

**课 程 实 验 报 告**

**课程名称： 大数据分析**

**专业班级：**

**学 号：**

**姓 名：**

**指导教师：**

**报告日期：**

**计算机科学与技术学院**

**目录**

[实验四 kmeans算法及其实现 1](#_Toc58793873)

[**4.1实验目的** 1](#_Toc58793874)

[**4.2 实验内容** 1](#_Toc58793875)

[**4.3 实验过程** 2](#_Toc58793876)

[4.3.1 编程思路 2](#_Toc58793877)

[4.3.2 遇到的问题及解决方式 2](#_Toc58793878)

[4.3.3 实验测试与结果分析 2](#_Toc58793879)

[**4.4 实验总结** 2](#_Toc58793880)

# 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，进一步掌握聚类算法的算法原理和实现流程。

2、分析常用的聚类算法kmeans的操作流程,探究聚类算法原理。

3、掌握kmeans算法的代码实现，优化算法实现细节。

4、将kmeans算法运用于实际，解决现实中的无监督分类问题并掌握其评估指标和性能优化策略。

## **4.2 实验内容**

提供葡萄酒识别数据集，数据集已经被归一化。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1.1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 系统整体框架

K-means是一种常见的无监督学习算法，也是常见的聚类算法，用于将一组无标注的点按照距离分成多个类别或者簇别。特别的，其衡量距离的方式为常见的欧氏距离。K-means算法的主要步骤如图4-1所示：

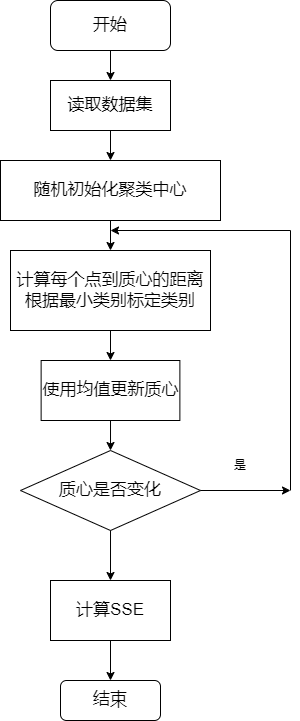


图4-1 K-means算法流程图

### 4.3.2 编程思路及实现

**1）数据处理**

原始数据（Wine.csv）需要先经过归一化处理，消除各个特征之间的量纲差异。不过贴心的学长姐为我们准备了归一化数据，可以直接使用，部分截图如图3-2所示。数据共有14列，第一类代表其类别，方便获得准确率，其余13列为13个特征。特别地，在读取特征数据时需要将第一行去掉。

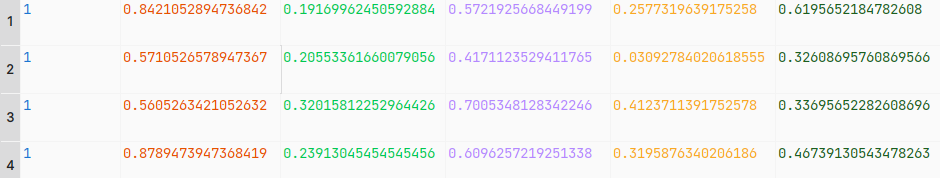


图4-2 归一化数据截图



图4-3 读取数据代码展示

**2）随机初始化质心**

初始化质心有两种方式，一是在[0,1]直接随机取随机数，二是在这一列坐标的最大值和最小值之间随机取值。我更偏向于第二种方法，在保证随机性的同时，也能够尽可能地靠近最终质心位置。



