K-means clustering

import library

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as colors
from matplotlib import cm
```

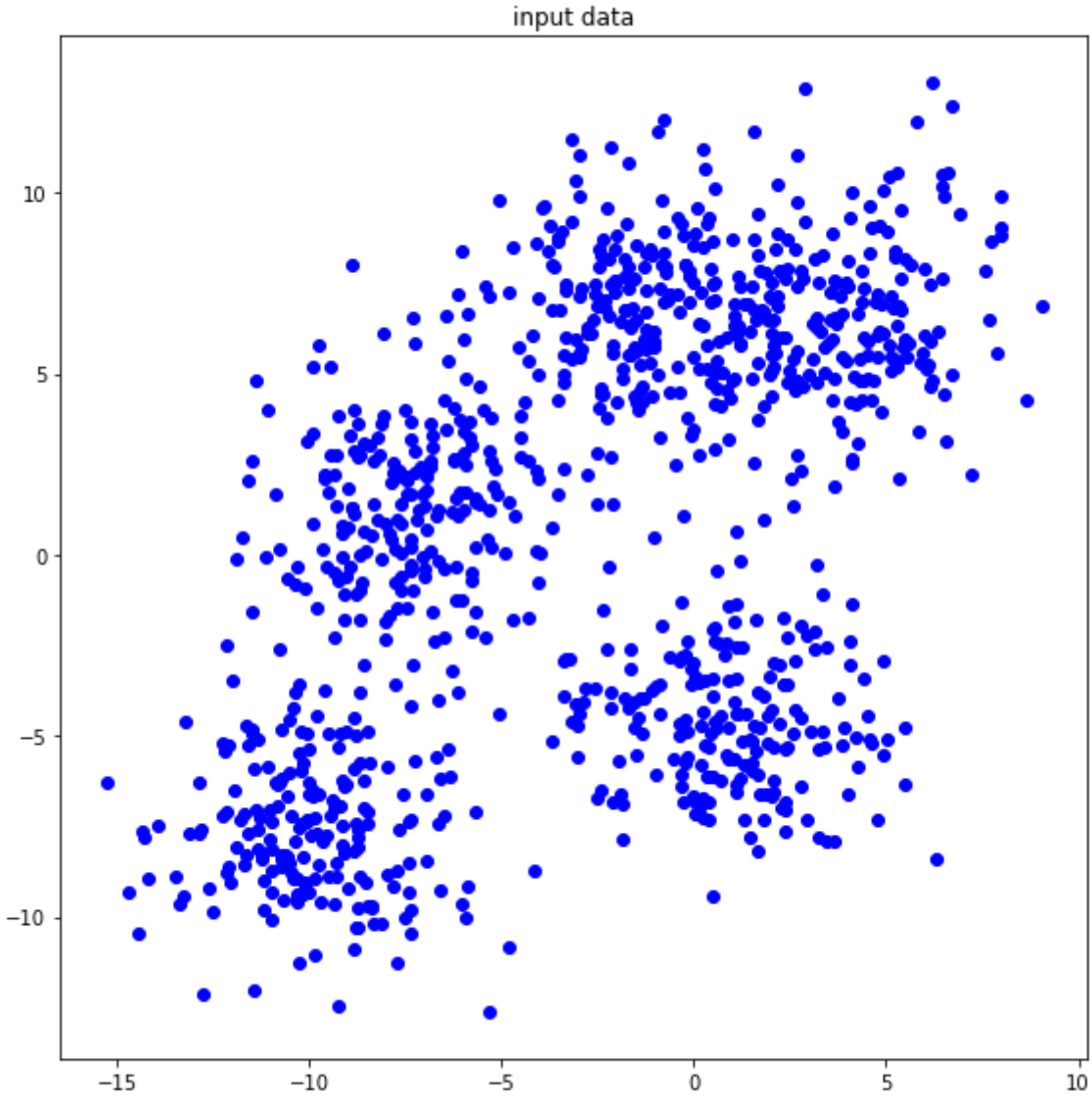
load data

In [ ]:
```
fname_data = 'assignment_11_data.csv'

feature = np.genfromtxt(fname_data, delimiter=',')

x = feature[:,0]
y = feature[:,1]

number_data     = np.size(feature, 0)
number_feature  = np.size(feature, 1)

print('number of data : {}'.format(number_data))
print('number of feature : {}'.format(number_feature))
```

```
number of data : 1000
number of feature : 2
```

plot the input data

In [ ]:
```
plt.figure(figsize=(8,8))
plt.title('input data')

plt.scatter(x, y, color='blue')

plt.tight_layout()
plt.show()
```



compute distance

- feature : $n \times m$, center : $1 \times m$, distance : $n \times 1$
- n : number of data, m : number of features

In [ ]:

```
def compute_distance(feature, center):

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    distance = np.zeros(np.size(feature, 0))
    for i in range(np.size(feature, 0)):
        temp = (feature[i,0]- center[0])**2 + (feature[i,1]- center[1])**2
        distance[i]= np.sqrt(temp)

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return distance

center=np.zeros(2)
distance=compute_distance(feature,center)
# print(distance)
# for i in range(4):
#     print(i)
```

compute centroid

- feature : $n \times m$, label_feature : $n \times 1$, value_label : 1×1, centroid : $1 \times m$
- n : number of data, m : number of features

In [ ]:
```
def compute_centroid(feature, label_feature, label):

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    centroid = np.zeros((1, np.size(feature, 1)))
    count = 0

    for i in range(np.size(feature, 0)):
        if label_feature[i] == label:
            centroid += feature[i]
            count += 1
    if count != 0:
        centroid /= count

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return centroid
```

compute label

- distance : $n \times k$, label_feature : $n \times 1$
- n : number of data, k : number of clusters

In [ ]:
```
def compute_label(distance):

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    label_feature = np.argmin(distance, axis=1)

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return label_feature
```

the number of clusters $K = 2$

In [ ]:
```
number_cluster          = 2
number_iteration        = 80     # you can modify this value
loss_iteration_02       = np.zeros(number_iteration)
centroid_iteration_02   = np.zeros((number_iteration, number_cluster, number_feature))
label_feature_02        = np.random.randint(0, number_cluster, size=(number_data))
```

In [ ]:
```
# ++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#
centroid = np.zeros((number_cluster, number_feature))
```

```
distance = np.zeros((number_data, number_cluster))
for i in range(number_iteration):
    for k in range(number_cluster):
        centroid[k] = compute_centroid(feature, label_feature_02, k)
        if centroid[k][1] and centroid[k][0] == 0:
            centroid[k]= centroid_iteration_02[i-1][k]
    centroid_iteration_02[i]= centroid

    for k in range(number_cluster):
        distance[:,k] = compute_distance(feature, centroid[k])

    label_feature_02 = compute_label(distance)
    label_feature_02 = label_feature_02.astype('uint32')

    loss = 0
    for j in range(number_data):
        loss += (feature[j,0]-centroid[label_feature_02[j],0])**2+(feature[j,1]-centr
    loss /= number_data
    loss_iteration_02[i]= loss

#
# +++++++++++++++++++++++++++++++++++++++++++++++++++
```

the number of clusters $K = 4$

In [ ]:

```
number_cluster          = 4
number_iteration        = 80     # you can modify this value
loss_iteration_04       = np.zeros(number_iteration)
centroid_iteration_04   = np.zeros((number_iteration, number_cluster, number_feature))
label_feature_04        = np.random.randint(0, number_cluster, size=(number_data))
```

In [ ]:

```
# +++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#
centroid = np.zeros((number_cluster, number_feature))
distance = np.zeros((number_data, number_cluster))
for i in range(number_iteration):
    for k in range(number_cluster):
        centroid[k] = compute_centroid(feature, label_feature_04, k)
        if centroid[k][1] and centroid[k][0] == 0:
            centroid[k]= centroid_iteration_04[i-1][k]
    centroid_iteration_04[i]= centroid

    for k in range(number_cluster):
        distance[:,k] = compute_distance(feature, centroid[k])

    label_feature_04 = compute_label(distance)
    label_feature_04 = label_feature_04.astype('uint32')
    loss = 0
    for j in range(number_data):
        loss += (feature[j,0]-centroid[label_feature_04[j],0])**2+(feature[j,1]-centr
    loss /= number_data
    loss_iteration_04[i]= loss
#
# +++++++++++++++++++++++++++++++++++++++++++++++++++
```

the number of clusters $K = 8$

In [ ]:
```
number_cluster          = 8
number_iteration        = 80     # you can modify this value
loss_iteration_08       = np.zeros(number_iteration)
centroid_iteration_08   = np.zeros((number_iteration, number_cluster, number_feature))
label_feature_08        = np.random.randint(0, number_cluster, size=(number_data))
```

In [ ]:
```
# +++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#
centroid = np.zeros((number_cluster, number_feature))
distance = np.zeros((number_data, number_cluster))
for i in range(number_iteration):
    for k in range(number_cluster):
        centroid[k] = compute_centroid(feature, label_feature_08, k)
        if centroid[k][1] and centroid[k][0] == 0:
            centroid[k]= centroid_iteration_08[i-1][k]
    centroid_iteration_08[i]= centroid

    for k in range(number_cluster):
        distance[:,k] = compute_distance(feature, centroid[k])

    label_feature_08 = compute_label(distance)
    label_feature_08 = label_feature_08.astype('uint32')
    loss = 0
    for j in range(number_data):
        loss += (feature[j,0]-centroid[label_feature_08[j],0])**2+(feature[j,1]-centr
    loss /= number_data
    loss_iteration_08[i]= loss

#
# +++++++++++++++++++++++++++++++++++++++++++++++++++
```

the number of clusters $K = 16$

In [ ]:
```
number_cluster          = 16
number_iteration        = 40     # you can modify this value
loss_iteration_16       = np.zeros(number_iteration)
centroid_iteration_16   = np.zeros((number_iteration, number_cluster, number_feature))
label_feature_16        = np.random.randint(0, number_cluster, size=(number_data))
```

In [ ]:
```
# +++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#
centroid = np.zeros((number_cluster, number_feature))
distance = np.zeros((number_data, number_cluster))
for i in range(number_iteration):
    for k in range(number_cluster):
        centroid[k] = compute_centroid(feature, label_feature_16, k)
        if centroid[k][1] and centroid[k][0] == 0:
            centroid[k]= centroid_iteration_16[i-1][k]
    centroid_iteration_16[i]= centroid

    for k in range(number_cluster):
        distance[:,k] = compute_distance(feature, centroid[k])

    label_feature_16 = compute_label(distance)
    label_feature_16 = label_feature_16.astype('uint32')
    loss = 0
    for j in range(number_data):
```

```
            loss += (feature[j,0]-centroid[label_feature_16[j],0])**2+(feature[j,1]-centr
        loss /= number_data
        loss_iteration_16[i]= loss

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
```

---

functions for presenting the results

In [ ]:
```
def function_result_01():

    print("final loss (K=2) = {:13.10f}".format(loss_iteration_02[-1]))
```

In [ ]:
```
def function_result_02():

    print("final loss (K=4) = {:13.10f}".format(loss_iteration_04[-1]))
```

In [ ]:
```
def function_result_03():

    print("final loss (K=8) = {:13.10f}".format(loss_iteration_08[-1]))
```

In [ ]:
```
def function_result_04():

    print("final loss (K=16) = {:13.10f}".format(loss_iteration_16[-1]))
```

In [ ]:
```
def function_result_05():

    plt.figure(figsize=(8,6))
    plt.title('loss (K=2)')

    plt.plot(loss_iteration_02, '-', color='red')
    plt.xlabel('iteration')
    plt.ylabel('loss')

    plt.tight_layout()
    plt.show()
```

In [ ]:
```
def function_result_06():

    plt.figure(figsize=(8,6))
    plt.title('loss (K=4)')

    plt.plot(loss_iteration_04, '-', color='red')
    plt.xlabel('iteration')
    plt.ylabel('loss')
```

```
    plt.tight_layout()
    plt.show()
```

In [ ]:
```
def function_result_07():

    plt.figure(figsize=(8,6))
    plt.title('loss (K=8)')

    plt.plot(loss_iteration_08, '-', color='red')
    plt.xlabel('iteration')
    plt.ylabel('loss')

    plt.tight_layout()
    plt.show()
```

In [ ]:
```
def function_result_08():

    plt.figure(figsize=(8,6))
    plt.title('loss (K=16)')

    plt.plot(loss_iteration_16, '-', color='red')
    plt.xlabel('iteration')
    plt.ylabel('loss')

    plt.tight_layout()
    plt.show()
```

In [ ]:
```
def function_result_09():

    plt.figure(figsize=(8,8))
    plt.title('centroid (K=2)')

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    x_initial=np.zeros(2)
    y_initial=np.zeros(2)
    x_final=np.zeros(2)
    y_final=np.zeros(2)

    for i in range(2):
        plt.plot(centroid_iteration_02[:,i,0], centroid_iteration_02[:,i,1], '-',labe
        x_initial[i]=centroid_iteration_02[0,i,0]
        y_initial[i]=centroid_iteration_02[0,i,1]
        x_final[i]=centroid_iteration_02[number_iteration-1,i,0]
        y_final[i]=centroid_iteration_02[number_iteration-1,i,1]


    plt.scatter(x_initial, y_initial, color='blue', marker='o', label='initial')
    plt.scatter(x_final, y_final, color='red', marker='s', label='final')

    plt.legend(loc='upper right')
    plt.tight_layout()
    plt.show()
    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:
```
def function_result_10():
```

```
    plt.figure(figsize=(8,8))
    plt.title('centroid (K=4)')

    # +++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    x_initial=np.zeros(4)
    y_initial=np.zeros(4)
    x_final=np.zeros(4)
    y_final=np.zeros(4)

    for i in range(4):
        plt.plot(centroid_iteration_04[:,i,0], centroid_iteration_04[:,i,1], '-',labe
        x_initial[i]=centroid_iteration_04[0,i,0]
        y_initial[i]=centroid_iteration_04[0,i,1]
        x_final[i]=centroid_iteration_04[number_iteration-1,i,0]
        y_final[i]=centroid_iteration_04[number_iteration-1,i,1]


    plt.scatter(x_initial, y_initial, color='blue', marker='o', label='initial')
    plt.scatter(x_final, y_final, color='red', marker='s', label='final')

    plt.legend(loc='upper right')
    plt.tight_layout()
    plt.show()

    #
    # +++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:

```
def function_result_11():

    plt.figure(figsize=(8,8))
    plt.title('centroid (K=8)')

    # +++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    x_initial=np.zeros(8)
    y_initial=np.zeros(8)
    x_final=np.zeros(8)
    y_final=np.zeros(8)

    for i in range(8):
        plt.plot(centroid_iteration_08[:,i,0], centroid_iteration_08[:,i,1], '-',labe
        x_initial[i]=centroid_iteration_08[0,i,0]
        y_initial[i]=centroid_iteration_08[0,i,1]
        x_final[i]=centroid_iteration_08[number_iteration-1,i,0]
        y_final[i]=centroid_iteration_08[number_iteration-1,i,1]


    plt.scatter(x_initial, y_initial, color='blue', marker='o', label='initial')
    plt.scatter(x_final, y_final, color='red', marker='s', label='final')

    plt.legend(loc='upper right')
    plt.tight_layout()
    plt.show()

    #
    # +++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:

```
def function_result_12():

    plt.figure(figsize=(8,8))
    plt.title('centroid (K=16)')

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    x_initial=np.zeros(16)
    y_initial=np.zeros(16)
    x_final=np.zeros(16)
    y_final=np.zeros(16)

    for i in range(16):
        plt.plot(centroid_iteration_16[:,i,0], centroid_iteration_16[:,i,1], '-',labe
        x_initial[i]=centroid_iteration_16[0,i,0]
        y_initial[i]=centroid_iteration_16[0,i,1]
        x_final[i]=centroid_iteration_16[number_iteration-1,i,0]
        y_final[i]=centroid_iteration_16[number_iteration-1,i,1]


    plt.scatter(x_initial, y_initial, color='blue', marker='o', label='initial')
    plt.scatter(x_final, y_final, color='red', marker='s', label='final')

    plt.legend(loc='upper right')
    plt.tight_layout()
    plt.show()

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:

```
def function_result_13():

    plt.figure(figsize=(8,8))
    plt.title('cluster (K=2)')

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #

    number_cluster = 2
    xCoord = feature[:,0]
    yCoord = feature[:,1]
    plt.scatter(xCoord, yCoord, c = label_feature_02, cmap= cm.get_cmap('jet', number
    plt.colorbar(ticks = range(number_cluster), label= 'cluster')
    plt.clim(-0.5, number_cluster - 0.5)

    plt.tight_layout()
    plt.show()

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:

```
def function_result_14():

    plt.figure(figsize=(8,8))
    plt.title('cluster (K=4)')

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
```

```
    number_cluster = 4
    xCoord = feature[:,0]
    yCoord = feature[:,1]
    plt.scatter(xCoord, yCoord, c = label_feature_04, cmap= cm.get_cmap('jet', number
    plt.colorbar(ticks = range(number_cluster), label= 'cluster')
    plt.clim(-0.5, number_cluster - 0.5)

    plt.tight_layout()
    plt.show()

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:

```
def function_result_15():

    plt.figure(figsize=(8,8))
    plt.title('cluster (K=8)')

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    number_cluster = 8
    xCoord = feature[:,0]
    yCoord = feature[:,1]
    plt.scatter(xCoord, yCoord, c = label_feature_08, cmap= cm.get_cmap('jet', number
    plt.colorbar(ticks = range(number_cluster), label= 'cluster')
    plt.clim(-0.5, number_cluster - 0.5)

    plt.tight_layout()
    plt.show()

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:

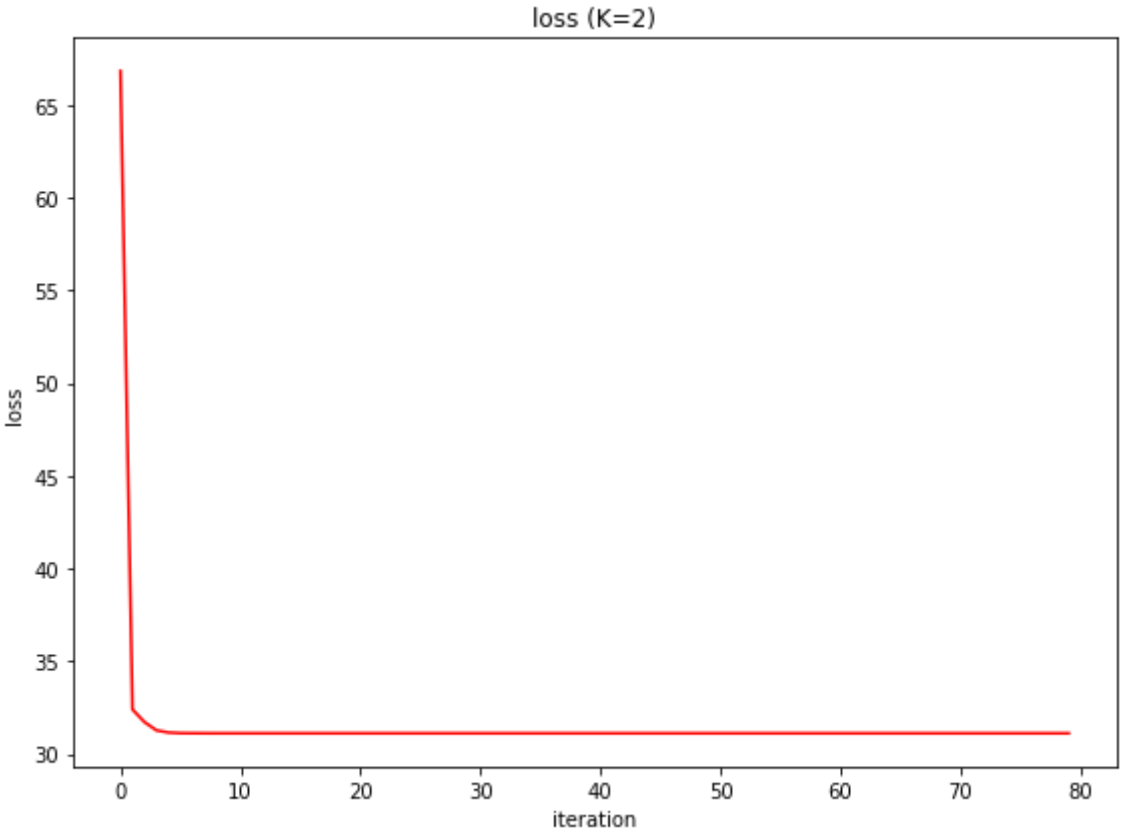
```
def function_result_16():

    plt.figure(figsize=(8,8))
    plt.title('cluster (K=16)')

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    number_cluster = 16
    xCoord = feature[:,0]
    yCoord = feature[:,1]
    plt.scatter(xCoord, yCoord, c = label_feature_16, cmap= cm.get_cmap('jet', number
    plt.colorbar(ticks = range(number_cluster), label= 'cluster')
    plt.clim(-0.5, number_cluster - 0.5)

    plt.tight_layout()
    plt.show()

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
```

---

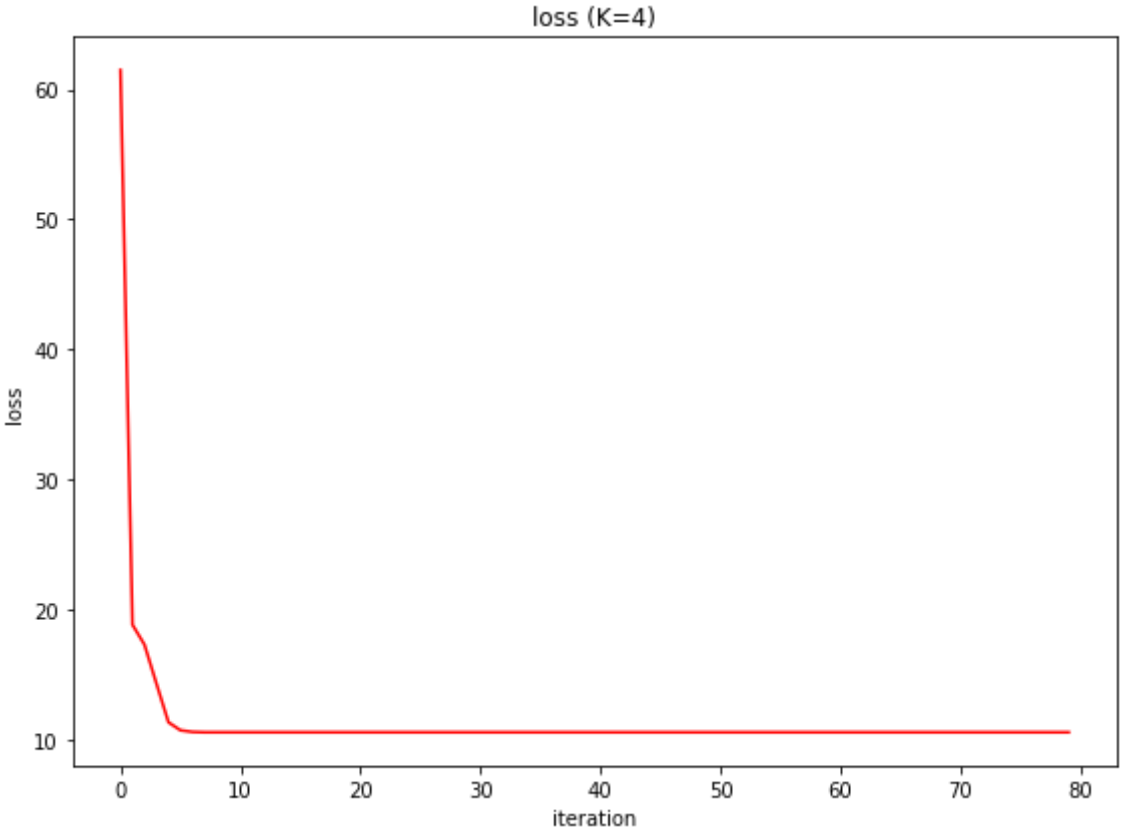
results

In [ ]:
```
number_result = 16

for i in range(number_result):
    title = '## [RESULT {:02d}]'.format(i+1)
    name_function = 'function_result_{:02d}()'.format(i+1)

    print('************************************************')
    print(title)
    print('************************************************')
    eval(name_function)
```

