**Lab 7 k-Means应用实践**

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 姓名 | 年级 | 专业 |
| 18353070 | 谭嘉伟 | 2018 | 软件工程 |

1. 实验目的：
2. 熟悉kMeans算法的程序编写与应用
3. 熟悉DBSCAN算法的程序编写与应用
4. 熟悉轮廓系数的定义与计算函数的编写
5. 进一步熟悉tensorflow，尝试用tensorflow实现kMean算法
6. 了解百度地图的API接口的使用
7. 实验环境：

1、硬件环境：PC 1.60GHz四核处理器 8.00G内存

2、软件环境：Windows10 64位操作系统

Python3.5.6 tensorflow1.4.0

1. 实验步骤：
2. 获取地图数据
3. 注册成为开发者

在浏览器进入lbsyun.baidu.com，使用百度账号登录网页，点击导航栏的“控制台”，进行姓名、手机、邮箱验证后即可注册成为开发者。在点击“获取秘钥”后，完善“创建应用”信息，将电脑ip地址填入白名单列表，点击“提交”即可获得秘钥。

1. 使用API获取数据

按照API格式正确书写网址，填入刚刚所获得的秘钥，调整输出文件的路径，运行api.py的代码，可以在txt文件中得到地点数据。

1. kMean算法实现
2. 建立辅助函数

导入kmean.py的代码，用import numpy as np替换from numpy import \*，进行函数名称的少许修改，进行kmean算法所用函数的测试。

修改后代码：

#kmean

import numpy as np

def loadDataSet(fileName):

dataMat = []

fr = open(fileName)

for line in fr.readlines():

curLine = line.strip().split('\t')

#将所有数据转换为float类型

fltLine = list(map(float,curLine))

dataMat.append(fltLine)

return dataMat

def distEclud(vecA, vecB):

return np.sqrt(np.sum(np.power(vecA - vecB, 2)))

def randCent(dataSet, k):

#得到数据集的列数

n = np.shape(dataSet)[1]

#得到一个K\*N的空矩阵

centroids = np.mat(np.zeros((k,n)))

#对于每一列

for j in range(n):

#得到最小值

minJ = min(dataSet[:,j])

#得到当前列的范围

rangeJ = float(max(dataSet[:,j]) - minJ)

#在最小值和最大值之间取值

centroids[:,j] = np.mat(minJ + rangeJ \* np.random.rand(k,1))

return centroids

def distSLC(vecA, vecB):

#pi为圆周率，在导入numpy时就会导入的了

#sin(),cos()函数输出的是弧度为单位的数据

#由于输入的经纬度是以角度为单位的，故要将其除以180再乘以pi转换为弧度

#设所求点A ，纬度β1 ，经度α1 ；点B ，纬度β2 ，经度α2。则距离

#距离 S=R·arc cos[cosβ1cosβ2cos（α1-α2）+sinβ1sinβ2]

a = np.sin(vecA[0,1]\*np.pi/180) \* np.sin(vecB[0,1]\*np.pi/180)

b = np.cos(vecA[0,1]\*np.pi/180) \* np.cos(vecB[0,1]\*np.pi/180) \* \

np.cos(np.pi \* (vecB[0,0]-vecA[0,0]) /180)

return np.arccos(a + b)\*6371.0 #6371.0为地球半径

def kMeans(dataSet, k, distMeas=distEclud, createCent=randCent):

# 数据集的行数，即数据的个数

m = np.shape(dataSet)[0]

# 簇分配结果矩阵

clusterAssment = np.mat(np.zeros((m,2)))

# 第一列储存簇索引值

# 第二列储存数据与对应质心的误差

# 先随机生成k个随机质心的集合

centroids = createCent(dataSet, k)

clusterChanged = True

# 当任意一个点的簇分配结果改变时

while clusterChanged:

clusterChanged = False

# 对数据集中的每一个数据

for i in range(m):

minDist = np.inf; minIndex = -1

# 对于每一质心

for j in range(k):

# 得到数据与质心间的距离

distJI = distMeas(centroids[j,:],dataSet[i,:])

#print(type(dijtJI))

# 更新最小值

if distJI < minDist:

minDist = distJI; minIndex = j

# 若该点的簇分配结果改变

if clusterAssment[i,0] != minIndex: clusterChanged = True

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

# print centroids

# 对于每一个簇

for cent in range(k):

# 通过数组过滤得到簇中所有数据

ptsInClust = dataSet[np.nonzero(clusterAssment[:,0].A==cent)[0]]

# .A 方法将matrix类型元素转化为array类型

# 将质心更新为簇中所有数据的均值

centroids[cent,:] = np.mean(ptsInClust, axis=0)

# axis=0表示沿矩阵的列方向计算均值

return centroids, clusterAssment

import matplotlib

import matplotlib.pyplot as plt

def clusterPlaces(inputfile,numClust=5):

datList = []

for line in open(inputfile).readlines():

lineArr = line.split('\t')

datList.append([float(lineArr[0]),float(lineArr[1])])

datMat = np.mat(datList)

# 进行聚类

myCentroids, clustAssing = kMeans(datMat, numClust, distMeas=distSLC)

fig = plt.figure()

# 创建一个矩形

rect = [0.1,0.1,0.8,0.8]

# 用来标识簇的标记

scatterMarkers = ['s' , 'o', '^', '8', 'p', \

'd', 'v', 'h', '>', '<']

axprops = dict(xticks=[], yticks=[])

ax0 = fig.add\_axes(rect, label='ax0', \*\*axprops)

ax1 = fig.add\_axes(rect, label='ax1', frameon=False)

for i in range(numClust):

ptsInCurrCluster = datMat[np.nonzero(clustAssing[:,0].A==i)[0],:]

markerStyle = scatterMarkers[i % len(scatterMarkers)]

ax1.scatter(ptsInCurrCluster[:,0].flatten().A[0], ptsInCurrCluster[:,1].flatten().A[0], marker=markerStyle, s=90)

ax1.scatter(myCentroids[:,0].flatten().A[0], myCentroids[:,1].flatten().A[0], marker='+', s=300)

plt.show()

