# 模式识别实验报告

## 实验一 K-均值聚类

学院：计算机科学与技术学院

姓名：曾钰城

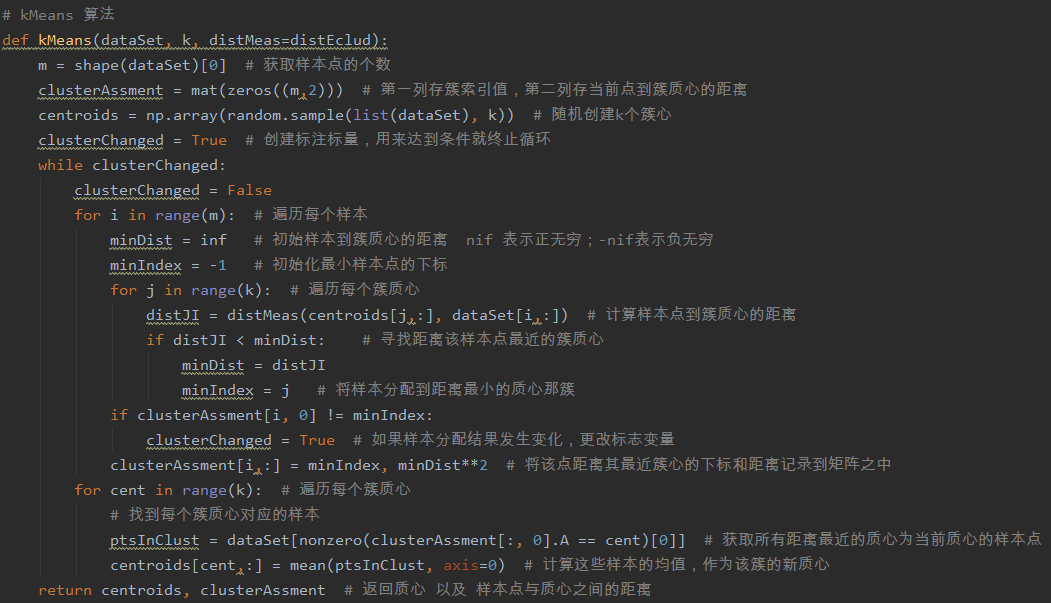
学号：1173710105

1. **实验内容**
2. 使用python或Matlab编程实现K-均值聚类算法：要求独立完成算法编程，禁止调用已有函数库或工具箱中的函数；
3. 使用仿真数据测试算法的正确性：将下列19个样本聚成2个聚类：

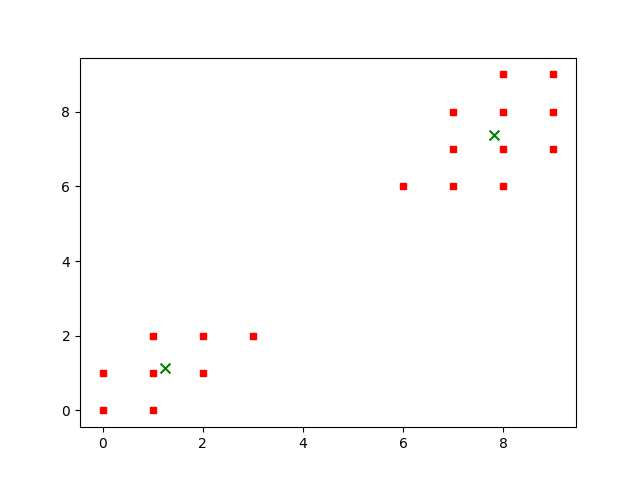


1. MNIST数据集测试：ClusterSamples中的10000个784维特征手写数字样本聚类为10个类别，根据SampleLabels中的标签统计每个聚类中不同样本的数量。测试不同初始值对聚类结果的影响。
2. **程序代码**

（K-均值算法部分代码）



1. **实验结果**
2. 仿真数据实验结果：（可以列出每个聚类中包含的样本，也可以画图显示不同聚类）



聚簇一：

簇心：(1.25, 1.25)

包含的样本：X1, X2, X3, X4, X5, X6, X7, X8

聚簇二：

簇心：(7.82, 7.36)

包含的样本：X9, X10, X11, X12, X13, X14, X15, X16, X17, X18, X19

1. MNIST数据集实验结果：

**每个聚类中包含不同类别样本数量统计表**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 聚类0 | 465 | 0 | 1 | 4 | 1 | 14 | 14 | 1 | 5 | 5 |
| 聚类1 | 26 | 3 | 57 | 748 | 0 | 364 | 7 | 1 | 370 | 18 |
| 聚类2 | 1 | 1 | 29 | 23 | 476 | 42 | 17 | 163 | 30 | 417 |
| 聚类3 | 4 | 414 | 55 | 22 | 47 | 163 | 54 | 74 | 154 | 20 |
| 聚类4 | 418 | 0 | 18 | 30 | 1 | 83 | 34 | 1 | 10 | 0 |
| 聚类5 | 12 | 2 | 5 | 44 | 380 | 223 | 1 | 36 | 322 | 473 |
| 聚类6 | 1 | 2 | 723 | 35 | 5 | 4 | 14 | 7 | 23 | 3 |
| 聚类7 | 0 | 702 | 62 | 65 | 26 | 16 | 37 | 51 | 45 | 28 |
| 聚类8 | 1 | 2 | 11 | 3 | 2 | 3 | 0 | 722 | 2 | 49 |
| 聚类9 | 32 | 0 | 33 | 16 | 21 | 29 | 801 | 2 | 18 | 1 |

