K-means算法

**一、算法流程：**

输入：数据集data，聚类数k   
输出：聚类集合c

1. 创建k个点作为起始质心
2. 当任意一个点的簇分配结果发生改变时
   * 对数据集中的每个数据点，对每个质心，计算质心与数据点之间的距离
   * 将数据点分配到离它距离最近的簇
   * 对每一个簇，计算簇中所有的点的均值并将均值作为新的聚类中心

**二、算法实现：**

首先绘制二维数据的散点图，可以观察到，数据可以聚成3类，绘图代码以及结果如下：

import matplotlib

import matplotlib.pyplot as plt

data = [[0,0], [3,8], [2,2], [1,1], [5,3], [4,8], [6,3], [5,4], [6,4], [7,5]]

x = []; y = []

for d in data:

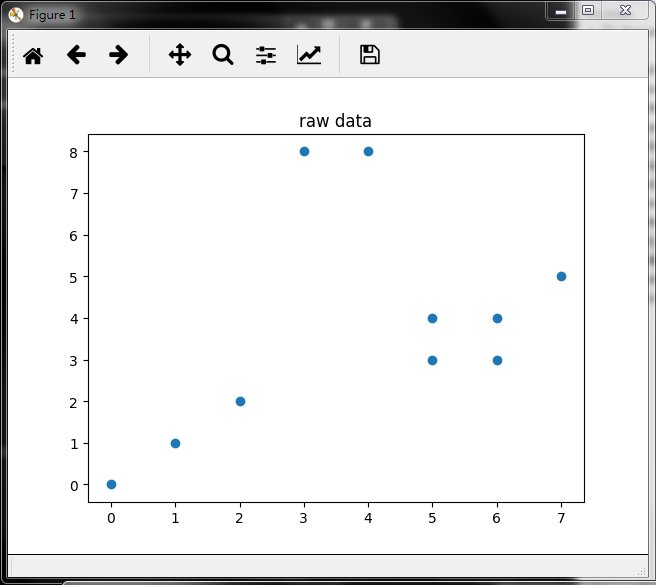
x.append(d[0])

y.append(d[1])

plt.figure(1)

plt.scatter(x,y)

plt.title('raw data')



k均值聚类函数如下，其中data为原始数据，k为聚类的类别数量，max\_iteration为最大迭代次数，函数返回聚类结果以及各个点到其对应的簇的中心的距离总和变化表。

def k\_means(data, k, max\_iteration):

"""select k centers"""

centers = []

for i in range(0,k):

centers.append(data[i])

cost\_trend = []

cluster\_result = []

iternumber = 0

while True:

iternumber += 1

"""cluster"""

clusters = []

for i in range(0,k):

clusters.append([])

for i in range(0,len(data)):

center\_index = find\_center(data[i], centers)

clusters[center\_index].append(i)

"""calculate the cost"""

total\_cost = 0

for i in range(0,k):

total\_cost += cal\_cost(centers[i], clusters[i], data)

cost\_trend.append(total\_cost)

print(total\_cost)

"""判断一下是否有必要继续"""

if iternumber == max\_iteration:

cluster\_result = clusters

break

"""update the center"""

for i in range(0,k):

center = cal\_center(data, clusters[i])

centers[i] = center

return cost\_trend, cluster\_result

设置k = 3, iteration = 20,得到如下聚类结果，绘制出coss\_trend可以发现，算法俺已经收敛。

