# 密度峰值聚类算法实验报告

**学号：PB21000224　　姓名：陈鸿绪　　日期：12.7.2023**

## 一. 实验目的：

在课程中，我们学习了 k-means 聚类和 DBSCAN 聚类，密度峰值聚类结合了这两种算法，具有以下优点：在处理高维非线性数据时具有更好的稳健性，对异常值和噪声数据具有一定的鲁棒性。实现简单，所需参数少。能处理非凸数据，聚类效果良好等优点，密度峰值聚类算法为很多现实问题提供了新的解决思路，通过这次实验旨在理解密度峰值聚类的算法实现。

## 二. 密度峰值聚类算法原理：

密度峰值聚类算法（Density Peak Clustering Algorithm）是一种无监督的聚类算法，它能够自动发现数据中的密度峰值点，并根据这些峰值点将数据进行聚类。该算法由 Alex Rodriguez 和 Alessandro Laio 于 2014 年提出，其原理相对简单但非常有效。在密度峰值聚类算法中，局部密度的计算方法如下：

1. 计算数据点 i 和 j 之间的距离 dist(i, j)。
2. 定义一个核函数，用来衡量 dist(i, j)对数据点 i 和 j 之间距离的影响，个人理解类似于加权处理局部密度。

该算法的核心是基于两个假设：

1. 簇心的密度比其周围的点高。
2. 簇心距离其他密度大的数据点相对更远。

为了达到这个目的，密度峰值聚类算法计算了以下几个量：

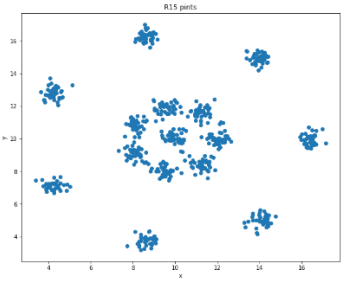
1. 数据点之间的距离。这个距离可以使用多种距离度量方式，如欧氏距离。
2. 每个数据点的局部密度 $\rho$。局部密度的度量方式也可以有多种选择，如直接对每个点周围一定范围内的点计数，或者使用高斯核函数等方法。

3. 相对距离 δ。点 i 的 δ 表示 i 到局部密度大于 i 且距离 i 最近点的距离。

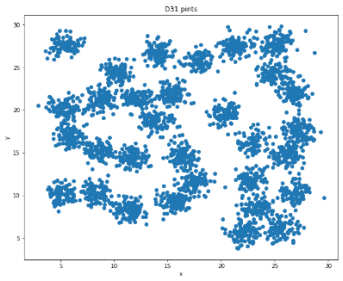
有了以上的量，就可以做出一个以 ρ 为横轴， δ 为纵轴的图（即决策图，decision graph）。在决策图中，簇心的密度比其周围的点高，所以 ρ 大；簇心距离其他密度大的数据点相对更远，所以 δ 大。因此被看做簇心的数据点都会被选取出来。在实验中，我们采取采取 ρ 与 δ 相乘作为指标，相乘结果越大代表越能选取为簇心。

## 三. 实验步骤：

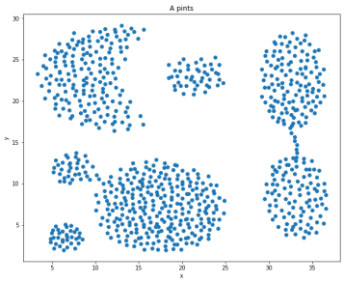
1. 逐行读入 D31.txt、R15.txt、Aggregation.txt，并分别绘出其二维图像，初步判断其包含的 cluster 数目，可以看出来三个数据集分别大致有 15、31、7 个簇，具体图像如下：（该部分对应于 Load the dataset and preprocess the data 底下的代码块）




