 **南昌大学实验报告**

**---（4）k-means算法实现**

实验类型：□ 验证 ■ 综合 □ 设计 □ 创新 实验日期： 11.30 实验成绩：

**一、实验目的**

学会利用k-means算法对数据集进行分类，掌握k-means算法的思想。

**二、实验内容**

根据k-means算法原理实现对数据集的分类，编译、运行程序并观察程序的输出，并分析实验结果，写出实验报告。

**三、实验要求**

1. 分析实验代码；

2. 分析运行结果；

3. 画出程序流程图。

**四、实验步骤**

1. 编写实验代码；

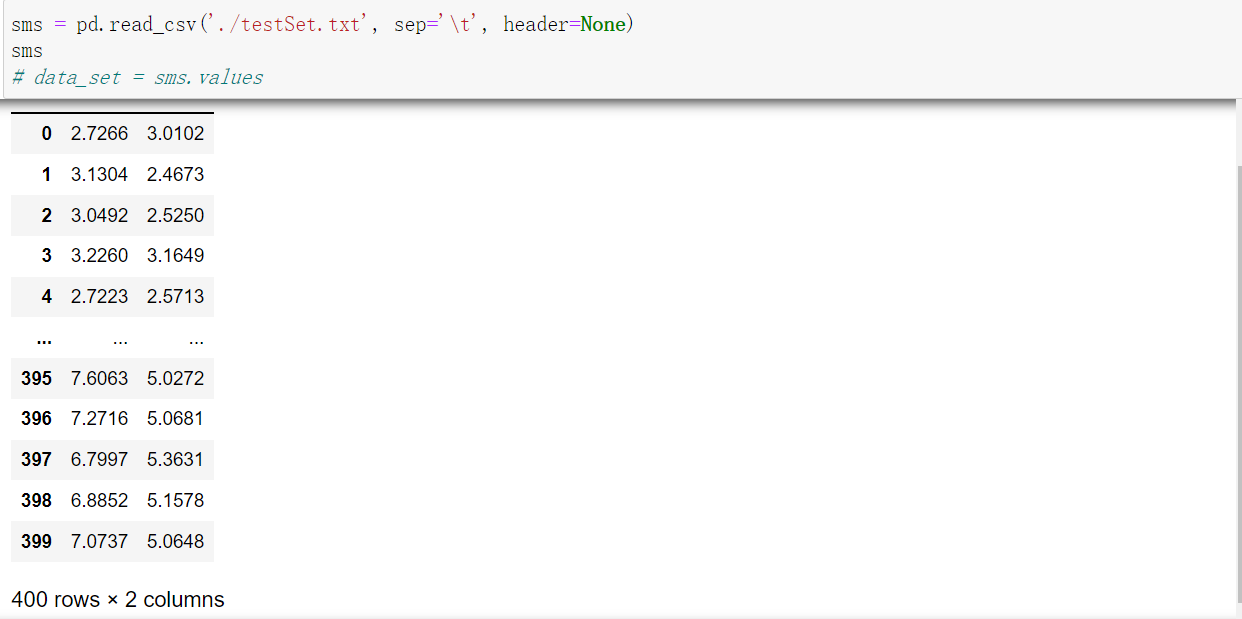
2. 编译、运行程序；

3. 观察程序的输出并分析实验结果。

**五、实验数据及处理结果**

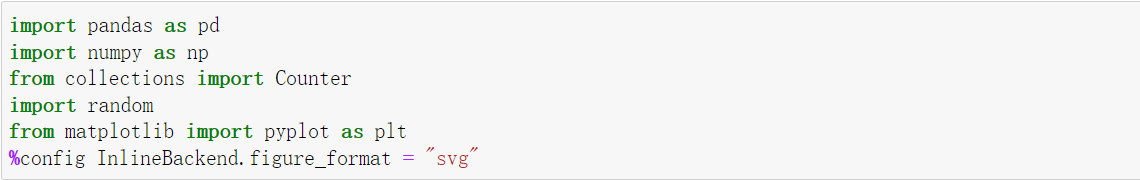
**1. 查看数据集**

使用Jupyter Notebook查看数据集。代码如下如下：

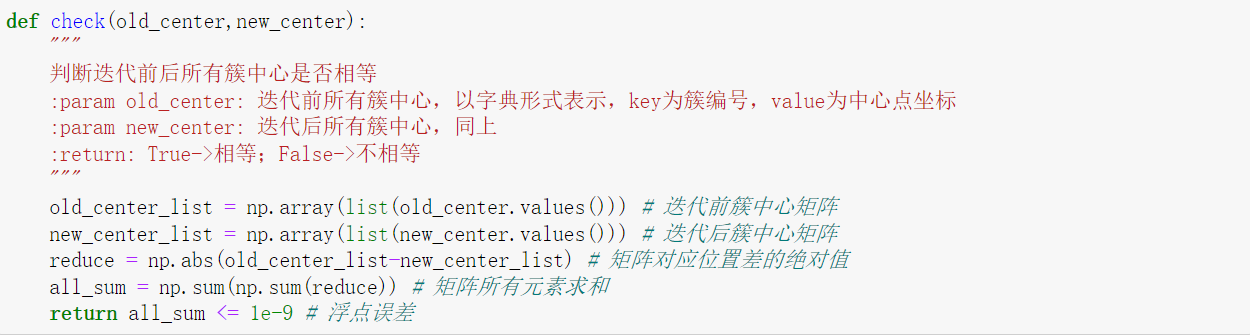
 由上图可知数据集共包含400行数据。每行数据拥有2个维度，可将每个数据视为平面直角坐标系中的点。实验的目的是要对这些点进行聚类。

