LAB4 Density Peak Clustering

袁雨 PB20151804

一、实验目的

本次实验需要实现《**Clustering by fast search and find of density peaks**》一文中的算法（以下简称**DPC**）。总体流程是完成 DPC 算法的代码实现，并在给定数据集上进行可视化实验。

二、实验原理

算法思想

集成了 k-means 和 DBSCAN 两种算法的思想

- 聚类中心周围密度较低，中心密度较高
- 聚类中心与其它密度更高的点之间通常都距离较远

算法流程

1. Hyperparameter: a distance threshold d_c
2. For each data point i, compute two quantities:
    - Local density: $\rho_i = \sum_j \chi(d_{ij} - d_c)$,where $\chi(x) = 1$ if $x < 0$ and $\chi(x) = 0$ otherwise
    - Distance from points of higher density: $\delta_i = \min\limits_{j:\rho_j > \rho_i} d_{ij}$ · For the point with highest density, take $\delta_i = \max\limits_j d_{ij}$
3. Identify the cluster centers and out-of-distribution (OOD) points

- Cluster centers: with both high ρ_i and δ_i
- OOD points: with high δ_i but low ρ_i
- Draw a decision graph, and make decisions manually

三、实验数据

本次实验采用 3 个 2D 数据集（方便可视化）

- Datasets/D31.txt
- Datasets/R15.txt
- Datasets/Aggregation.txt

数据格式

- 每个文件都是普通的 txt 文件，包含一个数据集
- 每个文件中，每一行表示一条数据样例，以空格分隔

注意事项

- 允许对不同的数据集设置不同的超参数

四、实验任务及要求

（一）任务

实验简介

本次实验的总体流程是完成 DPC 算法的代码实现，并在给定数据集上进行可视化实验。具体来说，同学们需要实现以下步骤

1. 读取数据集，（如有必要）对数据进行预处理
2. 实现 DPC 算法，计算数据点的 δ_i 和 ρ_i
3. 画出决策图，选择样本中心和异常点
4. 确定分簇结果，计算评价指标，画出可视化图

助教除大致浏览代码外，以以下输出为评价标准：

- 可视化的决策图
- 可视化的聚类结果图
- 计算出的评价指标值 （DBI）
- 输出只要在**合理范围**内即可，不作严格要求

实验结果需要算法代码和实验报告

- 助教将通过可视化结果和代码来确定算法实现的正确性
- 助教将阅读实验报告来检验同学对实验和算法的理解

评价指标

- 本次实验采用 Davis-Bouldin Index (DBI) 作为评价指标
- 建议统一调用 sklearn.metrics. davies_bouldin_score 进行计算

数据可视化

- 本次实验需要画两个二维散点图：决策图和聚类结果图
- 可视化库推荐 pyplot (也可自行选择别的工具，此处只做教程)

（二）要求

- 禁止使用`sklearn`或者其他的机器学习库，你只被允许使用`numpy`，`pandas`，`matplotlib`，和 Standard Library，你需要从头开始编写这个项目。
- 你可以和其他同学讨论，但是你不可以剽窃代码，我们会用自动系统来确定你的程序的相似性，一旦被发现，你们两个都会得到这个项目的零分。

五、实验步骤

1. 读取数据集，对数据进行预处理

```
dfA = pd.read_csv('Datasets//Aggregation.txt', header=None , sep = ' ')
dfA = np.array((dfA-dfA.min())/(dfA.max()-dfA.min()))
```

以数据集Aggregation.txt为例。使用`pd.read_csv`进行读取，因为源文件中没有列索引，故设置`header=None`。数据集中列以空格相隔，故设置分隔符`seq=' '`。

对数据进行预处理，将其进行Min_Max标准化，消除特征量纲等的影响，并将数据类型转换为numpy数组。

2. 实现 DPC 算法

（1）计算距离d_{ij}

```
def GetDistance(self):
    self.d = np.zeros([self.m,self.m])
    for i in range(self.m):
        for j in range(i+1,self.m):
            self.d[i][j] = np.sqrt(np.sum((self.X[i] - self.X[j]) ** 2))
            self.d[j][i] = self.d[i][j]
```

使用欧氏距离，计算距离$d_{ij}(i<j)$，并令$d_{ji}=d_{ij}$，保存在二维数组`d`中。

（2）确定截断距离d_c

```
dA = np.triu(dpc1.d,0)
dA = dA[np.where(dA!=0)].flatten()
dA = np.sort(dA)
print(dA[int(len(dA)*0.01)])
print(dA[int(len(dA)*0.02)])
```

论文中提供了对d_c选取的一个建议： As a rule of thumb, one can choose dc so that the average number of neighbors is around 1 to 2% of the total number of points in the data set.