R15.txt




D31.txt




Aggregation.txt

2. 构建没有采用核函数的算法 DPC 函数

a. 计算点点之间的距离存储在距离矩阵中，同时计算每一个点的局部密度，这里采取论文中描述小于某个阈值的距离中所有点的个数作为局部密度。

b. 按照局部密度对数据点的索引进行降序排序，按照论文中的方法计算出任意点 A 对应的 δ 和其所对应的密度较大的点 B（记录在 pre_point 列表中，为了记作 B 为 A 的前驱点）。

c. 将每一个点相乘 ρ 、 δ 得到 ρ δ，将其从大到小排序，取前若干个点作为密度中心。

d. 对不在密度中心的点 A 不断递归寻找其前驱结点（recurse 函数即为辅助递归函数，用于这里的递归搜索），直到前驱结点是密度中心之一为止。将 A 归属于该密度中心所对应的簇。

c. 根据需求决定是否鉴别出噪点，方法就是先对一个选取的簇划分出边界，然后选取边界上最大局部密度，将该簇中密度小于边界上最大局部密度的点视为噪点，并将其剔除出分类簇。

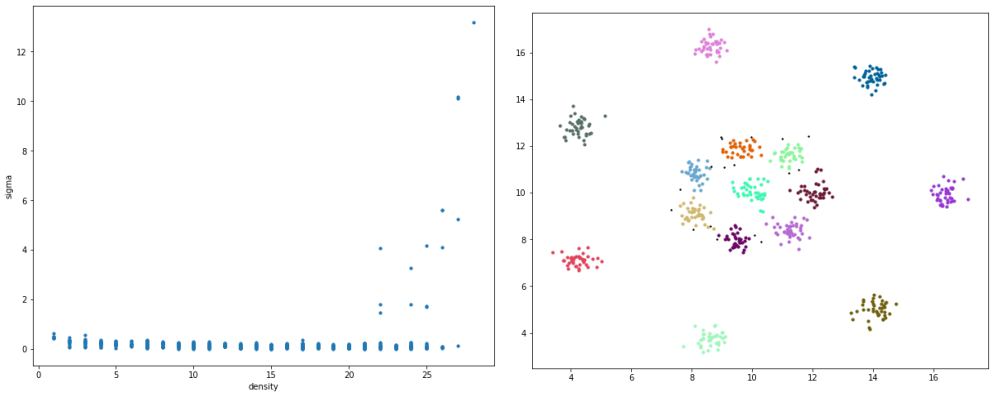
3. 构建采用高斯核函数的 DPC 函数，唯一与之前的不同之处在于计算局部密度时采取了高斯核函数，所以需要加入带宽和协方差矩阵的参数，其他部分保持不变。但这个 DPC_gauss_kernel 函数并没有实现剔除噪点的功能。
4. 设置参数分别用 DPC 函数和 DPC_gauss_kernel 函数对 R15.txt 的数据进行训练，画出决策图，按照 davies_bouldin_score 计算出内部平均指标，不断调整参数直到获得较优的 davies_bouldin_score 为止。最后得聚类结果图像。
5. D31txt、Aggregation.txt 进行类似 4 操作，得决策图、聚类结果图像。

## 四． 实验结果与分析：

注意实验结果中纯黑色数据点代表数据噪点，对于 R15.txt 与 Aggregation.txt 的普通 DCP 聚类考虑了数据噪点的存在，且源代码文件中的图像清晰度更高。

1. 对于 R15.txt，决策图和聚类结果如下图所示

普通 DPC 聚类：davies_bouldin_score=0.29688　噪点占比：0.001667



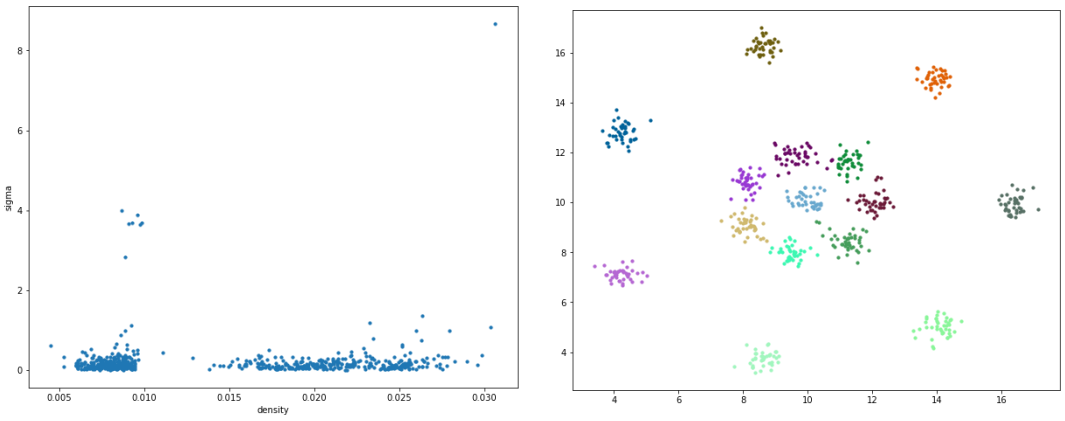
参数：DPC(0.4,R15_points,k_number,1)，类别数：15