**2. k-means算法实现**

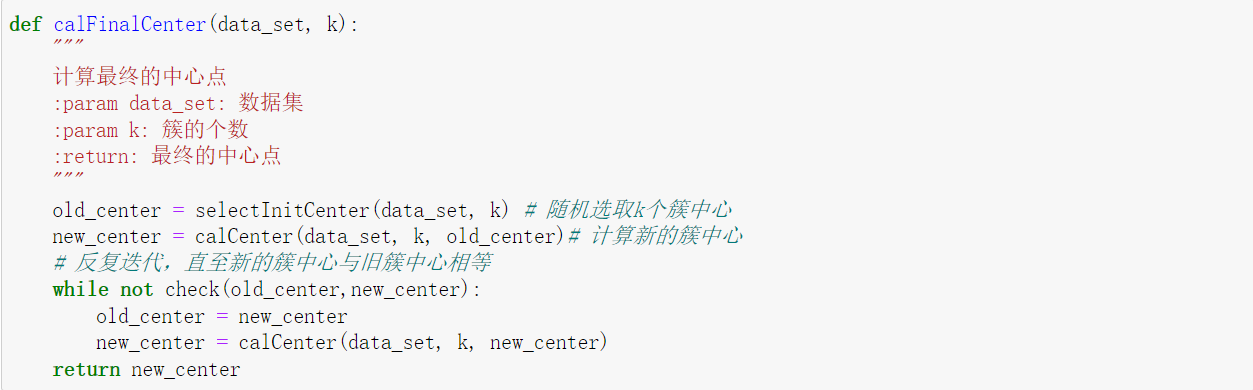
代码实现如下：















算法的流程图如下：



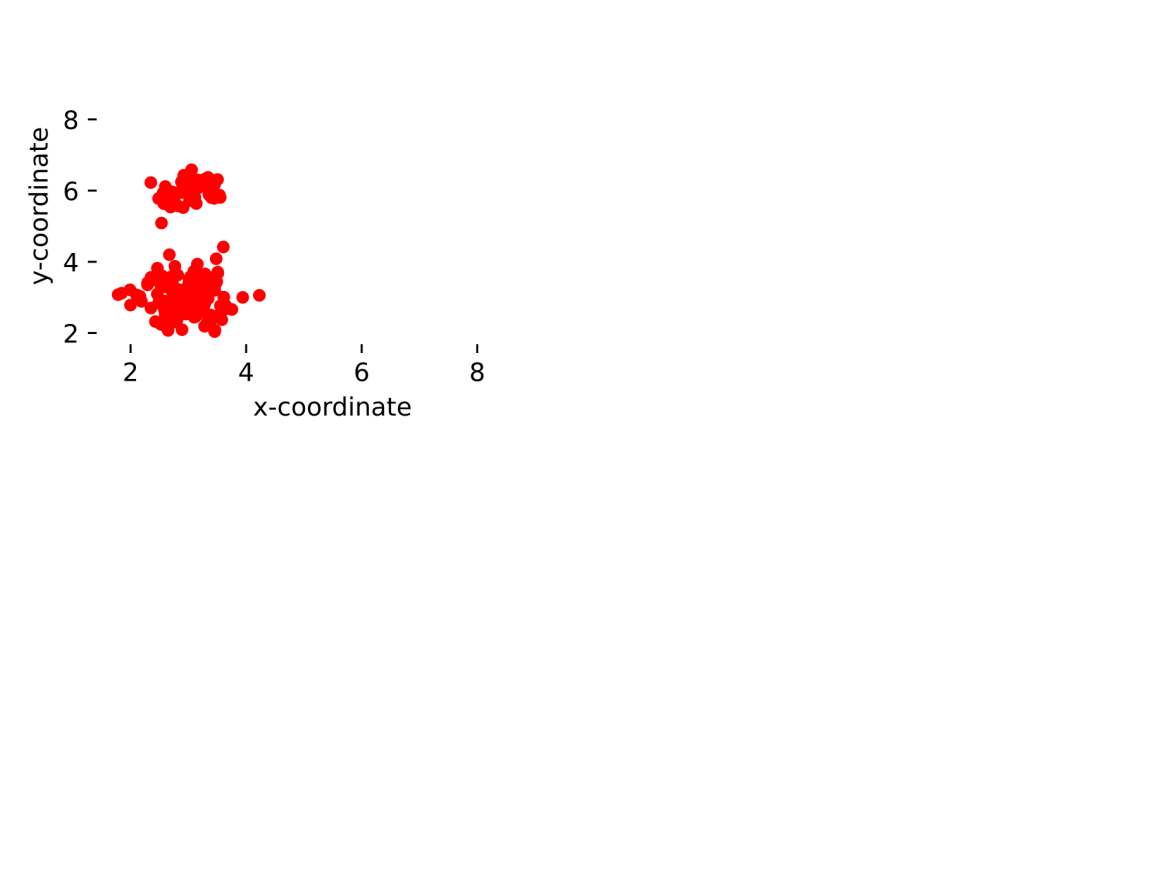
1. **聚类结果**
2. **运行结果**

将k设置为4，运行上述代码得到聚类结果并可视化，如下：

从图中可以看到，数据集中的点被成功分为了4个簇，且各簇内的点都是在距离上相近的点，满足了聚类的需要。

**(2) 不同K值下的聚类结果**

调整K值，使其从2递增到5，得到不同的K值下的聚类结果，如下：



从图中可以看到，当K=2和K=3时，达成了聚类的目的，但是还缺少进一步的划分；当K=4时，聚类效果最佳，距离相近的点都被归到同一簇中；当K=5时，有划分过度的倾向。

1. **问题讨论**

上述实验表明，k-means算法描述容易、实现简单、快速，但是存在如下不足：

1. k-means算法中簇个数k需要预先给定。
2. 算法对初始值的选取依赖性极大以及算法常陷入局部最优解。
3. 由于将簇的质心(即均值)作为簇中心进行新一轮计算，远离数据密集区的离群点呵噪声点会导致聚类中心偏离真正的数据密集区，所以k-means算法对噪声点和离群点很敏感。

下面三节将分别探讨并依次解决上述问题。

1. **轮廓系数(Silhouette Coefficient)**

轮廓系数（Silhouette Coefficient）,是[聚类](https://baike.baidu.com/item/%E8%81%9A%E7%B1%BB/593695" \t "https://baike.baidu.com/item/%E8%BD%AE%E5%BB%93%E7%B3%BB%E6%95%B0/_blank)效果好坏的一种评价方式。最早由 Peter J. Rousseeuw 在 1986 提出。它结合内聚度和分离度两种因素。可以用来在相同原始数据的基础上用来评价不同算法、或者算法不同运行方式对聚类结果所产生的影响。

