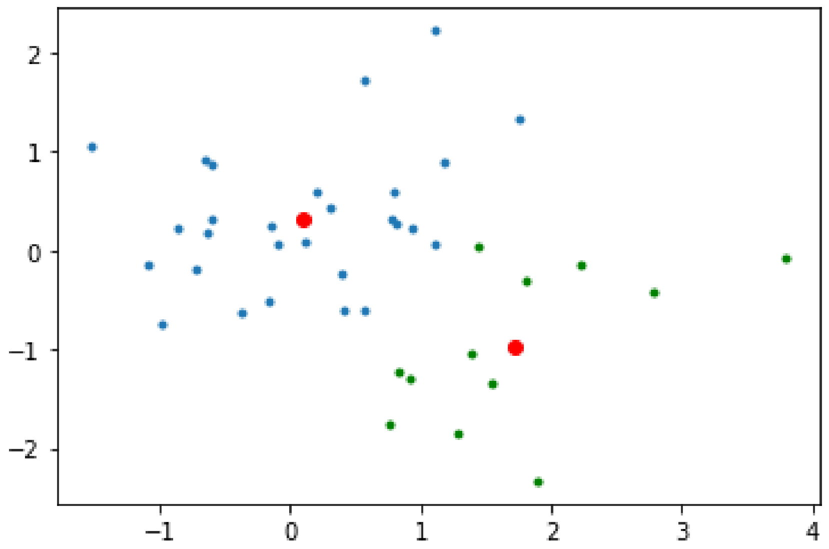


- The two groups are clustered together while the outliers are clustered to form another group.
- Reason: mean is sensitive to outliers (Re-call the updating step of K-means clustering)

Comparison

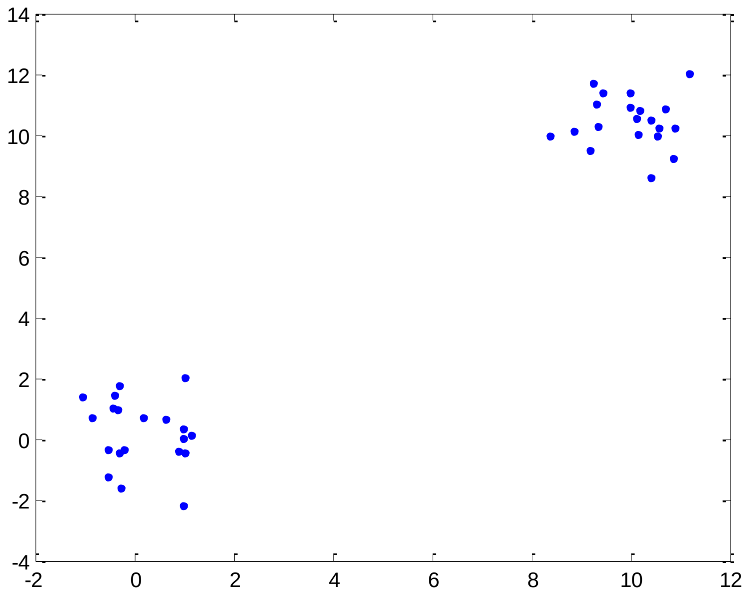
- These three examples show several characteristics of K-means clustering algorithm.
- Example 1: there are two groups but they are merged.
- Cluster results: can’t reveal the two groups

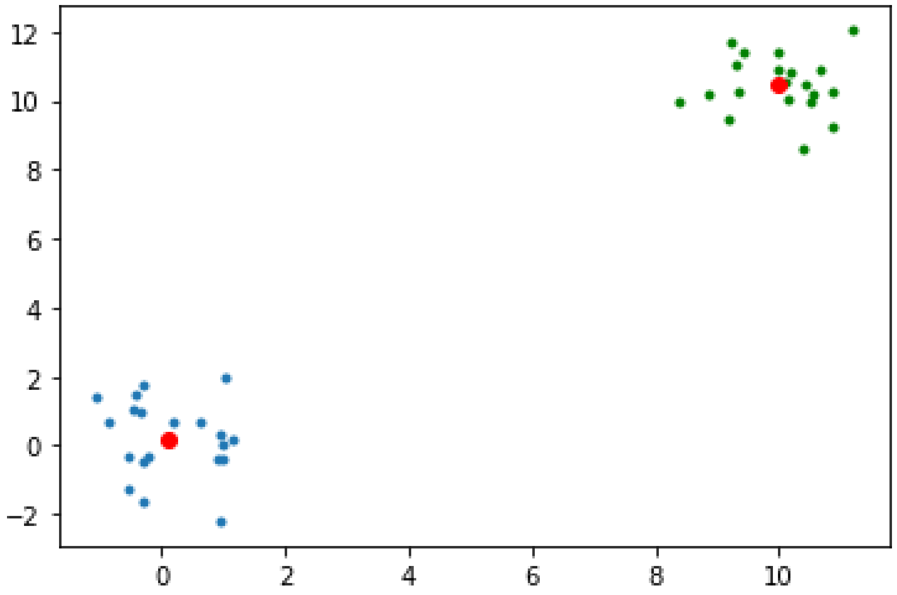




Comparison

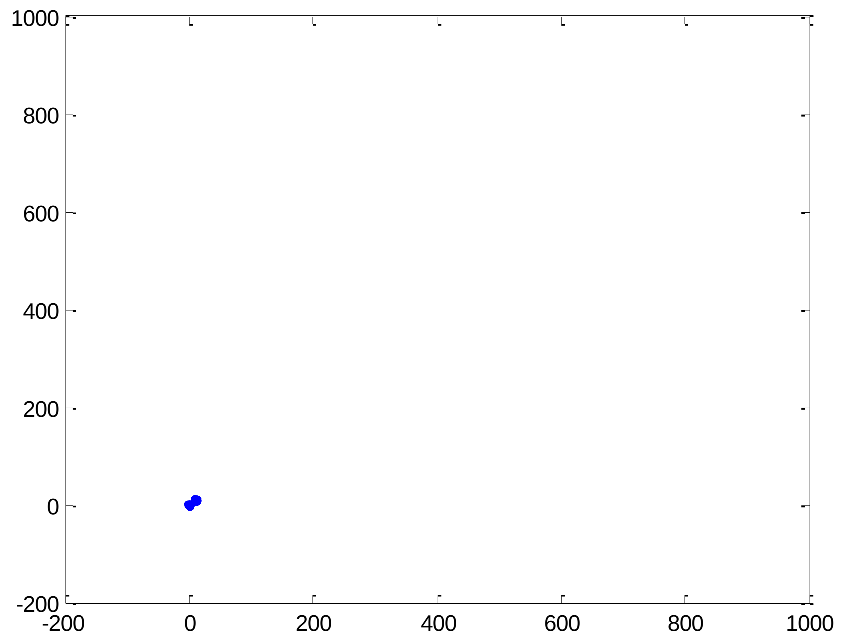
- Example 2: there are two groups and they are well-separated
- Cluster results: perfect

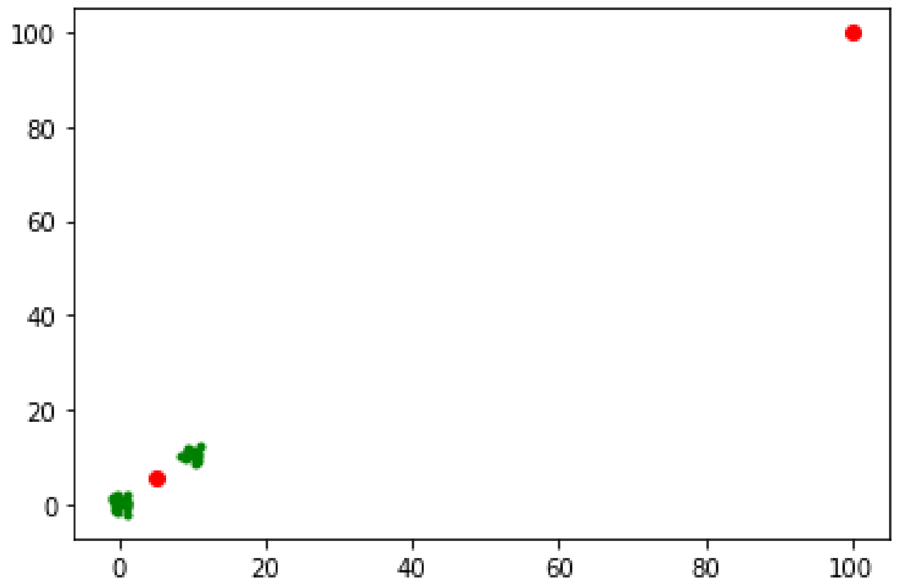




Comparison

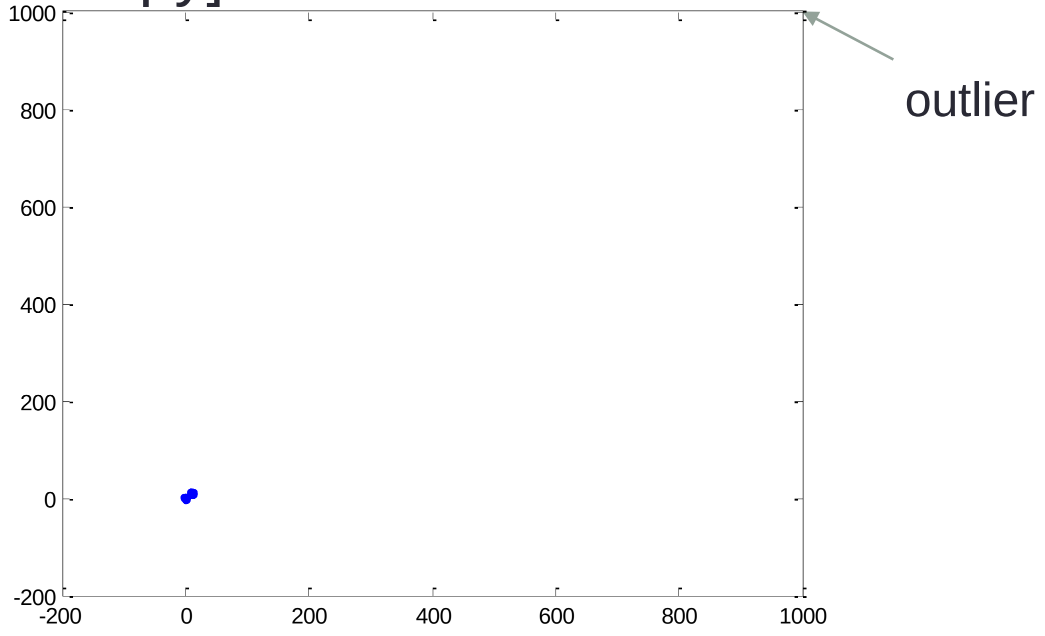
- Example 3: there are two groups with the presence of outliers
- Cluster results: the original two groups are combined and the outliers form a group.





Example 4 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data3.csv).
- 20 samples for group 1 while 20 samples for group 2. 10 samples are outliers.
- Apply K-means clustering algorithm with Manhattan distance to partition the data into two different clusters. [need kmeans.py]



Example 4

- **Answer**: Python code for K-means clustering

```
import pandas as pd #Import pandas module
from kmeans import Kmeans_manh #import kmeans.py
import matplotlib.pyplot as plt #Import the visualization module
data = pd.read_csv('Data\\kmeans_data3.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
labels,centroids=Kmeans_manh(data,n_clusters=2)
plt.figure(); #Plot the figure
plt.plot(data.iloc[labels==0,0],data.iloc[labels==0,1],'g.');
#plot the data with label = 0 (color: g)
plt.plot(data.iloc[labels==1,0],data.iloc[labels==1,1],'.');
#plot the data with label = 1 (color: b)
plt.plot(centroids[:,0],centroids[:,1],'ro') #plot the cluster centroid
```

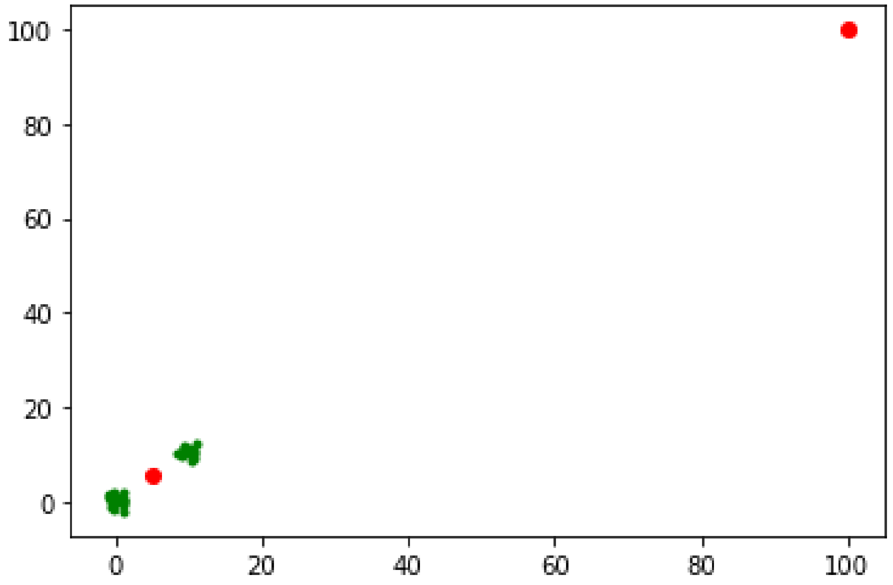
Answer [Reference]: Python code for Kmeans.py

```
import numpy as np

def Kmeans_manh(data,n_clusters):
    epsilon = 1e-4; iter = 0; residual = 1;
    randint = np.random.randint(0,data.shape[0],size=n_clusters)
#Randomly select two random integers
    X_new = data.iloc[randint,:].values
#Find random points as the initial guesses
    while (residual > epsilon) & (iter <100):
        X_old = X_new.copy(); #Assign the updated one to the old one
        #Update cluster labels
        #Compute distance
        dist = [];
        for x in X_new:
            dist.append(np.linalg.norm(data-x,axis=1,ord=1))
        labels = np.argmin(np.array(dist),axis=0)

        #Update cluster centroids
        for ii,x in enumerate(X_old):
            X_new[ii,:] = np.median(data[labels==ii],axis=0)
        residual = np.linalg.norm(X_new-X_old)
        iter +=1
        return labels,X_new
```

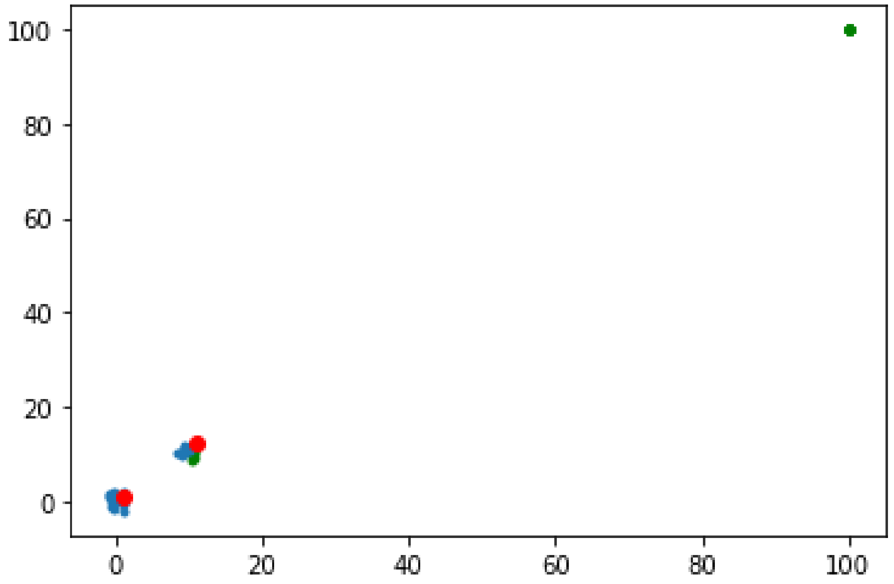
Example 3 vs Example 4

- Manhattan distance is more robust to the presence of outliers.

Example 3. Results



Example 4. Results



Example 5 - Question

- Cluster the Iris dataset with K-means clustering algorithm.
- There are 3 groups with 150 samples and 4 attributes.
- The three groups are Iris-setosa, Iris-versicolor and Iris-virginica.
- Suppose we know that the dataset has 3 groups.

