Clustering in Python
k-means clustering

Dr. Chun-Hao Chen

Outline

1、Import Library

2、Generate Clustering Instances

3、Call KMeans() for Clustering

4、Show Clustering Result

5、Find Suitable Number of Clusters

1. Import Library

✓import matplotlib.pyplot as plt # Draw figures

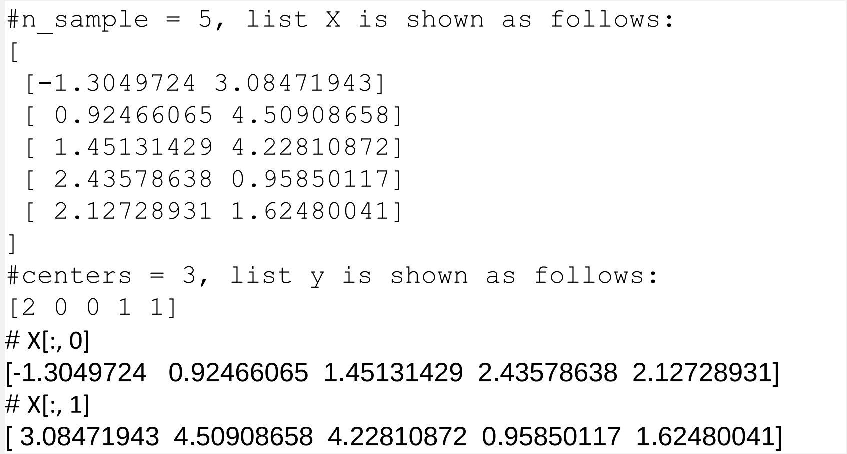
✓from sklearn.datasets import make_blobs # make_blobs: generate clustering instances

✓from sklearn.cluster import KMeans # Kmeans: k-means clustering algorithm

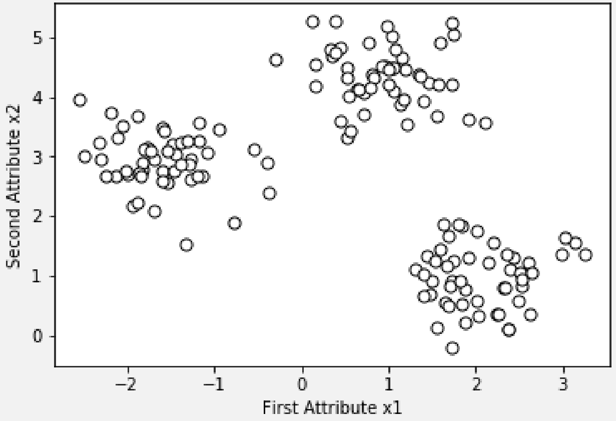
2. Generate Clustering Instances

```
# create instances
# X: attribute values
# y: group label
X, y = make_blobs(
    n_samples=150, n_features=2,
    centers=3, cluster_std=0.5,
    shuffle=True, random_state=0
)
print(X)
print(X[:, 0])
print(X[:, 1])

# Draw scatter of the instances
plt.scatter(
    X[:, 0], X[:, 1],
    c='white', marker='o',
    edgecolor='black', s=50
)
plt.xlabel('First Attribute x1')
plt.ylabel('Second Attribute x2')
plt.show()
```



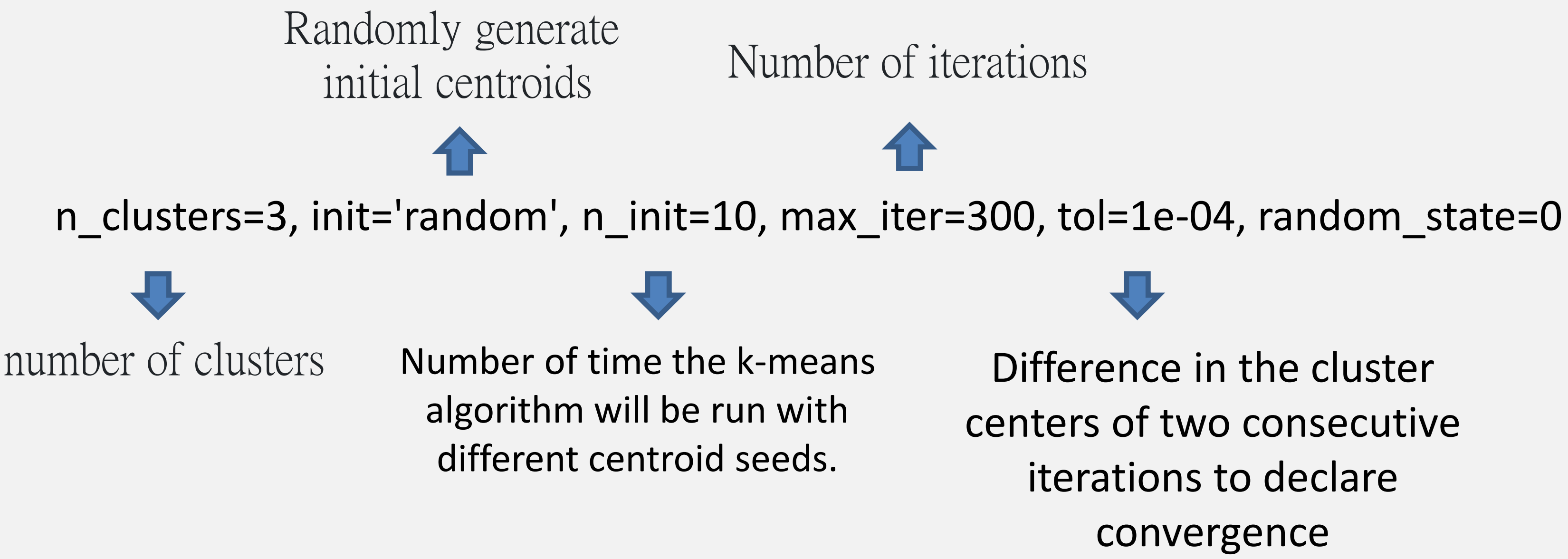
```
#n_sample = 5, list X is shown as follows:
[
 [-1.3049724 3.08471943]
 [ 0.92466065 4.50908658]
 [ 1.45131429 4.22810872]
 [ 2.43578638 0.95850117]
 [ 2.12728931 1.62480041]
]
#centers = 3, list y is shown as follows:
[2 0 0 1 1]
# X[:, 0]
[-1.3049724  0.92466065  1.45131429  2.43578638  2.12728931]
# X[:, 1]
[ 3.08471943  4.50908658  4.22810872  0.95850117  1.62480041]
```