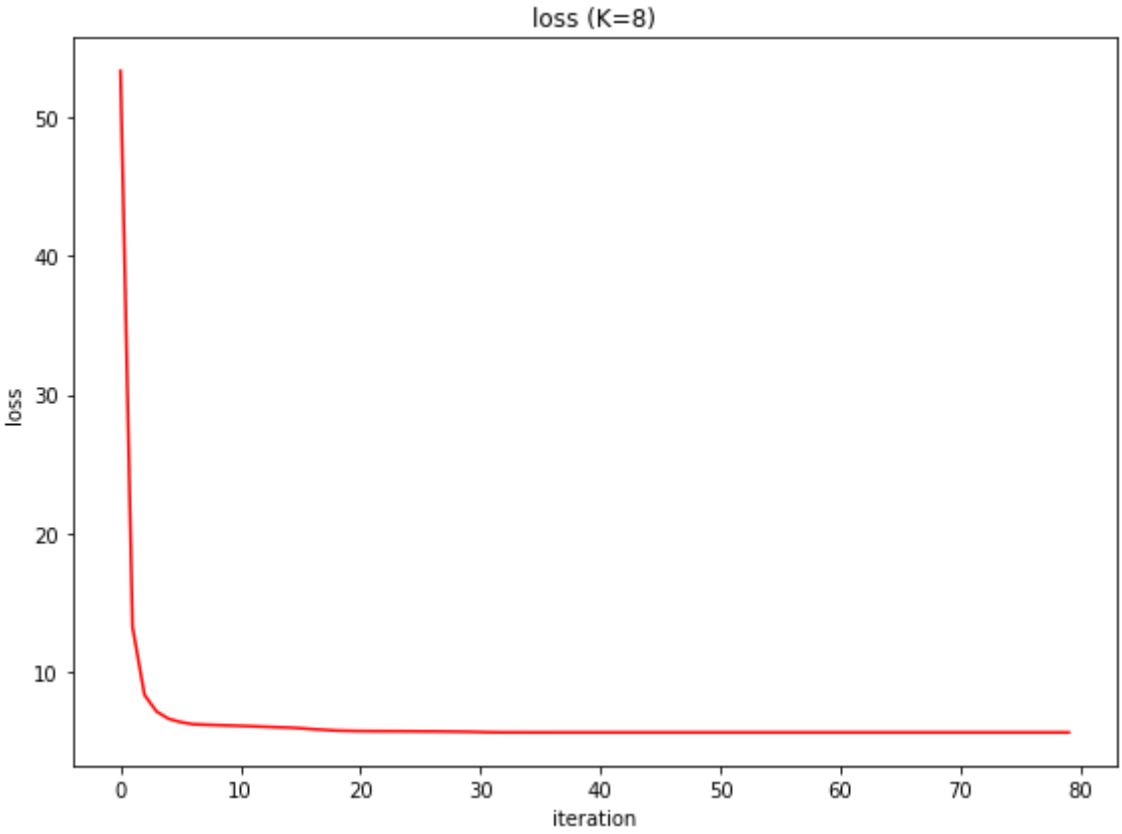
```
************************************************
## [RESULT 01]
************************************************
final loss (K=2) = 31.1123356206
************************************************
## [RESULT 02]
************************************************
final loss (K=4) = 10.5831291650
************************************************
## [RESULT 03]
************************************************
final loss (K=8) =  5.6797490414
************************************************
## [RESULT 04]
************************************************
final loss (K=16) =  3.0397969150
************************************************
## [RESULT 05]
************************************************
```



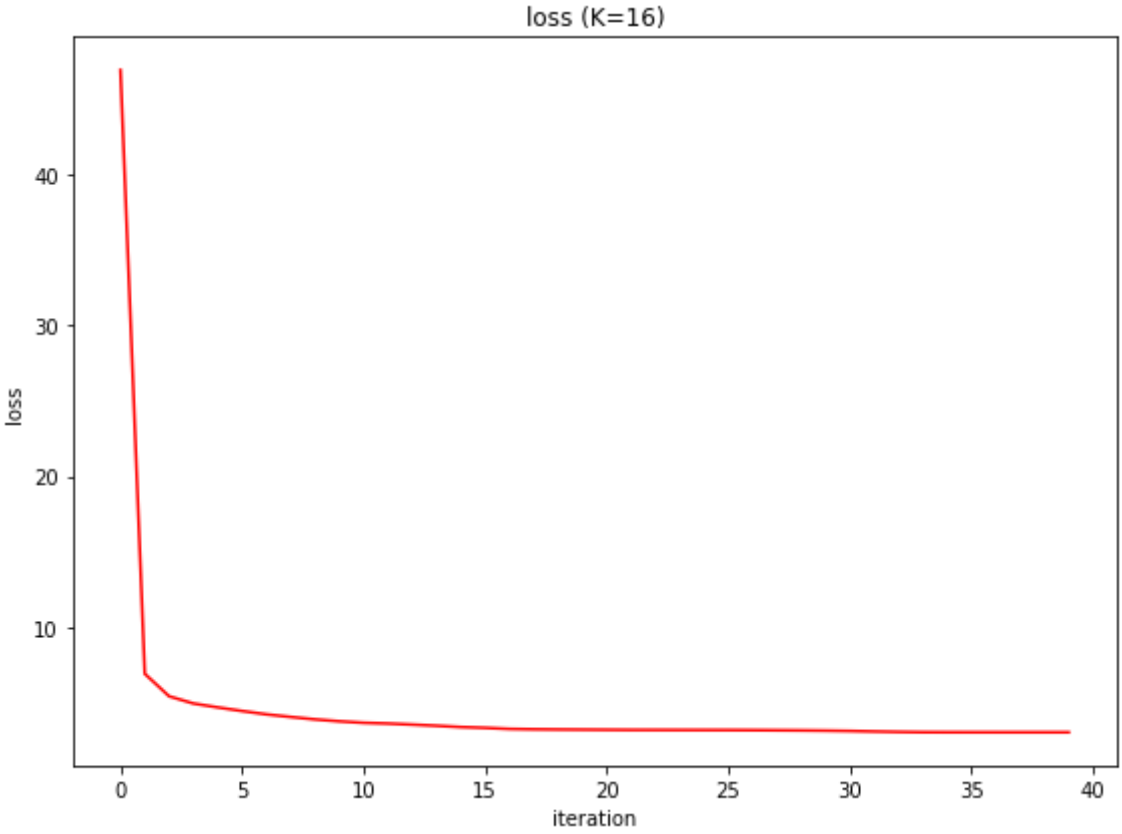

```
************************************************
## [RESULT 06]
************************************************
```