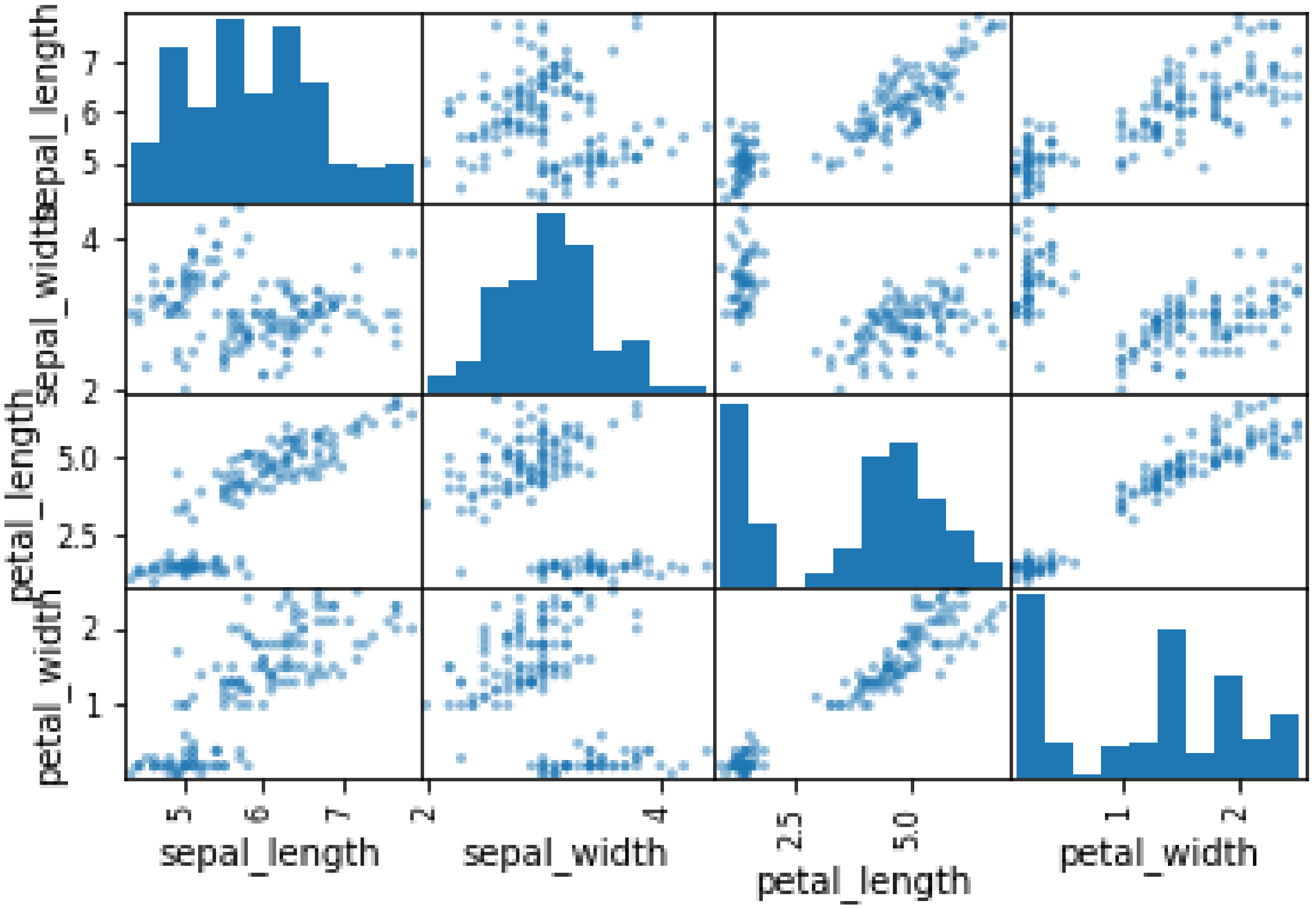
Example 5

- **Answer**: Python code for K-means clustering

```
#Example 5
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
from sklearn.metrics import confusion_matrix
#Import confusion matrix module
from sklearn.preprocessing import LabelEncoder
data = pd.read_csv('Data\\iris.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
kmeans = KMeans(n_clusters=3).fit(data.iloc[:,:-1])
#perform K-means clustering with number of clusters = 3
centroids = kmeans.cluster_centers_ #Extract the cluster centroids
labels = kmeans.labels_ #Extract the labels of clusters
plt.figure(); #Plot the figure
plt.plot(data.iloc[labels==0,0],data.iloc[labels==0,1],'g.’);
#plot the data with label = 0 (color: g)
plt.plot(data.iloc[labels==1,0],data.iloc[labels==1,1],'.');
#plot the data with label = 1 (color: b)
plt.plot(centroids[:,0],centroids[:,1],'ro') #plot the cluster centroid
label_encoder = LabelEncoder()
confusion_matrix(label_encoder .fit_transform(data.iloc[:,-1]),labels)
```

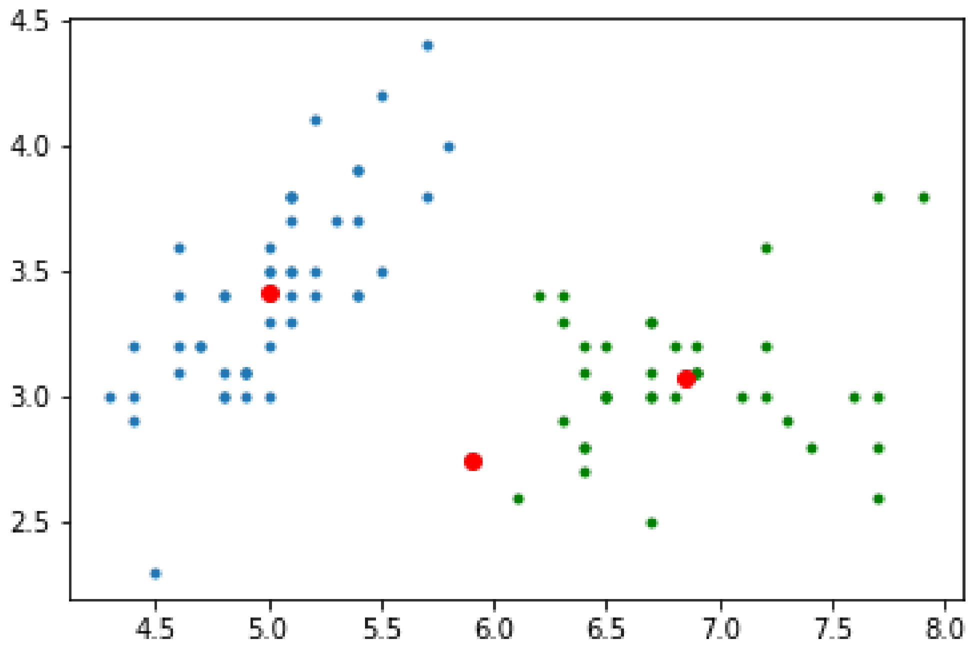
Example 5

- Data Visualization



Example 5

- Clustering results (first two dimensions)



Example 5 (Different Evaluation)

- Confusion matrix: it is used to evaluate the performance of a model on a testing set, which the true labels are known.
- In our example: the confusion matrix is

	Cluster Labels		
	0	1	2
Iris-Setosa	50	0	0
Iris-versicolor	0	2	48
Iris-virginica	0	36	14

Example 5 (Different Evaluation)

36 samples from cluster 1 belong to Iris-virginica

	Cluster Labels		
	0	1	2
Iris-Setosa	50	0	0
Iris-versicolor	0	2	48
Iris-virginica	0	36	14

50 samples from cluster 0 belong to Iris-Setosa

48 samples from cluster 2 belong to Iris-versicolor

Example 5 (Different Evaluation)

- The total number of incorrectly clustered instances: 2 + 14
- Percentage of incorrectly clustered instances: $\frac{(2+14)}{150} = 10.667\%$
- Note: there are totally 150 samples in the dataset.

	Cluster Labels		
	0	1	2
Iris-Setosa	50	0	0
Iris-versicolor	0	2	48
Iris-virginica	0	36	14

Example 6 - Question

- Cluster the prima_diabetes dataset with K-means clustering algorithm.
- There are 2 groups with 768 samples and 9 attributes.
- The two groups are tested negative and tested positive.
- Suppose we know that the dataset has 2 groups.

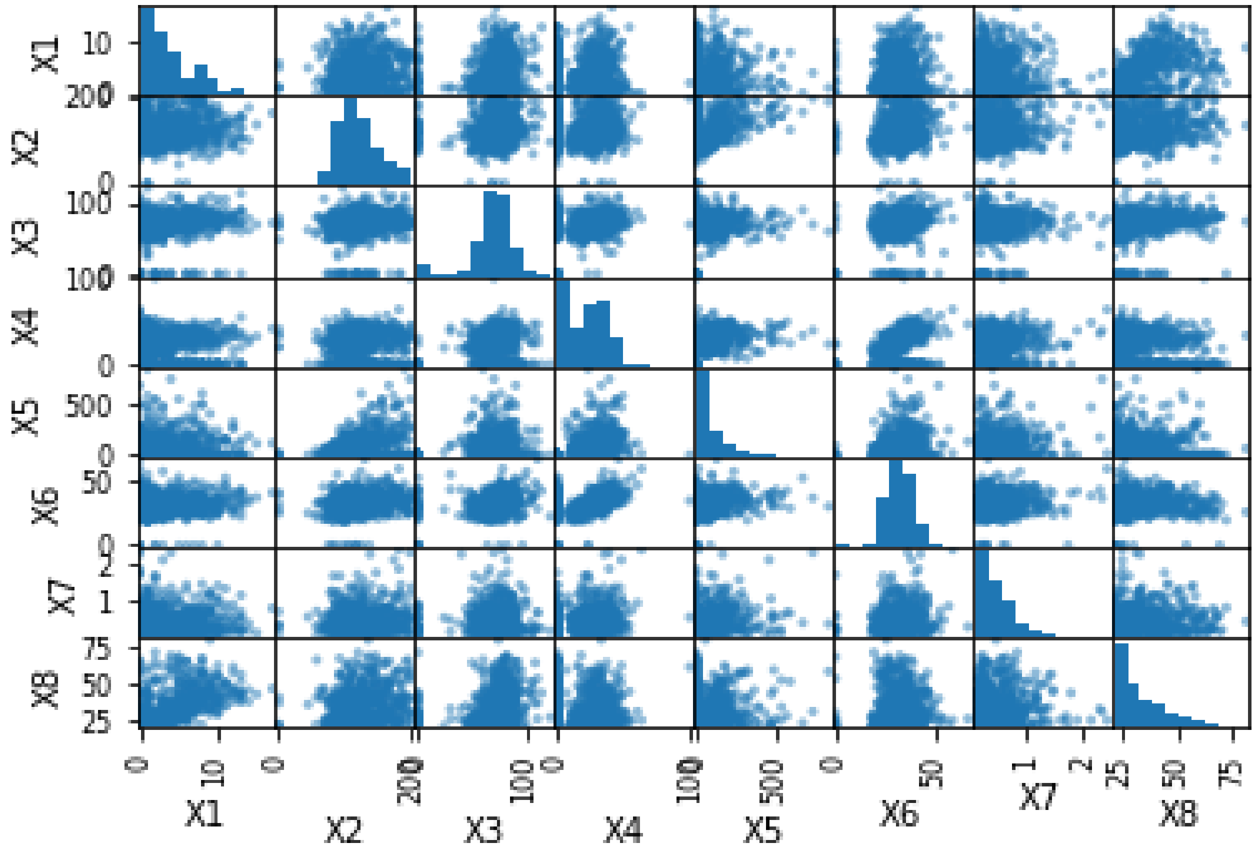
Example 6

- **Answer**: Python code for K-means clustering

```
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
from sklearn.metrics import confusion_matrix
#Import confusion matrix module
from sklearn.preprocessing import LabelEncoder
data = pd.read_csv('Data\\diabetes.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
kmeans = KMeans(n_clusters=2).fit(data.iloc[:,:-1]) #perform K-means
clustering with number of clusters = 3
centroids = kmeans.cluster_centers_ #Extract the cluster centroids
labels = kmeans.labels_ #Extract the labels of clusters
plt.figure(); #Plot the figure
plt.plot(data.iloc[labels==0,0],data.iloc[labels==0,1],'g.');
#plot the data with label = 0 (color: g)
plt.plot(data.iloc[labels==1,0],data.iloc[labels==1,1],'.');
#plot the data with label = 1 (color: b)
plt.plot(centroids[:,0],centroids[:,1],'ro') #plot the cluster centroid
label_encoder = LabelEncoder()
confusion_matrix(label_encoder .fit_transform(data.iloc[:,-1]),labels)
```

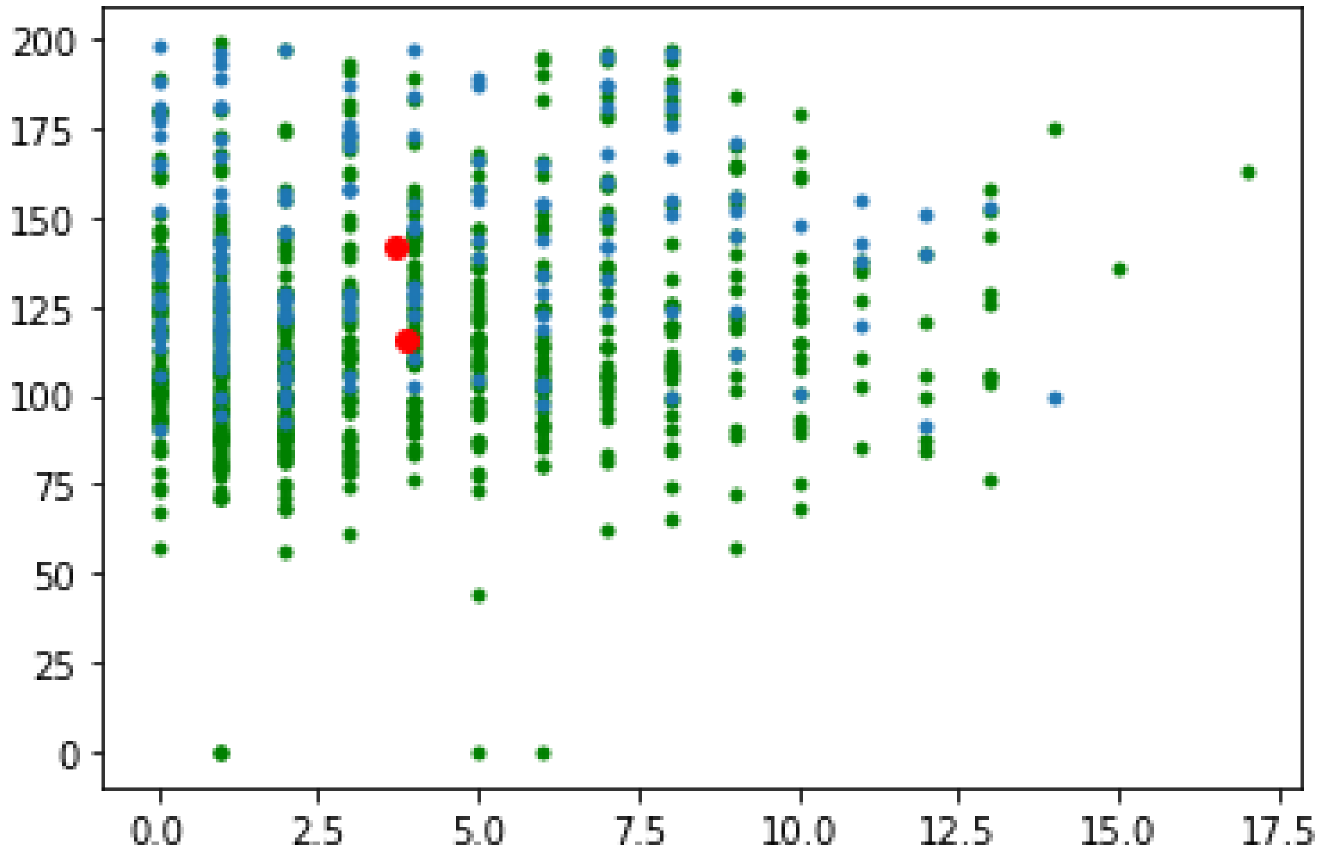
Example 6

- Data Visualization



Example 6

- Clustering results (first two dimensions)



Example 6

- The confusion matrix is

	Cluster Labels	
	0	1
Tested Negative	421	79
Tested Positive	182	86

- The total number of incorrectly clustered instances: 182 + 79=261
- Percentage of incorrectly clustered instances: $\frac{261}{768} = 33.98\%$
- Note: there are totally 768 samples in the dataset.