故取`d`中`i<j`的部分，并对其进行升序排序，取前1%~2%的数作为d_c，实际调参时根据决策图与聚类结果等再做调整与选择。

（3）计算局部密度ρ_i

```
def GetRho(self):
    self.rho = np.zeros(self.m)
    if self.kernel == 'cut-off':
        dd = self.d - self.dc
        for i,di in enumerate(dd):
            self.rho[i] = di[di<0].size
    if self.kernel == 'gauss':
        self.rho = np.sum(np.exp(-self.d/self.dc) ** 2, axis = 1)
```

论文中使用Cut-off kernel： $\rho_i=\sum_j \chi(d_{ij}-d_c),\ \chi(x)=\begin{cases}1, & x<0\\ 0, & x\geq 0\end{cases}$

论文中还提到了Gaussian kernel：$\rho_i=\sum_j e^{-\left(\frac{d_{ij}}{d_c}\right)^2}$。

其中 cut-off kernel 为离散值，Gaussian kernel 为连续值，故后者产生冲突（不同的数据点具有相同的局部密度值）的概率更小。且 Gaussian kernel 满足：与x_i距离小于d_c的数据点越多，ρ_i值越大。

将计算结果保存在数组`rho`中。

（4）计算距离 δ_i

```
def GetDelta(self):
    self.delta = np.zeros(self.m)
    sort_index = np.argsort(-self.rho)
    self.index = sort_index
    self.delta[sort_index[0]] = np.max(self.d[sort_index[0]])
    for i in range(1,self.m):
        self.delta[sort_index[i]] = np.min(self.d[sort_index[i]]
[sort_index[0:i]])
```

论文中对δ_i的计算方法为：δ_i is measured by computing the minimum distance between the point i and any other point with higher density: $\delta_i = \min\limits_{j:\rho_j>\rho_i} d_{ij}$. For the point with highest density, take $\delta_i = \max\limits_{j} d_{ij}$

使用`sort_index`（以下用s代替）数组保存 $\{\rho_i\}_{i=1}^m$ 的一个降序排列下标序，则可定义

$$\delta_{s_i} = \begin{cases} \min_{j<i} \left\{d_{s_i s_j}\right\}, & i \geq 1 \\ \max_j \left\{d_{s_0,j}\right\}, & i = 0. \end{cases}$$

将计算结果保存在数组`delta`中。

（5）确定聚类中心和异常点

```
self.c = np.zeros(self.m)
    self.center = []
    cn = 1
    for i in range(self.m):
        # 聚类中心
        if self.rho[i]>=self.thr_rho and self.delta[i]>=self.thr_delta:
            self.center.append(i)
            self.c[i]=cn
            cn+=1
        # 异常点
        elif self.rho[i]<self.thr_rho and self.delta[i]>=self.thr_delta:
            self.c[i] = -1

print(f'共{cn}个cluster')
```

- Cluster centers: with both high ρ_i and δ_i
- OOD points: with high δ_i but low ρ_i

将$\rho >= thr_rho,\ \delta >= thr_delta$ 的点定义为聚类中心，将$\rho < thr_rho,\ \delta >= thr_delta$ 的点定义为异常点，其中$thr_rho,\ thr_delta$ 根据决策图人工选择。

使用数组`c`保存数据点归类属性标记，定义为：

$$c_i = \begin{cases} k, & \text{若 } x_i \text{ 为聚类中心，且归属于第 } k \text{ 个类；} \\ -1, & \text{若 } x_i \text{ 为异常点.} \end{cases}$$

初始化数组 c 为全零。

使用 cn 保存当前为第几个cluster。并按照定义对聚类中心和异常点的c_i进行赋值。

（6）对剩余数据点进行归类

```
for i in range(self.m):
    if self.c[self.index[i]] == 0:
        j = np.argmin(self.d[self.index[i]][self.index[:i]])
        self.c[self.index[i]] = self.c[self.index[j]]
```

论文中的归类方法：After the cluster centers have been found, each remaining point is assigned to the same cluster as its nearest neighbor of higher density.