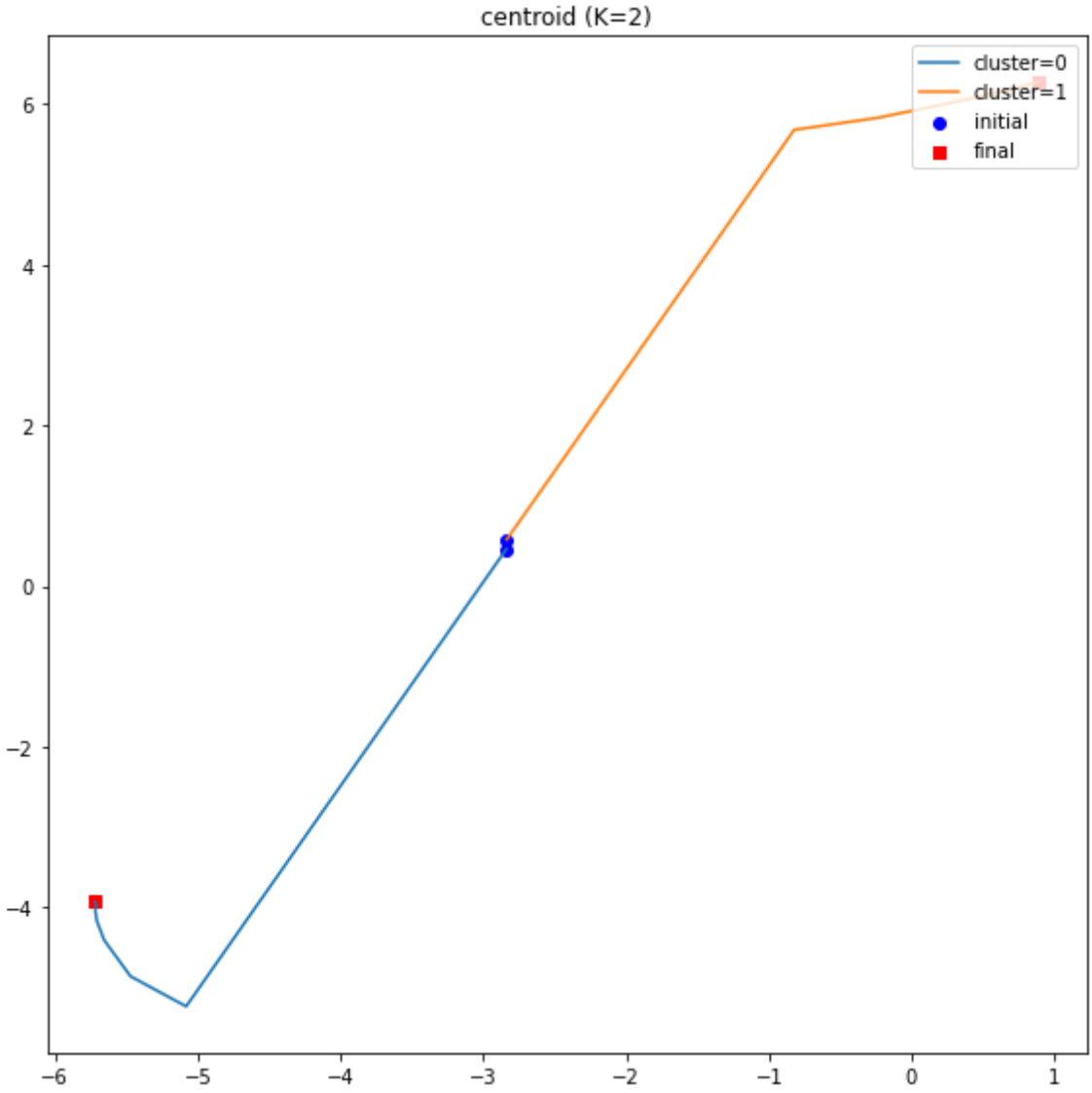

```
**************************************************
## [RESULT 07]
**************************************************
```



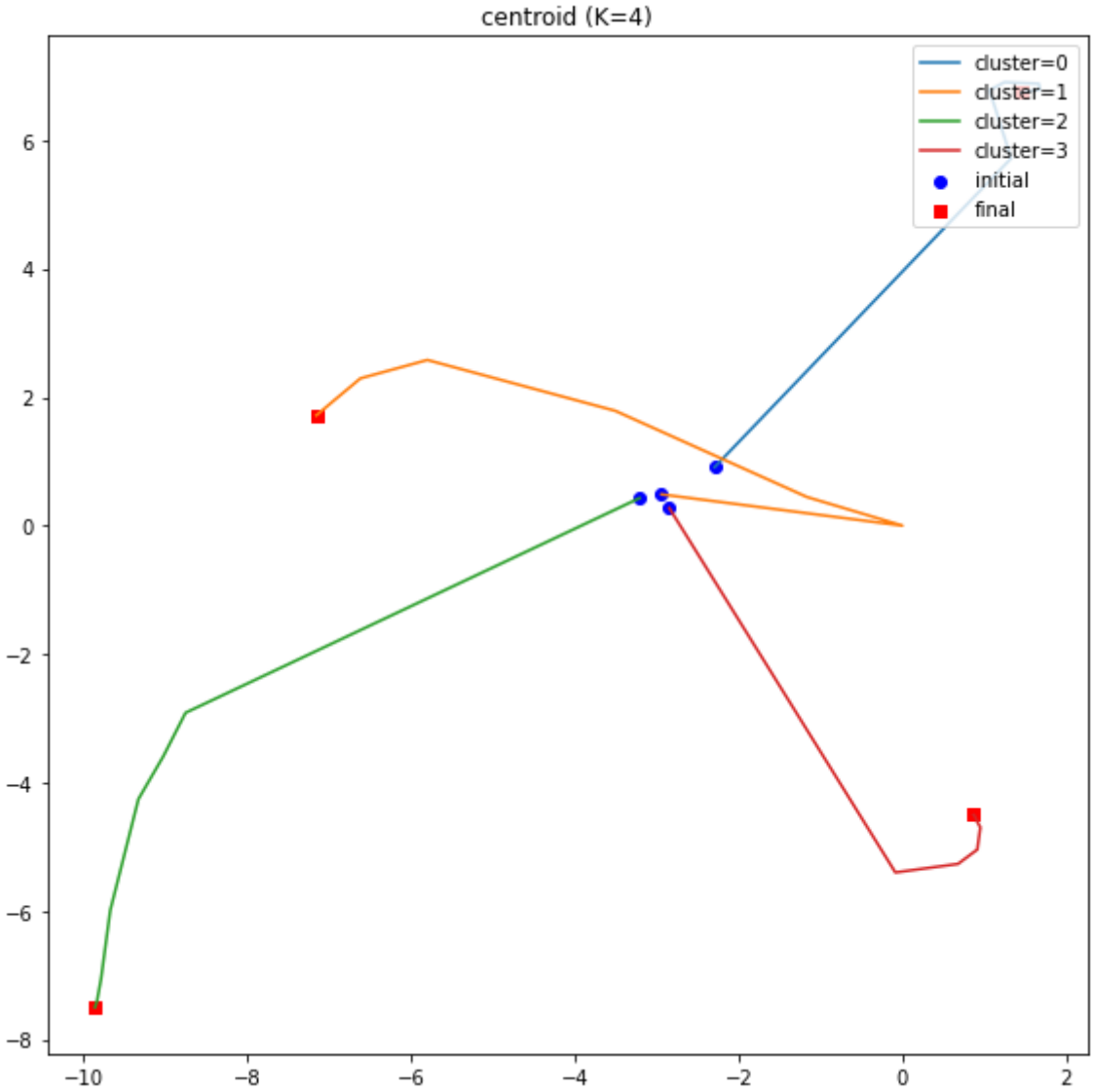

```
**************************************************
## [RESULT 08]
**************************************************
```