Note on Applying Clustering Algorithm to Labelled Data

- In many cases, we do not apply clustering algorithm to data with class labels. Analysis labelled data is called supervised learning. This is not the purpose of unsupervised learning.
- However, in some situations, different datasets can have similar characteristics. For example, researchers want to label a flower dataset. They know iris dataset has lablels and have a similar characteristics with the flower dataset. They also know that k-means can guess the labels of the iris dataset well. In this case, we can apply k-means to guess the labels of the flower dataset.

Summary

- Properties of K-means clustering:
  1. If the groups are well-separated, the results can be perfect.
  2. If outliers are present, the results can be very poor.
  3. K-means clustering with Manhattan distance can handle data with outliers.

PROBABILISTIC CLUSTERING VIA EM ALGORITHM

Review of Probabilistic clustering

- It consists of two steps: E-step and M-step.
- E-step:

$$z_{ik} = \frac{Probability\ (ith\ pt)\ c_k}{\sum_k Probability\ (ith\ pt)\ c_k}$$

- M-step:

$$\max_{\mathrm{c_k}} \sum_{i,k} z_{ik} \log(p_{ik})$$

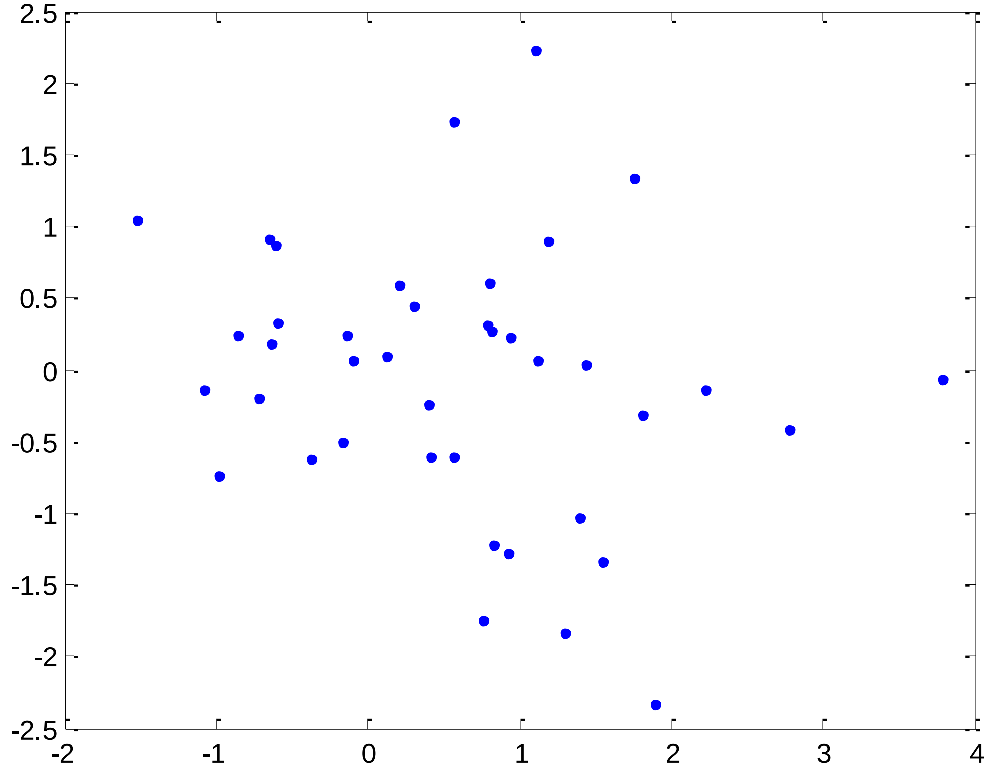
If p_{ik} is Gaussian, we have $p_{ik} = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - c_k)^2}{2\sigma^2}\right)$

Case Study

- Example 1: A data with two groups but they are closed to each other.
- Example 2: A data with two well-separated clusters. (with the use of cross validation)
- Example 3: A data with two clusters and outliers. (with the use of cross validation)
- Comparing K-means and GMM: Data used in example 1
- Comparing K-means and GMM: Data used in example 2
- Comparing K-means and GMM: Heterogeneous cluster

Example 1 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data1.csv).
- Apply GMM clustering algorithm to partition the data into two different clusters.



Example 1

- **Answer**: Python code for K-means clustering

```
#Example 1
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.mixture import GaussianMixture #Import GMM module
data = pd.read_csv('Data\\kmeans_data1.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
gmm = GaussianMixture(n_components=2).fit(data)
#perform EM clustering with number of clusters = 2
centroids = gmm.means_#Extract the cluster centroids
labels = gmm.predict(data)
plt.figure(); #Plot the figure
plt.plot(data.iloc[labels==0,0],data.iloc[labels==0,1],'g.');
#plot the data with label = 0 (color: g)
plt.plot(data.iloc[labels==1,0],data.iloc[labels==1,1],'.');
#plot the data with label = 1 (color: b)
plt.plot(centroids[:,0],centroids[:,1],'ro') #plot the cluster centroid
```

Example 1

- The default GMM model is

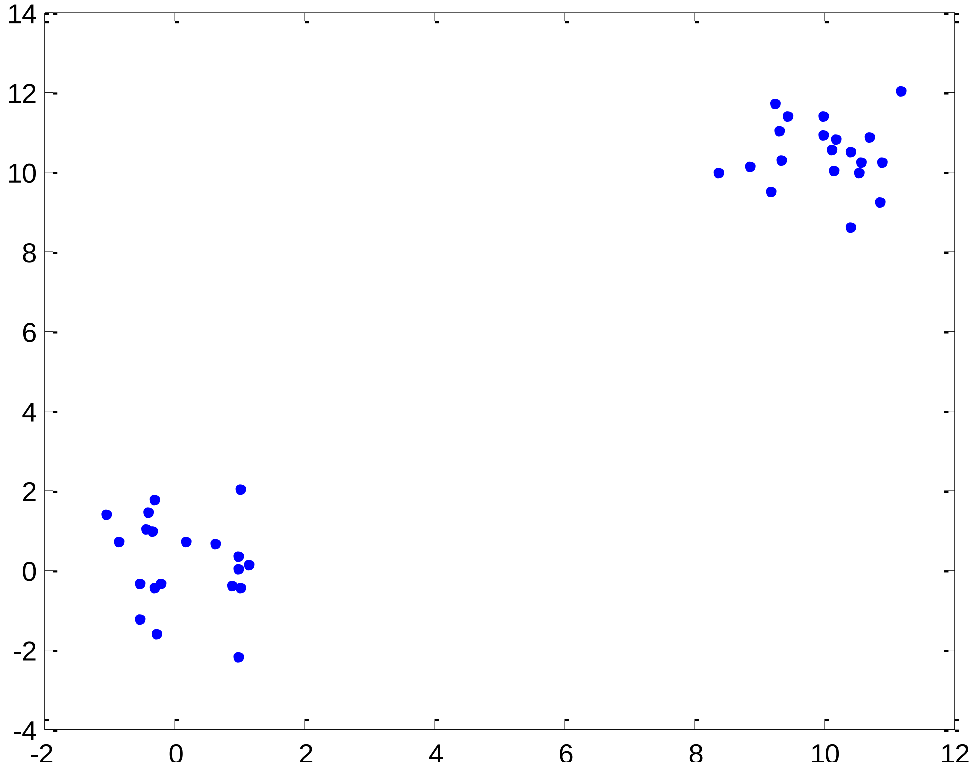
$$\max_{c_k} \sum_{i,k} z_{ik} \log(p_{ik})$$

where $p_{ik} = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - c_k)^2}{2\sigma^2}\right)$

- For each cluster, it has two parameters and they are the mean and standard deviation.
- Type: gmm.means_ (for means) and gmm.covariances_ (for covariance)
- The results also report the value of AIC and BIC.
- Type: gmm.aic(data) and gmm.bic(data)
- Smaller values imply better clustering results.

Example 2 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data2.csv).
- Apply GMM clustering algorithm to partition the data into two different clusters.



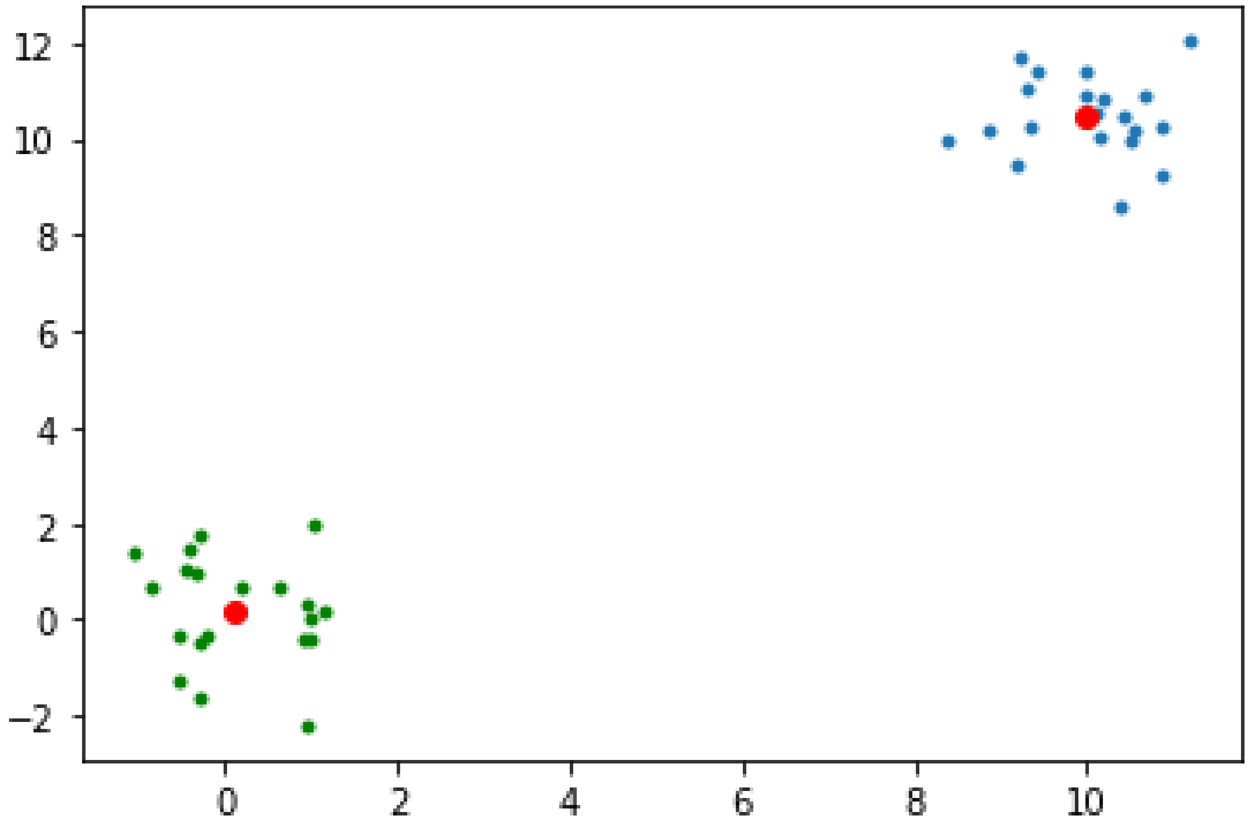
Example 2

- **Answer**: Python code for K-means clustering

```
#Example 2
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.mixture import GaussianMixture #Import GMM module
data = pd.read_csv('Data\\kmeans_data2.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
gmm = GaussianMixture(n_components=2).fit(data)
#perform GMM clustering with number of clusters = 2
centroids = gmm.means_#Extract the cluster centroids
labels = gmm.predict(data)
plt.figure(); #Plot the figure
plt.plot(data.iloc[labels==0,0],data.iloc[labels==0,1],'g.');
#plot the data with label = 0 (color: g)
plt.plot(data.iloc[labels==1,0],data.iloc[labels==1,1],'.');
#plot the data with label = 1 (color: b)
plt.plot(centroids[:,0],centroids[:,1],'ro') #plot the cluster centroid
```

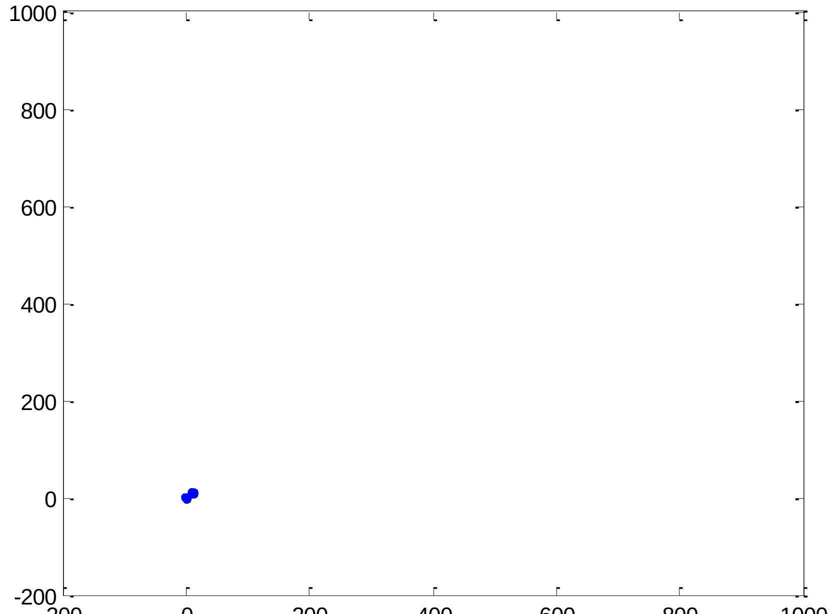
Example 2

- Clustering results using GMM



Example 3 - Question

- Consider the following two-dimensional data (The data is in the file: kmeans_data3.csv).
- 20 samples for group 1 while 20 samples for group 2. 10 samples are outliers.
- Apply GMM clustering algorithm to partition the data into two different clusters.