按照 ρ 值从大到小的顺序进行遍历，逐层扩充每一个cluster，按照定义对剩余数据点的c_i进行赋值。

3. 画出决策图，选择样本中心和异常点

```
from matplotlib import pyplot as plt
def draw_decision_graph(self):
    plt.scatter(self.rho, self.delta,alpha=0.5)
    plt.hlines(self.thr_delta,0,np.max(self.rho),linestyles='--
',colors="lightcoral")
    plt.vlines(self.thr_rho,0,np.max(self.delta),linestyles='--
',colors='lightcoral')
    plt.xlabel("rho")
    plt.ylabel("delta")
    plt.title(f'决策图\n dc={self.dc}, thr_rho={self.thr_rho}, thr_delta=
{self.thr_delta}')
    plt.show()
    print("聚类中心：")
    for i in self.center:
        print("rho:",self.rho[i]," delta:",self.delta[i],"数据点:",self.X[i])
    print("异常点：")
    print("rho: ",self.rho[np.where(self.c==-1)]," delta:
",self.delta[np.where(self.c==-1)], "数据点:", self.X[np.where(self.c==-1)])
```

调用 pyplot 库对决策图进行可视化。

人工根据决策图选择聚类中心和异常点，即选择thr_rho、thr_delta。

打印聚类中心和异常点。

4. 确定分簇结果，计算评价指标（DBI），画出可视化图

```
def plot_result(self):
    plt.scatter(x=self.X[:, 0], y=self.X[:, 1],
c=self.c,cmap='tab20',alpha=0.4)
    plt.scatter(x=self.X[self.center][:, 0], y=self.X[self.center][:, 1],
marker="x", s=50, c="r")
    plt.title(f'分簇结果\n dc={self.dc}, thr_rho={self.thr_rho}, thr_delta=
{self.thr_delta}')
    plt.show()
```

```
from sklearn.metrics import davies_bouldin_score as dbs
print('DBI得分为：', dbs(dfA, dpc1.c))
```

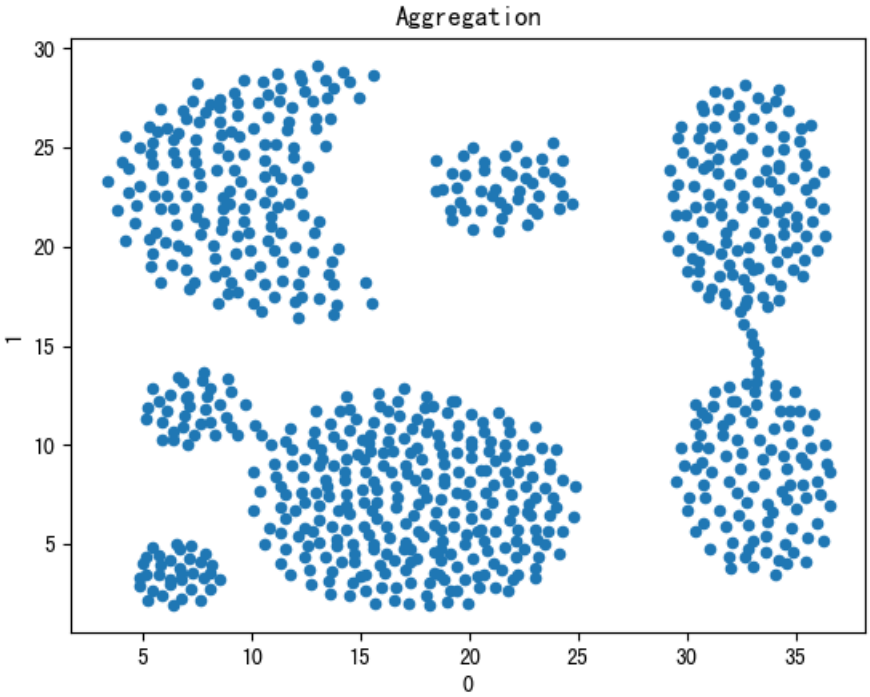
调用 pyplot 库对分簇结果进行可视化，调用sklearn.metrics. davies_bouldin_score计算DBI指数。