即，距离阈值为 0.4，且考虑数据噪点的存在。

分析：由决策图中，不难发现密度和 δ 都相对较大的总共有 15 个点，这是采取 15 个点作为簇中心的理由，同时不难发现聚类效果也是不错的，其中的噪点可以发现大致处于簇的边界处，且占比也不大，说明这是一个比较理想的数据集。

若采用高斯核函数的 DPC 聚类：davies_bouldin_score=0.32779

（注意这里高斯核函数计算出来的密度有常数倍的缩放，但是并不影响结果）



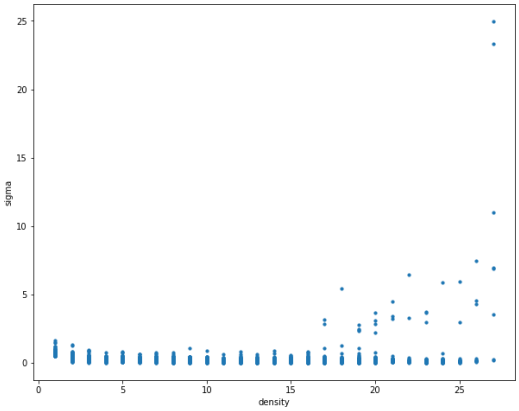

参数：DPC_gauss_kernel(0.05,R15_points,k_number,600) 类别数：15

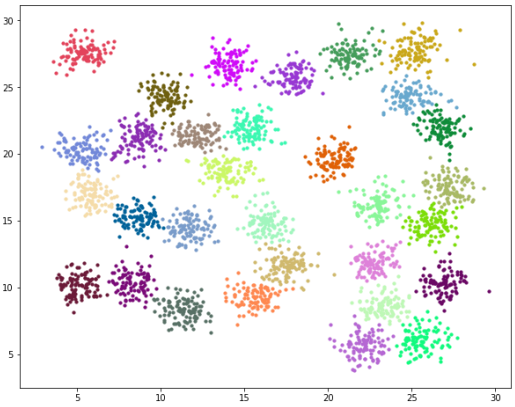
即，高斯分布的带宽为 0.05，且其协方差矩阵的方差参数取 600。

分析：用高斯核函数聚类的效果并没有原始的方法好，这可以从内部评价指标大小可以看出（但也许是调参没有细调的问题）。但是仍然划分出来 15 个簇。

2. 对于 D31.txt，决策图和聚类结果如下图所示

普通 DPC 聚类：davies_bouldin_score=0.55180





参数： DPC(0.49,D31_points,k_number,0) 类别数：31

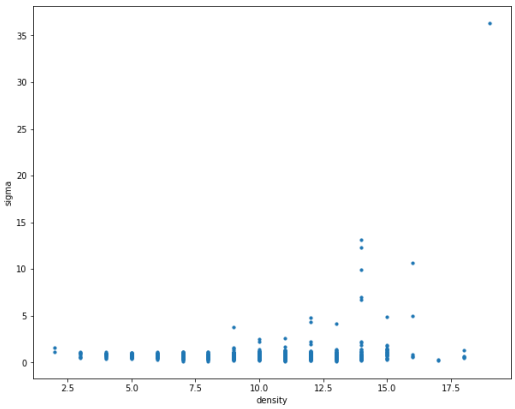
即，距离阈值为 0.49，且不考虑数据噪点的存在。

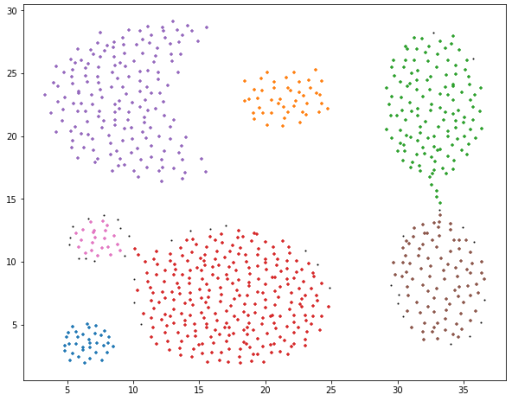
分析：通过直观感受，可以发现划分效果不错，但是在实验中我尝试着将噪点也鉴别出来，通过调试发现如果想让噪点比例较小，会使得整体分类评估指标的下降，所以最终决定不鉴别噪点进行处理。(如果想要噪点尽量减少，那么距离阈值也需要减少，这会使得局部密度的估计过于离散，所以聚类效果不佳)。