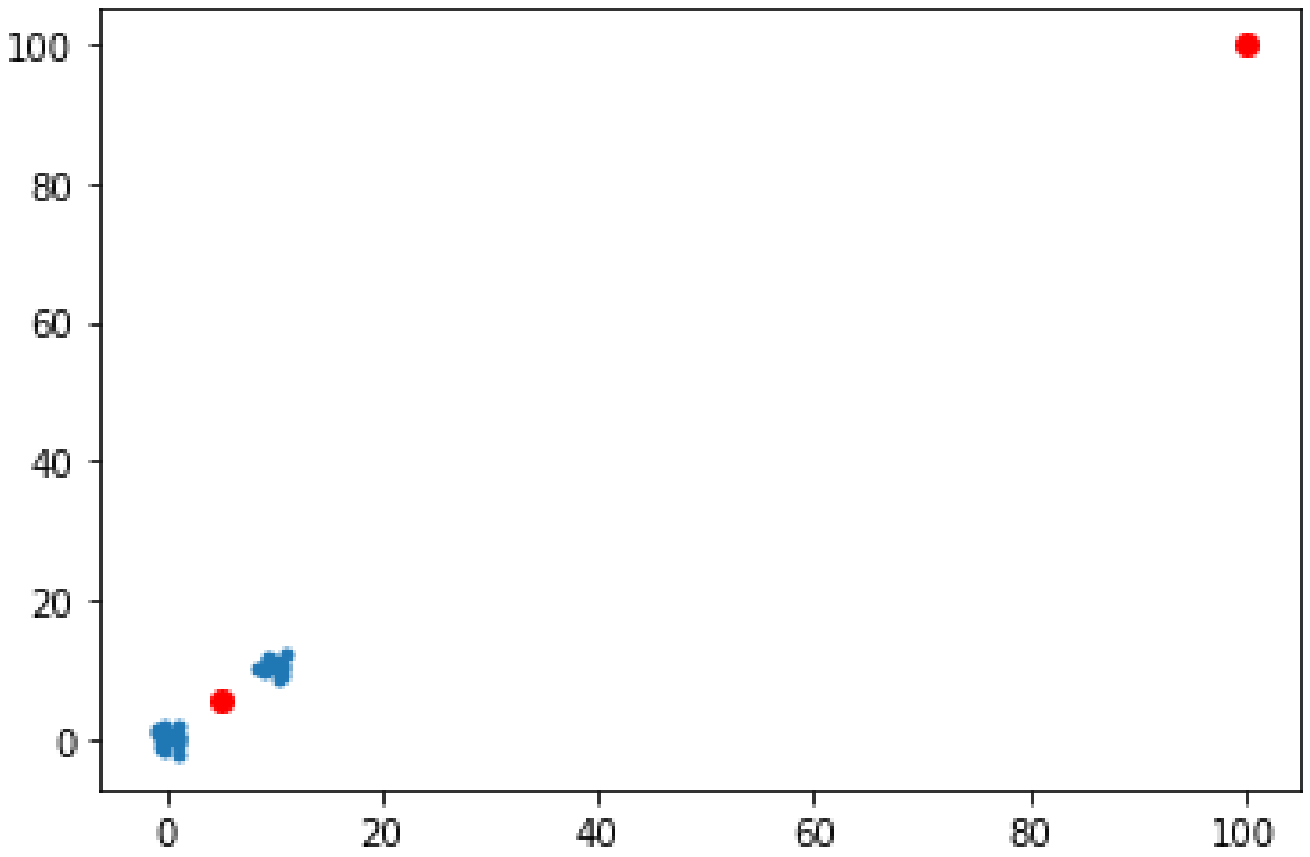
Example 3

- **Answer**: Python code for GMM clustering

```
#Example 3
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.mixture import GaussianMixture #Import GMM module
data = pd.read_csv('Data\\kmeans_data3.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
gmm = GaussianMixture(n_components=2).fit(data)
#perform GMM clustering with number of clusters = 2
centroids = gmm.means_#Extract the cluster centroids
labels = gmm.predict(data)
plt.figure(); #Plot the figure
plt.plot(data.iloc[labels==0,0],data.iloc[labels==0,1],'g.');
#plot the data with label = 0 (color: g)
plt.plot(data.iloc[labels==1,0],data.iloc[labels==1,1],'.');
#plot the data with label = 1 (color: b)
plt.plot(centroids[:,0],centroids[:,1],'ro') #plot the cluster centroid
```

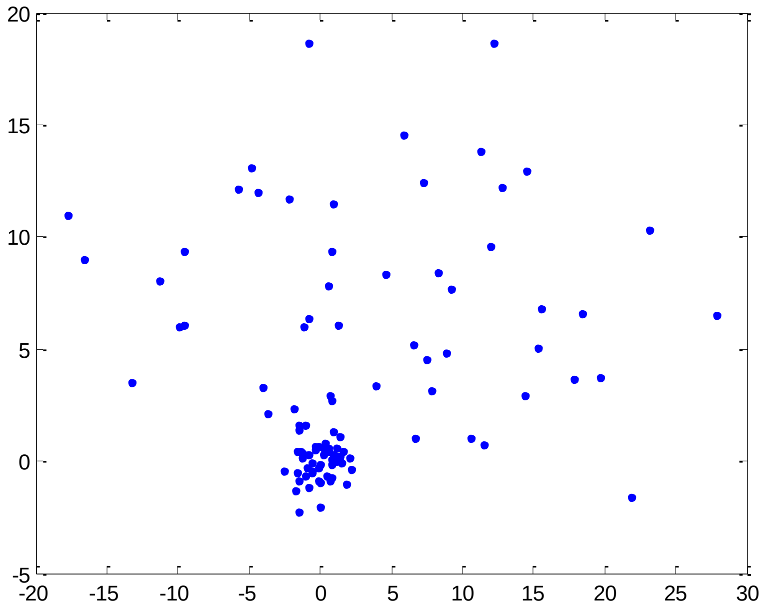
Example 3

- Clustering results



Example – Question (Heterogeneous Data)

- Consider the following two-dimensional data (The data is in the file: heterogeneous_data.csv).
- It consists of two groups. Each of which has 50 samples. The first group follows a standard normal distribution. The second group follows a normal distribution with mean 10 and 10 and 5 standard deviations for the x and y axes.
- Apply GMM clustering algorithm to partition the data into two different clusters.



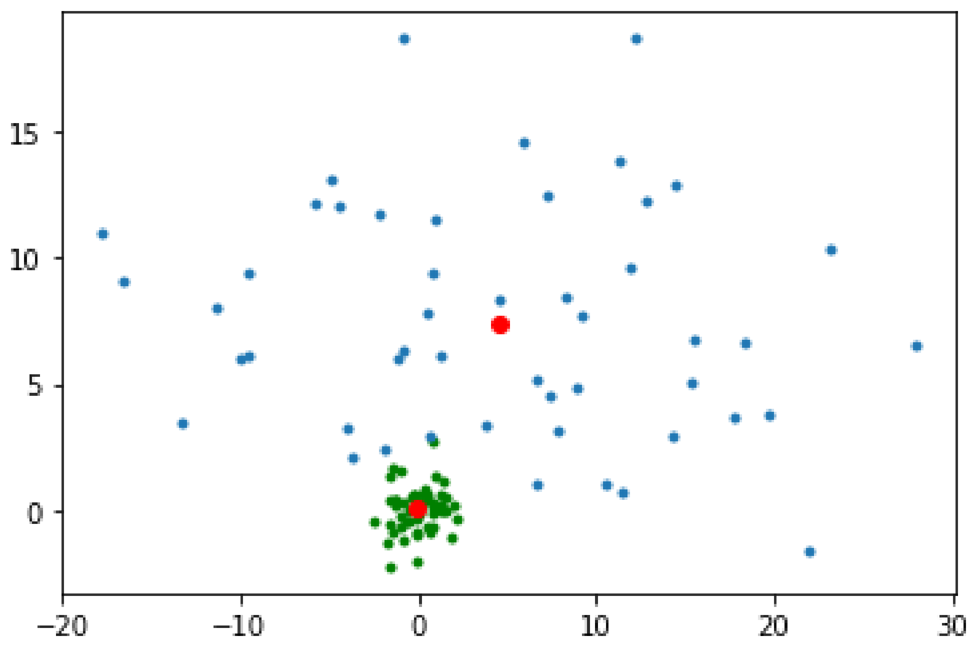
Example –Heterogeneous Data

- **Answer**: Python code for GMM clustering

```
#Example Heterogeneous Data
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.mixture import GaussianMixture #Import GMM module
data = pd.read_csv('Data\\heterogeneous_data.csv') #Load the data file
pd.plotting.scatter_matrix(data); #Visualize the data
gmm = GaussianMixture(n_components=2).fit(data.iloc[:,:-1])
#perform GMM clustering with number of clusters = 2
centroids = gmm.means_#Extract the cluster centroids
labels = gmm.predict(data.iloc[:,:-1])
plt.figure(); #Plot the figure
plt.plot(data.iloc[labels==0,0],data.iloc[labels==0,1],'g.');
#plot the data with label = 0 (color: g)
plt.plot(data.iloc[labels==1,0],data.iloc[labels==1,1],'.');
#plot the data with label = 1 (color: b)
plt.plot(centroids[:,0],centroids[:,1],'ro') #plot the cluster centroid
```

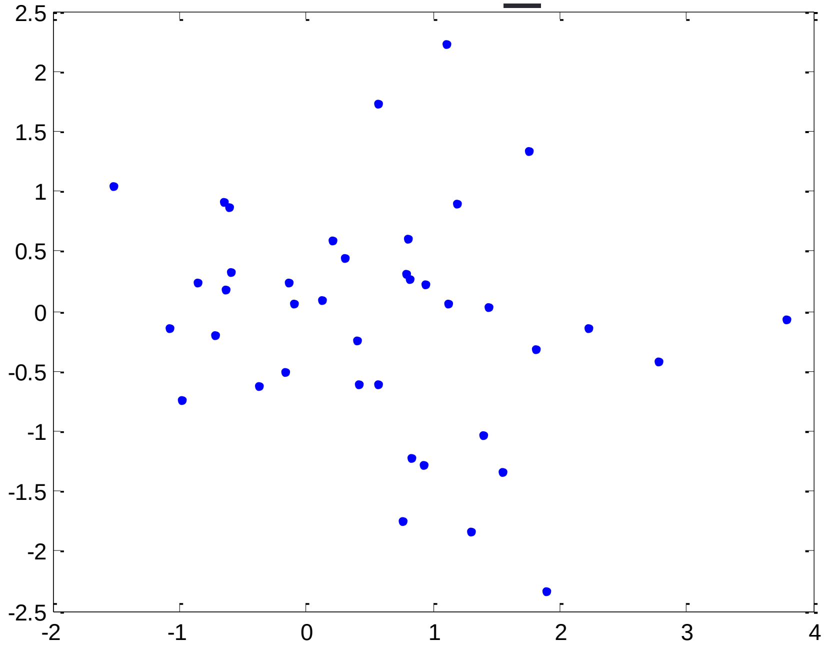
Example –Heterogeneous Data

- The clustering results are shown in graph form.



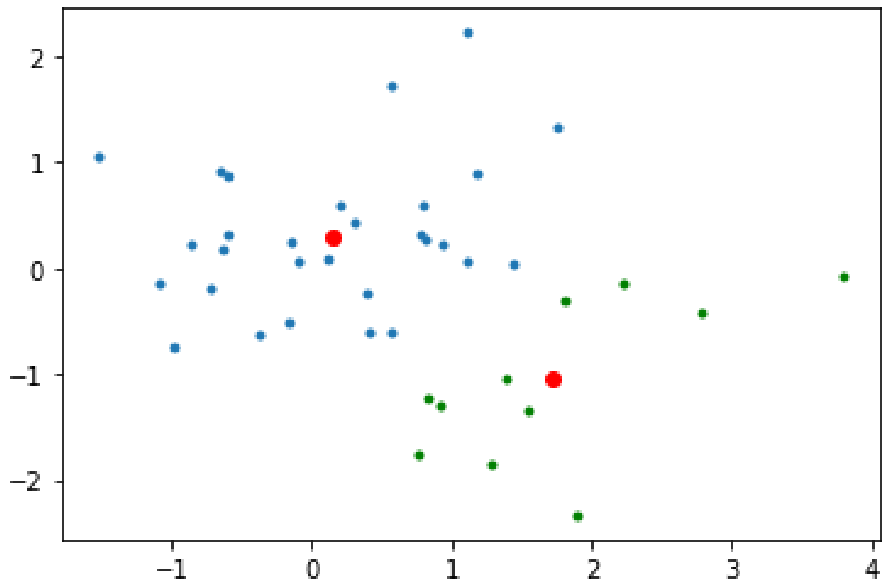
Comparing GMM and K-means Clustering Example 1 - Question

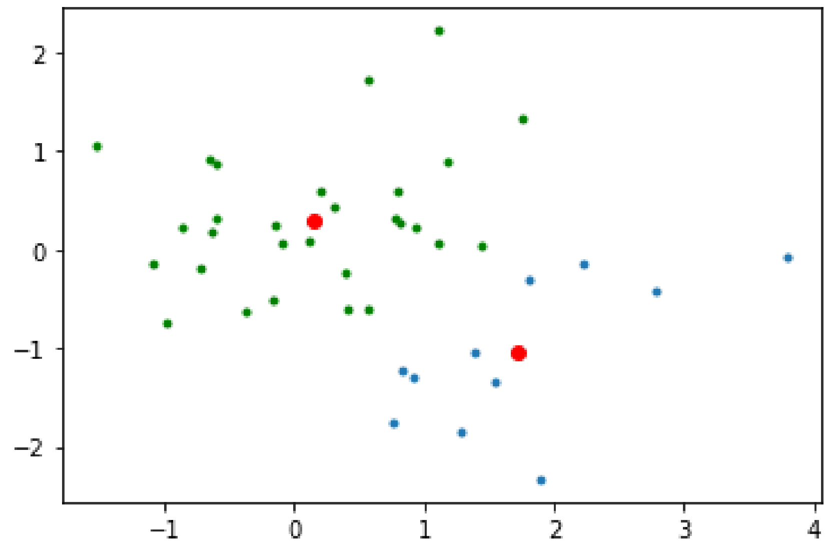
- We compare the performance of GMM and K-means clustering by the dataset shown in Example 1.
- There are totally 40 samples. First 20 samples belong to class 1 while the last 20 samples belong to class 2.
- We re-run both methods to the data with label. The filename of the data is “kmeans_data1 – with label.csv”.



Comparing GMM and K-means Clustering Example 1

Answer: K-means GMM





	Cluster Labels	
	0	1
True Label a	18	2
True Label b	10	10

Incorrect instances (%)	12 (30.0%)

	Cluster Labels	
	0	1
True Label a	18	2
True Label b	11	9

Incorrect instances (%)	13 (32.5%)

Comparing GMM and K-means Clustering Example 1

- In this example, K-means clustering and GMM algorithm have similar performances.
- Compare the two objective functions:
- For K-means clustering:

$$\min_{I_{ik},\boldsymbol{c}_k} J(I_{ik},\boldsymbol{c}_k), where\ J(I_{ik},\boldsymbol{c}_k) = \sum_{i=1}^{n}\sum_{k=1}^{c} I_{ik}\left|\left|\boldsymbol{x}_i - \boldsymbol{c}_k\right|\right|^2$$

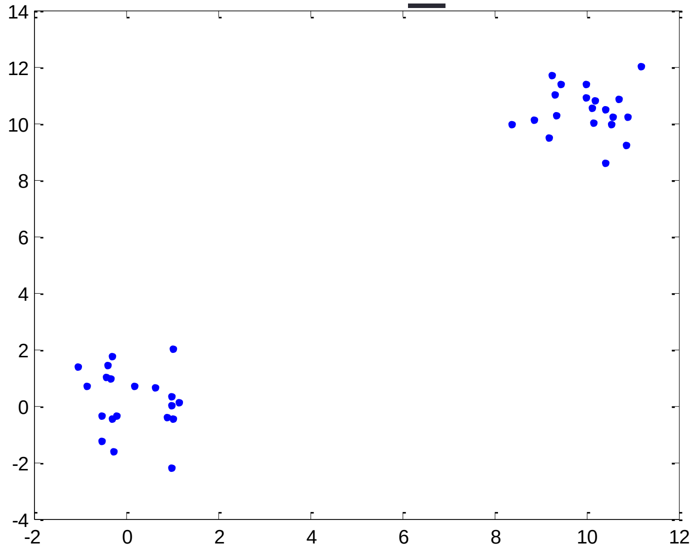
- For GMM:

$$\max_{\mathrm{c_k}} \sum_{i,k} z_{ik}\log(p_{ik})$$

where $p_{ik} = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x_i - c_k)^2}{2\sigma^2}\right)$

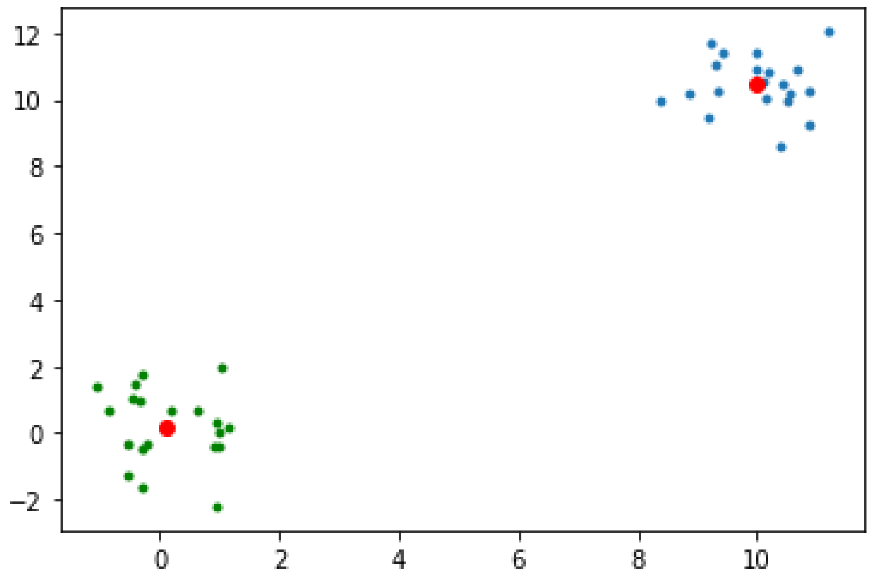
Comparing GMM and K-means Clustering Example 2 - Question

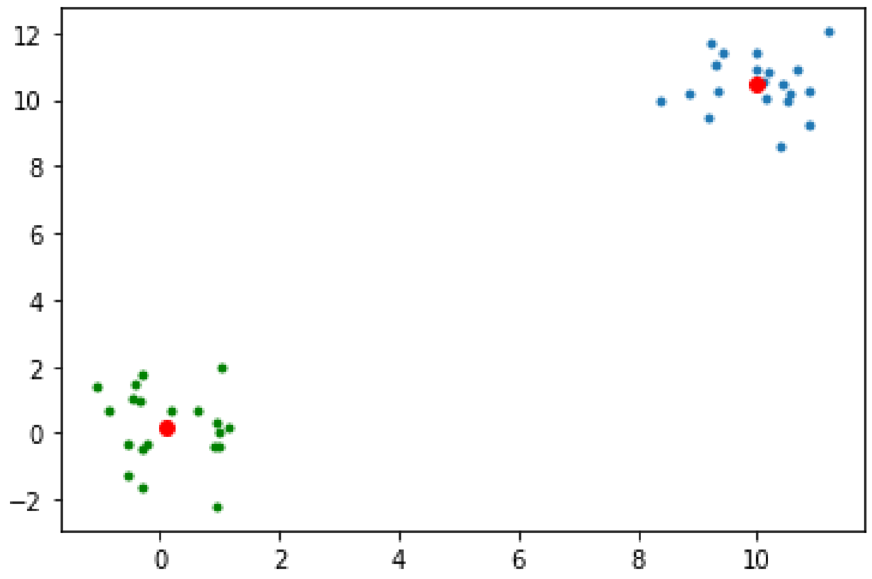
- We compare the performance of GMM and K-means clustering by the dataset shown in Example 2.
- There are totally 40 samples. First 20 samples belong to class 1 while the last 20 samples belong to class 2.
- We re-run both methods to the data with label. The filename of the data is “kmeans_data2 – with label.csv”.