图4-4 随机初始化质心代码展示

**3）k-means迭代过程**

A）计算每个点的类别

每个点的类别根据其到三个质心的距离远近进行判断，分别对三个质心均进行欧氏距离计算，取最近的距离进行类别的标注。

我采用数组mat存储每个点的信息。第一列表示其到最近质心的距离，第二列表示k-means算法的分类结果，第三列表示其真实的分类结果。

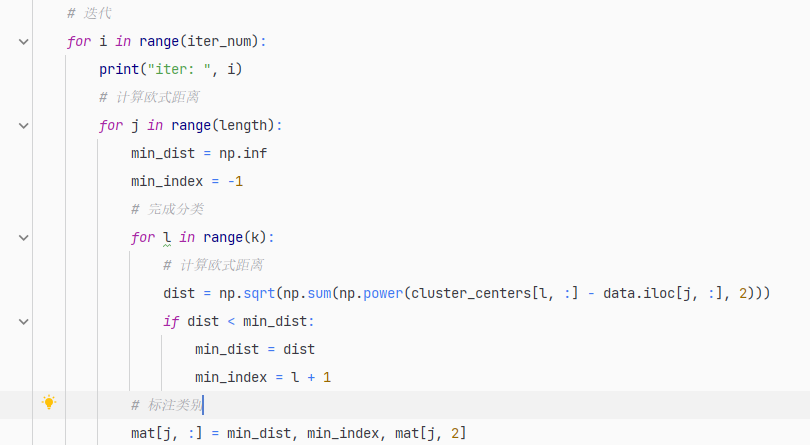


图4-5 计算距离标注类别代码展示

B）更新聚类中心

在标注完所有点的类别后，我们根据每个类别中的点，计算其均值，得到新的质心的坐标。当质心坐标不再变化（或变化值小于限制）或者超出预设迭代次数时，认为迭代完成，退出迭代。

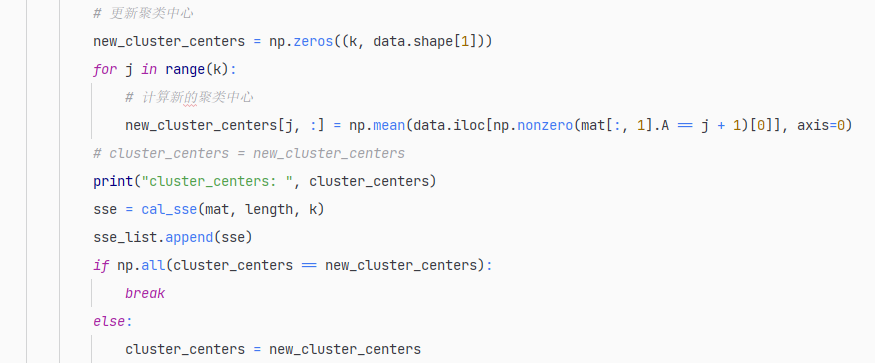


图4-6 更新聚类中心代码展示

**4）计算Acc（准确率）和SSE（残差平方和）**

在完成迭代后，我们将每个点的类别和其在数据集中的分类进行对比，得到分类的准确率。为了评估分类的效果，我们还计算SSE（Sum of Squared Errors），值越小代表分类的效果越好。SSE值的计算公式如图3-6所示，计算的代码如图3-7，3-8所示。



图3-6 SSE计算公式



图4-7 计算SSE代码展示

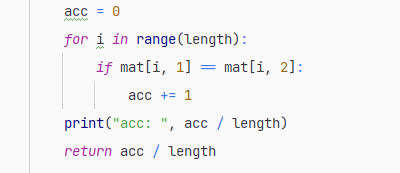


图4-8 计算准确率代码展示

**5）绘制分类图**

为了可视化分类结果，我采用matplotlib库进行绘制。由于源数据的特征为13维，此处任选两维（特征7，特征8）展现分类结果。图像结果将在4.3.3节进行展示。

