

**课 程 实 验 报 告**

**课程名称： 大数据分析**

**专业班级： ACM1901**

**学 号： U201914965**

**姓 名： 卫云泽**

**指导教师： 崔金华**

**报告日期： 2021年12月22日**

**计算机科学与技术学院**

**目录**

[实验四 kmeans算法及其实现 1](#_Toc58252477)

[**4.1实验目的** 1](#_Toc58252478)

[**4.2 实验内容** 1](#_Toc58252479)

[**4.3 实验过程** 2](#_Toc58252480)

[4.3.1 编程思路 2](#_Toc58252481)

[4.3.2 遇到的问题及解决方式 4](#_Toc58252482)

[4.3.3 实验测试与结果分析 5](#_Toc58252483)

[**4.4 实验总结** 6](#_Toc58252484)

# 实验四 kmeans算法及其实现

## **4.1实验目的**

1、加深对聚类算法的理解,进一步认识聚类算法的实现；

2、分析kmeans流程,探究聚类算法院里；

3、掌握kmeans算法核心要点；

4、将kmeans算法运用于实际，并掌握其度量好坏方式。

## **4.2 实验内容**

提供葡萄酒识别数据集(WineData.csv)，数据集已经被归一化(normalizedwinedata.csv)。同学可以思考数据集为什么被归一化，如果没有被归一化，实验结果是怎么样的，以及为什么这样。

同时葡萄酒数据集中已经按照类别给出了1、2、3种葡萄酒数据，在cvs文件中的第一列标注了出来，大家可以将聚类好的数据与标的数据做对比。

编写kmeans算法，算法的输入是葡萄酒数据集，葡萄酒数据集一共13维数据，代表着葡萄酒的13维特征，请在欧式距离下对葡萄酒的所有数据进行聚类，聚类的数量K值为3。

在本次实验中，最终评价kmean算法的精准度有两种，第一是葡萄酒数据集已经给出的三个聚类，和自己运行的三个聚类做准确度判断。第二个是计算所有数据点到各自质心距离的平方和。请各位同学在实验中计算出这两个值。

实验进阶部分：在聚类之后，任选两个维度，以三种不同的颜色对自己聚类的结果进行标注，最终以二维平面中点图的形式来展示三个质心和所有的样本点。效果展示图可如图4-1所示。



图4-1 葡萄酒数据集在黄酮和总酚维度下聚类图像（SSE为距离平方和，Acc为准确率）

## **4.3 实验过程**

### 4.3.1 编程思路

**随机初始化质心**

在执行kmeans聚类迭代操作之前，首先需要获得k个初始化随机质心。在数据规模范围内随机获取 k个数据编号，以此作为初始化的随机质心。质心初始化过程如下，其中centroids变量存放的是当前质心的信息。

1. init = random.sample(range(data.shape[0]), k)  # 初始化随机质心
2. **for** i **in** range(k):
3. centroids[i] = data[init[i]]

**kmeans聚类过程**

kmeans聚类是在数据集上反复迭代执行的过程，每一轮迭代都要计算出每个点与当前的三个质心的距离，并将该点划入距离其最近的质心的那一类。每一轮分类完毕后，再计算求出每一类样本点的质心，作为新的质心参与下一轮迭代，迭代过程以达到最大迭代次数（max\_iter）或质心（几乎）不再变化作为结束条件。kmeans聚类过程的实现如下。

1. **for** i **in** range(max\_iter):
2. predlabel = []  # 每一轮初始化聚类标签为空列表
3. resdist2 = []   # 初始化距离平方为空列表
4. **for** j **in** range(k):
5. cluster[j] = []     # 初始化每一轮聚类列表为空
6. **for** point **in** data:
7. dist = []   # 样本到每个质心的距离
8. **for** key, centroid **in** centroids.items():
9. thisdist = np.sqrt(np.sum((point - centroid) \*\* 2))  # 计算距离
10. dist.append(thisdist)
11. thisclass = np.argmin(dist)  # 获取最小距离下标
12. cluster[thisclass].append(point)
13. predlabel.append(thisclass + k) # predlabel和实际label不重叠
14. resdist2.append(np.sum((point - centroids[thisclass]) \*\* 2))
15. new\_centroids = {}  # 新的质心
16. **for** j **in** range(k):
17. new\_centroids[j] = np.mean(cluster[j], axis=0)
18. **print**("{}: {}".format(j, new\_centroids[j]))
19. maxdelta = 0.0
20. **for** j **in** range(k):
21. maxdelta = max(maxdelta, np.max(np.abs(new\_centroids[j] - centroids[j])))   # 获得最大差值maxdelta
22. **if** i > max\_iter **or** maxdelta < delta:
23. **break**
24. centroids = new\_centroids
25. i += 1

代码中实际用于每一轮迭代过程的数据结构分别是用于记录本次分类结果的cluster（存放list的dict类型）和用于存放当前质心信息的centroids。predlabel 和resdist2 记录了本次迭代的过程信息，其中predlabel是聚类的标签，由于聚类过程并不会生成“真实的标签”，故聚类标签使用了与实际标签没有交集的标签集合（即当前类别号+k）；resdist2记录了本次所有点距离其分类质心的距离平方，用于最终结果的SSE展示。

kmeans的每一轮迭代过程首先会初始化每一类为空集，如代码4-5行；然后遍历数据集中的所有点，计算该点与所有质心的距离并顺次存入临时list变量dist中，使用np.argmin可直接获得dist中最小值的所在位置（索引），由于dist是顺次存放的，最小距离所在位置（索引）即为该点本次的分类结果（0~k-1类），将该点加入cluster中的对应类别，同时在predlabel和resdist2中记录相关信息。遍历过程见代码6-14行。代码15-18行根据本次cluster聚类结果求出每一类所有点的质心，即新的质心，并在19-21行求出新质心和先前质心的最大差值，首先求出单个质心每一维度的最大差值，再求出所有质心的最大差值的最大值作为最大差值。若最大差值小于预设的delta值，或迭代次数超出预设的最大迭代次数max\_iter，则聚类过程结束。