Comparing GMM and K-means Clustering Example 2

Answer: K-means GMM





K-means:

	Cluster Labels	
	0	1
True Label a	0	20
True Label b	20	0

Incorrect instances (%)	0 (0%)

GMM:

	Cluster Labels	
	0	1
True Label a	0	20
True Label b	20	0

Incorrect instances (%)	0 (0%)

Comparing GMM and K-means Clustering Example 2

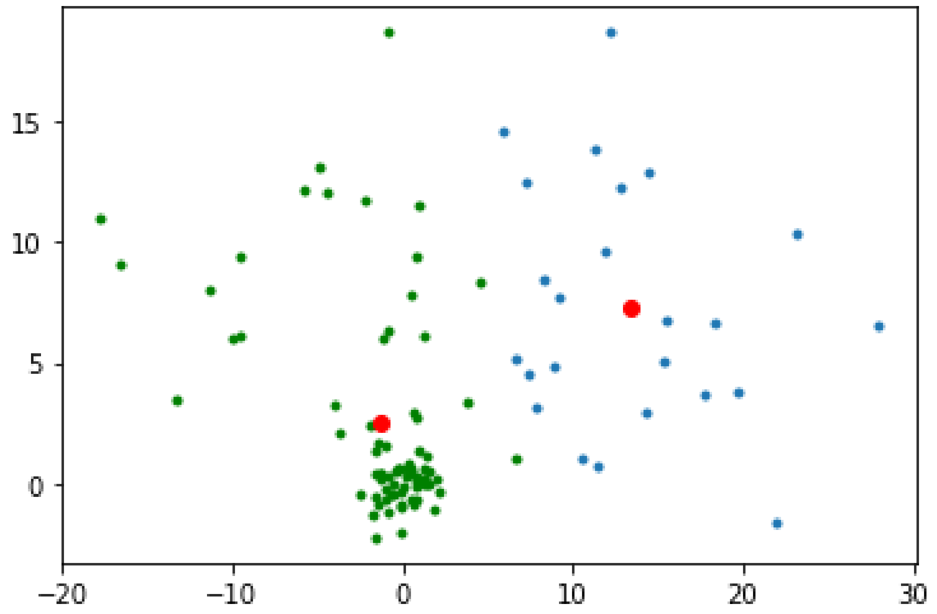
- The two clustering algorithms perform equally well.
- That means when the clusters are well-separated, the performance of GMM and K-means are the same.

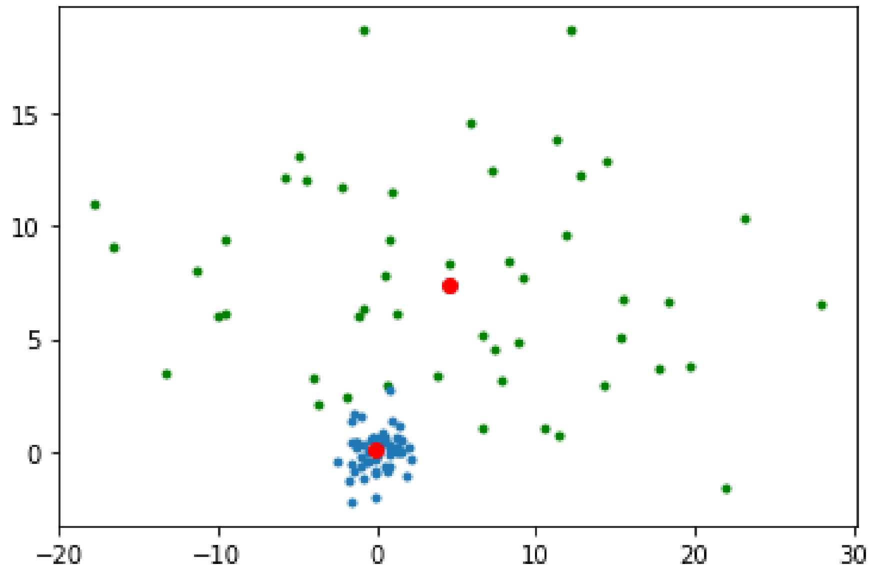
Comparing GMM and K-means Clustering Example (Heterogeneous Data) - Question

- We compare the performance of GMM and K-means clustering by the dataset shown in Example (Heterogeneous Data). (The data is in the file: heterogeneous_data.csv).
- It consists of two groups. Each of which has 50 samples. The first group follows a standard normal distribution. The second group follows a normal distribution with mean 10 and 10 and 5 standard deviations for the x and y axes.

Comparing GMM and K-means Clustering Example (Heterogeneous Data) - Question

Answer: K-means GMM





	Cluster Labels	
	0	1
True Label a	50	0
True Label b	26	24

Incorrect instances (%)	26 (26%)

	Cluster Labels	
	0	1
True Label a	2	48
True Label b	47	3

Incorrect instances (%)	5 (5%)

Comparing GMM and K-means Clustering Example (Heterogeneous Data) - Question

- GMM algorithm performs better than K-means algorithm.
- Generally speaking, GMM algorithm performs better when one of the clusters is heterogeneous. That is, the standard deviation of one cluster is very different from the other.

Summary

- The GMM algorithm employs a probabilistic approach to partition the data.
- It can handle heterogeneous data well. It outperforms K-means clustering algorithm.
- However, it cannot handle data with not well-separated data.
- It has an option to infer the number of clusters by the techniques called cross validation.

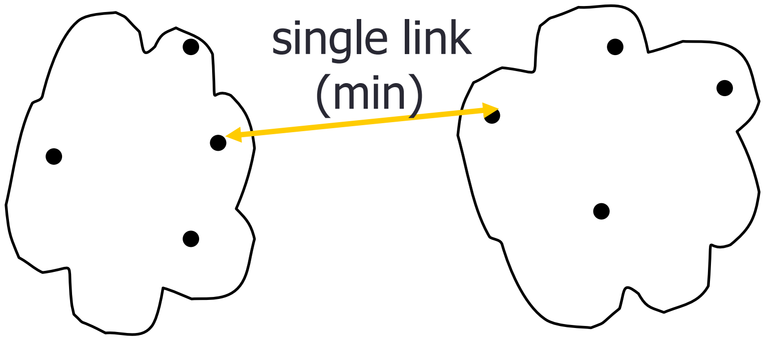
HIERARCHICAL CLUSTERING

Review of Hierarchical clustering

- **Single-link**
  - Distance of the *“closest” points* (single-link)
- **Complete-link**
  - Distance of the “furthest” points
- **Average-link**
  - Average distance between pairs of elements

Cluster Distance Measures

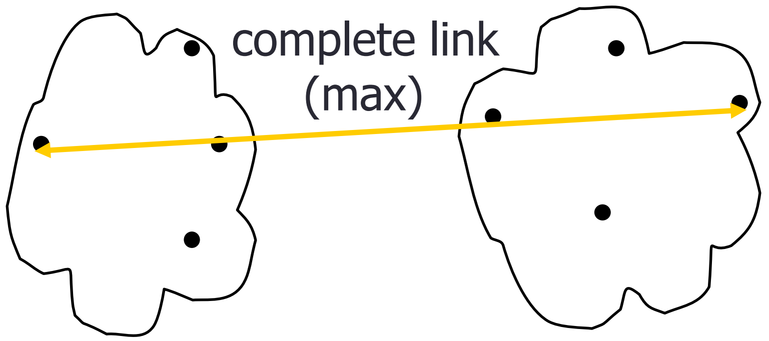
- Single link: smallest distance between an element in one cluster and an element in the other, i.e., $d(C_i, C_j) = \min\{d(x_{ip}, x_{jq})\}$



Obviously, d(C, C)=0

Cluster Distance Measures

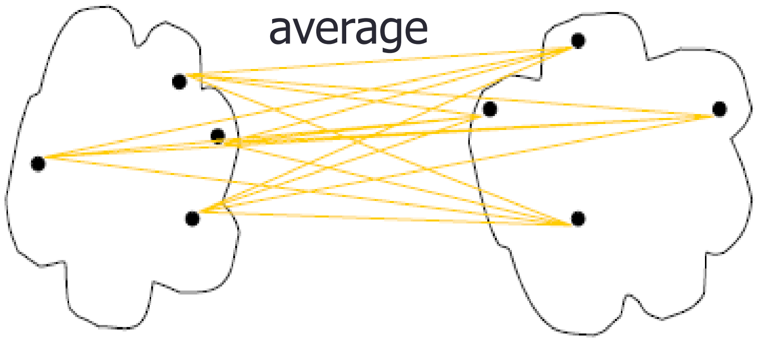
- Complete link: largest distance between an element in one cluster and an element in the other, i.e., $d(C_i, C_j) = \max\{d(x_{ip}, x_{jq})\}$



Obviously, d(C, C)=0

Cluster Distance Measures

- Average: avg distance between elements in one cluster and elements in the other, i.e., $d(C_i, C_j) = avg\{d(x_{ip}, x_{jq})\}$



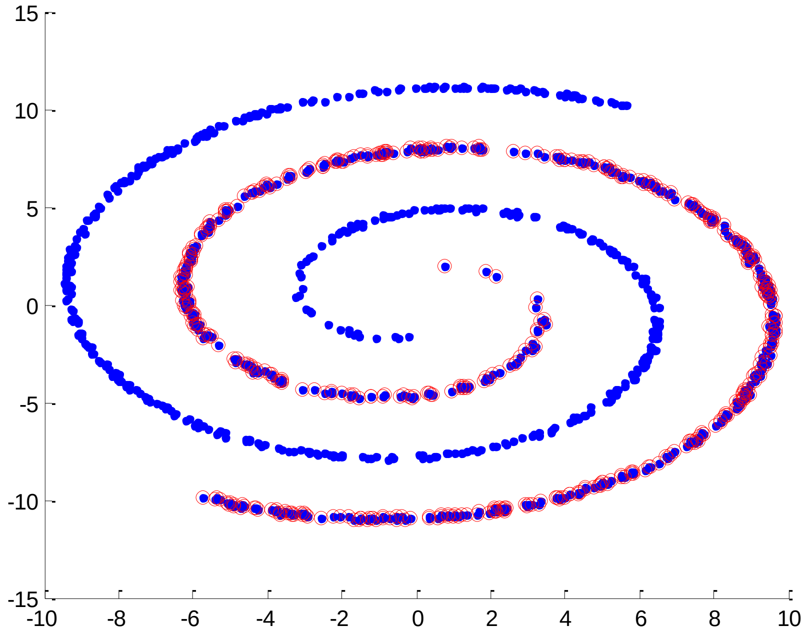
Obviously, d(C, C)=0

Case Study

- Example 1: Two spiral data
- Example 2: Two very closed clusters
- Example 3: Two very closed clusters with outliers

Example 1 – Spiral Data (Question)

- Consider the following two spiral data. It has 1000 samples.
- Each of the two groups has 500 samples.
- The data file is “spiral.csv”.
- Cluster the data into two groups by Hierarchical clustering algorithm with (i) single link; (ii) complete link and (iii) average link.



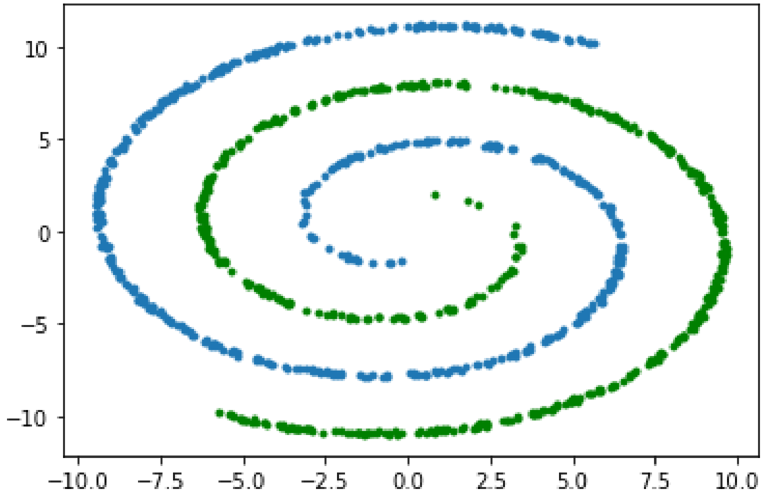
Example 1 – Spiral Data (Single Link)

- **Answer**: Python code for Hierarchical clustering

```
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import confusion_matrix
#Import confusion matrix module
from sklearn.preprocessing import LabelEncoder
data = pd.read_csv('Data\\spiral.csv') #Load the data file
cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean',
linkage='single').fit(data.iloc[:,:-1])
#apply Hierarchical clustering with single linkage
labels = cluster.labels_ #Extract the labels of clusters
label_encoder = LabelEncoder()
print(confusion_matrix(label_encoder .fit_transform(data.iloc[:,-1]),labels))
plt.figure(); #Plot the figure
plt.plot(data.iloc[labels==0,0],data.iloc[labels==0,1],'g.');
#plot the data with label = 0 (color: g)
plt.plot(data.iloc[labels==1,0],data.iloc[labels==1,1],'.');
#plot the data with label = 1 (color: b)
```

Example 1 – Spiral Data (Single Link)

- Clustering results



	Cluster Labels	
	0	1
True Label a	500	0
True Label b	0	500

It gives perfect clustering results!