测试代码：

dataset=np.mat(loadDataSet('D:\\testSet.txt'))

print(min(dataset[:,0]))

print(min(dataset[:,1]))

print(max(dataset[:,0]))

print(max(dataset[:,1]))

print(randCent(dataset,2))

distEclud(dataset[0],dataset[1])

1. kMean算法实现：

实现如下代码测试kMean算法

myCentroids,clustAssing=kMeans(dataset,4)

print(myCentroids)

print(clustAssing)

1. 用kMeans算法对地图上的点聚类：
2. 准备数据

导入数据：dataMat=loadDataSet('D:\\Restaurant\_Data\_Beijing.txt')

dataMat

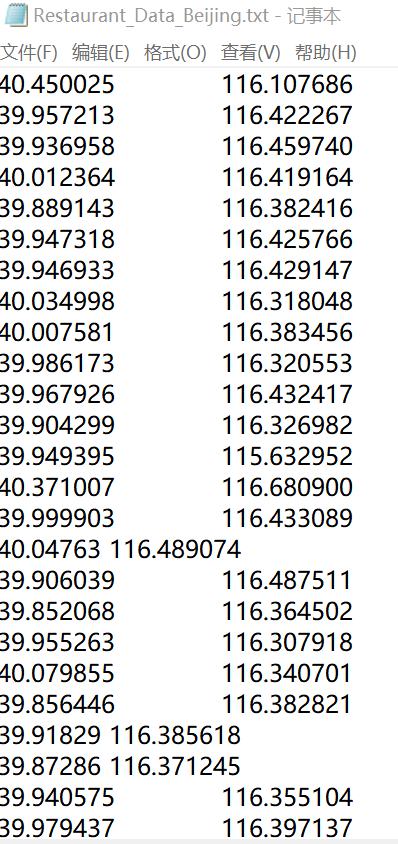
1. 对地理坐标进行聚类：

distSLC()返回的是地球两个经纬度的距离(km)，使用这个距离函数而不是欧几里得距离进行距离计算

使用clusterPlaces('D:\\Restaurant\_Data\_Beijing.txt',k)(k=3,4,5,6)语句观察聚类并可视化的结果

1. 实验结果与分析：
2. 获取地图数据

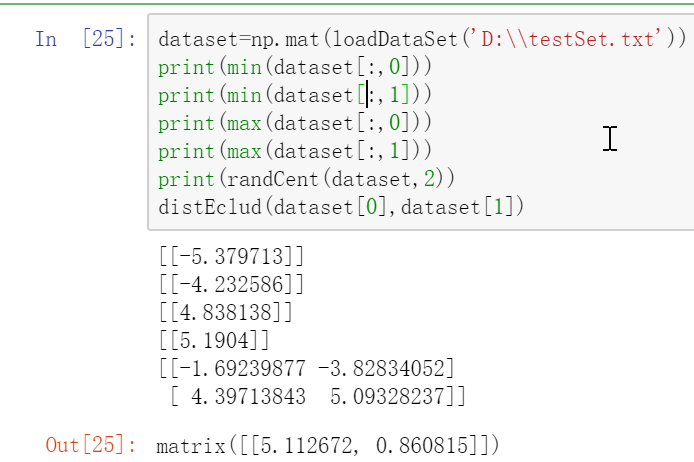
运行api.py后，在txt文件中得到以下数据：



与数据与代码文件夹内的数据相同

1. kMean算法实现：
2. 建立辅助函数：

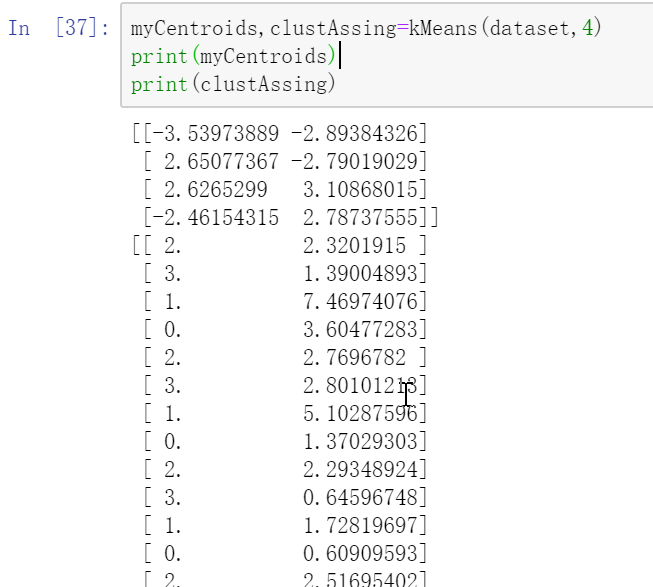
测试结果如下：



通过测试可见可以正确进行数据集的数据函数计算和欧几里得距离计算

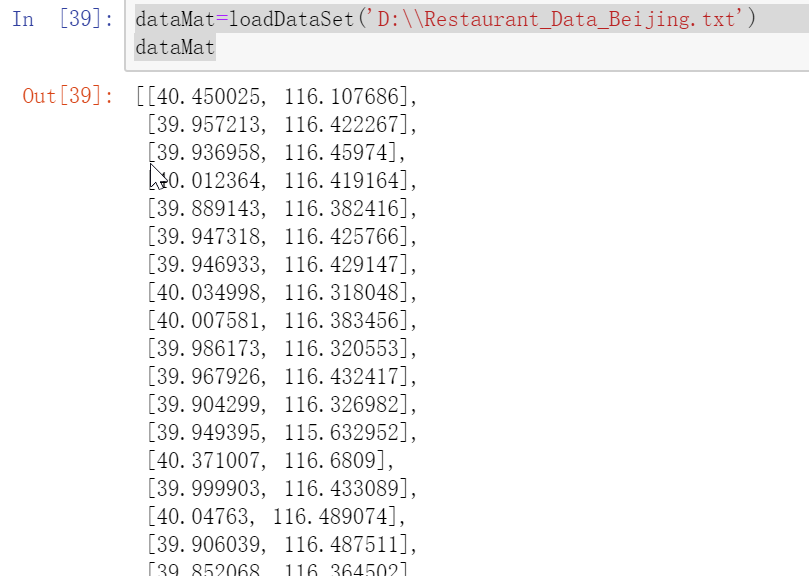
1. kMens算法实现：

可以查看到聚类结果如下：



