import matplotlib as plt

from numpy import \*

import numpy as np

def loadDataSet(fileName):#读入文件 格式是二维数组 每一行为一个散点的数据：位置x 位置y 所属类

data = []

fr = open(fileName)

for line in fr.readlines():

curLine = line.strip().split(',')

fltLine = map(float, curLine)

data.append(fltLine)

return data

def changeData(data):

PointNumber = shape(data)[0]

na = shape(data)[1]

temp = np.transpose(data)

da = mat(zeros((na,PointNumber)))

da = temp[:(na-1)]

da = np.transpose(da)

return da

def distEclud(a, b):#计算欧氏距离

return (sum(power(a - b, 2)))

def normalize(v):

norm=np.linalg.norm(v)

if norm==0:

return v

return v/norm

def Laplacian(sigma,data):#计算拉普拉斯矩阵

PointNumber = shape(data)[0]

WMatrix =mat(zeros((PointNumber,PointNumber)))

D =mat(zeros((PointNumber,PointNumber)))

L =mat(zeros((PointNumber,PointNumber)))

for i in range(0,PointNumber):#计算关系矩阵

for j in range(0,PointNumber):

if(i==j):

WMatrix[i,j] = 0

continue

else:

WMatrix[i,j] = exp(- distEclud(data[j, :], data[i, :]) / (2 \* sigma\*sigma))#get related matrix

continue

for i in range(PointNumber):#计算度矩阵

for j in range(PointNumber):

D[i,i] += WMatrix[i,j]

for i in range(PointNumber):#计算拉普拉斯矩阵

for j in range(PointNumber):

L[i,j] = D[i,j] - WMatrix[i,j]

return L

def TOPK(sigma,k,data):

LA = Laplacian(sigma,data)

a,b=np.linalg.eig(LA)

evals=a

evecs=b

indices = np.argsort(evals)[:k]

return normalize(evecs[:, indices])

# return evecs[:, indices]

def InitNewPoint(sigma,k,data):

PointNumber = shape(data)[0]

newpoint =mat(zeros((PointNumber,k)))

top\_evecs = mat(zeros((PointNumber,k)))

top\_evecs = TOPK(sigma,k,data)

for i in range(PointNumber):

for j in range(k):

newpoint[i,j] = top\_evecs[i,j]

return newpoint

def RandK(data, k):#随机给出k个中心点

n = shape(data)[1]#k个点的维数与数据一致

randk = mat(zeros((k,n)))#randk矩阵每一行为一个随机点

for j in range(n):

Min = min(data[:,j])

Range = float(max(array(data)[:,j]) - Min)

randk[:,j] = Min +(Range)\* random.rand(k,1)

print randk#一列一列地给出随机数 范围是在数据最大值和最小值之间 以免中心点偏离数据点

return randk

def kMeans(data, k):#k-means

PointNumber = shape(data)[0]

randk = RandK(data, k)

tap = mat(zeros((PointNumber,2)))#一个标签 记录上一次改变之后所有点所属的点和最小距离的平方

Changed = True#已经更新过中心点的位置

while Changed:#如果已经更新过 循环更新 标志改变

Changed = False

for i in range(PointNumber):

minDist = 1000000

minIndex = -1

for j in range(k):

distJI = distEclud(randk[j,:],data[i,:])#对于每一个数据点 计算它到中心点的距离 选取最近的一个 则该数据点所属一类

if distJI < minDist:

minDist = distJI;

minIndex = j

if tap[i,0] != minIndex: #如果此时算出来的选类和上一次的不同 则标记改变 需要更新中心点位置

Changed = True

tap[i,:] = minIndex,minDist\*\*2

for m in range(k):

sameclassofm = data[nonzero(tap[:,0].A==m)[0]]#求出所有和中心点同类的数据点

randk[m,:] = mean(sameclassofm, axis=0) #将他们求平均

#print randk

return randk, tap

def randRGB():

return (random.randint(0, 255)/255.0,

random.randint(0, 255)/255.0,

random.randint(0, 255)/255.0)

def show(data, k, randk, tap,colors):

from matplotlib import pyplot as plt

numSamples, dim = data.shape

for i in xrange(numSamples):

group = int(tap[i, 0])

print group

plt.plot(data[i, 0], data[i, 1],'o',color=colors[group])

#mark = ['Dr', 'Db', 'Dg', 'Dk', '^b', '+b', 'sb', 'db', '<b', 'pb','b','c','g','k','m','r','w','y']

#for i in range(k):

# plt.plot(randk[i, 0], randk[i, 1], mark[i], markersize = 12)

plt.show()

dataMat = mat(loadDataSet('C:/Users/BPEI/Desktop/shiyan/flame.txt'))

colors = []

data = changeData(dataMat)

kk = random.randint(1,10)#k值取随机10个点

kk=2

for i in range(kk):

colors.append(randRGB())

newpoint = InitNewPoint(1,10,data)

RANDK, TAP= kMeans(newpoint,kk)

show(dataMat, kk, RANDK, TAP,colors)