对于本实验来说，k-means算法将待分类数据分为了 k 个簇 。对于簇中的向量i，计算其轮廓系数公式如下：



***a(i)*** 表示向量i到同一簇内其他点不相似程度的平均值，计算方式如下： 

***b(i)*** 表示向量i到其他簇的平均不相似程度的最小值，计算方式如下： 

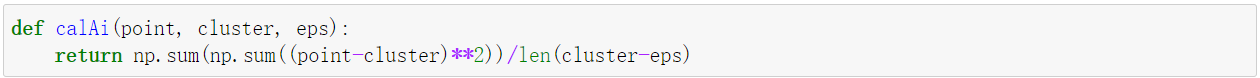
所以原***S(i)*** 可表示为：

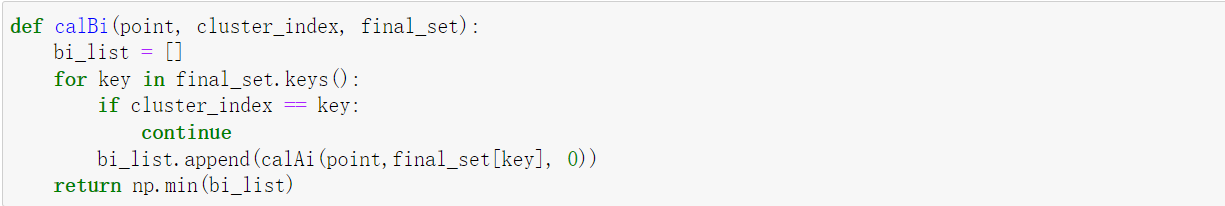


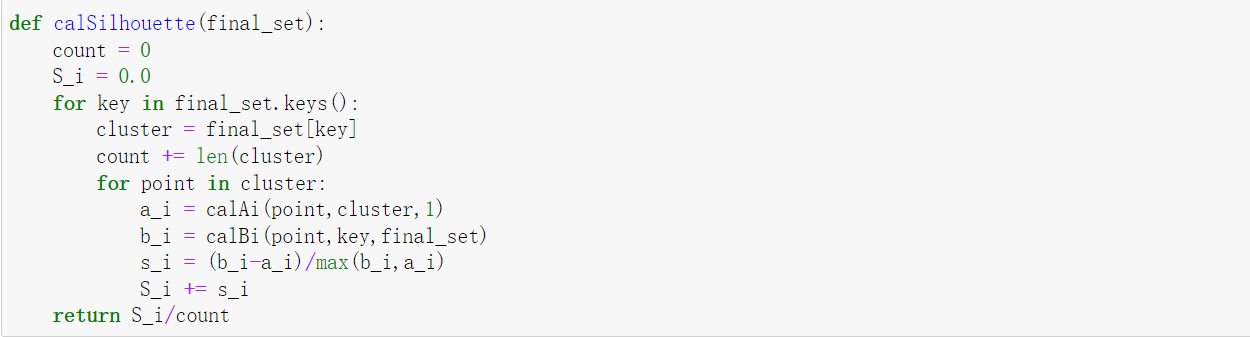
可见轮廓系数的值是介于 [-1,1] ，越趋近于1代表内聚度和分离度都相对较优。

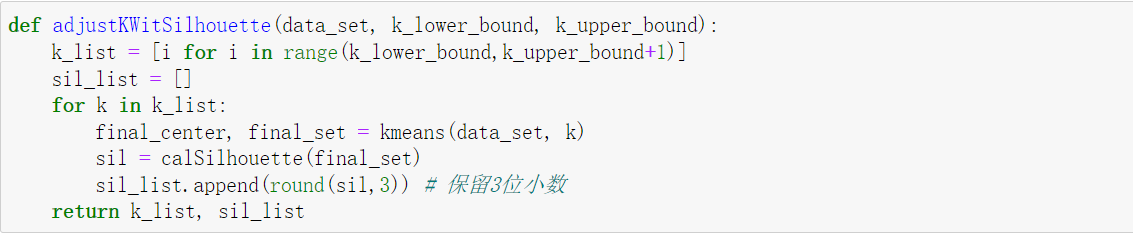
当***a(i)*** < ***b(i)***时，即类内的距离小于类间距离，则聚类结果更紧凑。S的值会趋近于1。越趋近于1代表轮廓越明显。相反，当***a(i)*** > ***b(i)***时，类内的距离大于类间距离，说明聚类的结果很松散。S的值会趋近于-1，越趋近于-1则聚类的效果越差。

轮廓系数能够用来衡量聚类好坏。在此基础上采用网格搜索，便能解决K值需要人为指定的缺点，具体操作为：枚举K从2到n(一般大于10)，在不同的K值下聚类并计算聚类结果的轮廓系数。选择最大的轮廓系数，该系数对应的K即为最合理的K。代码实现如下：