myCentroids为每个簇中心坐标，clustAssing返回每个元素所属簇和离簇中心的距离的平方

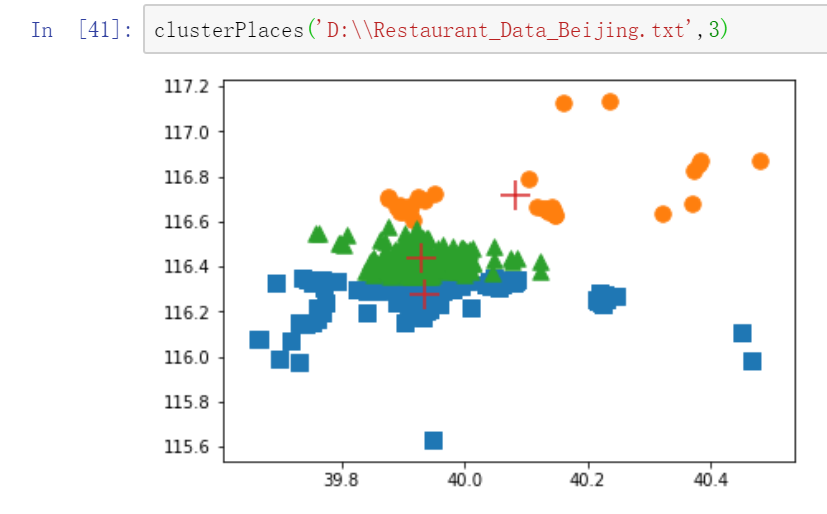
1. 用kMeans算法对地图上的点聚类
2. 准备数据



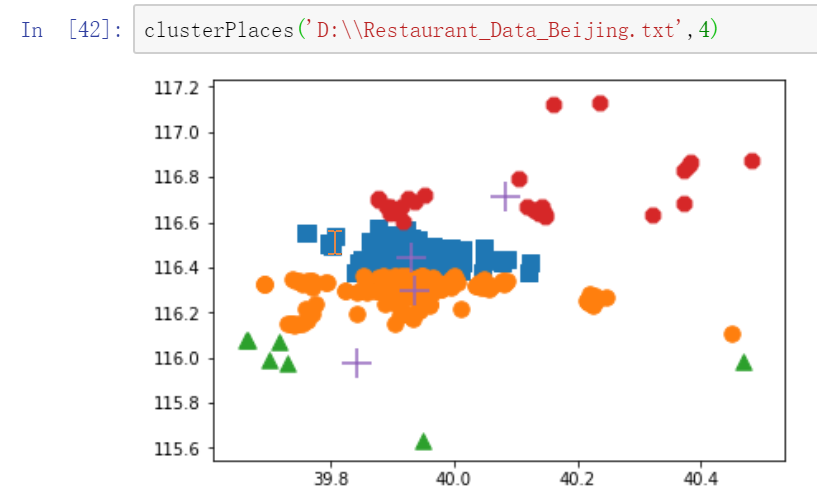
数据被导入到dataMat的二维数组里

1. 对地理坐标进行聚类：

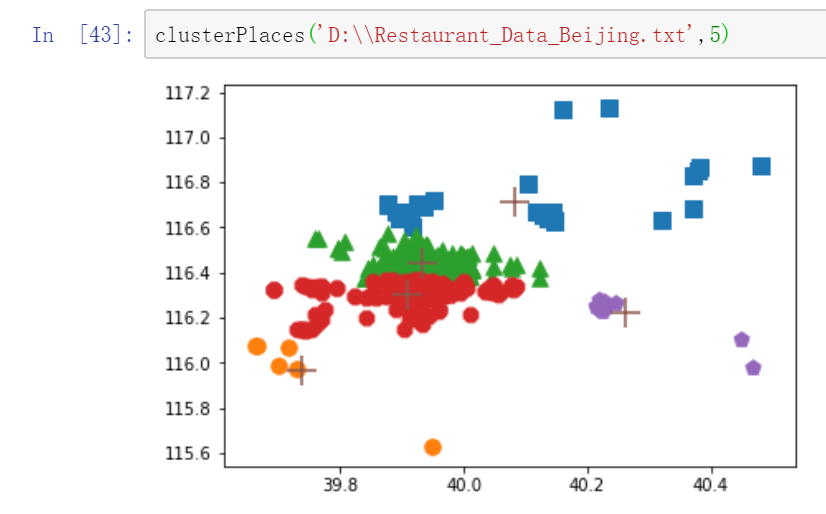
K=3时

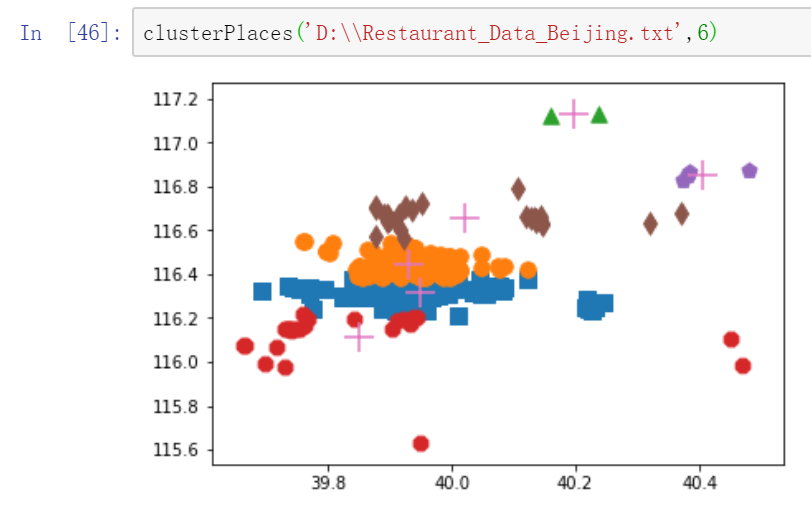


K=4时

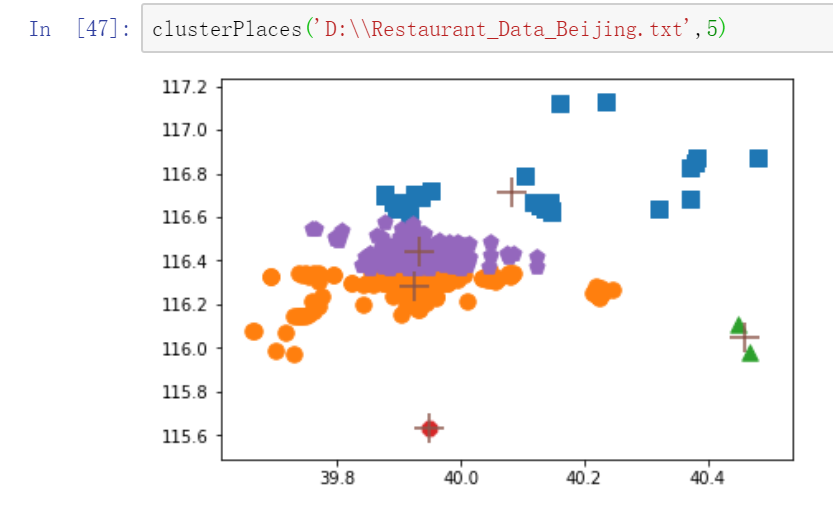


K=5时





由于起始簇中心是随机的，所以多次运行可能得到不完全一样的结果，有时有可能会把一个孤立的点单独作为一个簇，这样的结果就是一个比较差的结果，比如：



kMeans算法的迭代有可能因为初始起点不好而陷入局部极小点，为了解决这样的问题，可以考虑以下几种处理方法：

A 设置一个评估函数，多次运行kMeans迭代法以求得尽量好的结果

B 采用一些贪心算法比较好地选择一些初始簇中心，通过这些簇中心开始迭代来获得比较好的结果

C 引入模拟退火算法/遗传算法的思想：给簇中心在求得平均值后一定的扰动/变异，并且按照概率有可能迭代到一个可能比当前解稍差一点的解，避免迭代被困在局部极小值，让迭代有机会自动跳出搜寻到全局最小值。

D 时间允许的情况下，可以增大枚举量：网格化搜索簇中心，来尽可能多地增加包含最优解的机会。

1. 作业：
2. 尝试用DBSCAN聚类该数据集：

用sklearn的DBSCAN处理testSet.txt的数据

from sklearn.cluster import DBSCAN

pred=DBSCAN(eps=1.5).fit\_predict(dataset)

x=[]

y=[]

for i in range(dataset.shape[0]):

x.append(dataset[i,0])

y.append(dataset[i,1])