Incorrect instances (%)	0 (0%)

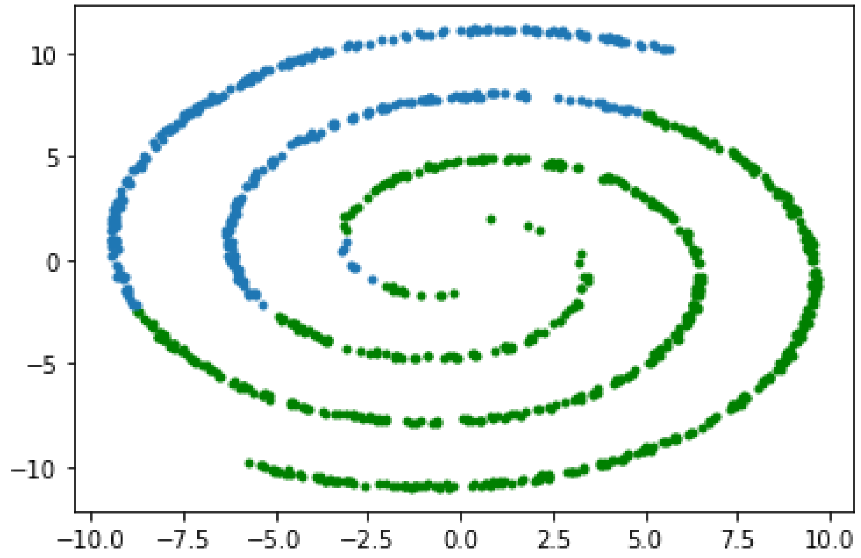
Example 1 – Spiral Data (Complete Link)

- **Answer**: Python code for Hierarchical clustering

```
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import confusion_matrix #Import confusion matrix module
from sklearn.preprocessing import LabelEncoder
data = pd.read_csv('Data\\spiral.csv') #Load the data file
cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean',
linkage='complete').fit(data.iloc[:,:-1])
#apply Hierarchical clustering with single linkage
labels = cluster.labels_ #Extract the labels of clusters
label_encoder = LabelEncoder()
print(confusion_matrix(label_encoder .fit_transform(data.iloc[:,-1]),labels))
plt.figure(); #Plot the figure
plt.plot(data.iloc[labels==0,0],data.iloc[labels==0,1],'g.');
#plot the data with label = 0 (color: g)
plt.plot(data.iloc[labels==1,0],data.iloc[labels==1,1],'.');
#plot the data with label = 1 (color: b)
```

Example 1 – Spiral Data (Complete Link)

- Clustering results

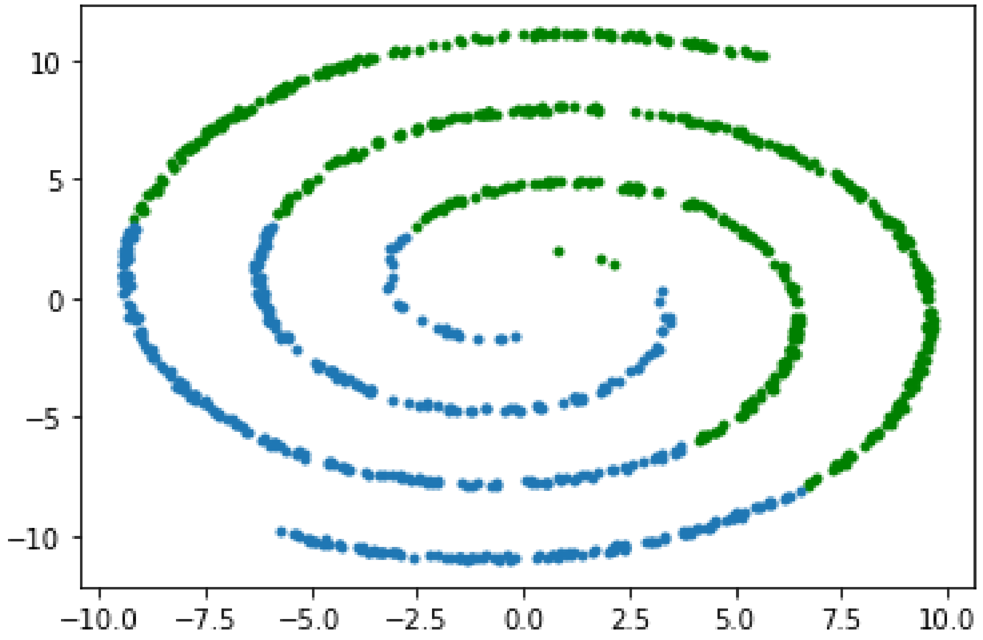


	Cluster Labels	
	0	1
True Label a	349	151
True Label b	283	217

Incorrect instances (%)	434 (43.4%)

Example 1 – Spiral Data (Average Link)

- Clustering results

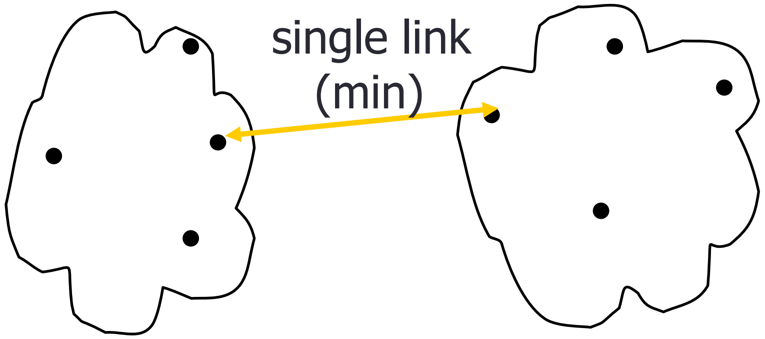


	Cluster Labels	
	0	1
True Label a	259	241
True Label b	309	191

Incorrect instances (%)	450 (45.0%)

Example 1 – Spiral Data

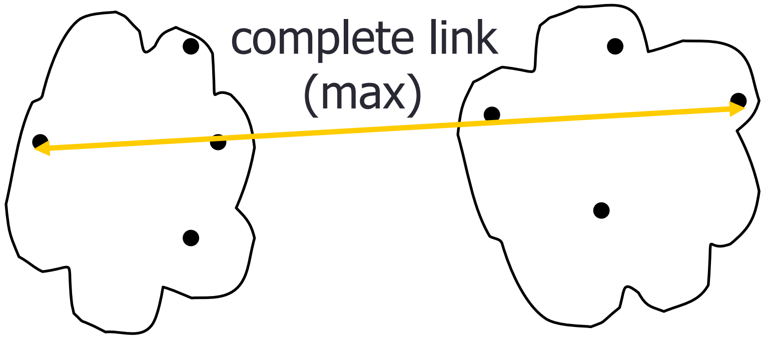
- Why single link performs the best? Why average and complete links not perform well?
- Single link: it measures the smallest distance between an element in one cluster and an element in the other, i.e., $d(C_i, C_j) = \min\{d(x_{ip}, x_{jq})\}$



- It can handle data with arbitrary shape.

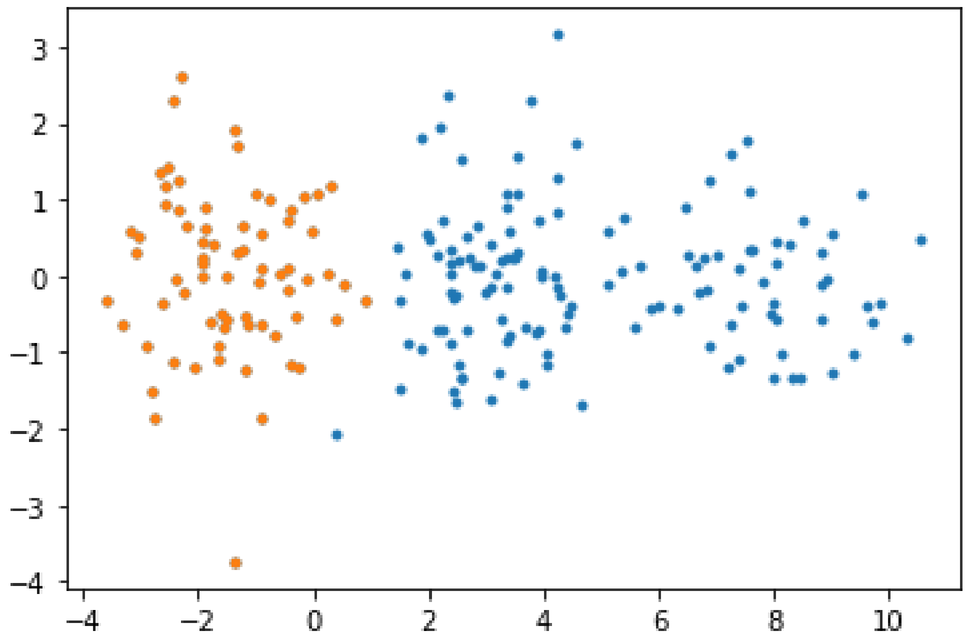
Example 1 – Spiral Data

- However, for complete and average linkages, they require that the cluster has spherical shape.
- Complete link: it measures largest distance between an element in one cluster and an element in the other, i.e., $d(C_i, C_j) = \max\{d(x_{ip}, x_{jq})\}$



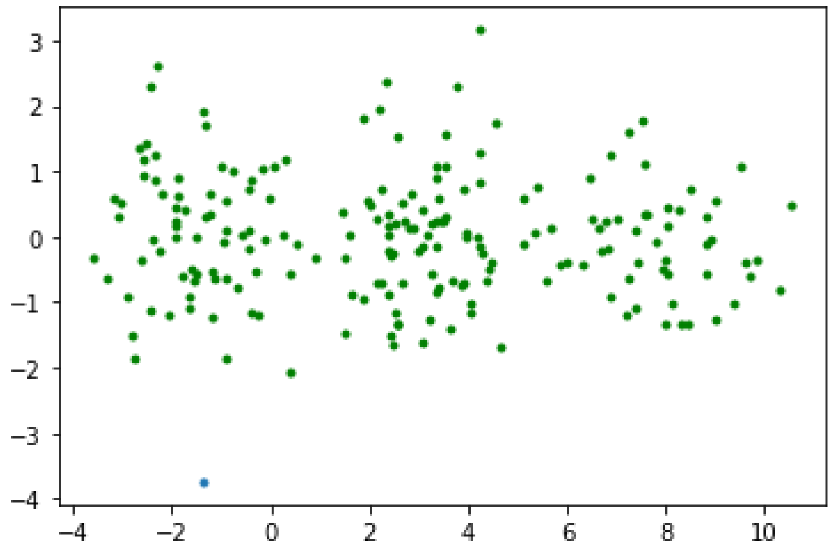
Example 2 - Question (Scatter)

- Consider the following two spherical data, which have 70 (left) and 130 (right) samples. But one cluster has some scatter points.
- The data file is “hierarchical0.csv”.
- Cluster the data into two groups by Hierarchical clustering algorithm with (i) single link; (ii) complete link and (iii) average link.



Example 2 - Question (Scatter)

- Answer: By following the steps show in previous example, we can obtain the clustering result using single link.

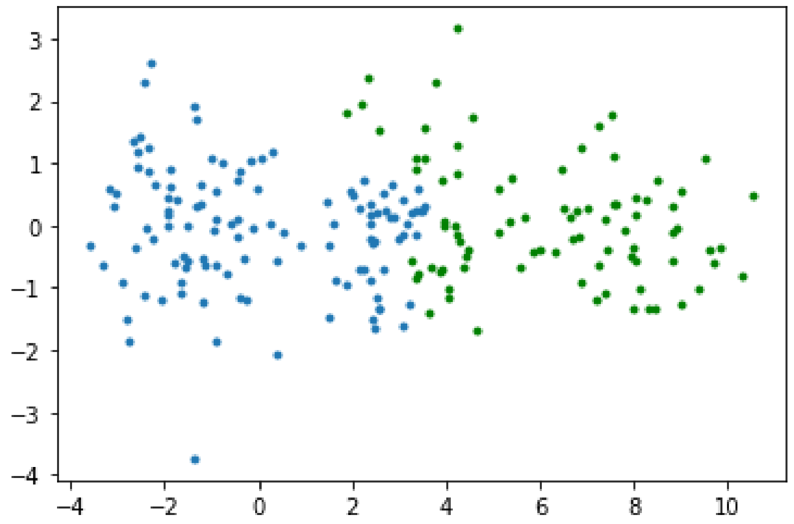


	Cluster Labels	
	0	1
True Label a	99	1
True Label b	100	0

Incorrect instances (%)	99 (49.5%)

Example 2 - Question (Scatter)

- The results of complete link are

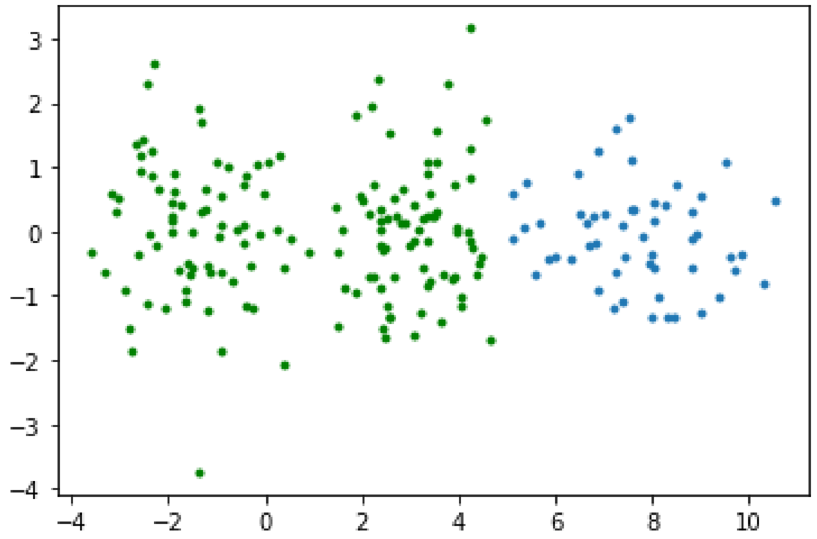


	Cluster Labels	
	0	1
True Label a	0	70
True Label b	85	45

Incorrect instances (%)	45 (22.5%)

Example 2 - Question (Scatter)

- The results of average link are



	Cluster Labels	
	0	1
True Label a	70	0
True Label b	77	53

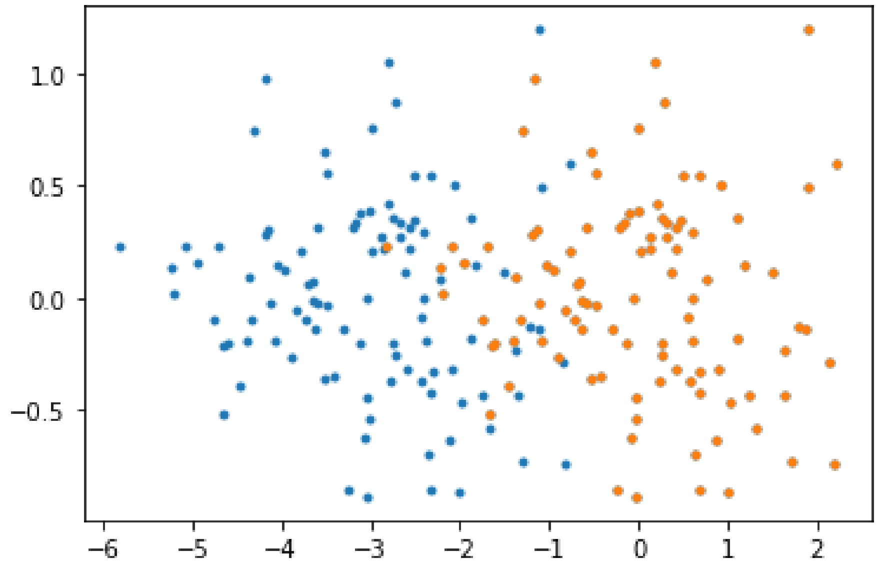
Incorrect instances (%)	77 (38.5%)

Example 2 - Question (Scatter)

- In this example, the complete link performs the best.
- Why?
- For complete link, it computes the largest distance between an element in one cluster and an element in the other. It tends to form a spherical cluster. The left cluster has a spherical-shape and the result is perfect.
- For average link, it computes the average distance among every pair between two clusters. It tends to partition the data and form clusters to have small variances.
- For single link, it has the chain effect. If there is no clear cut between two clusters, it is easy to get bad results.