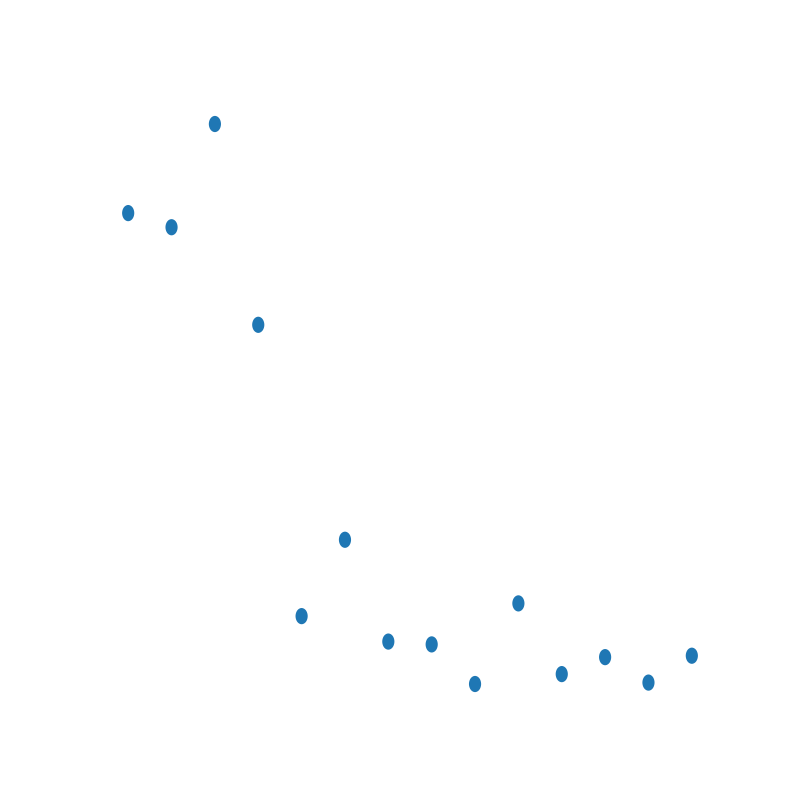








结果如下：



从图中可以看到，当簇数为4时，轮廓系数取值最大，故最佳簇数是4。

1. **k-means++**

由于k-means算法的初始质心是随机选择的，所以可能存在两初始质心过于接近从而导致k-means算法无法收敛到全局最小值的情况，并且这种情况的出现的概率不小。如下：



为了解决该问题，改进的k-means算法，即k-means++算法被提出。其主要解决初始化质心“碰撞”问题。算法描述如下：

· 在数据集中随机选择一个样本点作为第一个初始化的聚类中心。

· 计算样本中的每一个样本点与已经初始化的聚类中心之间的距离，并选择其中最短的距离，记为D(x)。

· 计算每个样本点被选为下一个聚类中心的概率P(x)，最后选择最大概率值所对应的样本点作为下一个簇中心。



对k个初始化的聚类中心，利用k-means算法计算最终的聚类中心。

不难发现，k-means++算法与k-means算法最本质的区别是在k个聚类中心的初始化过程，其他完全相同。于是代码实现方面，只需要修改选取初始化质心的函数即可，如下：



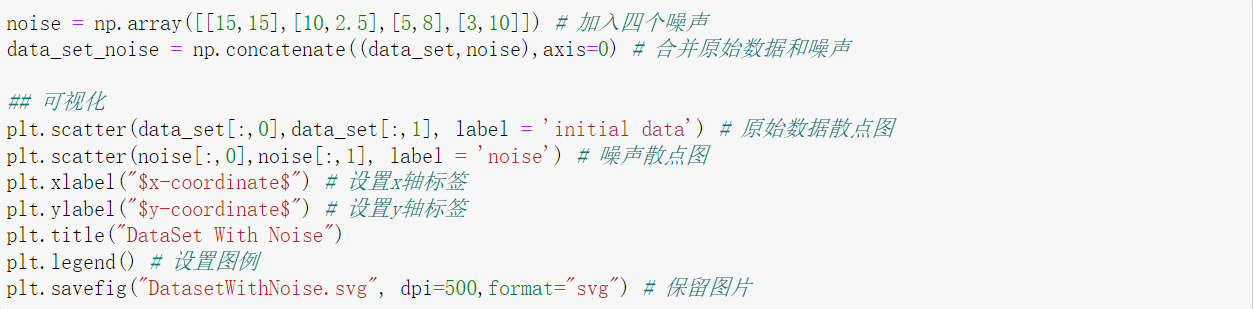
为了对比两个算法的性能，本小节在k=4的情况下，使用两种算法分别聚类10次并计算轮廓比。将计算结果采用折线图的形式呈现便于观察，结果如下：