1. **代码**

from numpy import \*  
import numpy as np  
import matplotlib.pyplot as plt  
import random  
import csv  
  
# 加载数据  
def loadDataSet(filename):  
 dataMat = []  
 fr = open(filename)  
 for line in fr.readlines():  
 curLine = line.strip().split('\t') # 将数据拆开返回一个列表  
 fltLine = list((map(float, curLine))) # 将列表里面的数字转化为浮点类型  
 dataMat.append(fltLine) # 添加到列表之中  
 return dataMat # 返回一个列表  
  
# 计算欧氏距离  
def distEclud(vecA, vecB):  
 return sqrt(sum(power(vecA - vecB, 2))) # 计算欧氏距离  
  
# kMeans 算法  
def kMeans(dataSet, k, distMeas=distEclud):  
 m = shape(dataSet)[0] # 获取样本点的个数  
 clusterAssment = mat(zeros((m,2))) # 第一列存簇索引值，第二列存当前点到簇质心的距离  
 centroids = np.array(random.sample(list(dataSet), k)) # 随机创建k个簇心  
 clusterChanged = True # 创建标注标量，用来达到条件就终止循环  
 while clusterChanged:  
 clusterChanged = False  
 for i in range(m): # 遍历每个样本  
 minDist = inf # 初始样本到簇质心的距离 nif 表示正无穷；-nif表示负无穷  
 minIndex = -1 # 初始化最小样本点的下标  
 for j in range(k): # 遍历每个簇质心  
 distJI = distMeas(centroids[j,:], dataSet[i,:]) # 计算样本点到簇质心的距离  
 if distJI < minDist: # 寻找距离该样本点最近的簇质心  
 minDist = distJI  
 minIndex = j # 将样本分配到距离最小的质心那簇  
 if clusterAssment[i, 0] != minIndex:  
 clusterChanged = True # 如果样本分配结果发生变化，更改标志变量  
 clusterAssment[i,:] = minIndex, minDist\*\*2 # 将该点距离其最近簇心的下标和距离记录到矩阵之中  
 for cent in range(k): # 遍历每个簇质心  
 # 找到每个簇质心对应的样本  
 ptsInClust = dataSet[nonzero(clusterAssment[:, 0].A == cent)[0]] # 获取所有距离最近的质心为当前质心的样本点  
 centroids[cent,:] = mean(ptsInClust, axis=0) # 计算这些样本的均值，作为该簇的新质心  
 return centroids, clusterAssment # 返回质心 以及 样本点与质心之间的距离  
  
# 可视化结果  
def plot(dataSet,centValue):  
  
 x1=dataSet[:,0]  
 x2=dataSet[:,1]  
 fig=plt.figure('k均值算法')  
 ax=fig.add\_subplot(111)  
  
 ax.scatter(list(x1),list(x2),s=15,c='red',marker='s')  
 ax.scatter(list(centValue[:,0]), list(centValue[:,1]), s=50, c='green', marker='x')  
 plt.show()  
  
def createDataByHand():  
 x1 = [0, 0]  
 x2 = [1, 0]  
 x3 = [0, 1]  
 x4 = [1, 1]  
 x5 = [2, 1]  
 x6 = [1, 2]  
 x7 = [2, 2]  
 x8 = [3, 2]  
 x9 = [6, 6]  
 x10 = [7, 6]  
 x11 = [8, 6]  
 x12 = [7, 7]  
 x13 = [8, 7]  
 x14 = [9, 7]  
 x15 = [7, 8]  
 x16 = [8, 8]  
 x17 = [9, 8]  
 x18 = [8, 9]  
 x19 = [9, 9]  
 dataset = []  
 dataset.append(x1)  
 dataset.append(x2)  
 dataset.append(x3)  
 dataset.append(x4)  
 dataset.append(x5)  
 dataset.append(x6)  
 dataset.append(x7)  
 dataset.append(x8)  
 dataset.append(x9)  
 dataset.append(x10)  
 dataset.append(x11)  
 dataset.append(x12)  
 dataset.append(x13)  
 dataset.append(x14)  
 dataset.append(x15)  
 dataset.append(x16)  
 dataset.append(x17)  
 dataset.append(x18)  
 dataset.append(x19)  
 return dataset  
  
def readDataFromCSV():  
 dataset = []  
 labels = []  
 with open('ClusterSamples.csv', 'r') as f:  
 reader = csv.reader(f)  
 for row in reader:  
 dataset.append(row)  
  
 with open('SampleLabels.csv', 'r') as f:  
 reader = csv.reader(f)  
 for row in reader:  
 labels.append(row)  
  
 return dataset, labels  
  
def classify(clusterList, labels):  
 resultMatrix = np.zeros((10, 10), dtype=int)  
 for i in range(0, len(clusterList)):  
 cluster = int(clusterList[i, 0])  
 label = int(labels[i, 0])  
 resultMatrix[cluster, label] = resultMatrix[cluster, label] + 1  
 for i in range(0, shape(resultMatrix)[1]):  
 sum = 0  
 for j in range(0, shape(resultMatrix)[0]):  
 sum = sum + resultMatrix[j, i]  
 return resultMatrix  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 dataset, labels = readDataFromCSV()  
 dataset = np.array(dataset).astype(float)  
 labels = np.array(labels).astype(int)  
 cenValue, clusterList = kMeans(dataset, 10)  
 print(classify(clusterList, labels))