

Clustering by fast search and find of density peaks

刘华坤

2017.12.03

Clustering by fast search and find of density peaks

核心思想

步骤

1. 读取点
2. 计算点与点之间的距离
3. 计算截断距离 d_c
二分法计算 d_c
测试
4. 计算局部密度 ρ_i
方法一：截断核函数
方法二：高斯核函数
分析上述两种方法
测试
5. 计算 δ_i
方法一：原论文方法。
方法二：改进后的方法
测试
6. 绘制 $\rho - \delta$ 或 $\gamma_i = \rho_i * \delta_i$ 图像
7. 确定聚类中心（难点二）
测试
测试结果
8. 分配点（聚类）
方法一：直接将数据点与他距离最近的聚类中心点分配到一类中
方法二：通过链式寻找最近高密度中心点
测试
9. 计算 halo
10. 绘制结果

总结

核心思想

通过局部密度和距离找出聚类中心点

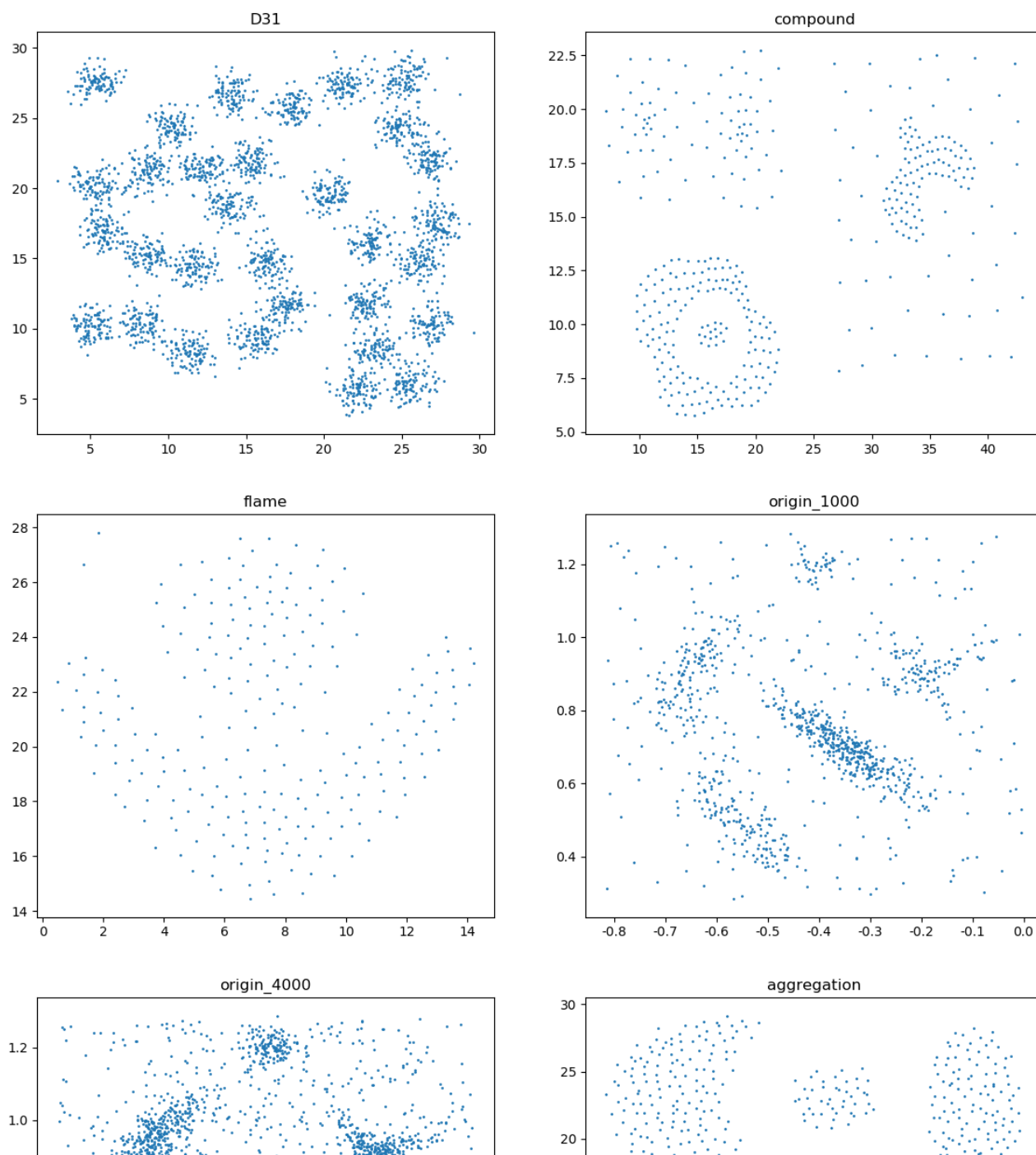
启发式聚类中心点的搜索。

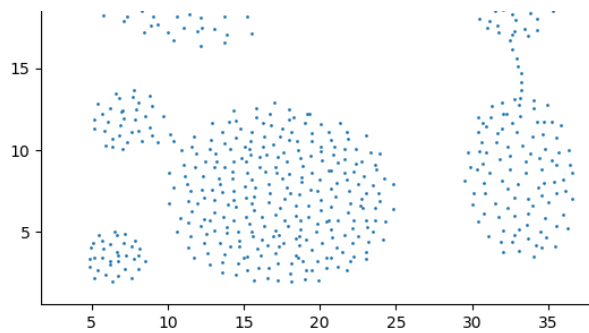
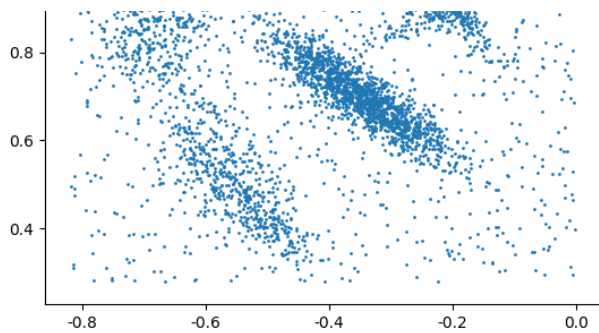
步骤