多次调参并没有将高斯核函数的 DPC 算法的结果调到一个较好的分类，而且训练花费时间较长，所以这里就舍去采取高斯核函数的情况。(数据过多，虽然高斯核函数的 DPC 时间复杂度实际上和普通 DPC 的量级一致，但是对于系数而言高斯核函数的 DPC 会比普通 DPC 要大很多，所以要慢一些)。

3. 对于 Aggregation.txt，决策图和聚类结果如下图所示

普通 DPC 聚类：davies_bouldin_score=0.48893　　噪点占比：0.000323



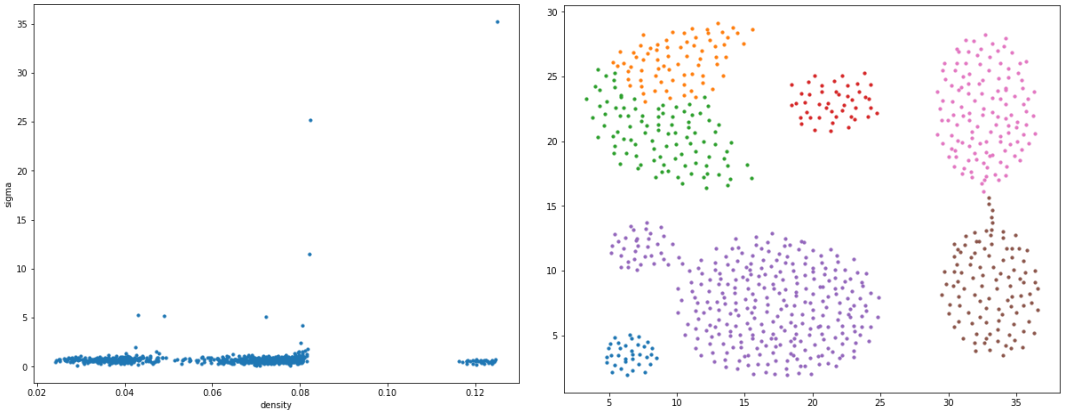


参数：DPC(1.39,A_points,k_number,1) 类别数：7

即，距离阈值为 1.39，且考虑数据噪点的存在。

若采用高斯核函数的 DPC 聚类：davies_bouldin_score=0.68625

（注意这里高斯核函数计算出来的密度有常数倍的缩放，但是并不影响结果）




参数：DPC_gauss_kernel(1,A_points,k_number,900) 类别数：7

即，高斯分布的带宽为 1，且其协方差矩阵的方差参数取 900。

分析：对于该参数的高斯核函数并没有完美的将其聚类，我们考察左上角的点集，可以发现理应划分成一类的被划分成了两类（绿色与橙色），左下的理应划分成两类的聚类成紫色一类（大概率是我调参没有将其调到合适的范围）。而普通 DPC 并没有发生这样的情况。

4. 综述，普通的 DPC 效果已经不错了，基于高斯核函数的 DPC 也许需要进一步优化调参才能达到更优的聚类效果。