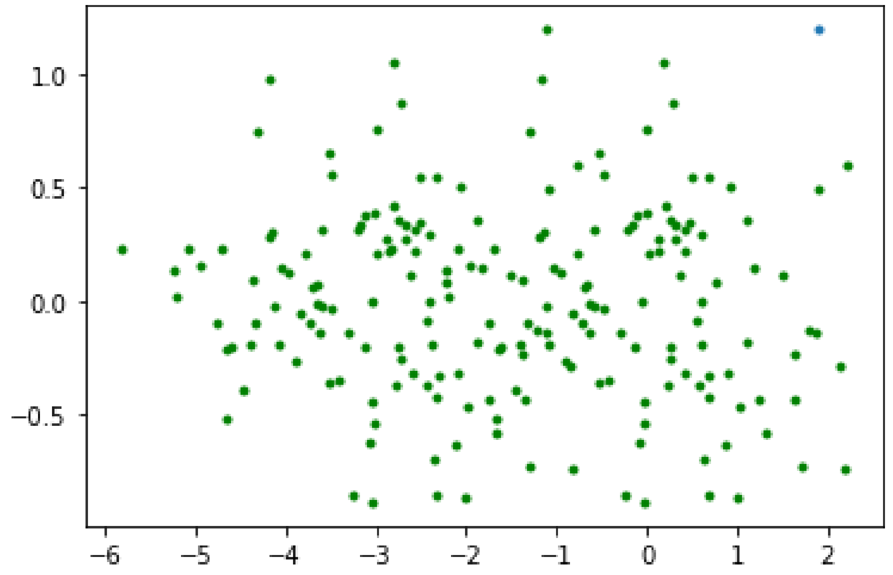
Example 3 - Question

- Consider the following two spherical data. It has 200 samples.
- Each of the two groups has 100 samples. They are close to each other.
- The data file is “hierarchical1.csv”.
- Cluster the data into two groups by Hierarchical clustering algorithm with (i) single link; (ii) complete link and (iii) average link.



Example 3

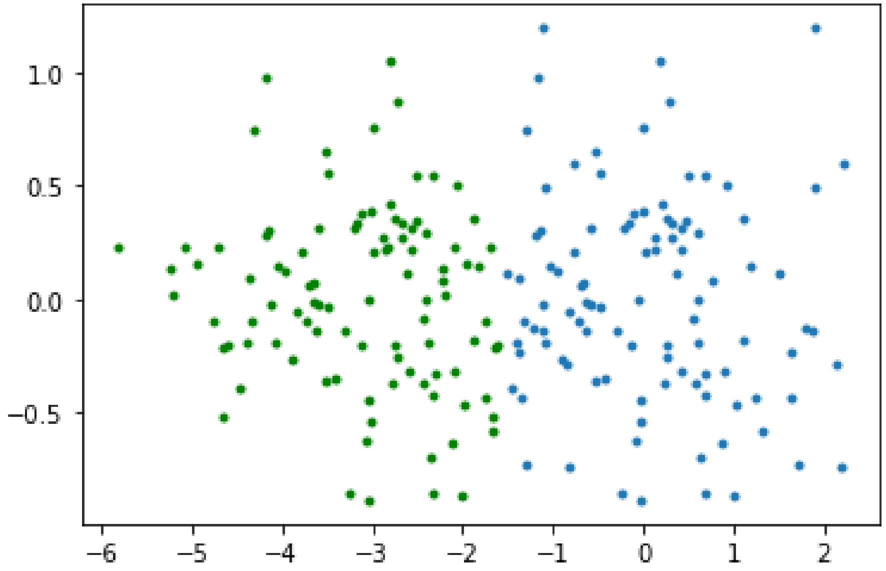
- Answer: By following the steps show in previous example, we can obtain the clustering result using single link.



	Cluster Labels	
	0	1
True Label a	99	1
True Label b	100	0

Example 3

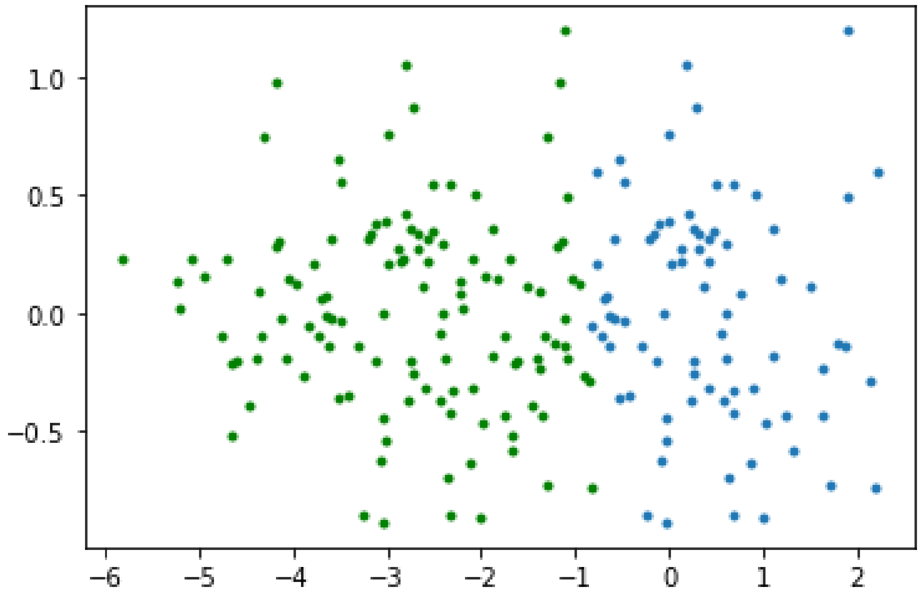
- The results of complete link are



	Cluster Labels	
	0	1
True Label a	10	90
True Label b	89	11

Example 3

- The results of average link are



	Cluster Labels	
	0	1
True Label a	23	77
True Label b	99	1

Example 3

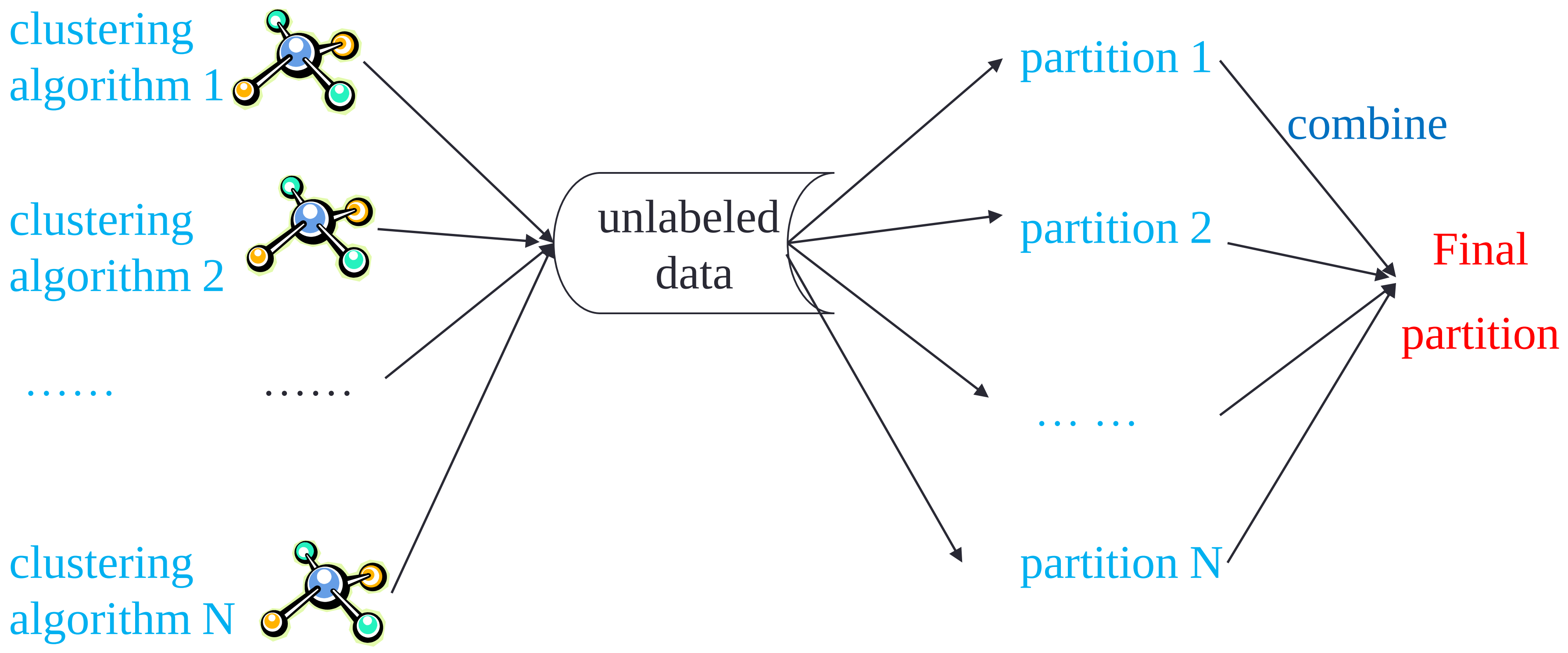
- In this example, the performance of complete and average links are similar. They outperforms single link.
- Why the performance of complete and average links are better?
- Reason: they require that the cluster has a spherical shape.
- However, both average and complete links did not get perfect solutions
- Reason: the two clusters are not well-separated.

Summary

- Single link is good at handling data with highly irregular shape. However, it is easy to get undesired result.
- Complete and average links are good at handling data with spherical shape.

ENSEMBLE CLUSTERING

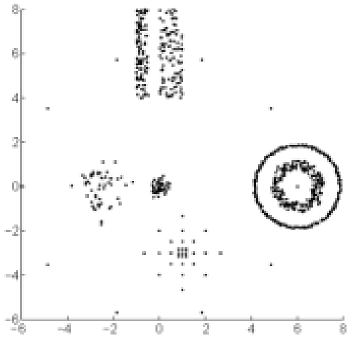
Ensemble Clustering

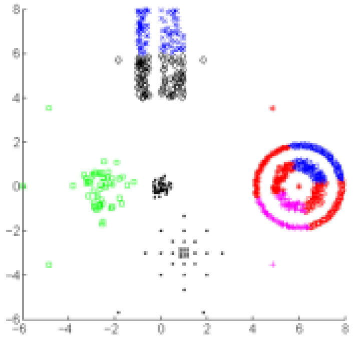


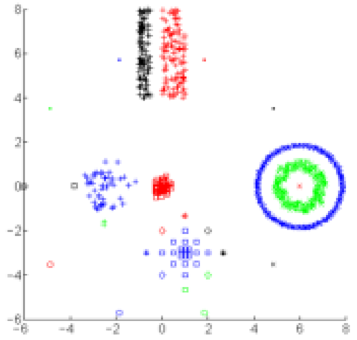
Combine multiple partitions of given data into a single partition of better quality

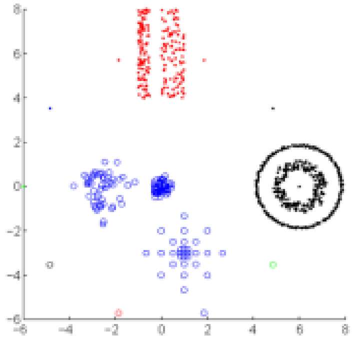
Why Ensemble Clustering?

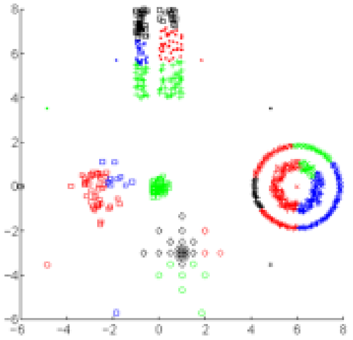
- Different clustering algorithms may produce different partitions because they impose different structure on the data; No single clustering algorithm is optimal

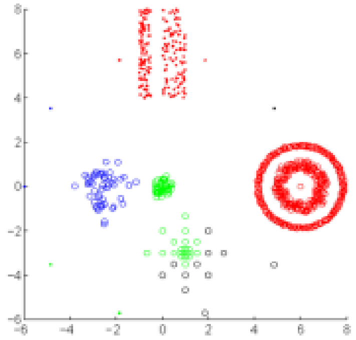

(a)Input data.


(b)K-means clustering, $k = 8$.


(c)Clustering with the SL method, threshold at 0.55, resulting in 27 clusters.


(d)Clustering with the SL method, forcing 8 clusters.


(e)Clustering with the CL method, threshold at 2.6, resulting in 22 clusters.

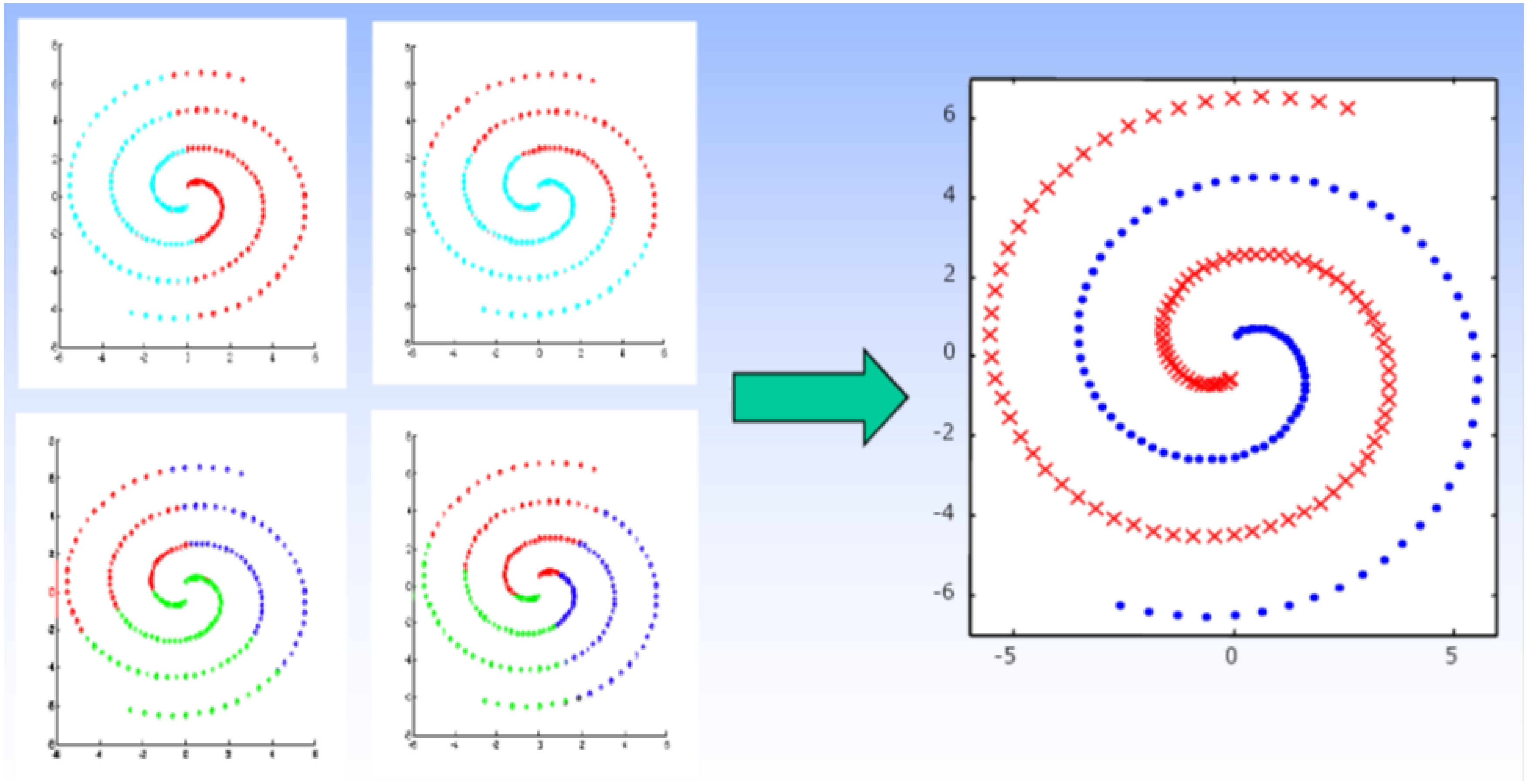

(f)Clustering with the CL method, forcing 8 clusters.