1. 读取点

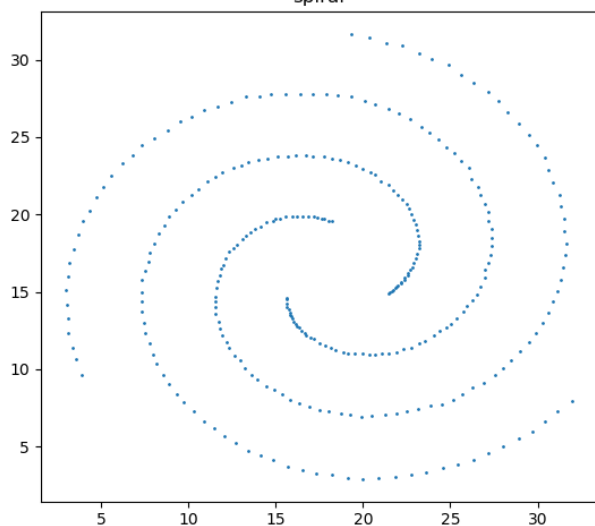
这里选取了 10 个 2维 数据集

如下图

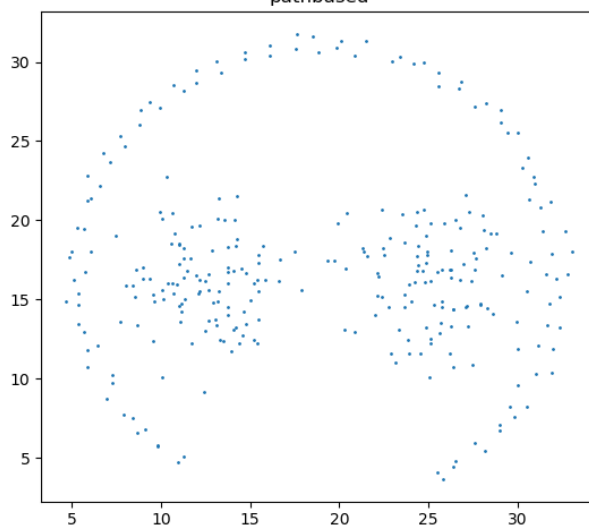




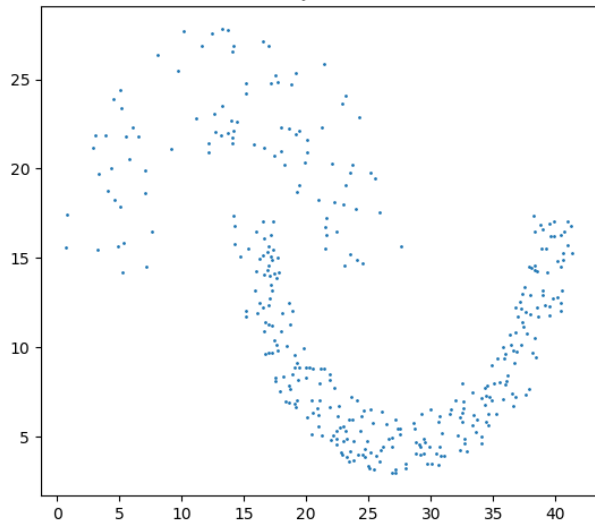
spiral



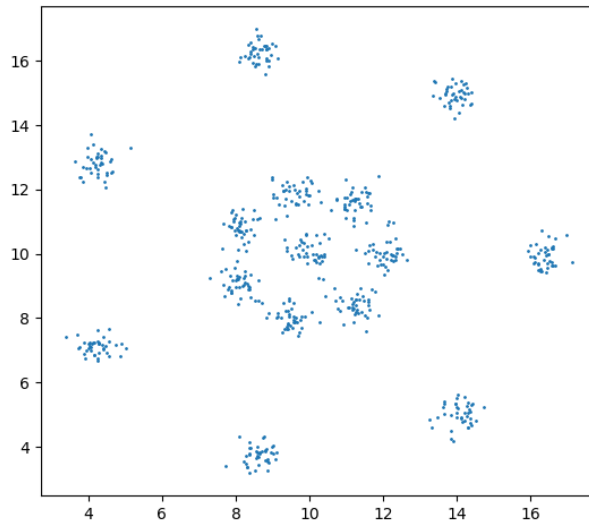
pathbased



jain



R15



2. 计算点与点之间的距离

这里采用了 欧几里德距离公式 计算两点间的距离。

暂时不对其他距离公式进行讨论。

3. 计算截断距离 d_c

介绍求解 d_c 的方式

论文中提到， d_c 理论的取值应使

$$avg(neighbors) = (1\% \sim 2\%) * N$$

N : 数据集点的总数

所以我们的方法应尽量满足此条件。

二分法计算 d_c

1. 初始化 d_c 和条件 $lower, upper$

$$d_c = (min_dis + max_dis) / 2$$

min_dis : 最小距离

max_dis : 最大距离

$$lower = 1\%$$

$$upper = 2\%$$

这里，我们设置百分比的计算方法为距离矩阵（上三角矩阵）小于 d_c 的数目与距离矩阵（上三角矩阵）总大小之比

$$percent = \frac{\sum_{dis_{ij} < d_c} 1}{(N-1)^2 / 2}$$

2. 二分法计算 d_c ，当在所给百分比范围内时即选为满足条件的 d_c

$$d_c = \begin{cases} \frac{(min_dis + d_c)}{2} & \text{if } percent > upper \\ \frac{(d_c + max_dis)}{2} & \text{if } percent < lower \\ correct\ d_c & \text{others} \end{cases}$$

此方法较精确的计算了 d_c ，求出来的结果满足论文中的条件。

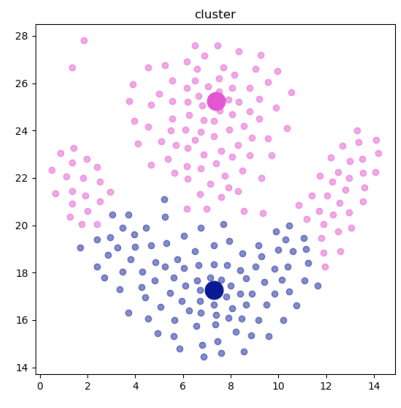
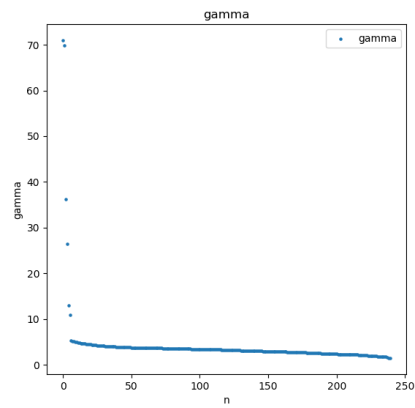
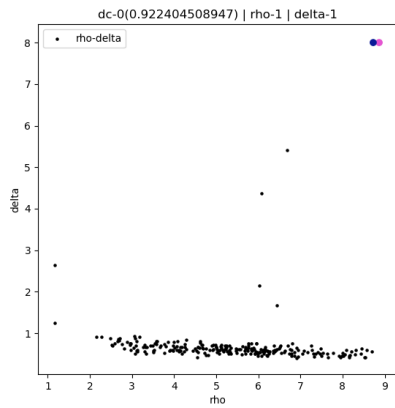
对于不同的数据集， d_c 的最佳取值范围都有所不同，具体问题需要具体测试。

测试

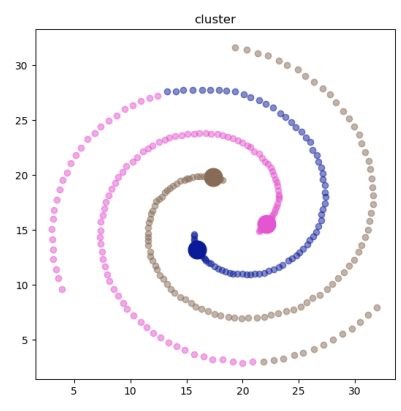
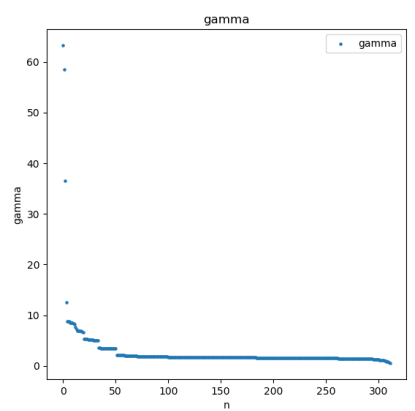
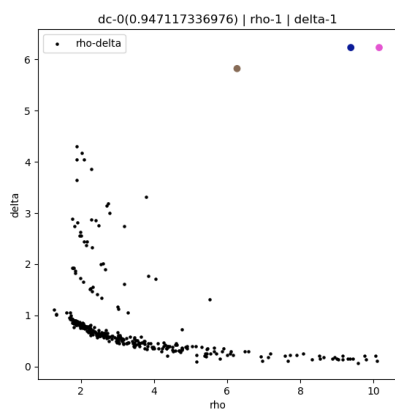
用 spiral 和 flame 两个数据集测试不同 d_c 值的影响。

ρ 的计算采用高斯核（见步骤4）， δ 的计算采用方法二（见步骤5）

- $lower = 1\%$ $upper = 2\%$
- flame 数据集

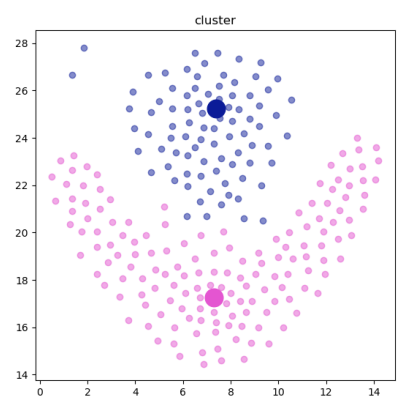
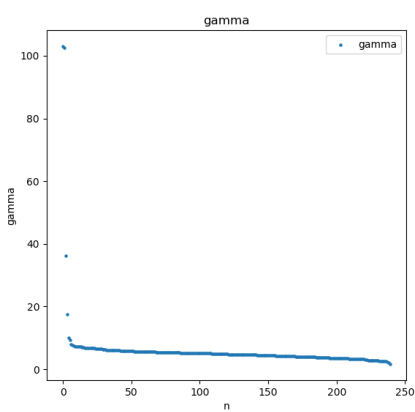
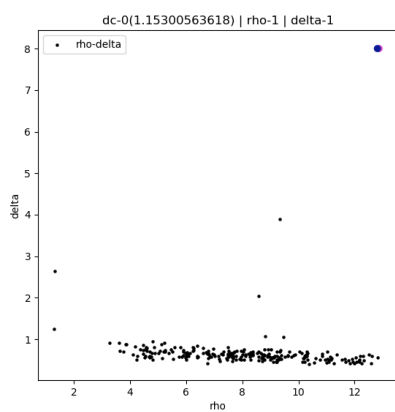


■ spiral 数据集

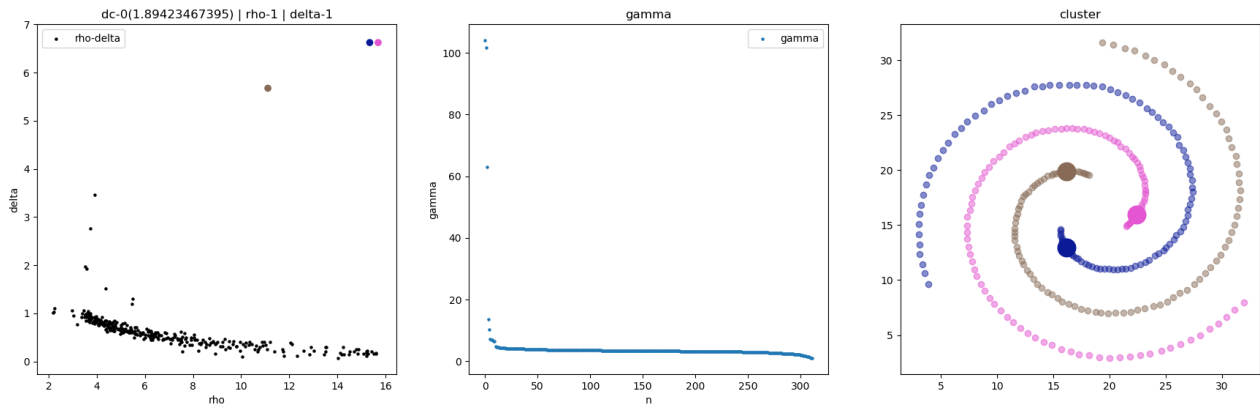


■ *lower = 3% upper = 4%*

■ flame 数据集



■ spiral 数据集



根据测试可知， d_c 暂时无法通过自动计算得到一个合理的值，而对于每个数据集都需要调试以达到一个较佳的值。

而根据原文可知， d_c 只要在一个合理的范围内（过大或过小均不行），其对中心点的选取影响不大（中心点不变），但是对于聚类的性能有不小的影响（因为改变了 d_c 即改变了 ρ ，对 δ 和 *assign* 等各种数据有所影响）。