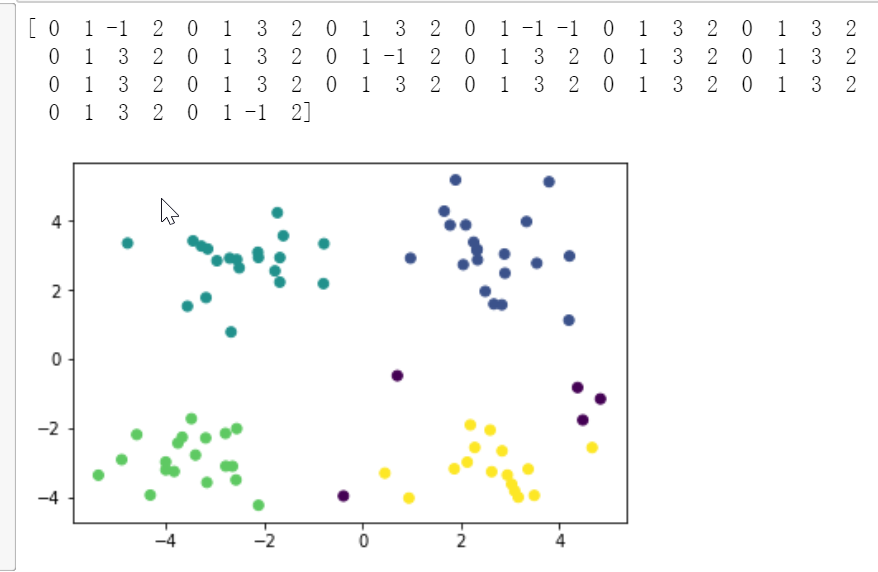
#rint(x)

#rint(y)

print(pred)

plt.scatter(x,y,c=pred)

plt.show()



用sklearn的DBSCAN处理Restaurant\_Data\_Beijing.txt的数据:

def distSLC2(vecA, vecB):

#pi为圆周率，在导入numpy时就会导入的了

#sin(),cos()函数输出的是弧度为单位的数据

#由于输入的经纬度是以角度为单位的，故要将其除以180再乘以pi转换为弧度

#设所求点A ，纬度β1 ，经度α1 ；点B ，纬度β2 ，经度α2。则距离

#距离 S=R·arc cos[cosβ1cosβ2cos（α1-α2）+sinβ1sinβ2]

a = np.sin(vecA[1]\*np.pi/180) \* np.sin(vecB[1]\*np.pi/180)

b = np.cos(vecA[1]\*np.pi/180) \* np.cos(vecB[1]\*np.pi/180) \* \

np.cos(np.pi \* (vecB[0]-vecA[0]) /180)

return np.arccos(a + b)\*6371.0 #6371.0为地球半径

#print(type(dataMat))

#print(dataMat)

#print(dataMat[0])

pred=DBSCAN(eps=4,metric=distSLC2).fit\_predict(dataMat)#

#print(x)

#print(y)

#print(pred)

x=[]

y=[]

for i in range(len(dataMat)):

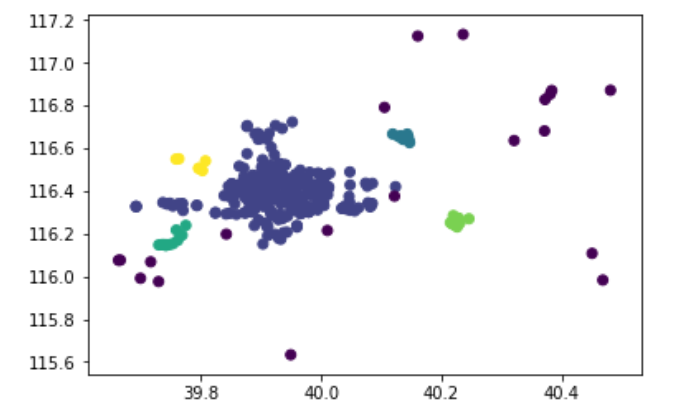
x.append(dataMat[i][0])

y.append(dataMat[i][1])

print(pred)

plt.scatter(x,y,c=pred)

plt.show()