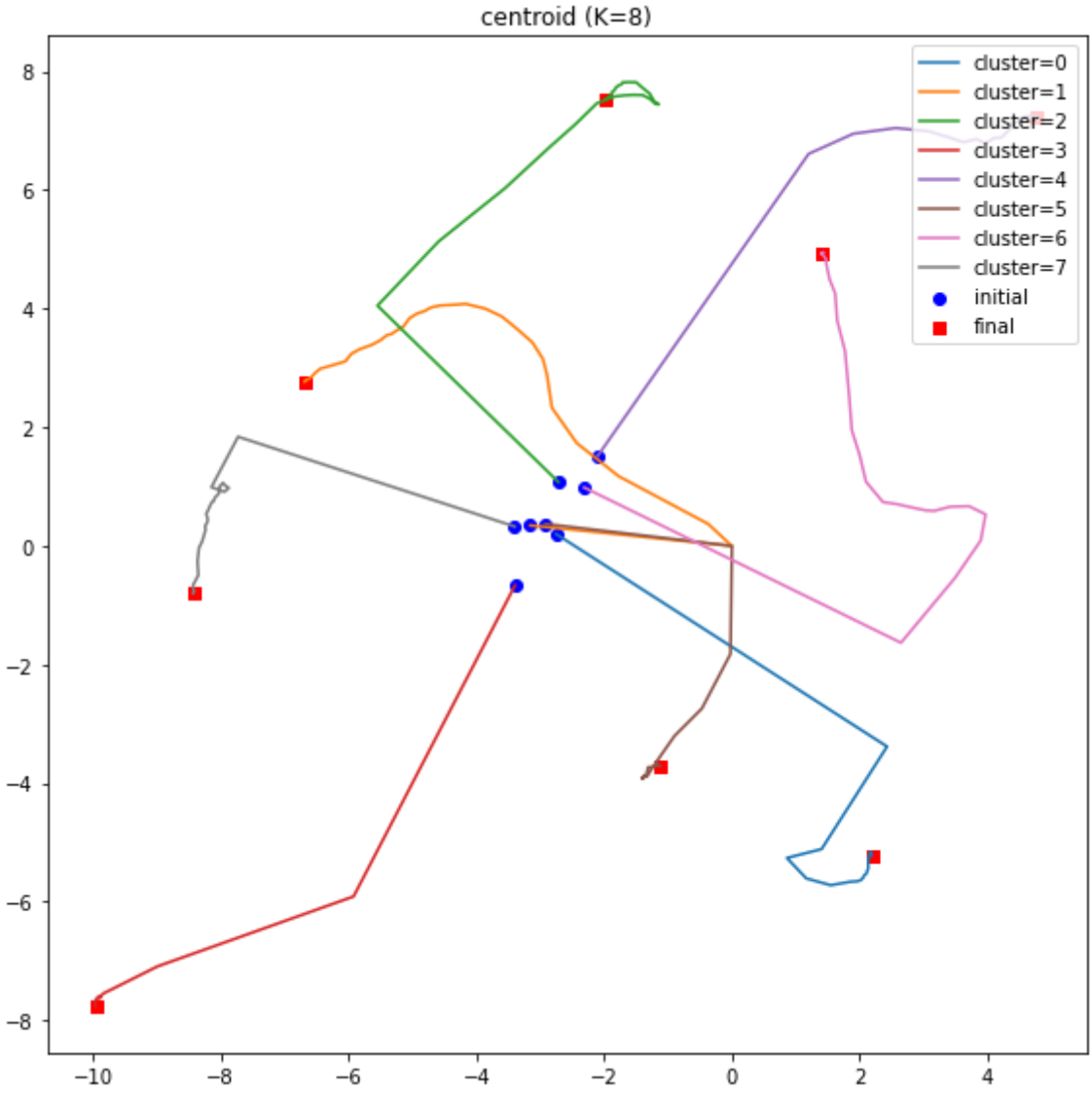

```
**************************************************
## [RESULT 09]
**************************************************
```


centroid (K=2)
cluster=0
cluster=1
initial
final
6
4
2
0
−2
−4
−6
−5
−4
−3
−2
−1
0
1

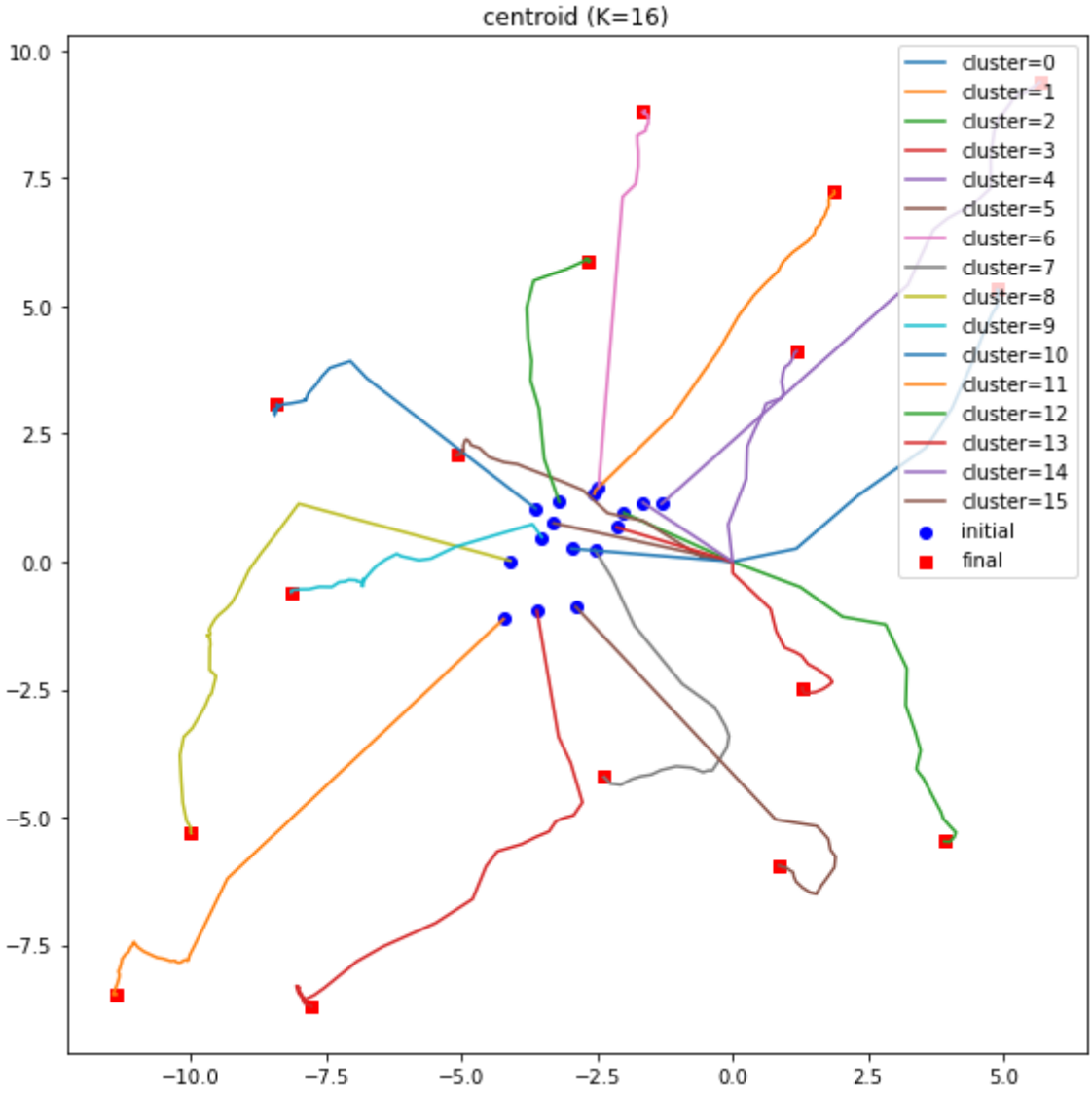
```
**************************************************
## [RESULT 10]
**************************************************
```


centroid (K=4)
cluster=0
cluster=1
cluster=2
cluster=3
initial
final
6
4
2
0
−2
−4
−6
−8
−10
−8
−6
−4
−2
0
2