可以看到，在进行的10次聚类中，k-means算法轮廓系数波动大，并且仅有3次聚类结果的轮廓系数与k-means++算法相同，其余均低于k-means++的轮廓系数。而k-means++轮廓系数稳定，无任何波动。由此不难说明k-means++算法较k-means算法拥有更优的聚类能力。

1. **k-medoids**

本实验数据集样本数量较少(样本数量仅为400)，并且样本中无过多离群点和噪声，数据集比较纯净。所以k-means++算法能取得不错的分类效果，但是仔细分析算法流程不难发现，虽然k-means++解决了质心初始化问题，但是其剩余步骤同k-means保持一致，所以如果样本中存在异常点，则聚类结果将会产生极大的偏差。为了更好地说明该问题，下面在将在原始数据集上加入若干噪声(离群点)。代码如下：



可视化：



使用k-means算法对带有噪声的数据集聚类，结果如下：



使用k-means++算法对带有噪声的数据集聚类，结果如下：



不难发现，当数据集中存在噪声或者离群点时，聚类中心将会被这些异常点严重带偏，降低聚类质量。本小节介绍k-medoids[聚类算法](https://so.csdn.net/so/search?q=%E8%81%9A%E7%B1%BB%E7%AE%97%E6%B3%95&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/xiaozhu_1024/article/details/_blank)，该算法能有效削弱异常点的影响。算法流程如下：

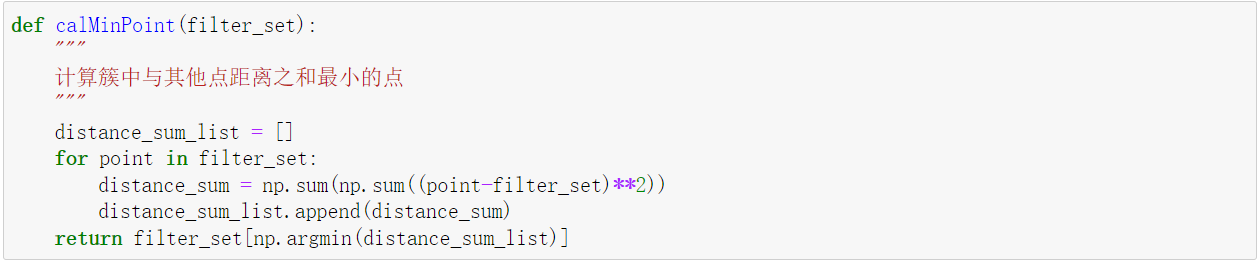
1、任意选取 k 个点作为 medoids(中心)。

2、按照与medoids最近的原则，将剩余点分配到当前最佳的medoids代表的类中。

3、在每一类中，计算每个成员点对应的准则函数，选取准则函数最小时对应的点作为新的 medoids。

4、重复2-3的过程，直到所有的 medoids 点不再发生变化，或已达到设定的最大迭代次数。

其中准则函数为，一类中，某个成员点和其他成员点的距离之和。k-medoids(中心点)算法不选用平均值，转而采用簇中位置最中心的对象，即中心点(medoids)作为参照点，算法步骤与k-means 类似，其实质上是对 k-means算法的改进和优化。关键代码如下(未展示的函数均与k-means保持一致)：