实现的DBSCAN，采用广度优先搜索算法，当前扫描的点是核心点时，将其未归类的邻居分到当前簇并入队进行后续扫描：

def myDBSCAN(data,eps,minpts,caldist):

n=len(data)

pred=np.zeros(n)#0表示噪声点

clo=0

for sc in range(n):

if pred[sc]==0:

nei=[]

neit=0

for i in range(n):

if pred[i]==0 and i!=sc:

dis=caldist(data[i],data[sc])

if dis<eps:

nei.append(i)

neit+=1

if neit>=minpts:

clo+=1

pred[sc]=clo

for i in range(neit):

pred[nei[i]]=clo

q=nei

he=0

ta=neit

while he<ta:

now=q[he]

nei=[]

neit=0

num=0

for i in range(n):

if i==now:

continue

if pred[i]==0 or pred[i]==clo:

dis=caldist(data[now],data[i])

if dis<eps:

num+=1

if pred[i]==0:

nei.append(i)

neit+=1

if num>=minpts:

for i in range(neit):

pred[nei[i]]=clo

q.append(nei[i])

ta+=1

he+=1

return pred

用自己实现的DBSCAN对testSet.txt进行聚类：

print(type(dataset))

pred=myDBSCAN(dataset,eps=1.3,minpts=3,caldist=distEclud)

x=[]

y=[]

for i in range(dataset.shape[0]):

x.append(dataset[i,0])

y.append(dataset[i,1])

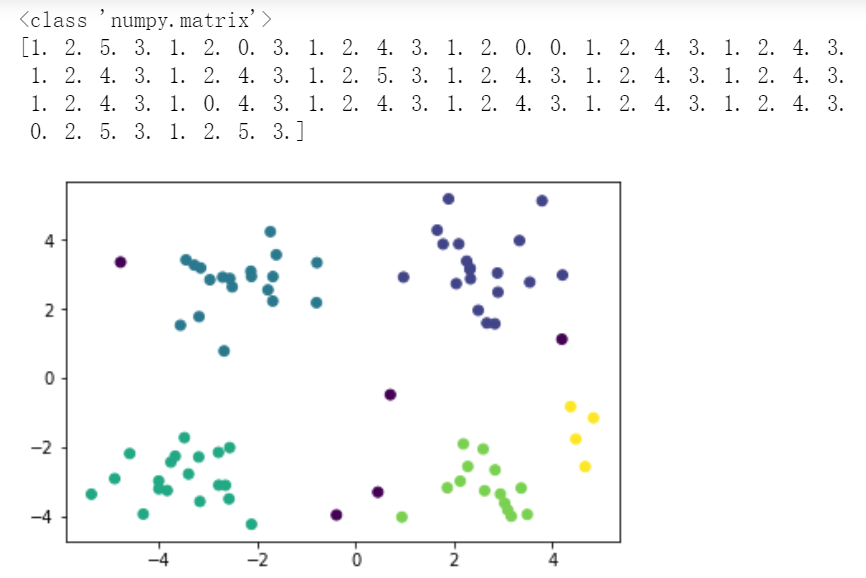
#print(x)

#print(y)

print(pred)

plt.scatter(x,y,c=pred)

plt.show()



用自己实现的DBSCAN对testSet.txt进行聚类：

pred=myDBSCAN(dataMat,eps=3,minpts=3,caldist=distSLC2)

x=[]

y=[]

for i in range(len(dataMat)):

x.append(dataMat[i][0])

y.append(dataMat[i][1])

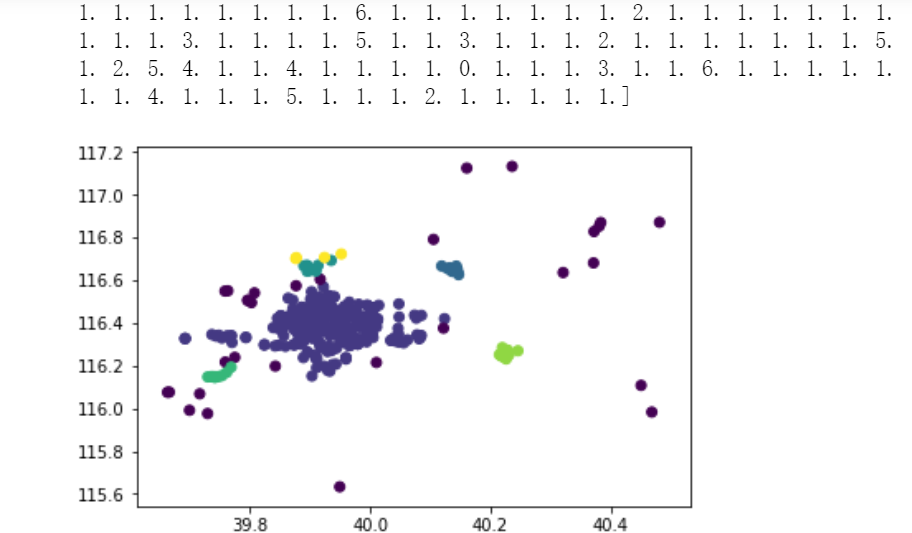
#print(x)

#print(y)

print(pred)

plt.scatter(x,y,c=pred)

plt.show()



1. 利用轮廓系数评估kMeans和DBSCAN对该数据集的聚类效果：

采用以下代码计算轮廓系数，使用python3的字典来统计不同簇的距离和与不同簇元素个数，来得到平均距离

def calSC(data,caldist,pred):