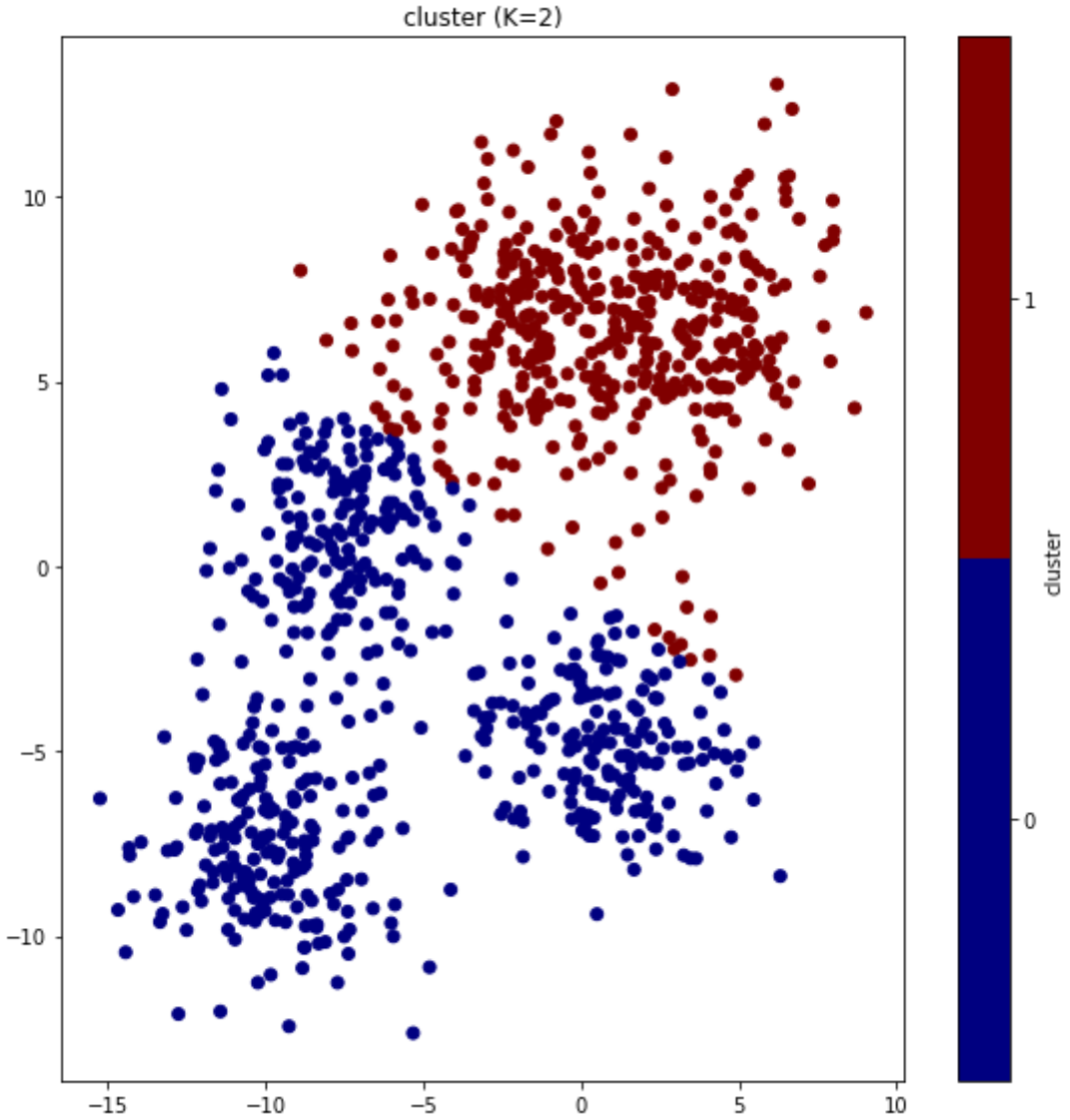
```
**************************************************
## [RESULT 11]
**************************************************
```


centroid (K=8)
cluster=0
cluster=1
cluster=2
cluster=3
cluster=4
cluster=5
cluster=6
cluster=7
initial
final
8
6
4
2
0
−2
−4
−6
−8
−10
−8
−6
−4
−2
0
2
4

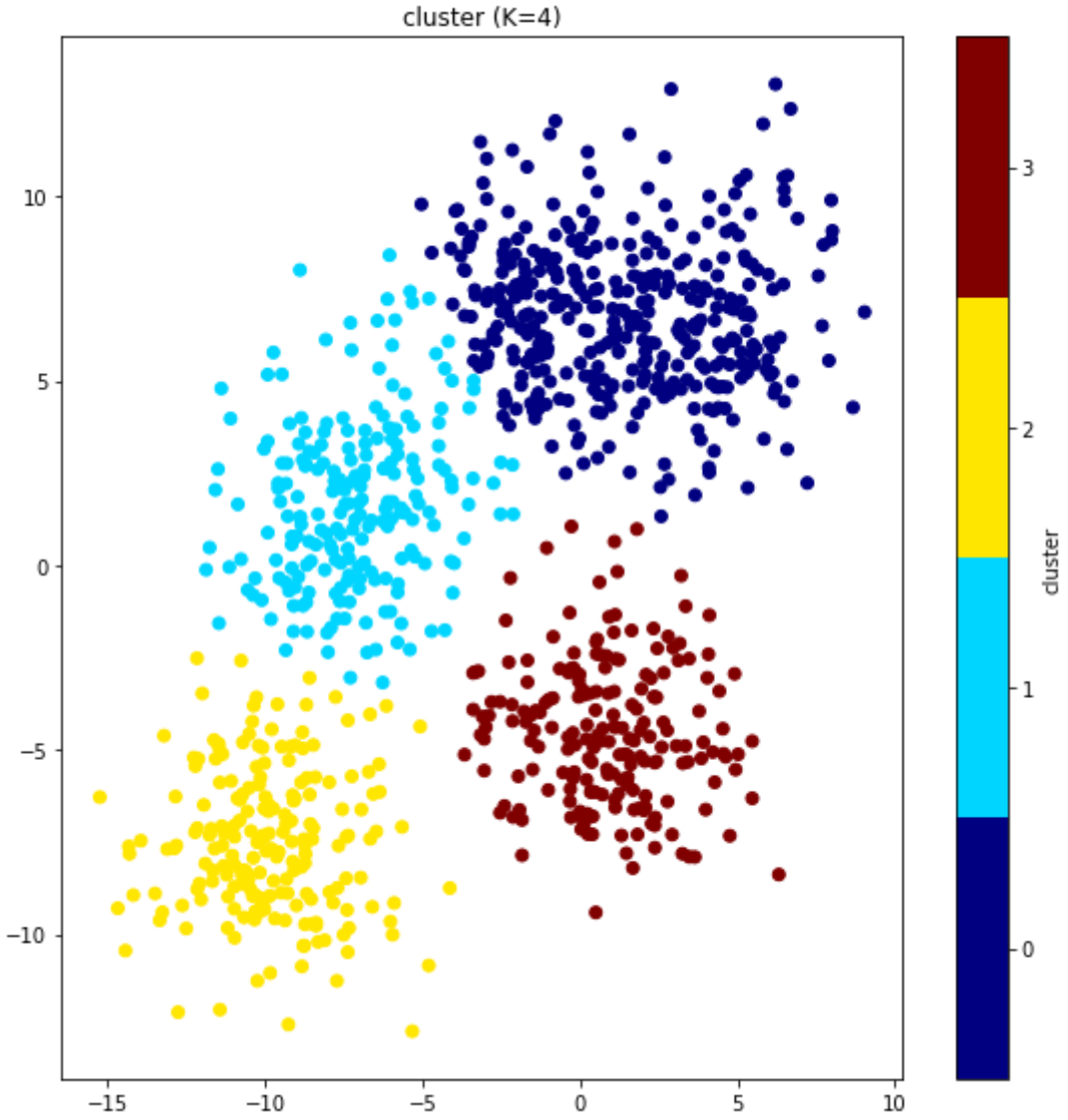
```
**************************************************
## [RESULT 12]
**************************************************
```


centroid (K=16)
cluster=0
cluster=1
cluster=2
cluster=3
cluster=4
cluster=5
cluster=6
cluster=7
cluster=8
cluster=9
cluster=10
cluster=11
cluster=12
cluster=13
cluster=14
cluster=15
initial
final
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0

