**实验四 k均值聚类算法**

聚类是一种无监督学习，它将相似的对象归到同一个簇中，将不相似的对象归到不同簇。“相似”这一概念取决于所选择的相似度计算方法。K均值聚类算法之所以称之为k均值，是因为它可以发现k个不同的簇，且每个簇的中心采用簇中所含值的均值计算而成。

**实验要求：**

应用k均值聚类算法，对给定的数据集进行聚类操作，打印聚类过程中的各个类的质心坐标。

**实验步骤：**

创建一个kMeans.py文件，在开头写下：

*#-\*-coding:utf-8-\*-*

*from numpy import \**

本文档所有代码都放到该文件中。

①导入数据集。

*def loadDataSet(fileName):*

# 初始化返回变量

*dataMat = []*

# 大致逻辑参考实验三的相关步骤

# 注意：

# a.要求函数返回矩阵的形式。可使用类似data=[]进行初始化，然后使

# 用data.append(XXX)的格式填充数据。

# b.数据集文件中每一行代表一个点的坐标，坐标与坐标之间使用制表

# 符“\t”分隔，在读取数据时要注意正确切割。

# c.将文本中的数值保存成float的数值格式，可使用map(float, XXX)函数

**#【代码待补全】**

*return dataMat*

②定义距离计算函数。这里使用欧氏距离，其它距离算法可自行了解。

*def distEclud(vecA, vecB):*

# 计算欧式距离

**#【代码待补全】**

③编写创建质心函数，为给定数据集创建包含k个随机质心的集合。

*def randCent(dataSet, k):*

# 获取数据集中坐标维数

*n = shape(dataSet)[1]*

# 创建一个矩阵保存随机生成的k个质心

*centroids = mat(zeros((k,n)))*

*for j in range(n):*

# 获取该列的最小值与最大值，得到该列的取值范围。

# 然后在该范围内随机生成坐标值

# 可使用numpy中的random.rand()函数，其生成的

# 随机数范围为[0,1)

**#【代码待补全】**

*return centroids*

④实现kMeans算法。

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

# 获取数据总量

*m = shape(dataSet)[0]*

# 使用一个矩阵辅助记录，第一列保存所属质心下标，

# 第二列保存到该质心的距离的平方

*clusterAssment = mat(zeros(m,2))*

# 调用上一步的函数随机生成k个质心保存为centroids

**#【代码待补全】**

# 使用一个标记记录质心是否发生变化

# 若没变化则说明算法已经收敛

*clusterChanged = True*

*while clusterChanged:*

*clusterChanged = False*

*for i in range(m):*

# 设置标记记录数据点到所有质心的最小距离及质心下标

*minDist = inf; minIndex = -1*

*for j in range(k):*

# 调用函数计算数据点到质心的距离

# 保存到变量distJI

# 并更新相关标记

**#【代码待补全】**

# 比较clusterAssment中这一行的第一列记录的下标是否等

# 于前面更新的质心下标，若相等则说明质心已收敛，否则

# 还没收敛，据此设置相关标记

# 然后记录更新clusterAssment中的数据

**#【代码待补全】**

# 打印质心

# python3使用print()函数

**#【代码待补全】**

# 根据新的聚类结果重新计算质心

# 因为nonzero()函数从参数需为列表，故使用.A进行转换

*for cent in range(k):*

*ptsInClust = \*

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

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

*return centroids, clusterAssment*

可以在python命令行中使用如下代码查看运行结果：

*from numpy import \**

*import kMeans*

*dataMat = mat(kMeans.loadDataSet(‘testSet.txt’))*

*centroids, clustAssing = kMeans.kMeans(dataMat, 4)*

这里的testSet.txt是数据集文件，4就是需要分成的类别数目。

**\*\*\*拓展延伸\*\*\***

K均值聚类算法有一个缺点就是容易收敛到局部最小值。针对这种情况，有人提出了二分k均值算法。该算法首先将所有点作为一个簇，然后将该簇一分为二。之后选择其中一个簇继续进行划分，选择哪一个簇进行划分取决于对其划分是否可以最大程度降低误差(点到质心距离的平方)。不断重复这个划分过程，直到得到用户指定的簇数目为止。伪代码形式如下：

将所有点看成一个簇

当簇数目小于k时

对每一个簇

计算总误差(即点到质心距离的平方)

在给定的簇上面进行k均值聚类(k=2)

计算将该簇一分为二之后的总误差

选择使得误差最小的那个簇进行划分操作