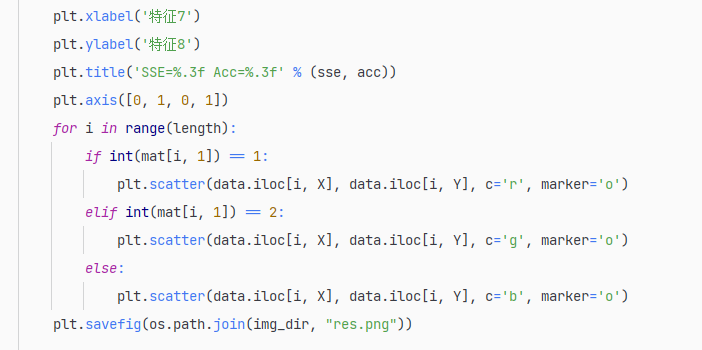


图4-9 绘图代码展示

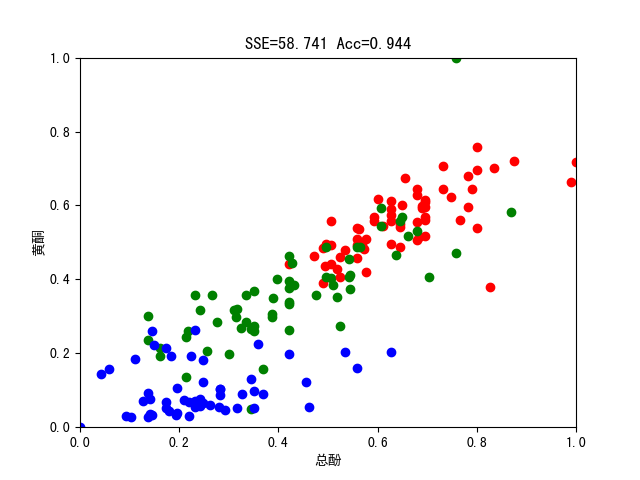
### 4.3.3 实验测试

1）k-means算法结果测试

我对k-means算法进行了多次测试，该数据集的迭代次数越6-10次，准确率约0.944，SSE值在[48,65]之间。实验结果总体符合预期。

2）算法效果可视化展示

图4-10，4-11展示了k-means分类的可视化效果，其中红色，蓝色，绿色分别为三个不同的类别，横纵坐标分别为两个特征（随机选取）。从效果可以看出，算法的分类结果较好，类别清晰，但是不同类别之间没有明显间距。这可能与绘图时特征选取有关。

  
图4-10 可视化展示

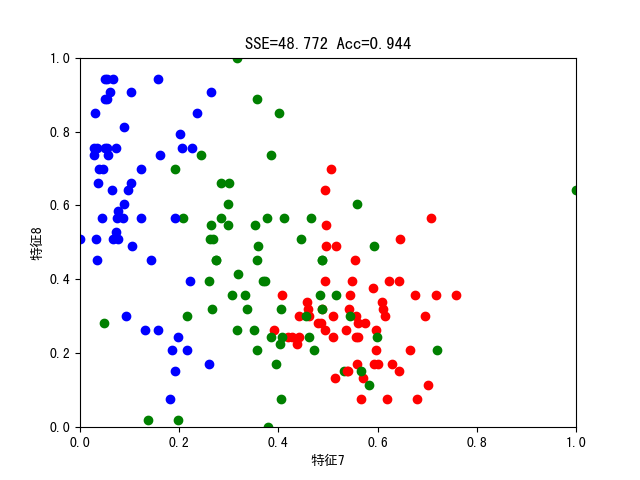


图4-11 可视化展示

### 4.3.4 对比试验及结果分析

1）不同的质心初始化方式

在4.3.2节中我提到过两种不同的质心初始化方案，我猜测不同的初始化可能对最后的分类效果影响较小，但是会影响到SSE值的变化。于是我分别绘制了两种不同方式SSE值的变化图。图4-12展示的是质心在[0,1]随机取值的SSE变化图，图4-13展示的是质心在该列最大最小值[min,max]之间随机取值的SSE变化图。