DBI表示聚类之间的平均“相似度”，其中相似度是将聚类之间的距离与聚类本身的大小进行比较的度量，越小越好。

六、实验结果与分析

1. 数据集 Aggregation.txt

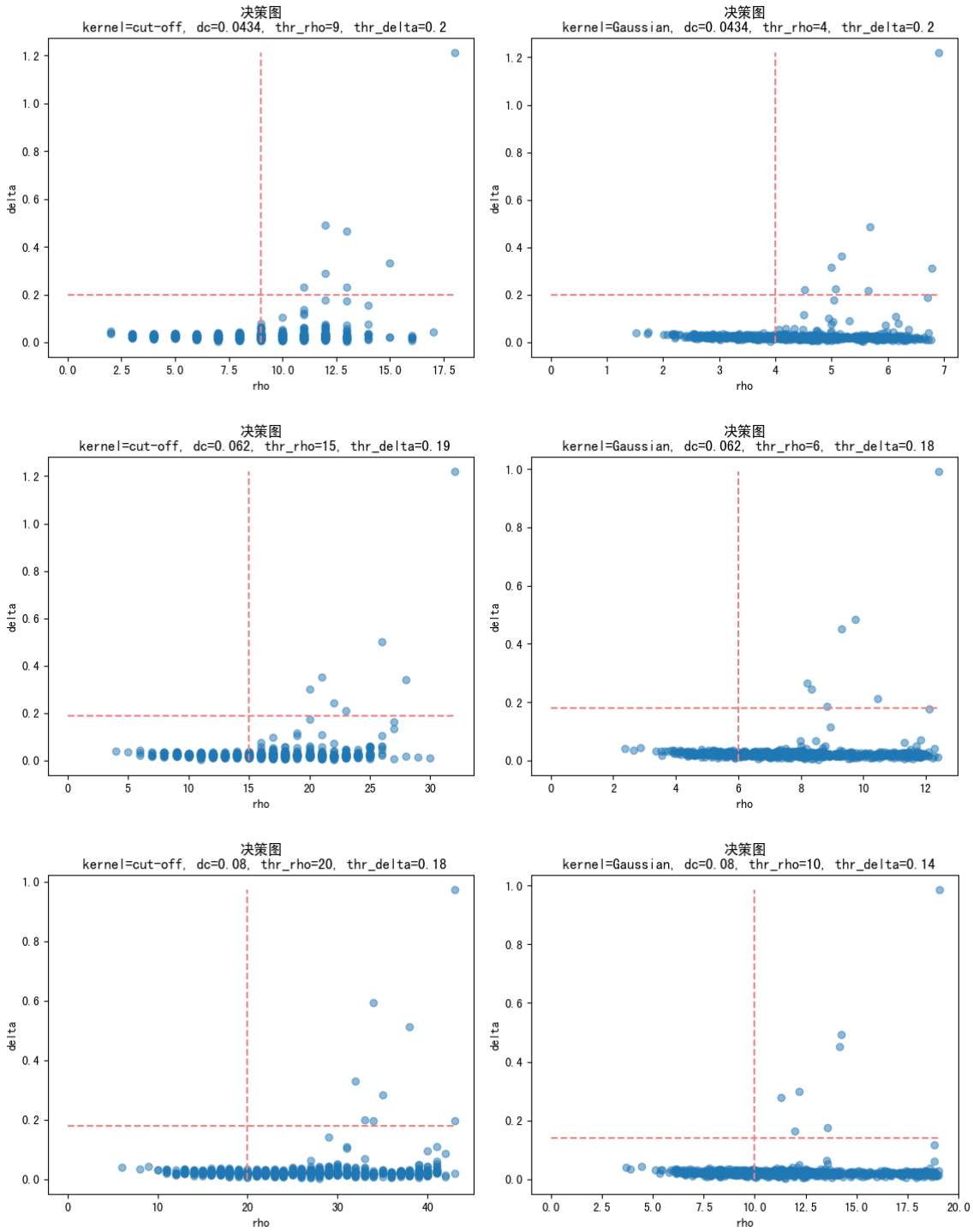
- 原始数据



- 调参

取`d`中`i<j`的部分，并对其进行升序排序，取前1%~2%位置的数，1%处为0.0434，2%处为0.0620。

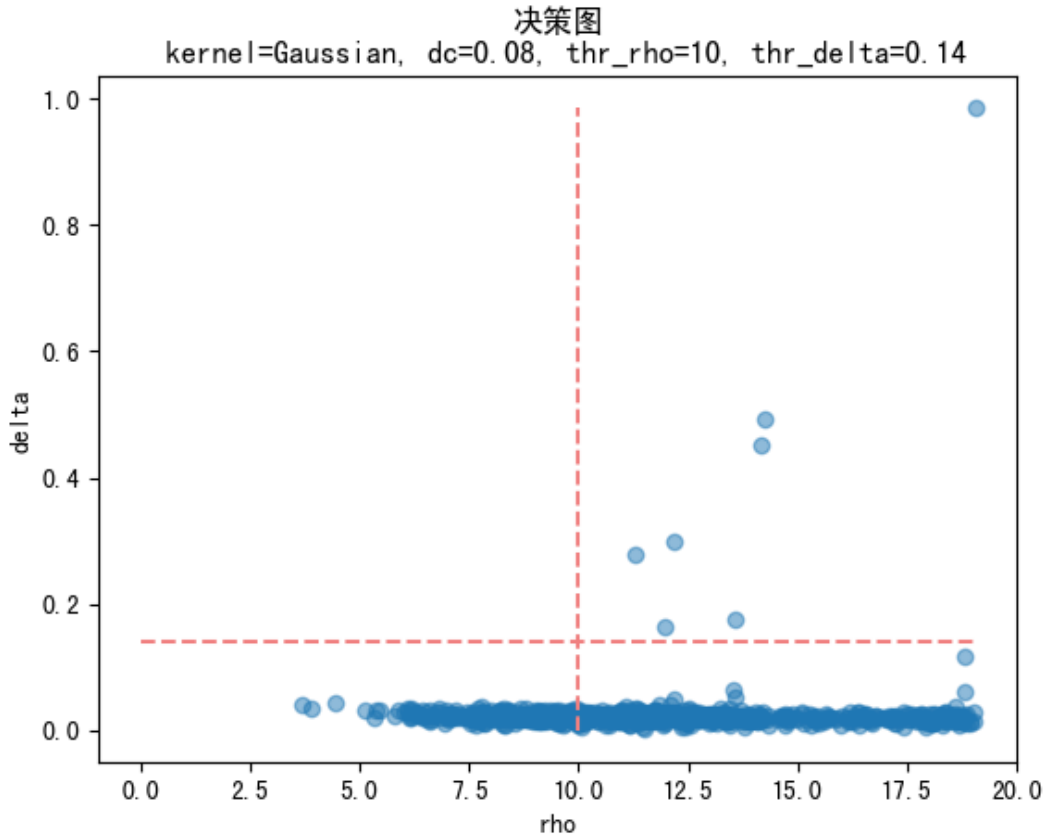
kernel	dc	thr_rho	thr_delta	DBI
cut-off	0.0434	9	0.2	0.5461
Gaussian	0.0434	4	0.2	0.6570
cut-off	0.0620	15	0.19	0.5435
Gaussian	0.0620	6	0.18	0.5435
cut-off	0.08	20	0.18	0.7141
Gaussian	0.08	10	0.14	0.5435