这里提出该算法的四个相互影响且起决定性作用的因素，即

- d_c ：截断距离
- ρ ：局部密度
- δ ：距高密度点的最小距离
- 分配方式：聚类方式

下面会对其他因素一一进行讨论。

4. 计算局部密度 ρ_i

ρ ：局部密度值

论文中提出了两种计算 ρ 的方法

方法一：截断核函数

计算和 i 的距离小于 d_c 的点的数量。

$$\rho_i = \sum_j \chi(d_{ij} - d_c)$$

$$\chi(x) = \begin{cases} 1 & x < 0 \\ 0 & otherwise \end{cases}$$

方法二：高斯核函数

$$\rho_i = \sum_j \exp(-(\frac{d_{ij}}{d_c})^2)$$

当 j 远离 i 时函数取值很小（接近0），靠近 i 时取值很大（接近1）

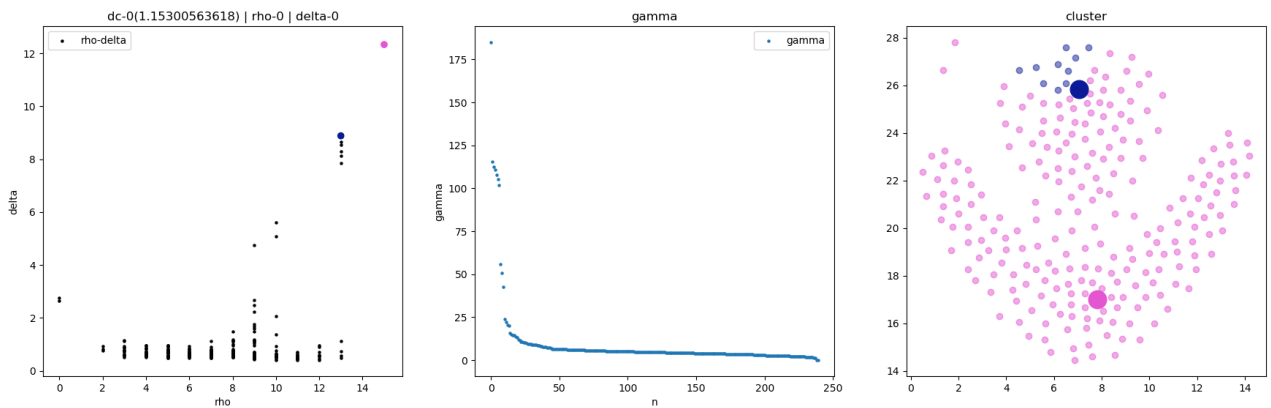
分析上述两种方法

- 第一种方法较为直接，其结果是离散的，实际使用时会发现较容易出现相同局部密度的点。因为没有考虑到邻居点（ $d_{ij} < d_c$ ， j 即为 i 的邻居点）和自身距离的因素。
- 第二种方法采用了高斯核函数，连续递减的函数可以将邻居点和自身的距离因素考虑进去，从而可以降低出现相同局部密度的点的概率。

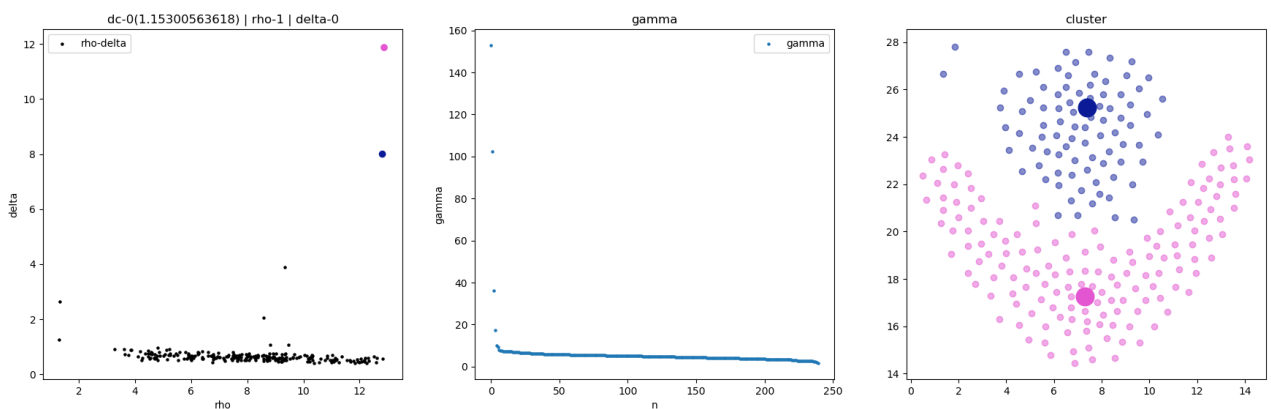
测试

这里用 flame 数据集(分布较均匀,aggregation 也可以) 进行测试

方法一



方法二



可以观察到在 $\rho \sim \delta$ 表中，方法一的点较聚集，很难分辨出集群中心。方法二很分散，效果较好。

综上，实现算法时我统一选择使用高斯核函数计算 ρ

5. 计算 δ_i

δ_i 用来衡量 i 和其他高密度的点的最小距离。

这里探讨两种 δ 的计算方法

方法一：原论文方法。

$$\delta_i = \begin{cases} \min_{j: \rho_j > \rho_i} (d_{ij}) & \text{if } \rho_i \neq \max(\rho) \\ \max_j (d_{ij}) & \text{if } \rho_i = \max(\rho) \end{cases}$$

这里会出现一个问题，如果 $\rho_i = \rho_j$ ，且 ρ_i and ρ_j 离得距离很近，则本应在一个聚类中的两个点，有可能会被分到两个聚类中，或成为两个聚类中心。针对这个问题，原文作者在[讨论](#)中提出了这么一种方法（方法二）。

方法二：改进后的方法

$$\text{order_rho_index} = \rho(\text{desc})_{\text{index}}$$

i, j 为 order_rho_index 的索引

$$m = \text{order_rho_index}[i]$$

$$n = \text{order_rho_index}[j]$$

$$\delta_i = \begin{cases} \min_{j: j < i} (d_{mn}) & i \geq 1 \\ \max(\delta) & i = 0 \end{cases}$$

这里，我们会发现当 $\rho_i = \rho_j$ 时，他们的索引会排在 order_rho_index 中相邻的两位，而由对 δ 的定义可得，此时，排在前面的首先会获取一个较大的 δ_i 值，而后者即使有较大的 ρ_j 值，其 δ_j 很可能为 $\min(d_{ji})$ ，而并不会获取一个较大的 δ 值。由此，可以有效的处理方法一中的问题，使的 i, j 中只会选择 i 为聚类中心（ ρ_j 较小）。