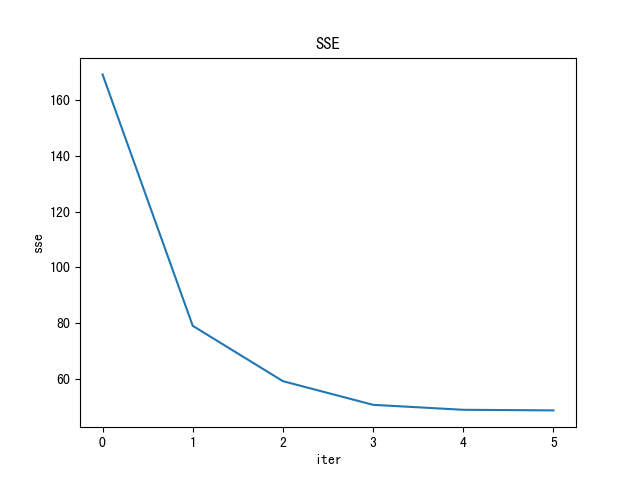


图4-12 在[0,1]间随机赋值

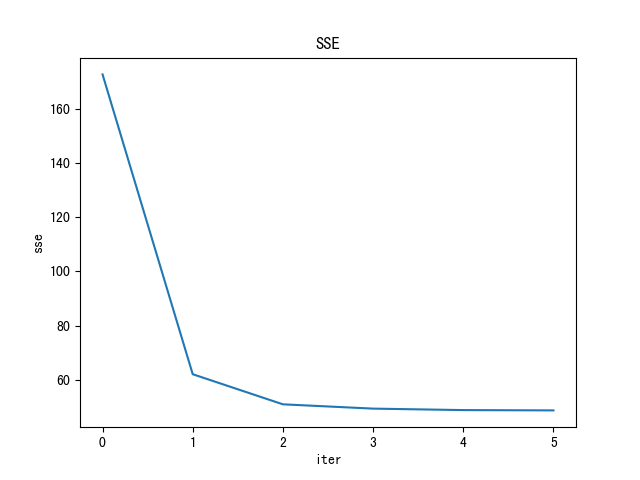


图4-13 在最大最小值间随机赋值

从图中可以得到，在[0,1]间随机赋值的SSE下降较慢，“肘部”也没有后一种方式那么明显，缩小随机取值空间确实能够使得质心更快接近最终答案。

## **4.4 实验总结**

### 4.3.2 实验中遇到的问题及解决方式

**1）准确率仅为2.2%**

在一次实验中，k-means算法正确率仅为2.2%，而同一代码多次实验的正确率可能发生浮动，34.5%，66.4%，94.4%均出现过，这样的问题引发了我的思考，但最终也找到了原因，问题并没有出现在代码上。

由于质心坐标的随机初始化，导致在分类时，每一个点分得的类别是随机的，而给定的类别是固定的。举个例子，比如A，B，C三个点，他们属于三个类别，k-means算法给他们标注的类别分别为0，1，2，但是在源数据中他们的类别为2，0，1。从结果来看，k-means算法的分类是正确的，但是在计算正确率是，他们和真实值并不匹配。这就造成了误判。所以发生了准确率的浮动，出现上述准确率均为正常，分别对应着一个，两个类别发生了误判。然而，即使发生误判，概率也会集中在0%，33%，66%，99%这几个概率上。

2）使用了14个特征

在一开始，我把其类别也当成了一个特征进行分类，这样就有14个特征，虽然从结果来看并没有非常明显的区别，甚至准确率更高（96%），但是明显不合理。我一开始并没有发现这个问题，直到我看到某一次实验中准确率达到了100%，这非常异常，经过检查才发现了多了一维特征。这也告诉我在使用数据集之前，必须先对数据集的结构进行分析。

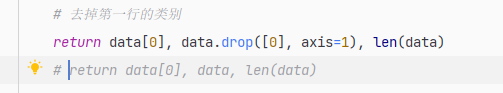


图4-14 使用正确的维度

### 4.3.3 实验总结

从实验的难度上来评判，k-means算法相对简单，流程清晰，运算简单，作为广泛使用的无监督聚类算法，k-means在几十年的发展过程中也有了许多改进和优化。我想起了一句名言，“其作始也简，其将毕也其必巨”，这是这样一个易于实现的算法，抛弃了奇淫技巧，纯粹而高效，得到了学术界和工业届的认可。这让我明白了我们需要的不仅是锦上添花，更是从头开始的原创性创新。

通过实现k-means算法，体会完整的迭代过程，我也将课堂上学习的内容同机器学习等知识进行了联系，这样课程直接的碰撞也帮助我从更多角度了解了k-means算法。