n_samples = 150
centers = 3

3、Call KMeans() for Clustering

```
km = KMeans(
    n_clusters=3, init='random', n_init=10, max_iter=300, tol=1e-04, random_state=0
)
y_km = km.fit_predict(X) # Compute cluster centers and predict cluster index for each sample.
print(y_km) # Continue previous example: y_km = [1 2 2 0 0]
```



Randomly generate initial centroids

Number of iterations

n_clusters=3, init='random', n_init=10, max_iter=300, tol=1e-04, random_state=0

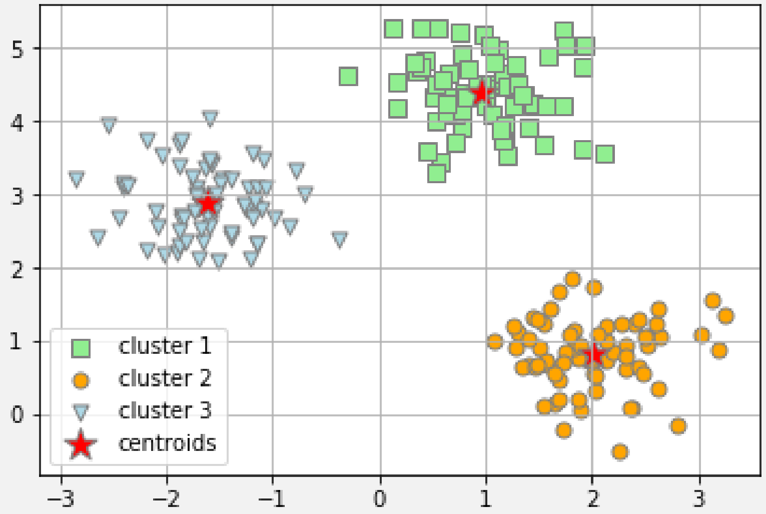
number of clusters

Number of time the k-means algorithm will be run with different centroid seeds.

Difference in the cluster centers of two consecutive iterations to declare convergence

4、Show Clustering Result

```
# Draw the 3 clusters
plt.scatter( X[y_km == 0, 0], X[y_km == 0, 1], s=50, c='lightgreen', marker='s', edgecolor='gray', label='cluster 1')
plt.scatter( X[y_km == 1, 0], X[y_km == 1, 1], s=50, c='orange', marker='o', edgecolor='gray', label='cluster 2')
plt.scatter( X[y_km == 2, 0], X[y_km == 2, 1], s=50, c='lightblue', marker='v', edgecolor='gray', label='cluster 3')
# Draw the centroids
plt.scatter(
km.cluster_centers_[:, 0], km.cluster_centers_[:, 1], s=250, marker='*', c='red', edgecolor='gray', label='centroids'
)
plt.legend(scatterpoints=1)
plt.grid()
plt.show()
```

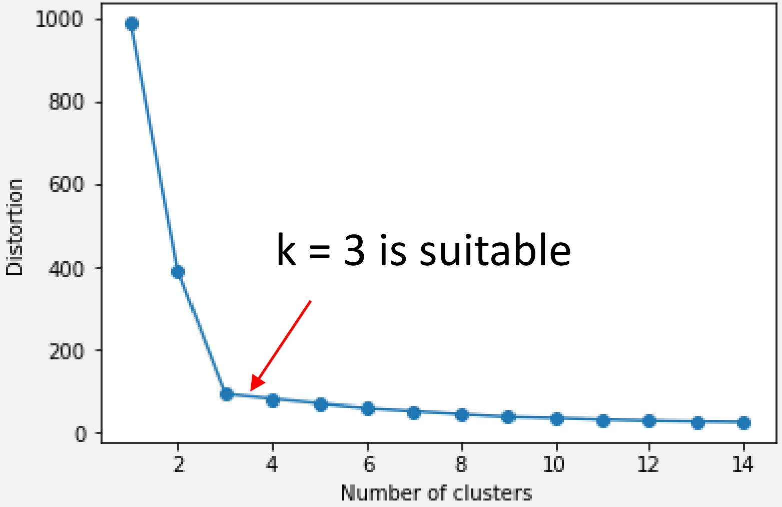


5、Find Suitable Number of Clusters

```
distortions = []
for i in range(1, 15):
    km = KMeans( n_clusters=i, init='random', n_init=10, max_iter=300, tol=1e-04, random_state=0 )
    km.fit(X)
    distortions.append(km.inertia_)

# Draw figures
plt.plot(range(1, 15), distortions, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.show()
```

Inertia_: Sum of squared distances of samples to their closest cluster center.



k = 3 is suitable