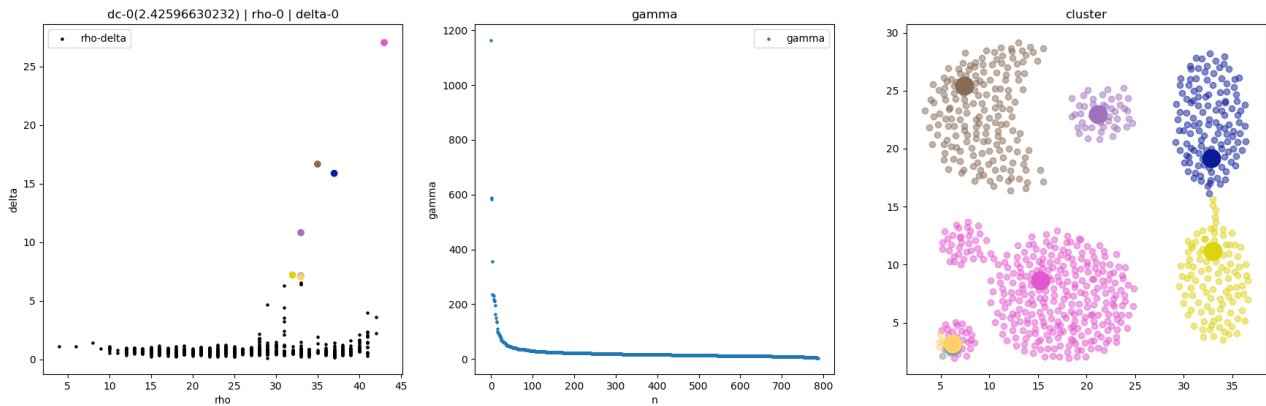
测试

我选用 aggregation 数据集进行测试

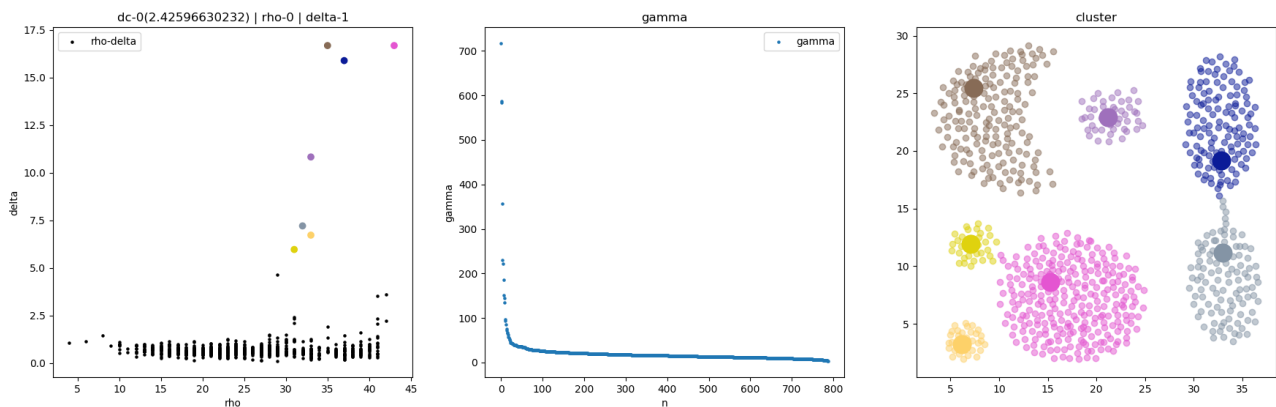
ρ 采用方法一（直接计算数量，不采用高斯核）

结果如下

- 方法一



方法二



因为我对中心点的选择使用的方法是：给定点的数量 N ，取 γ 值前 N 个，并且未对距离很小的两点进行筛选，所以方法一的结果导致左下角聚类中心点高度密集。

但由此也很好的表现出了方法二的效果。

综上，实现算法时我统一选择使用方法二。

6. 绘制 $\rho - \delta$ 或 $\gamma_i = \rho_i * \delta_i$ 图像

在论文中，作者在结尾提出了一种确定中心点数量的思路，即绘制 γ 图像

$$\gamma_i = \rho_i * \delta_i$$

但作者在讨论中也明确表达即使绘制出了图像，其距离中心点的数量，或 γ 图的聚类中心与非聚类中心的分隔线也很难自动确定，只能通过人们的观察和主观判断才能确定。

7. 确定聚类中心 (难点二)

通过 $\rho - \delta$ 决策图或 γ 图确定聚类中心。

这里是原论文中争议较大的部分，因为按照原文的意思，聚类中心可以自动的寻找到，尽管论文作者在讨论中表达出他所指的“自动”是指自动整理数据并生成 $\rho \sim \delta$ 的图，然后人们可以通过视觉人工地来确定中心点，但是此处仍有诸多问题，如取点的标准，这里我们探讨了如下的问题。

- $\rho - \delta$ 决策图中，如何确定聚类中心点？

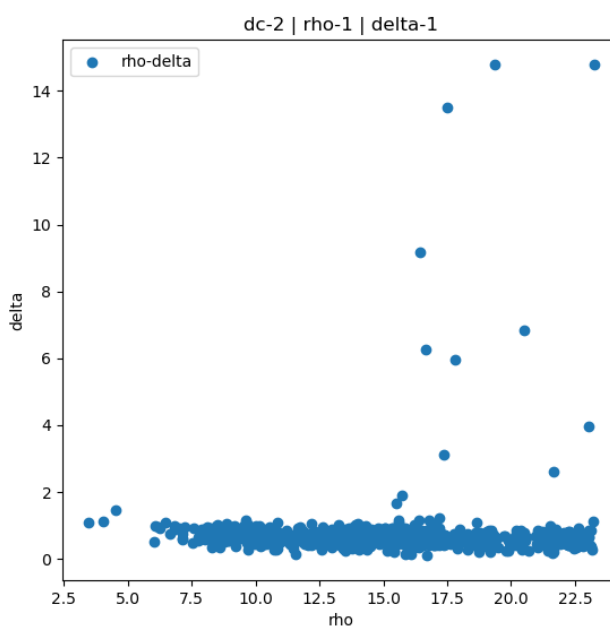
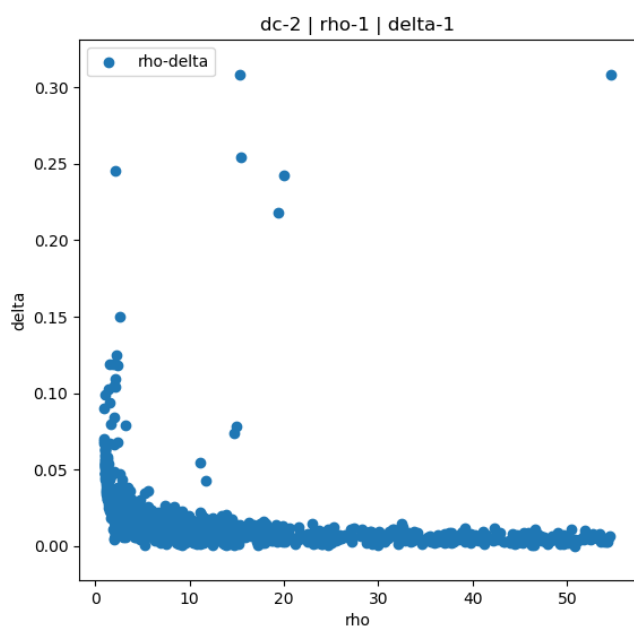
我们这里不再讨论其他因素产生的影响，如：相近且相似点的取舍。而只关注分散的中心点的选取。

所以我们 d_c 取为经调试后的较佳值， ρ 的计算采用方法二（高斯核函数）， δ 的计算采用方法二（改进后的方法）

数据集选取两种数据集进行讨论。（origin-1000，aggregation）

测试

- 下左图为 origin-1000 数据集的决策图，根据原文中提供的思想，发现在 $\rho > 10, \delta > 0.2$ 有5个点，其 δ 和 ρ 都较大，故可选取为聚类中心点。



- 但是观察到上右图（aggregation数据集），每个人对其取点的范围定义可能会有所不同，从而导致聚类中心的点不确定。

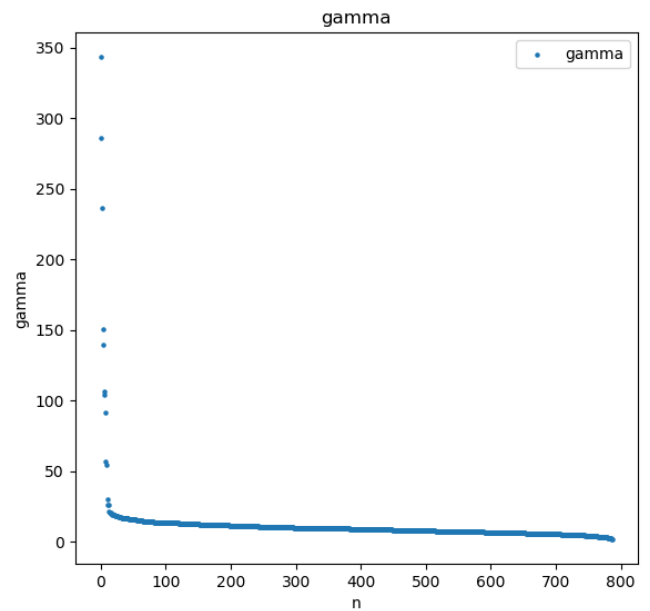
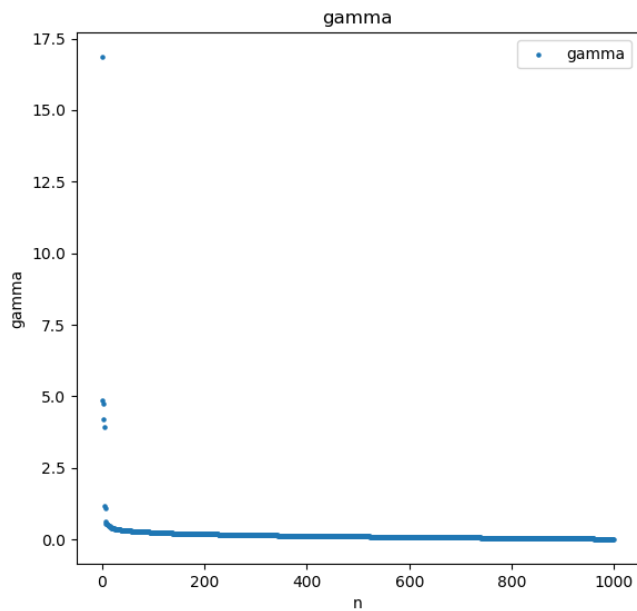
所以很容易发现，生成决策图后，对聚类中心的选取需要明显的人工干预（即给定 $\rho_{threshold}$ 和 $\delta_{threshold}$ ）。

- 原文中的最后，作者给出了一种获取中心点数量的方法。

$$\gamma_i = \rho_i * \delta_i$$

首先通过 (12) 获得 γ ，然后对 γ 进行降序排序，画出 $\gamma - n$ (索引) 图。如下：

(左：origin-1000，右：aggregation)