SCsum=0

n=len(data)

for i in range(n):

dis={}

num={}

for j in range(n):

if (j==i):

continue

if not(pred[j] in dis):

dis[pred[j]]=0

num[pred[j]]=0

dis[pred[j]]=dis[pred[j]]+caldist(data[i],data[j])

num[pred[j]]=num[pred[j]]+1

b=np.inf

for x in dis:

if (x==pred[i]):

a=dis[x]/num[x]

continue

te=dis[x]/num[x]

if te<b:

b=te

SCsum+=(b-a)/max(a,b)

return SCsum/n

对sklearn处理testSet.txt聚类计算轮廓系数：

pred=DBSCAN(eps=1.5).fit\_predict(dataset)

x=[]

y=[]

for i in range(dataset.shape[0]):

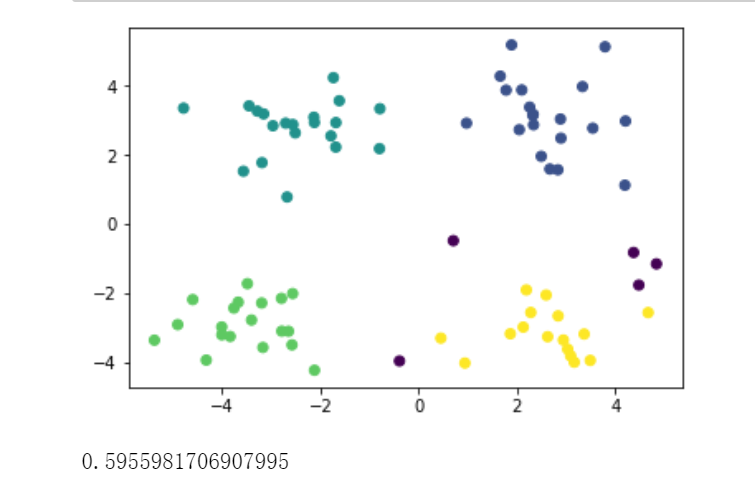
x.append(dataset[i,0])

y.append(dataset[i,1])

plt.scatter(x,y,c=pred)

plt.show()

print(calSC(dataset,distEclud,pred))



对自己的DBSCAN处理testSet.txt聚类计算轮廓系数：

pred=myDBSCAN(dataset,eps=1.3,minpts=3,caldist=distEclud)

x=[]

y=[]

for i in range(dataset.shape[0]):

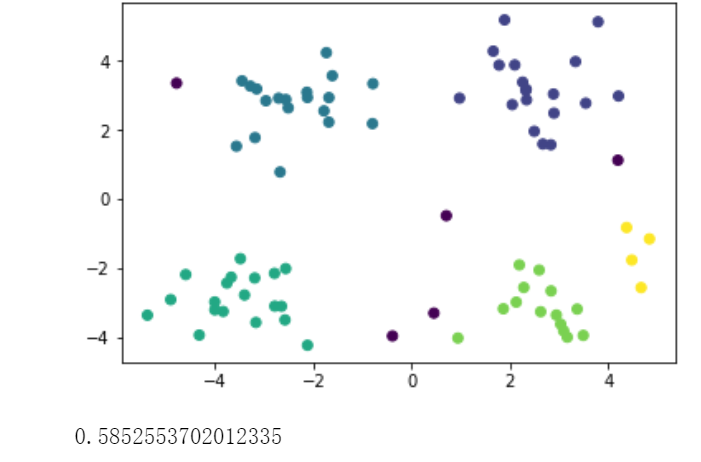
x.append(dataset[i,0])

y.append(dataset[i,1])

plt.scatter(x,y,c=pred)

plt.show()

print(calSC(dataset,distEclud,pred))



对sklearn处理Restaurant\_Data\_Beijing.txt聚类计算轮廓系数：

pred=DBSCAN(eps=4,metric=distSLC2).fit\_predict(dataMat)#

x=[]

y=[]

for i in range(len(dataMat)):

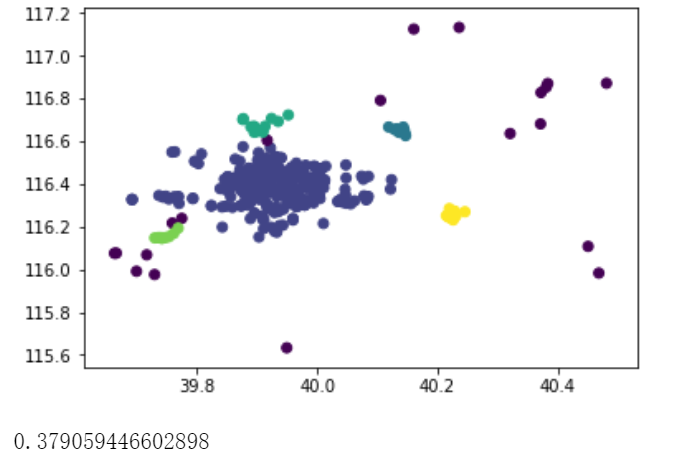
x.append(dataMat[i][0])

y.append(dataMat[i][1])

plt.scatter(x,y,c=pred)

plt.show()

print(calSC(dataMat,distSLC2,pred))



对自己的DBSCAN处理Restaurant\_Data\_Beijing.txt聚类计算轮廓系数：

pred=myDBSCAN(dataMat,eps=3.5,minpts=4,caldist=distSLC2)

x=[]

y=[]

for i in range(len(dataMat)):

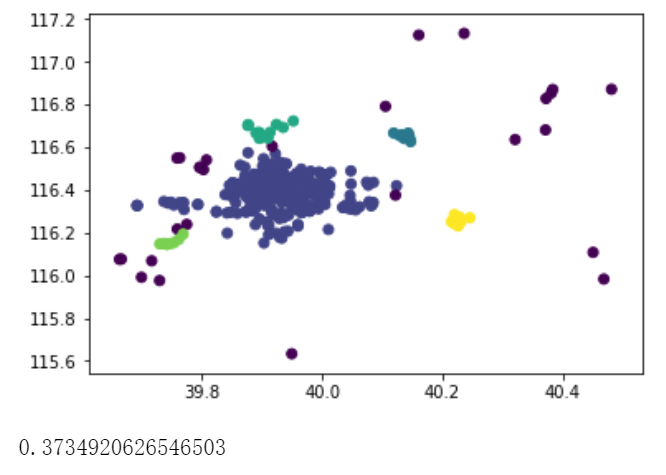
x.append(dataMat[i][0])

y.append(dataMat[i][1])

plt.scatter(x,y,c=pred)

plt.show()

print(calSC(dataMat,distSLC2,pred))



