k-means

July 24, 2020

0.1 Prepare modules and data.¶

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine

from sklearn.cluster import KMeans

wine = load_wine()

X = pd.DataFrame(wine.data, columns=wine.feature_names)
Y = pd.DataFrame(wine.target)
```

[2]:
```
X.head(5)
```

[2]:
```
   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
0    14.23        1.71  2.43               15.6      127.0           2.80
1    13.20        1.78  2.14               11.2      100.0           2.65
2    13.16        2.36  2.67               18.6      101.0           2.80
3    14.37        1.95  2.50               16.8      113.0           3.85
4    13.24        2.59  2.87               21.0      118.0           2.80

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
0        3.06                  0.28             2.29             5.64  1.04
1        2.76                  0.26             1.28             4.38  1.05
2        3.24                  0.30             2.81             5.68  1.03
3        3.49                  0.24             2.18             7.80  0.86
4        2.69                  0.39             1.82             4.32  1.04

   od280/od315_of_diluted_wines  proline
0                          3.92   1065.0
1                          3.40   1050.0
2                          3.17   1185.0
3                          3.45   1480.0
4                          2.93    735.0
```

[3]:
```
X.shape
```

[3]: (178, 13)

[4]:
```
model = KMeans(n_clusters=3, random_state=1)
```

[5]:
```
X['clusters'] = model.fit_predict(X)
```

[6]:
```
X
```

[6]:
```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
0      14.23        1.71  2.43               15.6      127.0           2.80
1      13.20        1.78  2.14               11.2      100.0           2.65
2      13.16        2.36  2.67               18.6      101.0           2.80
3      14.37        1.95  2.50               16.8      113.0           3.85
4      13.24        2.59  2.87               21.0      118.0           2.80
..       …           …     …                  …          …              …
173    13.71        5.65  2.45               20.5       95.0           1.68
174    13.40        3.91  2.48               23.0      102.0           1.80
175    13.27        4.28  2.26               20.0      120.0           1.59
176    13.17        2.59  2.37               20.0      120.0           1.65
177    14.13        4.10  2.74               24.5       96.0           2.05

     flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
0          3.06                  0.28             2.29             5.64  1.04
1          2.76                  0.26             1.28             4.38  1.05
2          3.24                  0.30             2.81             5.68  1.03
3          3.49                  0.24             2.18             7.80  0.86
4          2.69                  0.39             1.82             4.32  1.04
..          …                     …                …                …     …
173        0.61                  0.52             1.06             7.70  0.64
174        0.75                  0.43             1.41             7.30  0.70
175        0.69                  0.43             1.35            10.20  0.59
176        0.68                  0.53             1.46             9.30  0.60
177        0.76                  0.56             1.35             9.20  0.61

     od280/od315_of_diluted_wines  proline  clusters
0                            3.92   1065.0         1
1                            3.40   1050.0         1
2                            3.17   1185.0         1
3                            3.45   1480.0         1
4                            2.93    735.0         2
..                            …        …         …
173                          1.74    740.0         2
174                          1.56    750.0         2
175                          1.56    835.0         2
176                          1.62    840.0         2
177                          1.60    560.0         0
```

```
[178 rows x 14 columns]
```

[7]:
```python
for i in range(3):
    count = (X['clusters'] == i ).sum()
    print(f'cluster {i} count is {count}')
```

```
cluster 0 count is 69
cluster 1 count is 47
cluster 2 count is 62
```

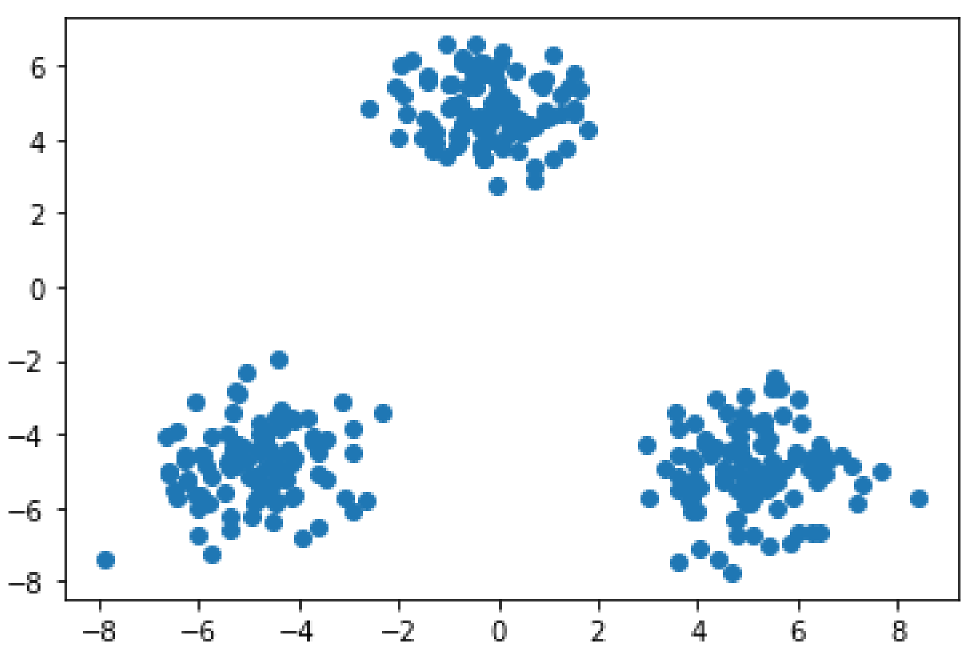
0.2 k-means with numpy

[8]:
```python
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