- 这里引申出来第二个问题，即使给出了 γ 图，对于人工判断的依赖依然很大，即很难找到一个标准，指定前 N 个为聚类中心点。

所以确定中心点这里有两种方法

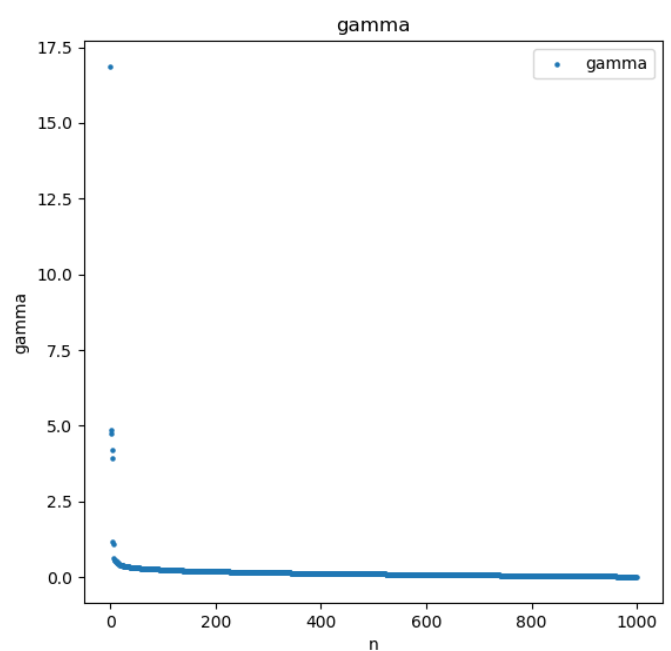
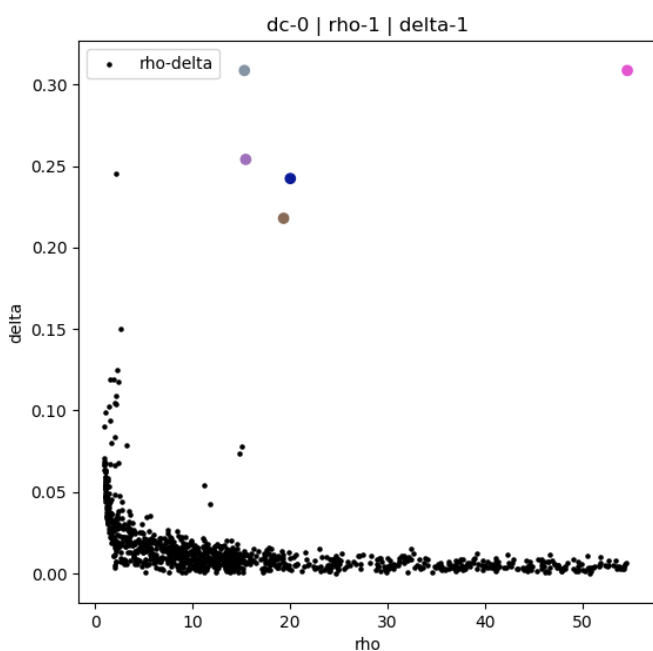
1. 给定： $\rho_{threshold}$ 和 $\delta_{threshold}$

$$i : \rho_i > \rho_{threshold} \text{ and } \delta_i > \delta_{threshold}$$

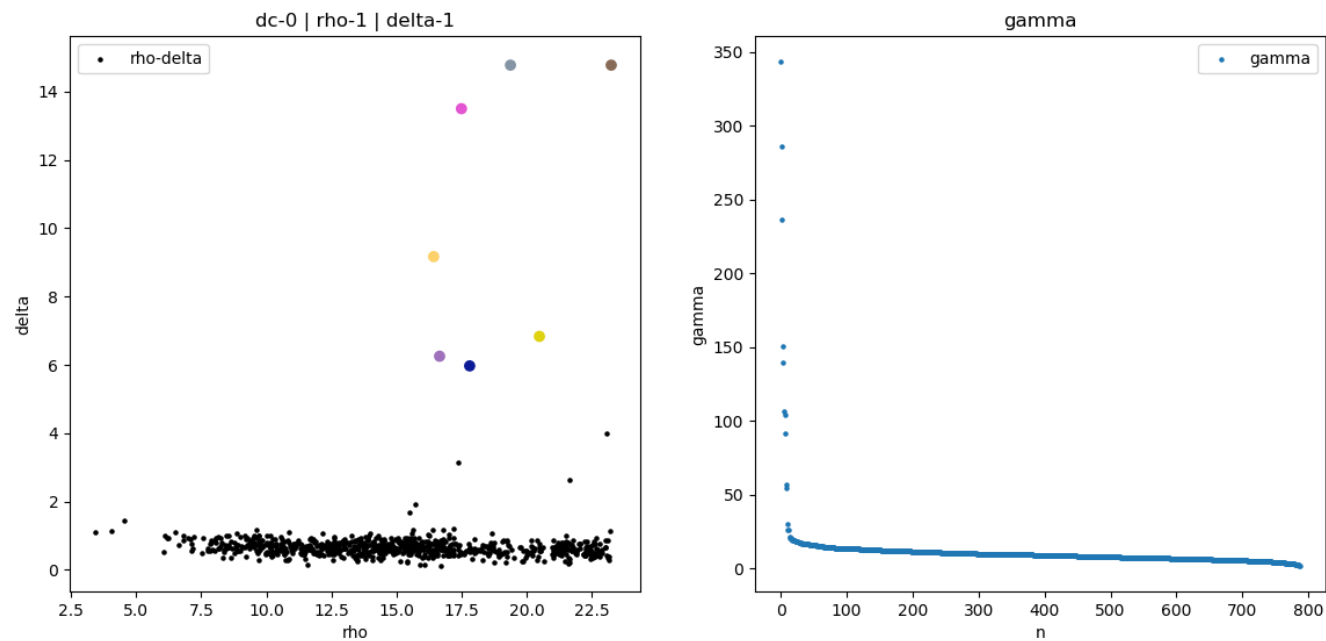
2. 给定 N ，取 γ_{desc_order} 前 N 个点（或给定 $\gamma_{threshold}$ ）

测试结果

- origin-1000 数据集



■ aggregation 数据集



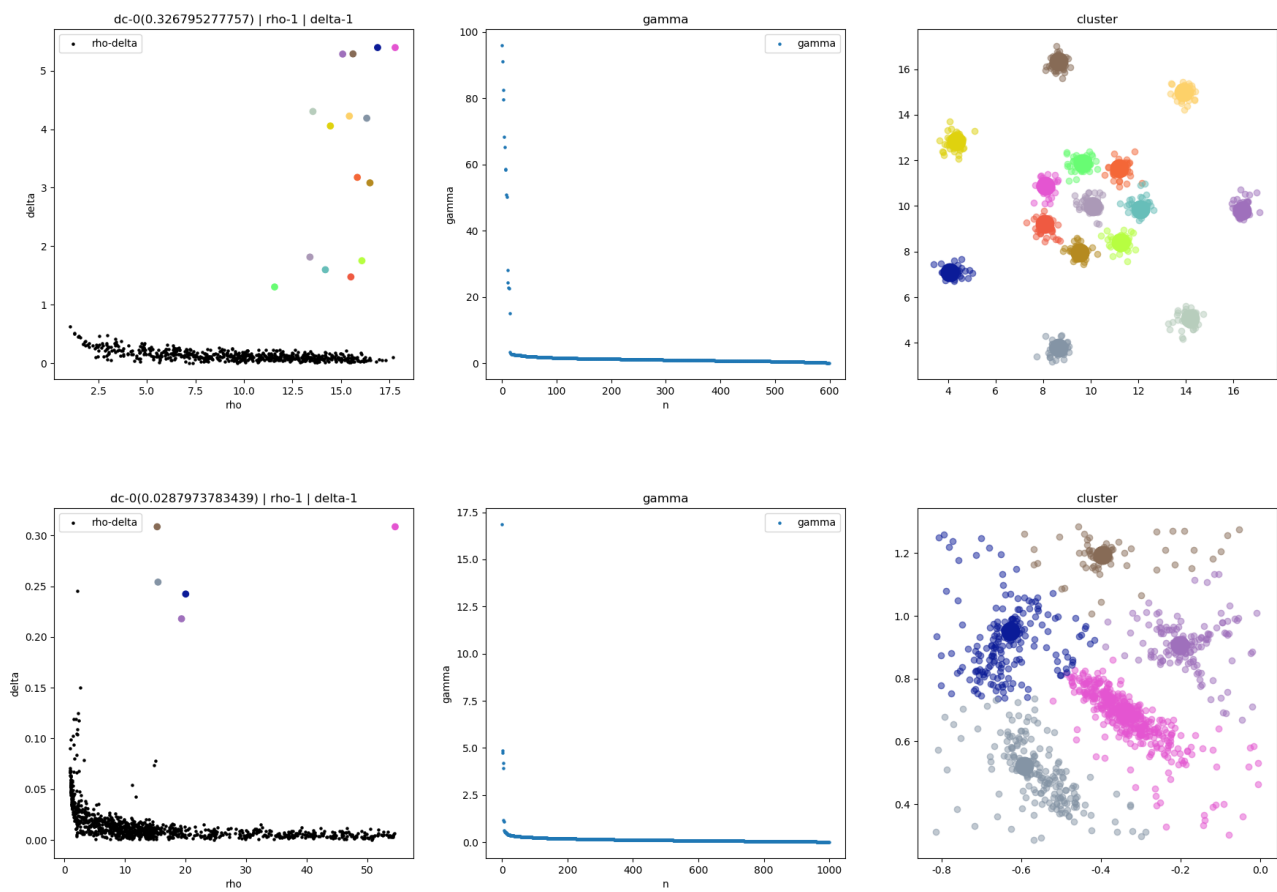
两种方法并无太大差异，
我的实现中采用了给定 N 值的方法。

8. 分配点 (聚类)