1. 尝试在tensorflow环境下实现kMeans：

使用tensorflow实现Kmean处理testSet.txt的代码

import tensorflow as tf

import random

def tfkmeans1(data,k,iterations):

n=len(data)

dim=len(data[0])

indices=list(range(n))

random.shuffle(indices)

graph=tf.Graph()

with graph.as\_default():

sess=tf.Session()

centroids=[tf.Variable(data[indices[i]]) for i in range(k)]

centroidv=tf.placeholder("float32",[dim])

cent\_assigns=[]

for centroid in centroids:

cent\_assigns.append(tf.assign(centroid,centroidv))

assignments=[tf.Variable(0) for i in range(n)]

assignmentv=tf.placeholder("int32")

cluster\_assigns=[]

for assignment in assignments:

cluster\_assigns.append(tf.assign(assignment,assignmentv))

mean\_input=tf.placeholder("float",[None,dim])

mean\_op=tf.reduce\_mean(mean\_input,0)

v1=tf.placeholder("float",[dim])

v2=tf.placeholder("float",[dim])

euclid\_dist=tf.sqrt(tf.reduce\_sum(tf.pow(tf.add(v1,tf.negative(v2)),2)))

centroid\_distances=tf.placeholder("float",[k])

cluster\_assignment=tf.argmin(centroid\_distances,0)

init\_op=tf.global\_variables\_initializer()

sess.run(init\_op)

for itern in range(iterations):

for vector\_n in range(n):

vect=data[vector\_n]

distances=[sess.run(euclid\_dist,feed\_dict={v1:vect,v2:sess.run(centroid)}) for centroid in centroids]

assignment=sess.run(cluster\_assignment,feed\_dict={centroid\_distances:distances})

sess.run(cluster\_assigns[vector\_n],feed\_dict={assignmentv:assignment})

for cluster\_n in range(k):

assigned\_vects=[data[i] for i in range(n) if sess.run(assignments[i])==cluster\_n]

new\_location=sess.run(mean\_op,feed\_dict={mean\_input:np.array(assigned\_vects)})

sess.run(cent\_assigns[cluster\_n],feed\_dict={centroidv:new\_location})

centroids=sess.run(centroids)

assignments=sess.run(assignments)

return centroids,assignments

#print(len(dataset[1]))

#dataset2=[]

#for i in range(len(dataset)):

# dataset2.append(dataset[i])

#print(dataset2)

#print(type(dataset))

dataset2=dataset.tolist()

center,pred=tfkmeans1(dataset2,4,50)

print(center,pred)

x=[]

y=[]

for i in range(dataset.shape[0]):

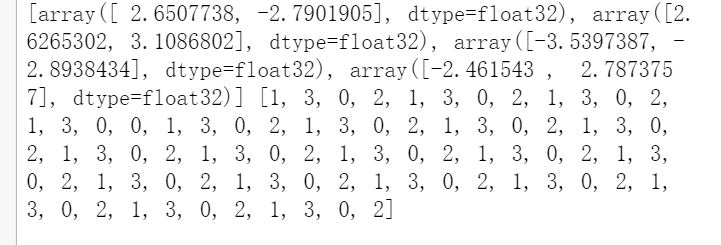
x.append(dataset[i,0])

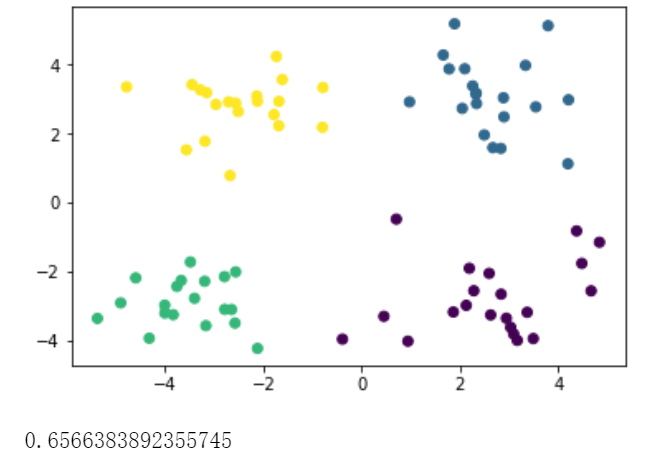
y.append(dataset[i,1])

plt.scatter(x,y,c=pred)

plt.show()

print(calSC(dataset,distEclud,pred))





使用tensorflow实现Kmean处理Restaurant\_Data\_Beijing.txt的代码：

import tensorflow as tf

import random

def tfkmeans2(data,k,iterations):

n=len(data)

dim=len(data[0])

indices=list(range(n))

random.shuffle(indices)

graph=tf.Graph()

with graph.as\_default():

sess=tf.Session()

centroids=[tf.Variable(data[indices[i]]) for i in range(k)]

centroidv=tf.placeholder("float32",[dim])

cent\_assigns=[]

for centroid in centroids:

cent\_assigns.append(tf.assign(centroid,centroidv))

assignments=[tf.Variable(0) for i in range(n)]

assignmentv=tf.placeholder("int32")

cluster\_assigns=[]

for assignment in assignments:

cluster\_assigns.append(tf.assign(assignment,assignmentv))

mean\_input=tf.placeholder("float",[None,dim])

mean\_op=tf.reduce\_mean(mean\_input,0)

v1=tf.placeholder("float",[dim])

v2=tf.placeholder("float",[dim])

euclid\_dist=tf.sqrt(tf.reduce\_sum(tf.pow(tf.add(v1,tf.negative(v2)),2)))

centroid\_distances=tf.placeholder("float",[k])

cluster\_assignment=tf.argmin(centroid\_distances,0)

init\_op=tf.global\_variables\_initializer()

sess.run(init\_op)

for itern in range(iterations):

for vector\_n in range(n):

vect=data[vector\_n]

distances=[sess.run(euclid\_dist,feed\_dict={v1:vect,v2:sess.run(centroid)}) for centroid in centroids]

assignment=sess.run(cluster\_assignment,feed\_dict={centroid\_distances:distances})

sess.run(cluster\_assigns[vector\_n],feed\_dict={assignmentv:assignment})

for cluster\_n in range(k):

assigned\_vects=[data[i] for i in range(n) if sess.run(assignments[i])==cluster\_n]

new\_location=sess.run(mean\_op,feed\_dict={mean\_input:np.array(assigned\_vects)})

sess.run(cent\_assigns[cluster\_n],feed\_dict={centroidv:new\_location})

centroids=sess.run(centroids)

assignments=sess.run(assignments)

return centroids,assignments

center,pred=tfkmeans2(dataMat,4,50)

print(center,pred)

x=[]

y=[]

for i in range(len(dataMat)):

x.append(dataMat[i][0])