Results of clusterings using different algorithms (K-means, single-link, and complete-link) with different parameters. (Fred & Jain, PAMI, 2005)

Different realizations of the same algorithm may generate different partitions

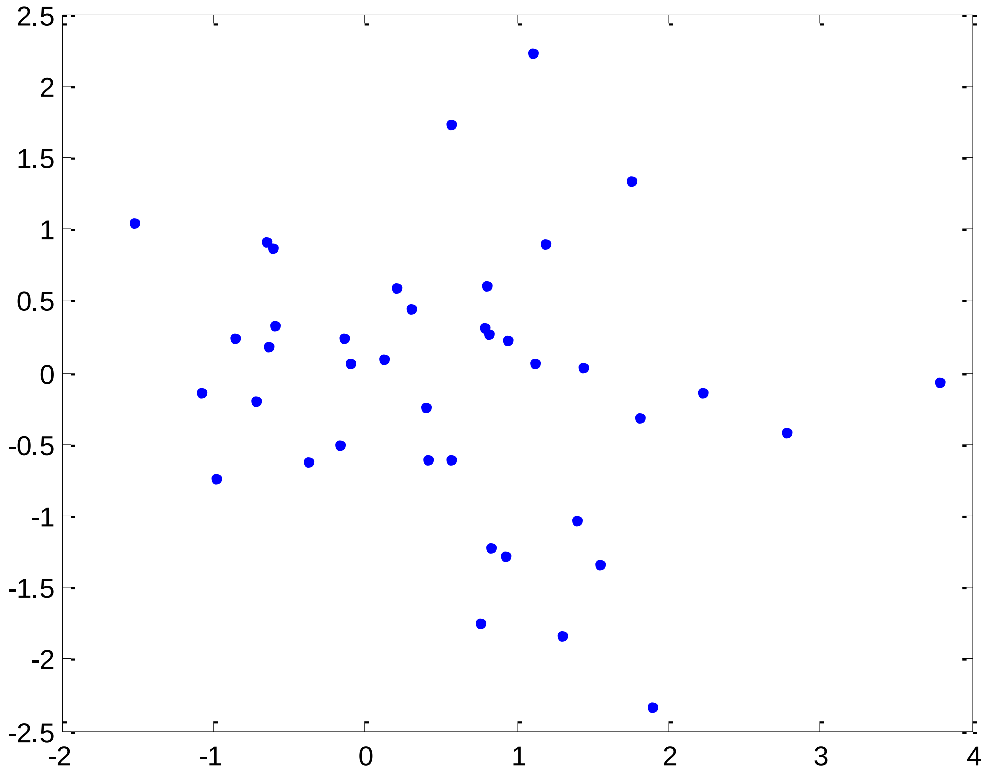
Why Ensemble Clustering?

- Goal
  - Exploit the complementary nature of different partitions
  - Each partition can be viewed as taking a different “look” or “cut” through data



Illustrative Example

- Apply K-means to ‘kmeans_data1.csv’ with K=5 and number of repeated runs = 1.
- Data ‘kmeans_data1.csv’:

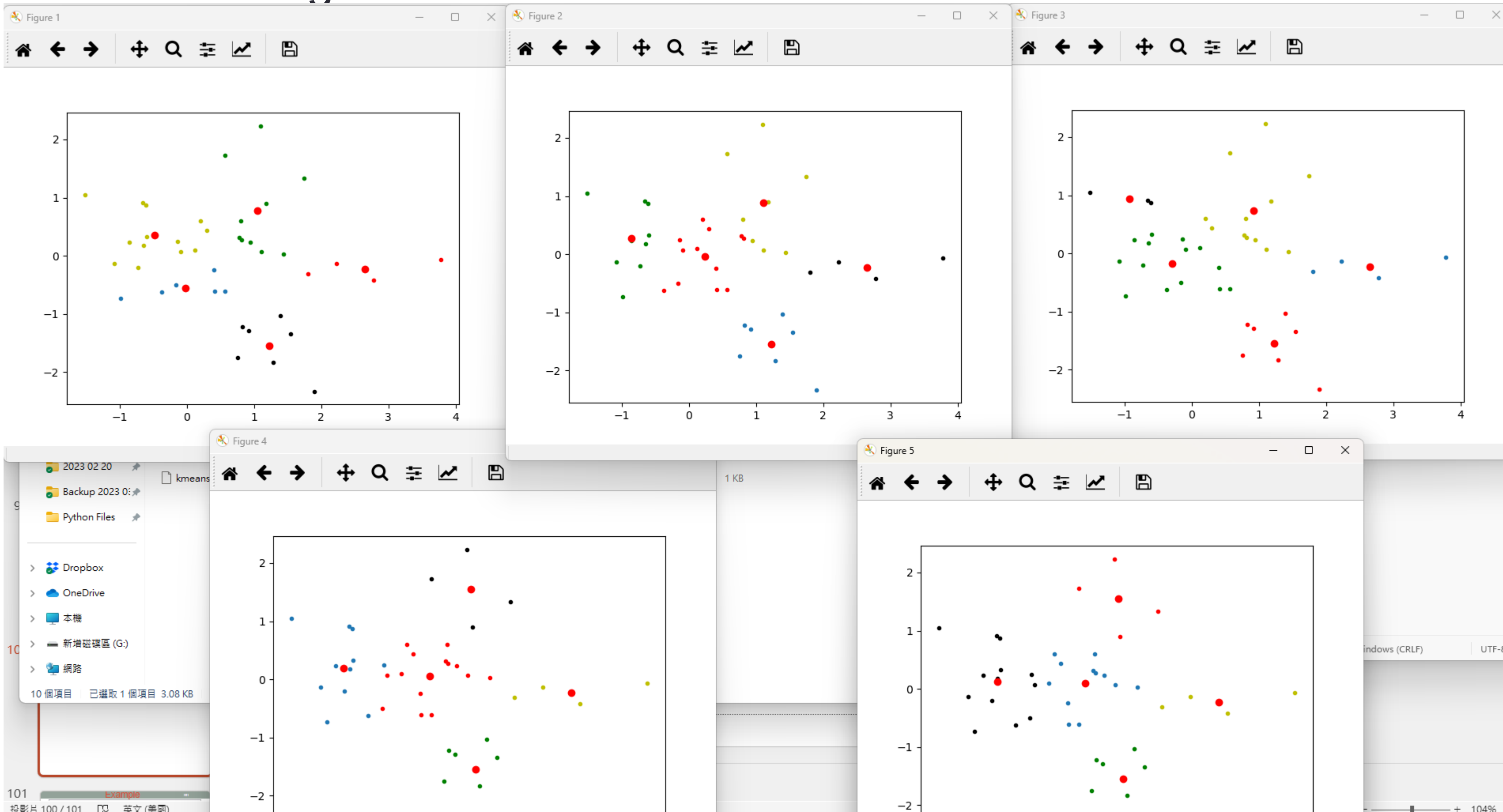


Illustrative Example

```
###Illustrative example
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
data = pd.read_csv('Data\\kmeans_data1.csv') #Load the data file
#pd.plotting.scatter_matrix(data); #Visualize the data
kmeans = KMeans(n_clusters=5,n_init=1).fit(data) #perform K-means clustering with number of clusters
= 2
centroids = kmeans.cluster_centers_ #Extract the cluster centroids
labels = kmeans.labels_ #Extract the labels of clusters
plt.figure(); #Plot the figure
plt.plot(data.iloc[labels==0,0],data.iloc[labels==0,1],'g.');  #plot the data with label = 0 (color: g)
plt.plot(data.iloc[labels==1,0],data.iloc[labels==1,1],'.');  #plot the data with label = 1 (color: b)
plt.plot(data.iloc[labels==2,0],data.iloc[labels==2,1],'y.');  #plot the data with label = 1 (color: b)
plt.plot(data.iloc[labels==3,0],data.iloc[labels==3,1],'r.');  #plot the data with label = 1 (color: b)
plt.plot(data.iloc[labels==4,0],data.iloc[labels==4,1],'k.');  #plot the data with label = 1 (color: b)
plt.plot(centroids[:,0],centroids[:,1],'ro') #plot the cluster centroid
```

Illustrative Example

- Clustering results



Illustrative Example

- Because different initial guesses are used, obtain different clustering results.
- But density samples are usually grouped into the same clusters.
- This is the key idea of ensemble clustering.

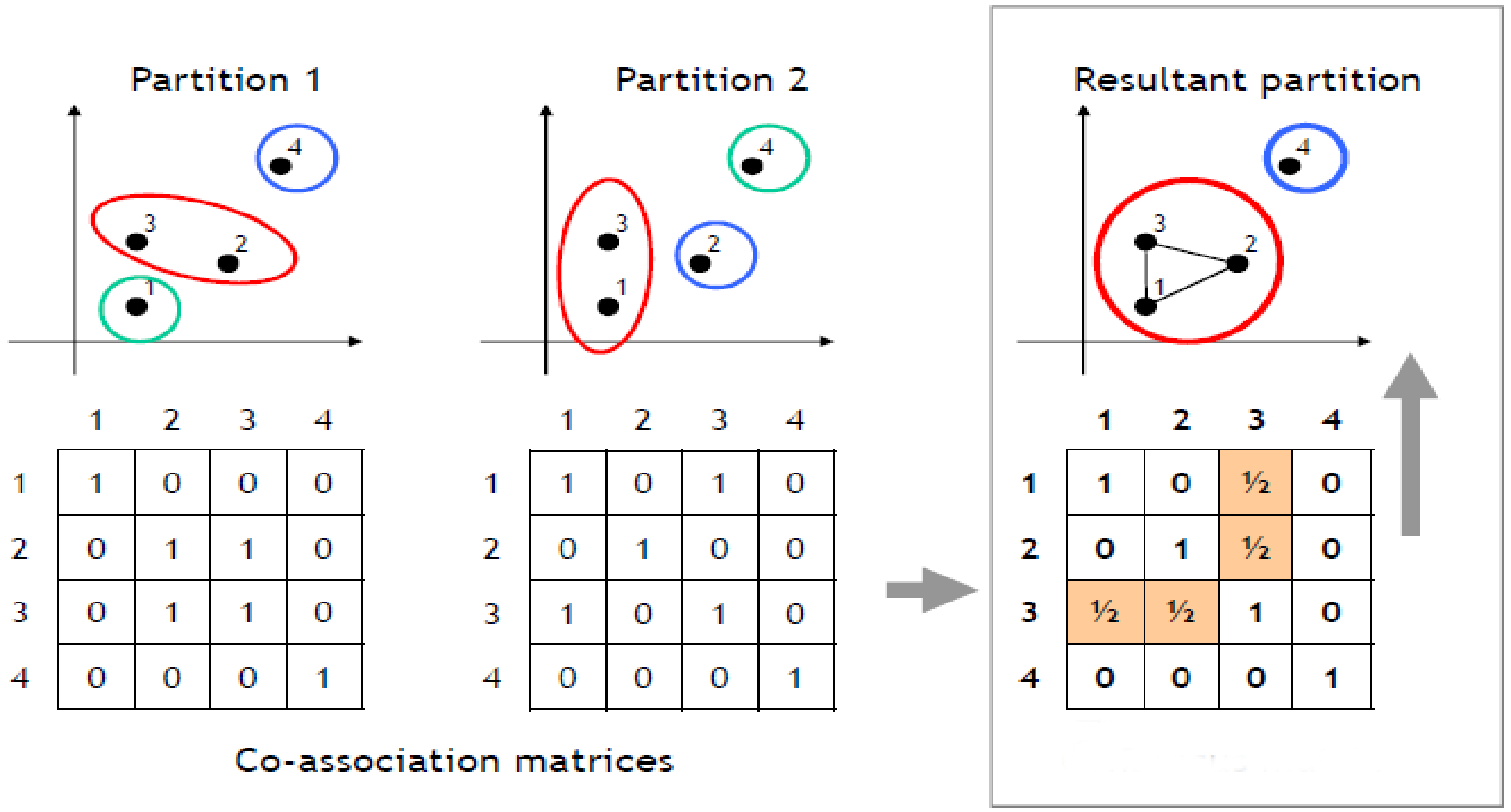
Example to Real Data

Apply the ensemble clustering method to ecoli dataset.

```
##################################
##################################
###Method 2
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
from sklearn.metrics import confusion_matrix #Import confusion matrix module
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import completeness_score
from sklearn import metrics
import numpy as np
from sklearn.cluster import AgglomerativeClustering
np.random.seed(8); n_clusters = 7
data = pd.read_csv('Data\\ecoli.csv') #Load the data file
#Apply K-means to Ecoli dataset
kmeans = KMeans(n_clusters=n_clusters).fit(data.iloc[:,:-1]) #perform K-means clustering with number of clusters = 3
centroids = kmeans.cluster_centers_ #Extract the cluster centroids
labels = kmeans.labels_ #Extract the labels of clusters
label_encoder = LabelEncoder()
results = confusion_matrix(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(results)
acc = metrics.completeness_score(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(acc)

#Apply Ensemble clustering to Ecoli dataset
all_labels = [];
data = pd.read_csv('Data\\ecoli.csv') #Load the data file
for ii in range(50):
kmeans = KMeans(n_clusters=10,init='random',n_init=1).fit(data.iloc[:,:-1]) #perform K-means clustering with number of clusters = 3
labels = kmeans.labels_ #Extract the labels of clusters
all_labels.append(labels)
all_labels = np.array(all_labels).T
cluster = AgglomerativeClustering(n_clusters=n_clusters, affinity='euclidean').fit(all_labels)
labels = cluster.labels_ #Extract the labels of clusters
label_encoder = LabelEncoder()
results = confusion_matrix(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(results)
acc = metrics.completeness_score(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(acc)
```

Co-association matrix based methods

- Basic idea: first compute a co-association matrix based on multiple data partitions, then apply a similarity-based clustering algorithm (e.g., single link and normalized cut) to the co-association matrix to obtain the final partition of the data.



	1	2	3	4
1	1	0	0	0
2	0	1	1	0
3	0	1	1	0
4	0	0	0	1

	1	2	3	4
1	1	0	1	0
2	0	1	0	0
3	1	0	1	0
4	0	0	0	1

	1	2	3	4
1	1	0	½	0
2	0	1	½	0
3	½	½	1	0
4	0	0	0	1

Example to Real Data

Apply the K-means to ecoli dataset.

```
###Method
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
from sklearn.metrics import confusion_matrix #Import confusion matrix module
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import completeness_score
from sklearn import metrics
import numpy as np
from sklearn.cluster import AgglomerativeClustering

np.random.seed(8); n_clusters = 7

data = pd.read_csv('Data\\ecoli.csv') #Load the data file
#Apply K-means to Ecoli dataset
kmeans = KMeans(n_clusters=n_clusters).fit(data.iloc[:,:-1]) #perform K-means clustering with number of clusters = 3
centroids = kmeans.cluster_centers_ #Extract the cluster centroids
labels = kmeans.labels_ #Extract the labels of clusters
label_encoder = LabelEncoder()
results = confusion_matrix(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(results)
acc = metrics.completeness_score(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(acc)
```

Example to Real Data

Apply the ensemble clustering method to ecoli dataset.

```
#Apply Ensemble clustering to Ecoli dataset
no_samples = data.shape[0]
no_estimators = 50
#Declare the weight of each vote
vote = 1/no_estimators
#co_association matrix is no_estimators X no_estimators (no_estimators patterns)
co_association = np.zeros((no_samples, no_samples))

#for each of your estimators
for est in range(no_estimators):
   #fit the data and grab the labels
   kmeans = KMeans(n_clusters=10,init='random',n_init=1).fit(data)
   labels = kmeans.labels_
   #find all associations and transform it into a numpy array
   res = [[int(i == j) for i in labels] for j in labels]
   res = np.array(res)
   #Vote and update the co_association matriz
   res = res * vote
   co_association = co_association + res
distance_matrix = 1-co_association
cluster = AgglomerativeClustering(n_clusters=n_clusters, affinity='euclidean',compute_distances=True).fit(distance_matrix)
labels = cluster.labels_ #Extract the labels of clusters
label_encoder = LabelEncoder()
results = confusion_matrix(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(results)
acc =  metrics.completeness_score(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(acc)
```

Apply K-means

Build co-association Matrix

Remark on Ensemble Clustering

- Other than K-means, can also use Gaussian Mixture Model or other clustering methods.
- The selected clustering method should generate different results when different random seeds are applied. If obtained same results, the ensemble clustering is meaningless.
- Have to set the inner number of clusters (i.e. have to set the number of clusters for K-means.)
- Set the number of estimators (i.e. no_estimators) in the code.