**将聚类标签映射到真实标签**

由于聚类过程并不能获得某一类对应的真实标签，故实验中的聚类结果以“聚类标签”的形式给出，聚类标签和真实标签不存在交集。完成kmeans聚类过程后，需要将聚类标签映射到真实标签，一种思路是统计每一类聚类标签对应的每一类真实标签的数目，对应真实标签数目最大的即为该类聚类标签所对应的真实标签。实现聚类标签到真实标签的映射，实现代码如下。

1. labelmap = {}   # 聚类标签集和真实标签的map
2. **for** i **in** range(k):
3. thismap = {}    # 用于统计标签i + k出现的真实标签的次数
4. predi = i + k
5. **for** index **in** range(label.shape[0]): # 遍历所有标签，记录
6. **if** predlabel[index] == predi:
7. thismap[label[index]] = thismap.get(label[index], 0) + 1
8. labelmax = 0
9. maxlabel = 0
10. **for** labeli, valuei **in** thismap.items():
11. **if** valuei > labelmax:
12. labelmax = valuei
13. maxlabel = labeli
14. labelmap[predi] = maxlabel

代码首先统计了每一类聚类标签样本对应的真实标签数目，如3-7行，然后求出该真实标签数目最多的那一个标签，并将该标签作为聚类标签对应的真实标签，记录在标签映射字典labelmap中。

**聚类结果的可视化展示**

在完成kmeans聚类和聚类标签映射后，统计聚类正确的样本数目，获得准确率Acc，并对kmeans过程获得的样本距离平方resdist2执行求和，获得所有样本的距离平方和。聚类结果进行可视化展示使用matplotlib.pyplot执行绘图，指定两个维度（如酒精和色调）给出聚类结果的可视化展现。绘图相关代码如下。

1. colors = ['r', 'b', 'g']
2. **for** key, centroid **in** centroids.items():
3. plt.scatter(centroid[0], centroid[10], marker = '+', c = colors[key],  s = 150)
4. **for** key, clusteri **in** cluster.items():
5. **for** point **in** clusteri:
6. plt.scatter(point[0], point[10], marker = '.', c = colors[key],  s = 100)
7. plt.xlabel("alcohol")   # 酒精
8. plt.ylabel("tonality")  # 色调
9. plt.title("Acc: {:.2f} SEE: {:.2f}".format(Acc, SEE))
10. plt.show()

### 4.3.2 遇到的问题及解决方式

**kmeans迭代过程无法停止问题：**

在初版代码中，kmeans过程的出口条件并未引入max\_iter，在测试时出现kmeans迭代过程无法停止的问题。引入max\_iter后发现迭代完成后聚类效果并不理想，进一步输出中间结果发现迭代过程的每一次聚类质心均未发生改变，但本次求出的前后两次质心的距离最大值却不为0，出错的中间结果如图4-1所示，图中iteration表示当前迭代次数，0,1,2分别代表本次迭代获得的三个质心。

**解决方案：**

分析出错的中间结果可知，maxdelta不为0，说明实际实现了新质心的求解，仔细审视代码发现，在求解出新的质心、判断迭代出口条件之后，遗漏了质心转换的过程，即忘记将新的质心值赋给原质心，导致聚类质心实际始终没有变化。在kmeans迭代结束前加入质心转换的赋值语句即可解决问题。

文本

描述已自动生成

图4-1 出现错误的中间结果（部分）

### 4.3.3 实验测试与结果分析

在解决了kmeans迭代过程无法停止的问题后，在数据集上对程序进行测试，获得的聚类结果如图4-2所示。

图表, 散点图

描述已自动生成

图4-2 聚类结果的可视化展示

聚类结果准确率（Acc）为0.95，SEE为48.99，图4-2选择了酒精、色调两个维度进行聚类结果的可视化展示，分别使用红、绿、蓝三种颜色表示三个类别的样本点，其中十字标记为该类别的质心位置，圆点表示该类别的样本点位置。由于仅选取2个维度展示，而实际数据维度多达13维，因此部分点距离该类别质心较远。聚类结果大体上形成了三种颜色（类别）的聚集态，

## **4.4 实验总结**

在本次实验中，我基于标准化后的红酒相关信息数据集实现了kmeans聚类算法，并给出了较好的预测结果。

在kmeans算法的实现过程中，我也进一步理解了数据的归一化的必要性。没有归一化的数据中，各维度范围差异可能较大，在kmeans过程中可能出现某一维度的对聚类结果的影响大于其他维度的情况，相当于是给某些特征维度增加了权重。若要保证每一个特征维度的权重一致，需要将不同数据的波动限制在相同范围内——即归一化操作。

此外，在执行标签映射的过程中，我也进一步理解了kmeans作为无监督学习聚类算法的特性、区别了“分类”和“聚类”两个概念的异同。在之后的学习中我也希望能够深入理解更多的无监督学习方法，进一步探索这类方法在现实中的应用价值。