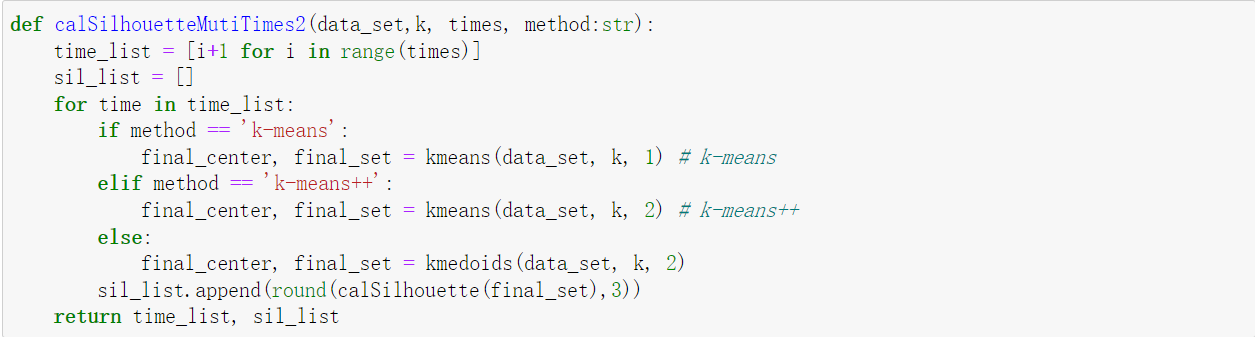


使用k-medoids算法对带有噪声的数据集聚类，结果如下：



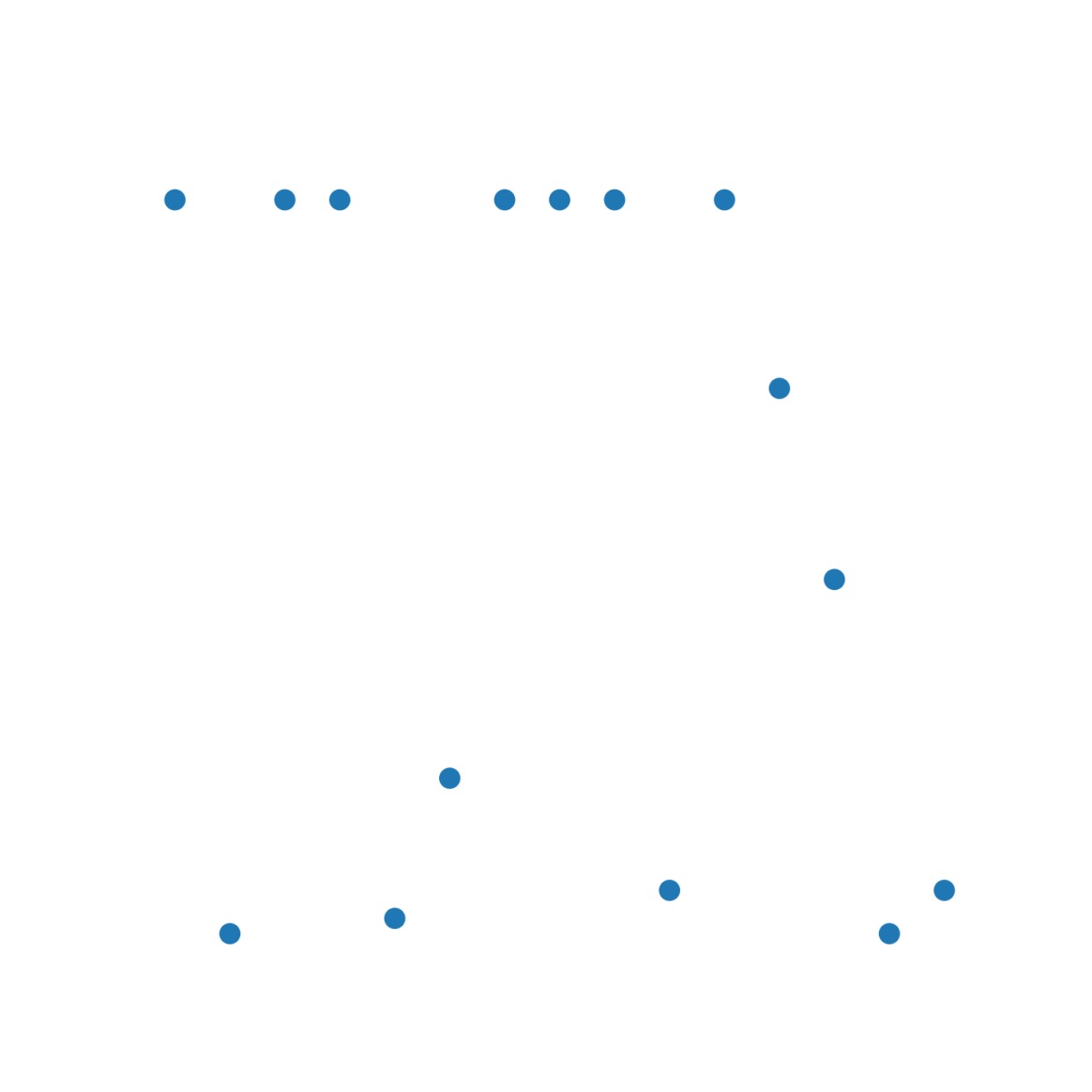
对比发现，k-medoids算法有效地减少了噪声或者离群点带来的干扰，提升了聚类质量。但值得注意一点的是，k-medoids算法对于小数据集工作得很好, 但其无法很好地用于大数据集，因为其更新质心的时间开销比k-means算法大得多。