cut-off kernel 为离散值，Gaussian kernel 为连续值。后者产生冲突的概率更小。

可见取 $kernel = Gaussian, d_c = 0.08$ 时，聚类中心较明显，且DBI较低。

- 最终聚类结果
  - 决策图

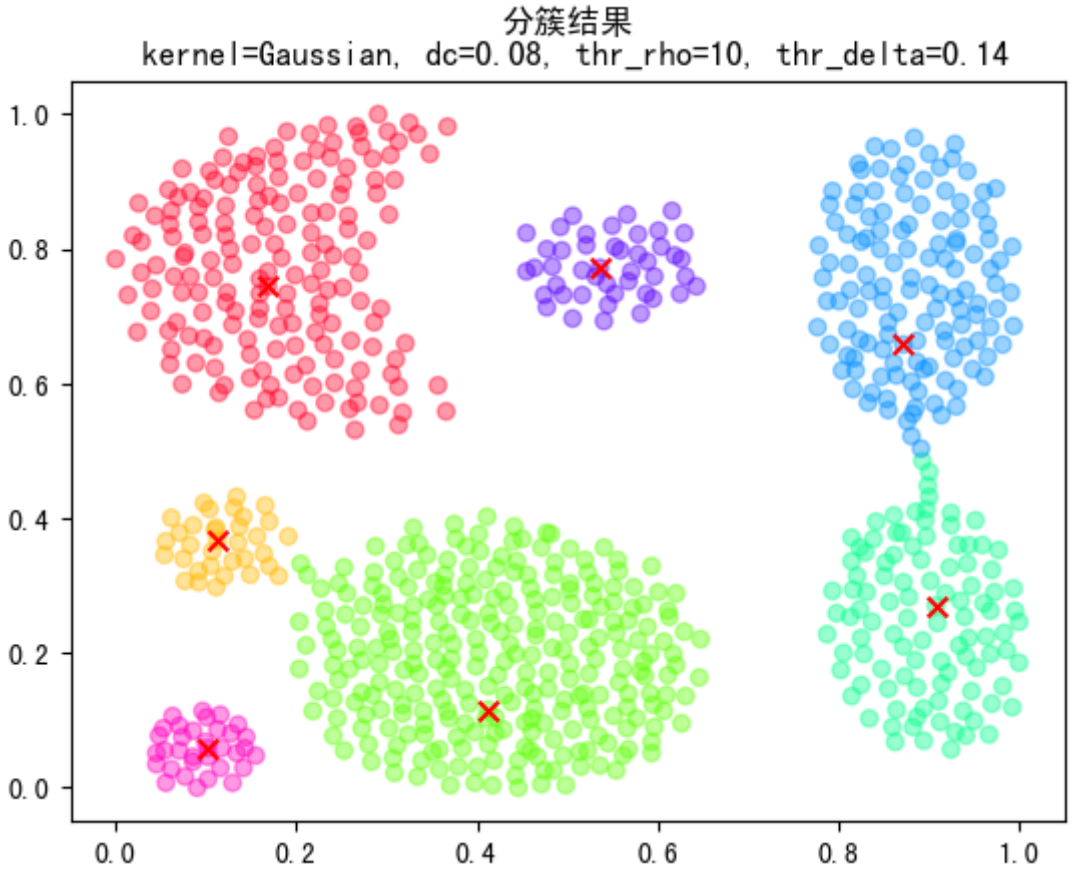


根据决策图，取$thr_rho = 10$，$thr\_delta = 0.14$。得到：

```
聚类中心：
rho: 14.163661793862584  delta: 0.45228153442147845 数据点: [0.1686747  0.74448529]
rho: 11.960983146456545  delta: 0.16345535310634343 数据点: [0.11295181 0.36764706]
rho: 19.053596557019606  delta: 0.9861867171163006 数据点: [0.4126506  0.11580882]
rho: 12.164807704686817  delta: 0.2993733475761586 数据点: [0.90813253 0.27022059]
rho: 14.258517403957905  delta: 0.49268833756639085 数据点: [0.87198795 0.65808824]
rho: 11.30565915823793  delta: 0.27907689865323926 数据点: [0.53614458 0.77205882]
rho: 13.59486821457438  delta: 0.17629866329748875 数据点: [0.10240964 0.05882353]
异常点：
rho:  []  delta:  [] 数据点: []
```