[9]:
```python
def gen_data():
    x1 = np.random.normal(size=(100, 2)) + np.array([-5, -5])
    x2 = np.random.normal(size=(100, 2)) + np.array([5, -5])
    x3 = np.random.normal(size=(100, 2)) + np.array([0, 5])
    return np.vstack((x1, x2, x3))
```

[10]:
```python
#
X_train = gen_data()
#
plt.scatter(X_train[:, 0], X_train[:, 1])
```

[10]:
```
<matplotlib.collections.PathCollection at 0x7f86af955f50>
```

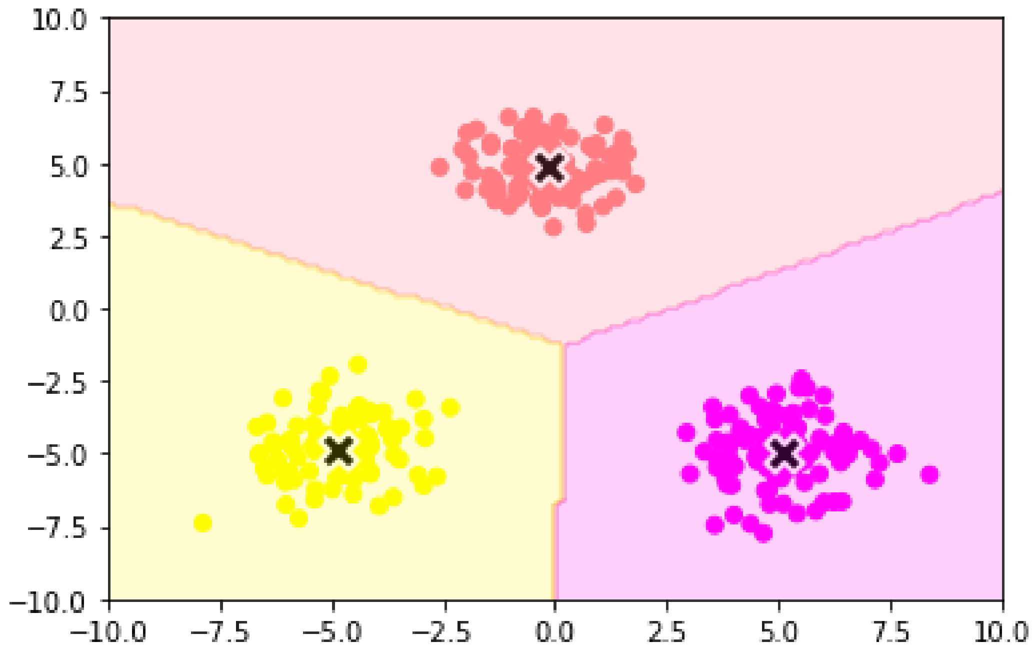


[11]:
```
def distance(x1, x2):
    return np.sum((x1 - x2)**2, axis=1)

n_clusters = 3
iter_max = 100

# Randomly initialize each cluster center
centers = X_train[np.random.choice(len(X_train), n_clusters, replace=False)]

for _ in range(iter_max):
    prev_centers = np.copy(centers)
    D = np.zeros((len(X_train), n_clusters))
    # For each data point, calculate the distance to the center of each cluster
    for i, x in enumerate(X_train):
        D[i] = distance(x, centers)
    # Assign the closest cluster to each data point
    cluster_index = np.argmin(D, axis=1)
    # Calculate the center of each cluster
    for k in range(n_clusters):
        index_k = cluster_index == k
        centers[k] = np.mean(X_train[index_k], axis=0)
    # convergent judgment
    if np.allclose(prev_centers, centers):
        break
```

[12]:
```
def plt_result(X_train, centers, xx):
    #  Visualize data
    plt.scatter(X_train[:, 0], X_train[:, 1], c=y_pred, cmap='spring')
    # Visualize the center
    plt.scatter(centers[:, 0], centers[:, 1], s=200, marker='X', lw=2,␣
 ↪c='black', edgecolor="white")
    # Area Visualization
    pred = np.empty(len(xx), dtype=int)
    for i, x in enumerate(xx):
        d = distance(x, centers)
        pred[i] = np.argmin(d)
    plt.contourf(xx0, xx1, pred.reshape(100, 100), alpha=0.2, cmap='spring')
```

[13]:
```
y_pred = np.empty(len(X_train), dtype=int)
for i, x in enumerate(X_train):
    d = distance(x, centers)
    y_pred[i] = np.argmin(d)
```

[14]:
```
xx0, xx1 = np.meshgrid(np.linspace(-10, 10, 100), np.linspace(-10, 10, 100))
xx = np.array([xx0, xx1]).reshape(2, -1).T

plt_result(X_train, centers, xx)
```



[15]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, random_state=0).fit(X_train)
```

[16]:
```
print("labels: {}".format(kmeans.labels_))
print("cluster_centers: {}".format(kmeans.cluster_centers_))
kmeans.cluster_centers_
```

```
labels: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
cluster_centers: [[-0.17020089  4.83880202]
 [-4.84396475 -4.83002982]
 [ 5.10112247 -4.97817271]]
```

[16]:
```
array([[-0.17020089,  4.83880202],
       [-4.84396475, -4.83002982],
       [ 5.10112247, -4.97817271]])
```

[17]:
```
plt_result(X_train, kmeans.cluster_centers_, xx)
```