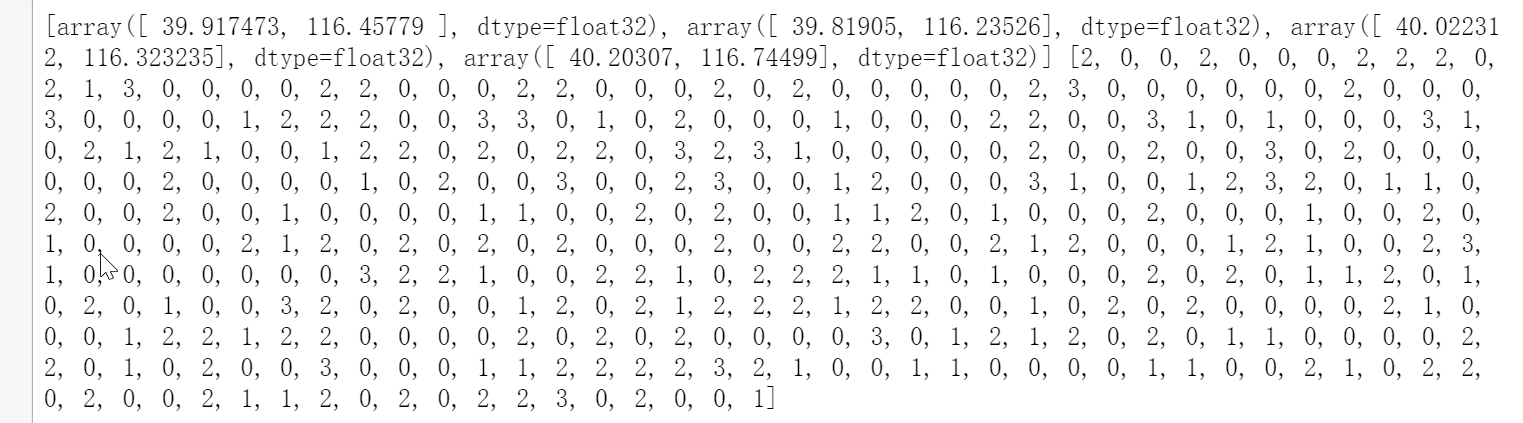
y.append(dataMat[i][1])

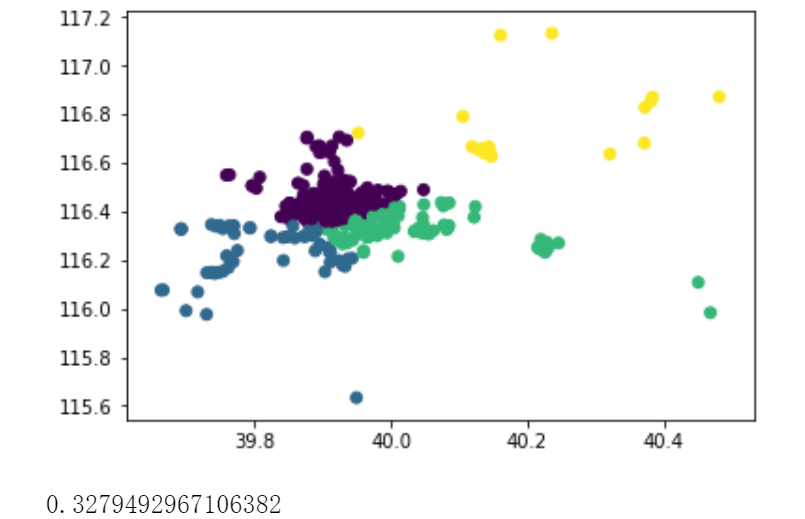
plt.scatter(x,y,c=pred)

plt.show()

print(calSC(dataMat,distSLC2,pred))

运行结果：





1. 实验总结：
2. 遇到实验指导中的问题：

api.py中命令为f = open(r'Restaurant\_Data\_Beijing.txt','w') ，但这个”r”似乎是误输入造成的，保留的话函数无法正常运行，需要将”r”去掉才可以正常运行代码得到地点数据。

1. 遇到问题：api.py获得的txt的数据中应该全为\t制表符来分隔数据，但实际获得的会显示出比\t宽度小的(疑似空格的)字符分隔数据，而代码只输出了\t和\n等字符，不清楚为何输出的数据在外观上没有都按\t对齐。
2. Python3里的sum有两种，一种是numpy里的，默认是矩阵全部数相加，用axis参数可以实现按行/按列相加；而另一种Python3的sum会将矩阵相加成向量。如果欧几里得距离计算中的sum不用np.sum的话会报错。
3. 学习到了球面距离(地球上给定经纬度两点的距离)的计算公式：dist=R·arc cos[cosβ1cosβ2cos（α1-α2）+sinβ1sinβ2]
4. DBSCAN算法是基于密度的分类算法，可以适用于较复杂形状的簇的分类，可以解决kMeans算法分类都是类球型域的问题，且能比较好地自适应地确定簇的个数，不需要事先预设簇的数量，能比较好地处理未知情形的簇的分类。
5. Python3的字典不能访问键值之前不存在的元素，如果字典不存在当前键值，需要先使用dict[key]=0来实现初始化这个键值对应的值，然后才可以访问。这里与C++的map不同，map可以直接访问之前键值没有出现过的元素，并且自动将其添加入字典。
6. 可以用for key in dict来实现字典的遍历，key会依次遍历dict的每个键值。
7. Matrix.tolist()方法可以将一个Matrix转换为一个list嵌套list，mat()方法则可以将list转换为Matrix