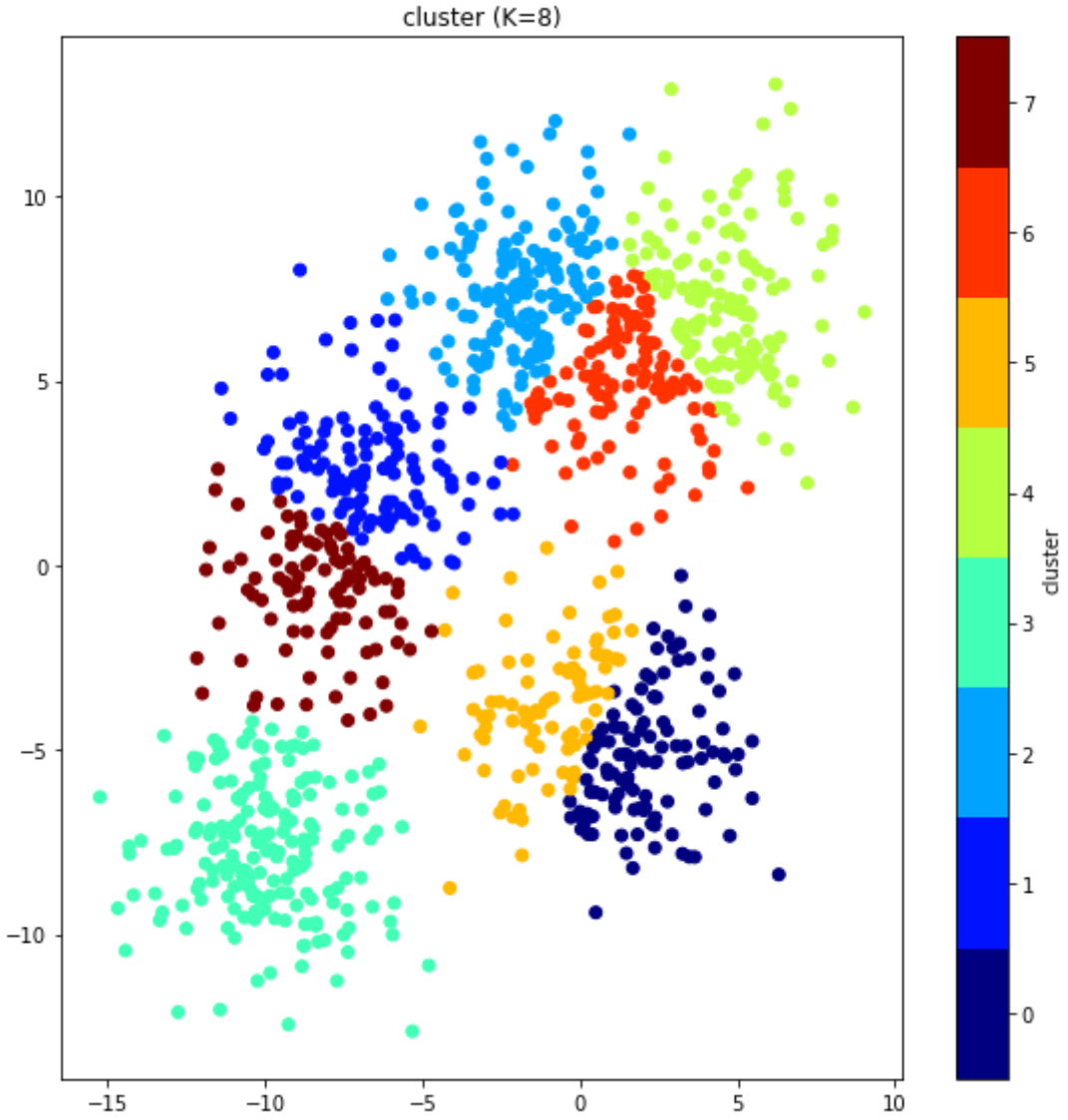
```
**************************************************
## [RESULT 13]
**************************************************
```



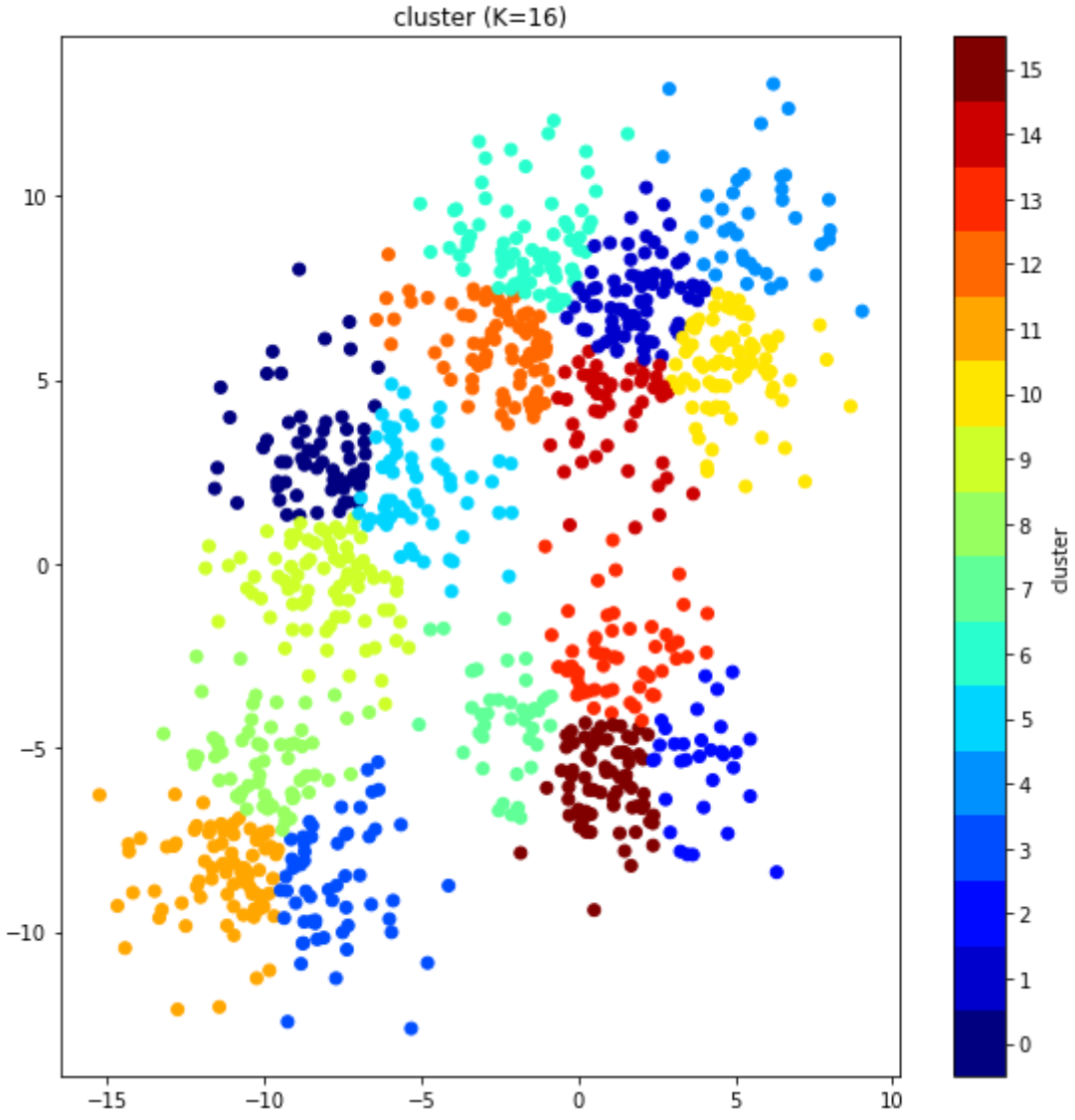

```
************************************************
## [RESULT 14]
************************************************
```




```
**************************************************
## [RESULT 15]
**************************************************
```




```
**************************************************
## [RESULT 16]
**************************************************
```


cluster (K=16)
10
5
0
−5
−10
−15
−10
−5
0
5
10
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
cluster