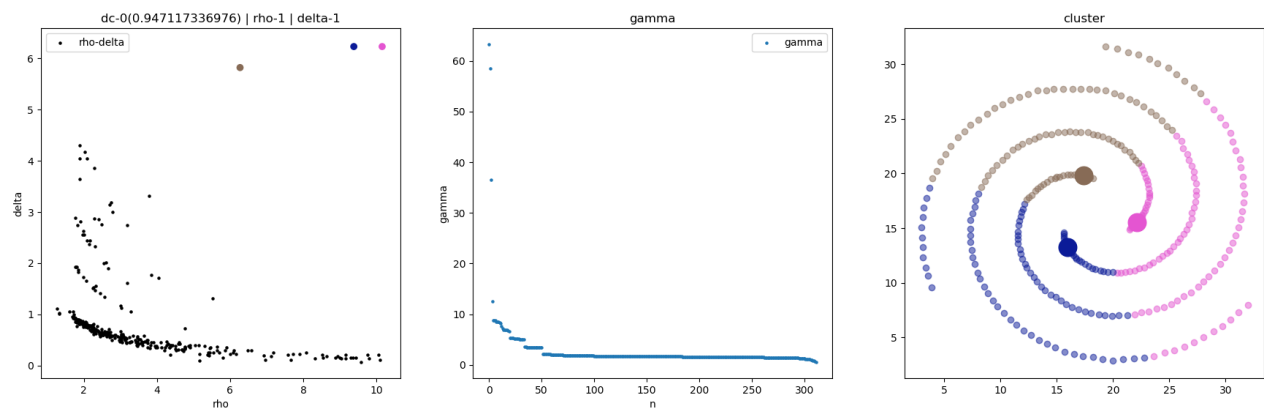
这里探讨两种分配点的方法。

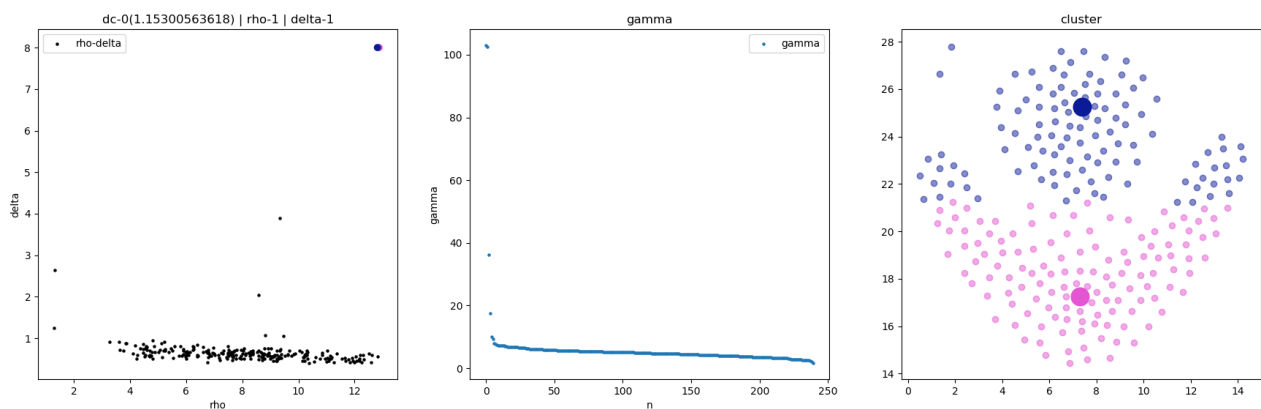
方法一：直接将数据点与他距离最近的聚类中心点分配到一类中

该方法简单直接
效果对于整体分布分散，局部分布集中的数据集效果较好。
如下 (origin-1000数据集，R15数据集， d_c 取 3%-4%， ρ 的计算采取高斯核， δ 的计算采取改进后的方法)



但是其弊端也十分明显，如下图（上：spiral数据集，下：flame数据集）





方法二：通过链式寻找最近高密度中心点

该方法中心思想为通过链式将每个点与中心点链接到。

1. 将 ρ 按降序排序，得到 $order_rho_index$ ：记录 ρ 降序的点编号，如 $order_rho_index[0]$ 为 ρ 最大的点编号。
2. 按照 $order_rho_index$ 设置 $link$ 。

$$i = order_rho_index[n], j = order_rho_index[m]$$

$$link[i] = \begin{cases} i & i \text{ is center} \\ j : \min(d_{j:m < n}(ij)) & i \text{ is not center} \end{cases}$$

3. 分配点

```

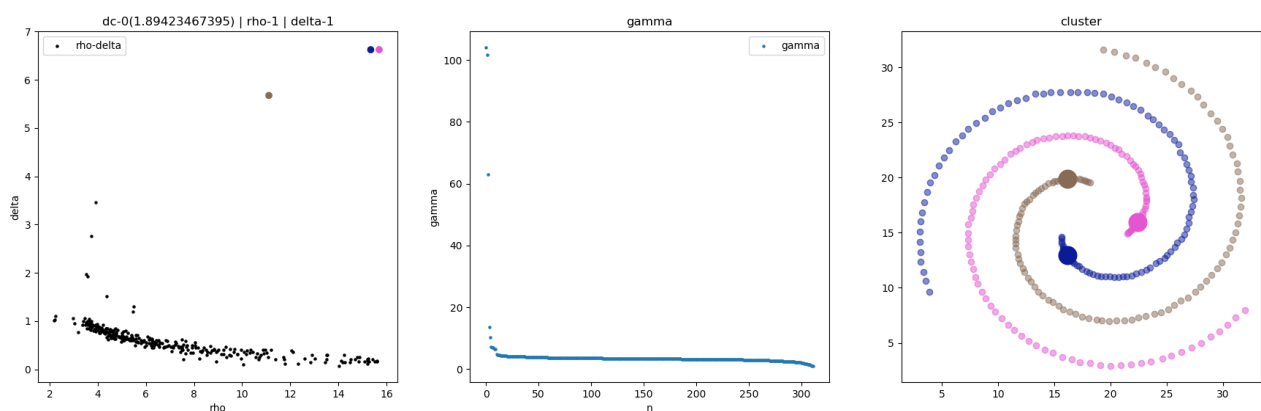
1 for i, v in link.items(): # i: 待分配点; v: 待分配点的链接点
2     c = v # c: 链接点
3     while c not in center: # 遍历直到中心点
4         c = link[c]
5     cluster[c].append(i) # 分配点