当然，上述聚类结果的好坏是依据观察评判，为了更加科学地分析对比k-medoids算法与k-means算法以及k-means++算法，下面使用三种算法对带有噪声的数据集聚类分别进行15次聚类，计算聚类结果的轮廓系数并进行分析。代码如下：





运行结果：



可以看到，k-means算法的轮廓曲线波动极大，这与其初始化质心的方式有关。如果随机的质心恰好为离群点，那么此次的聚类效果便不佳，轮廓系数低。这也充分说明了k-means算法极其依赖初始点的选取。k-means++算法的轮廓曲线与k-medoids算法的轮廓曲线稳定，但是后者的轮廓系数普遍比前者高。这是因为虽然k-means++算法解决了如何合理初始化质心的问题，但是同样无法抵抗噪声或者离群点带来的干扰，噪声或者离群点的存在仍会使质心偏移从而降低轮廓系数。而k-medoids算法通过不断替换中心点的贪心策略削弱了异常值带来的影响，使其获得了稳定的轮廓系数。

**六、实验总结**

实验首先依据k-means算法流程编写出python代码，并在实验的数据集上进行初步验证。通过将聚类结果可视化分析验证了实验代码能够完成基本的聚类任务。接下来的实验致力于解决k-means算法的三个不足：

1. k值需要预先给定;
2. 算法极其依赖初始质心的选取;
3. 算法噪声或者离群点异常敏感;

为了解决问题1，实验引入轮廓系数(Silhouette Coefficient)来衡量聚类的好坏。轮廓系数是内聚度(反映一个样本点与类内元素的紧密程度)与分离度(反映一个样本点与类内元素的紧密程度)的综合体现。有了衡量标准，便能通过调整k值大小进行网格搜索从而获得最佳聚类簇数。对于实验数据集来说，最佳聚类簇数为4。

为了解决问题2，实验采用全新的初始化质心的方法，也即k-means++算法。该算法的核心是选取的下一个质心尽量远离已选质心。算法思路虽然简单，但却有效解决了k-means算法随机初始化质心容易产生“碰撞”的问题。实验通过k-means++算法与k-means算法多次聚类结果的对比验证了这一点。

为了解决问题3，实验采用全新的更新质心的方法，即k-medoids算法。该算法通过反复替换质心以达到削弱异常值带来的质心偏移。为了验证该算法的优越性，实验在原始数据集上加入4个离群点，并对比k-means算法、k-means++算法以及k-medoids算法聚类结果的轮廓系数，结果表明k-medoids算法较k-means算法或者k-means++算法具有更强的鲁棒性。不过也需要注意，该算法的是将开销比k-means大。

同时，本次实验也让我意识到一个缺点的解决往往伴随着另一个缺点的诞生。k-means++算法虽然成功解决了k-means算法质心初始化的问题，但同时也让其对异常值更加敏感；k-medoids算法有较强的鲁棒性，但却带来了额外的时间开销。三种算法没有好坏，拥有各自特定的使用场合，需要根据实际情况选择合适的算法。至此，实验完成。

由于实验代码较长，实验报告仅展示核心代码。完整代码已放至

[ChenWeilinx/machine\_learning: 机器学习与数据挖掘实验 (github.com)](https://github.com/ChenWeilinx/machine_learning)

**七、参考资料**

《数据挖掘原理与实践 蒋盛益，李霞，郑琪编著》

David Arthur and Sergei Vassilvitskii *k-means++: The Advantages of Careful Seeding*