```

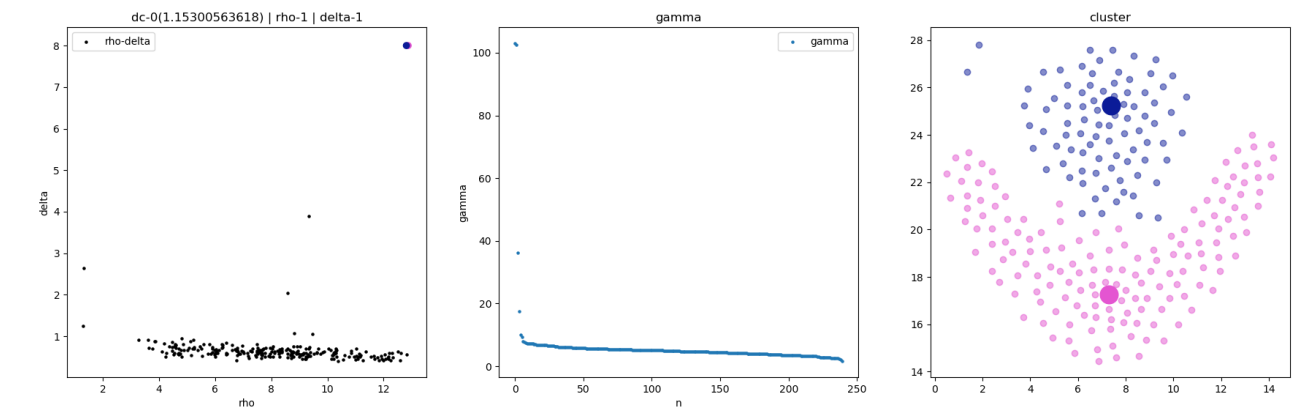
测试

方法二采用和方法一相同的数据集和方法进行测试。

■ spiral

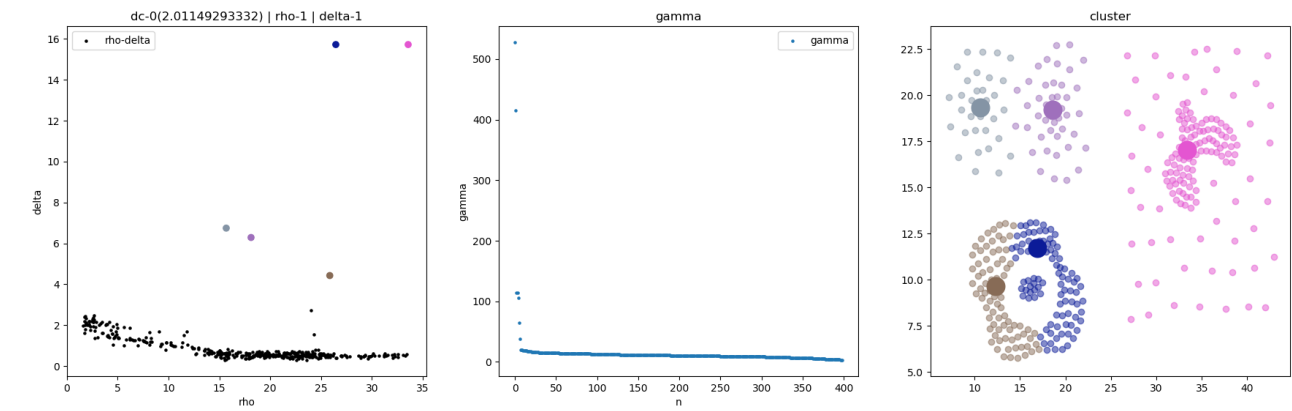


■ flame

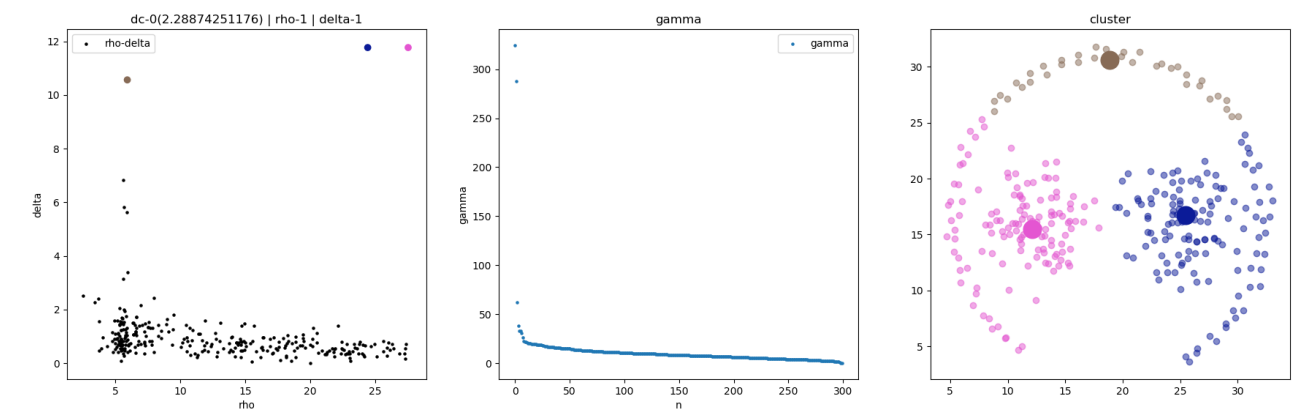


通过实验，可以明显感受到方法二较方法一更适合处理不规则数据集，但在测试所选的十个的数据集中，仍有两个数据集无法得到最佳聚类。

■ compound



■ path-based



综上，我在实现算法时采用了方法二（尽管仍有一些未处理的问题）。

9. 计算 halo

评测点的可靠性。

将聚类点分为 *core*, *halo* 。

$i \in \text{cluster}[c]$, $j \notin \text{cluster}[c]$, c : 聚类中心

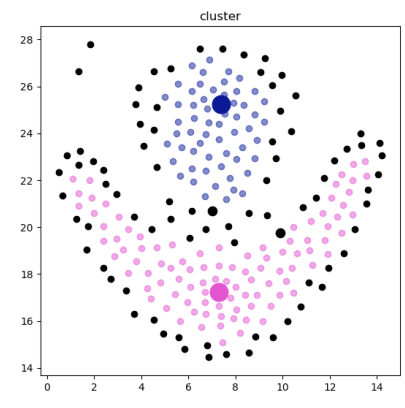
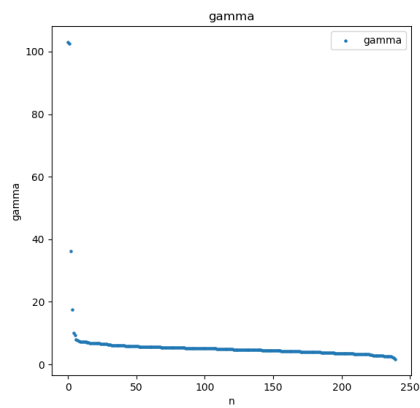
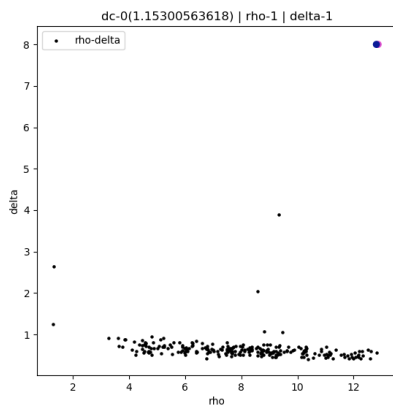
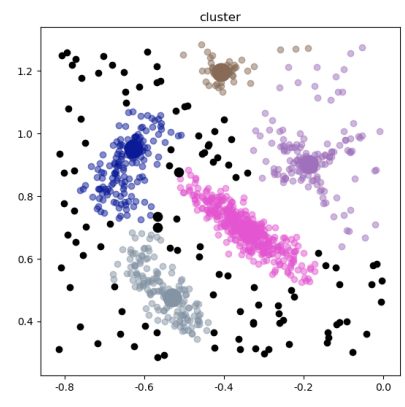
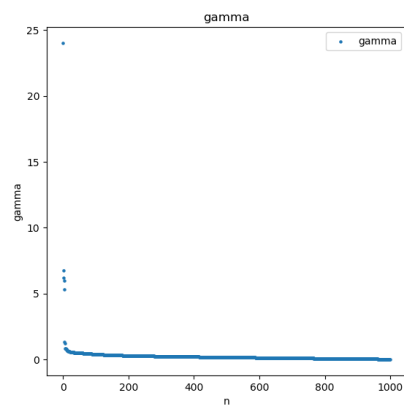
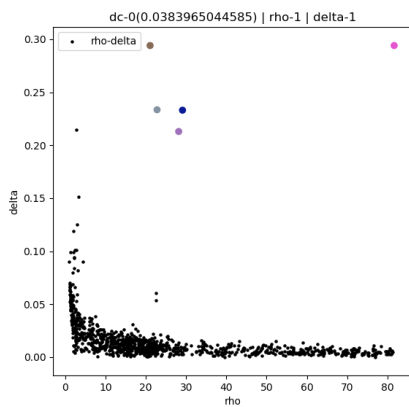
$\text{border}[c] = i : d(ij) < d_c$

$\rho_b = \max(\rho_k : k \in \text{border}[c])$

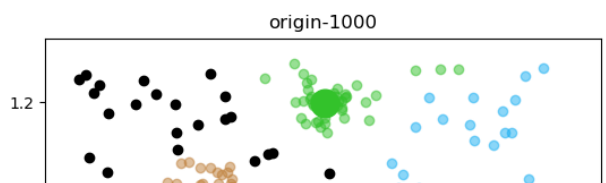
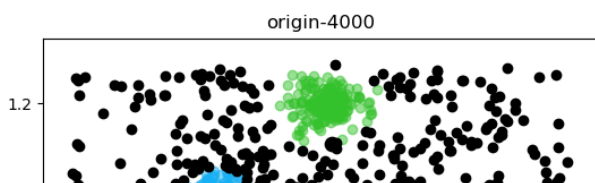
$\text{core}[c] = m : \rho_m \geq \rho_b$

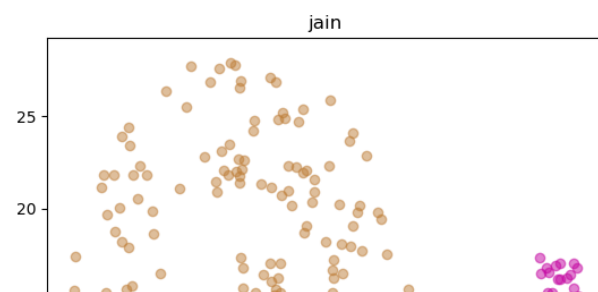
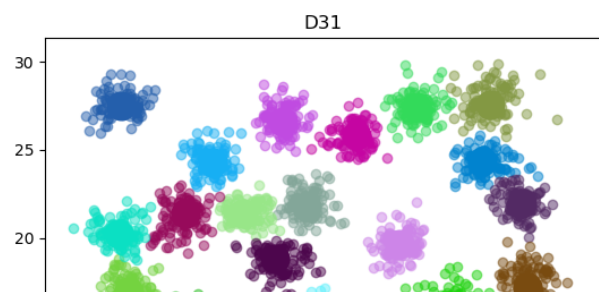
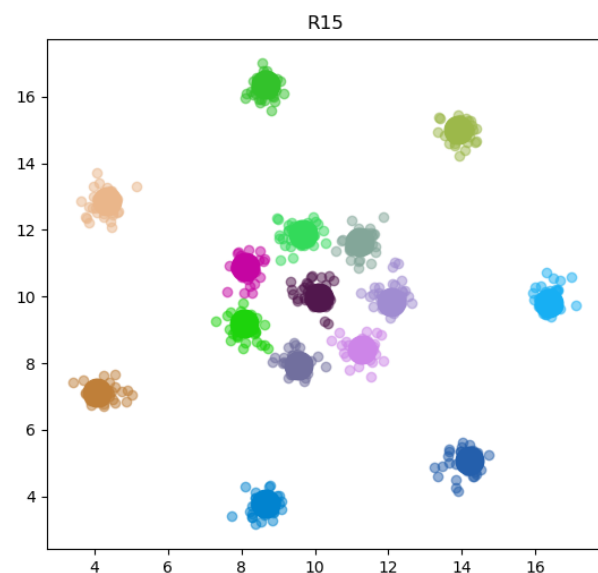
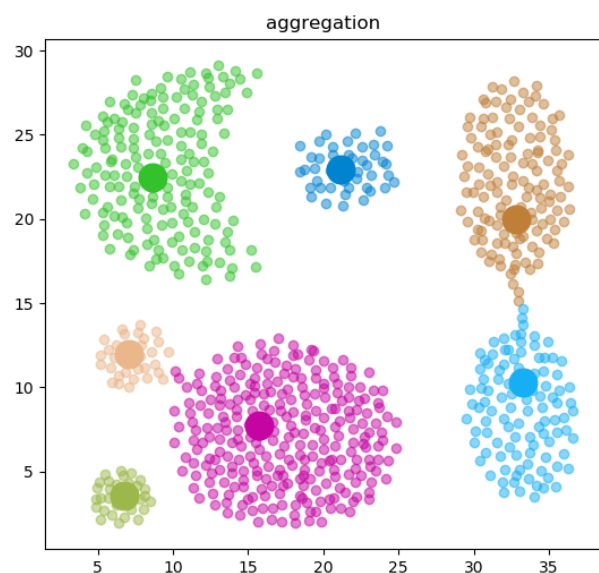
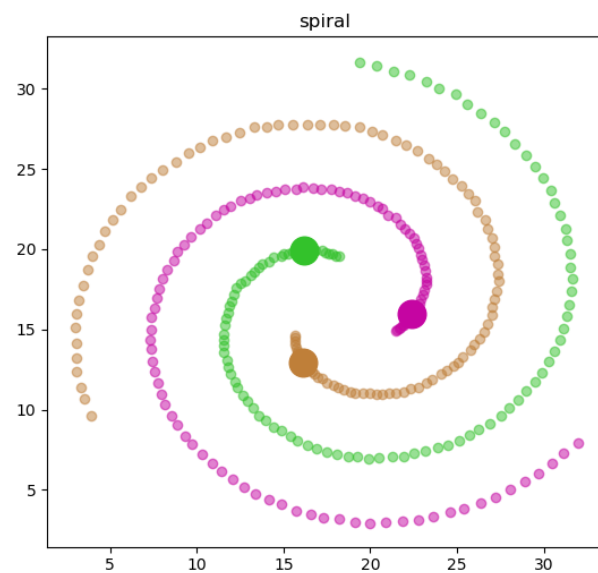
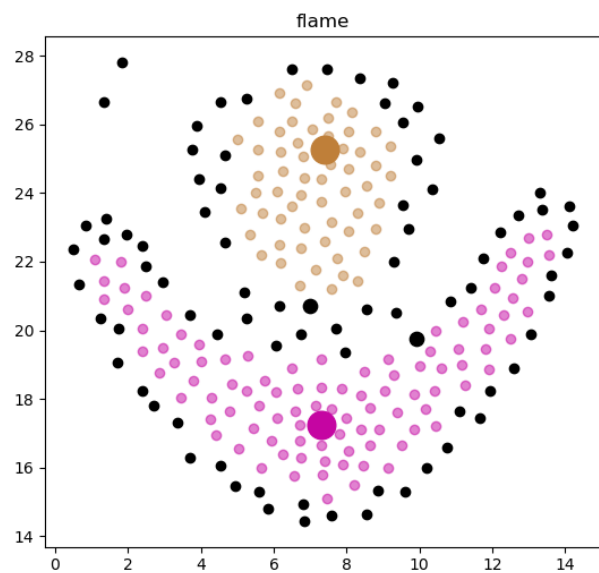
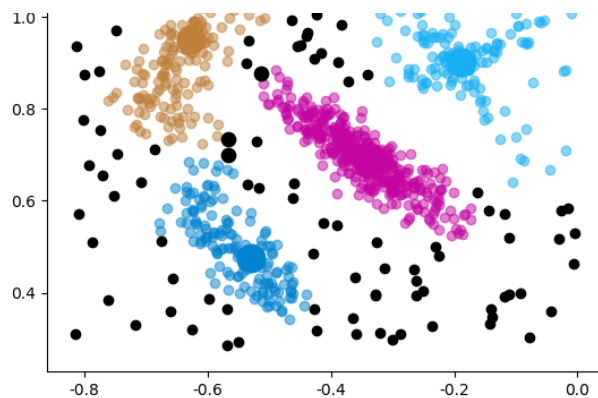
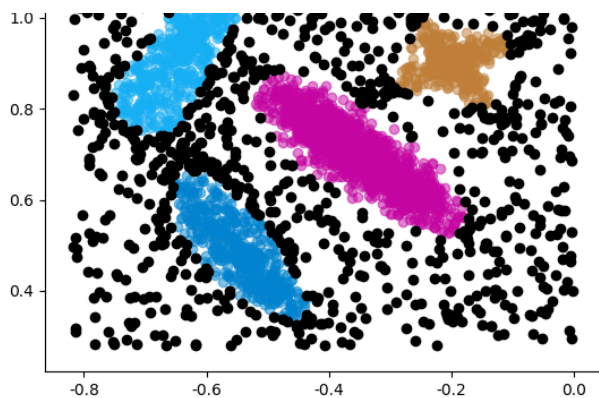
$\text{halo}[c] = n : \rho_n < \rho_b$

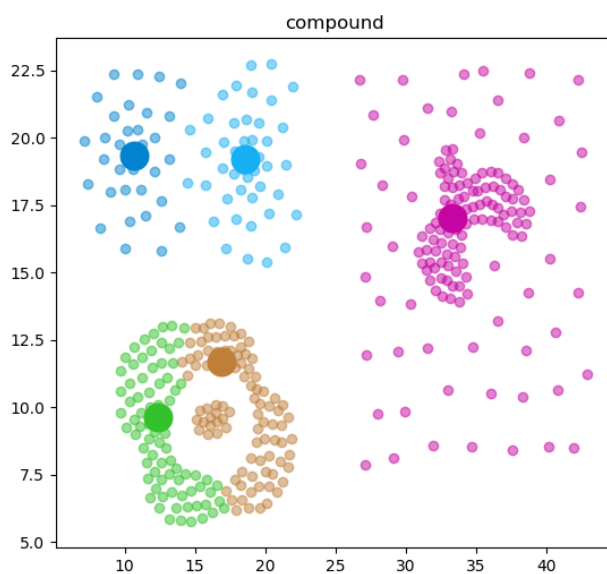
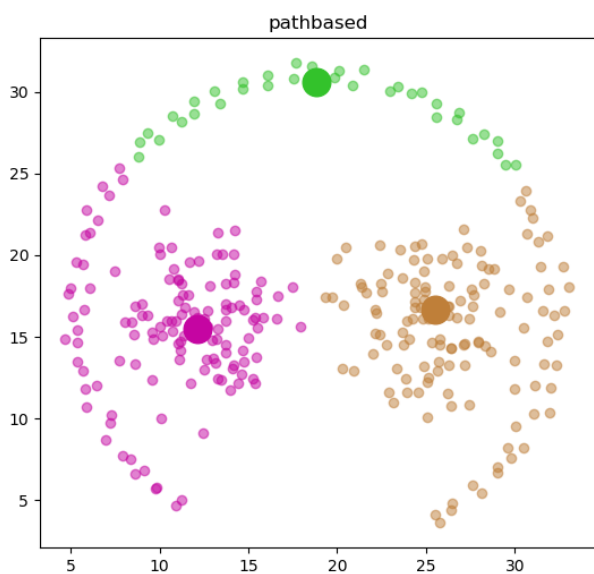
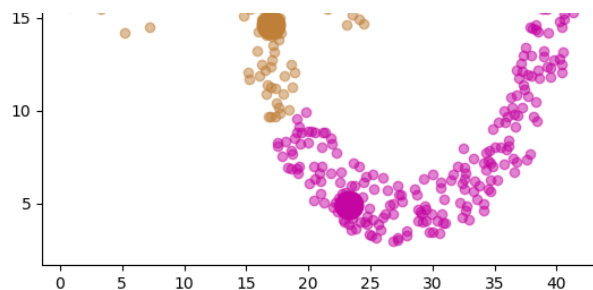
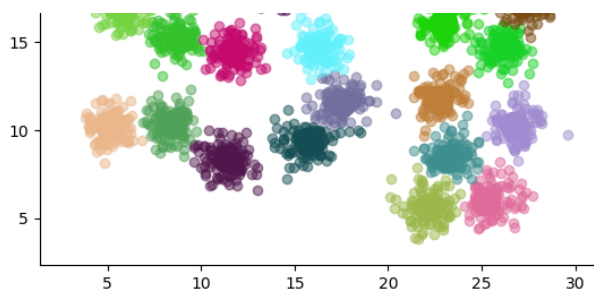
这里选取 origin-1000 数据集 和 flame 数据集



10. 绘制结果







总结

- DPC算法将 ρ 与 δ 结合并给予人们寻找聚类中心启发性的决策图。在当时聚类算法存在诸多问题时是很不错的思想。
- 尽管饱受人们的争议，例如作者的措辞有种被夸大的感觉，或者作者代码实现与论文原文所描述有所出入，又或者因为测试的不够全面而被批判，但是其算法的提炼程度及其核心思想仍受到大多数人们的称赞。
- 在未接触过其他著名的聚类算法的情况下阅读这篇论文并实现，收获颇丰。在质疑很多问题的同时也收获了很多知识。