共7个cluster，没有异常点。

  - 聚类结果图

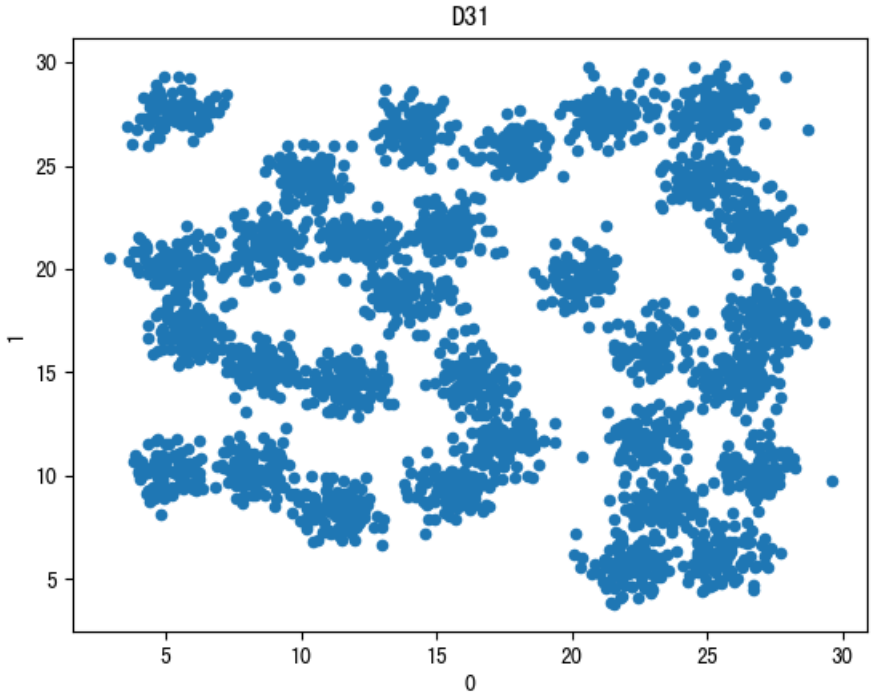


- DBI

```
DBI:  0.5435124431628843
```

2. 数据集 D31.txt

- 原始数据



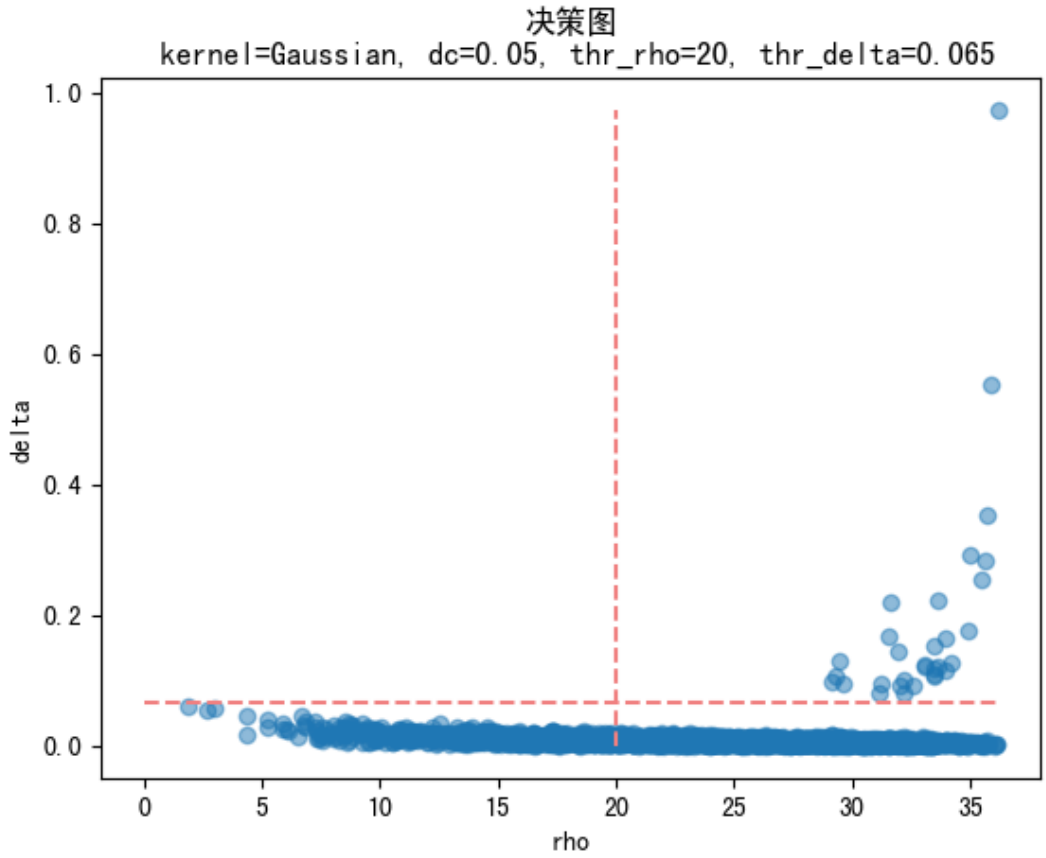
- 调参

取`d`中`i<j`的部分，并对其进行升序排序，取前1%~2%位置的数，1%处为0.0346，2%处为0.0542。

kernel	dc	thr_rho	thr_delta	DBI
cut-off	0.04	35	0.07	0.5527
Gaussian	0.04	15	0.065	0.5527
cut-off	0.05	40	0.07	0.5525
Gaussian	0.05	20	0.065	0.5520

4种参数的决策图差异不大，取$kernel = Gaussian, d_c = 0.05$时，DBI较低。

- 最终聚类结果
  - 决策图

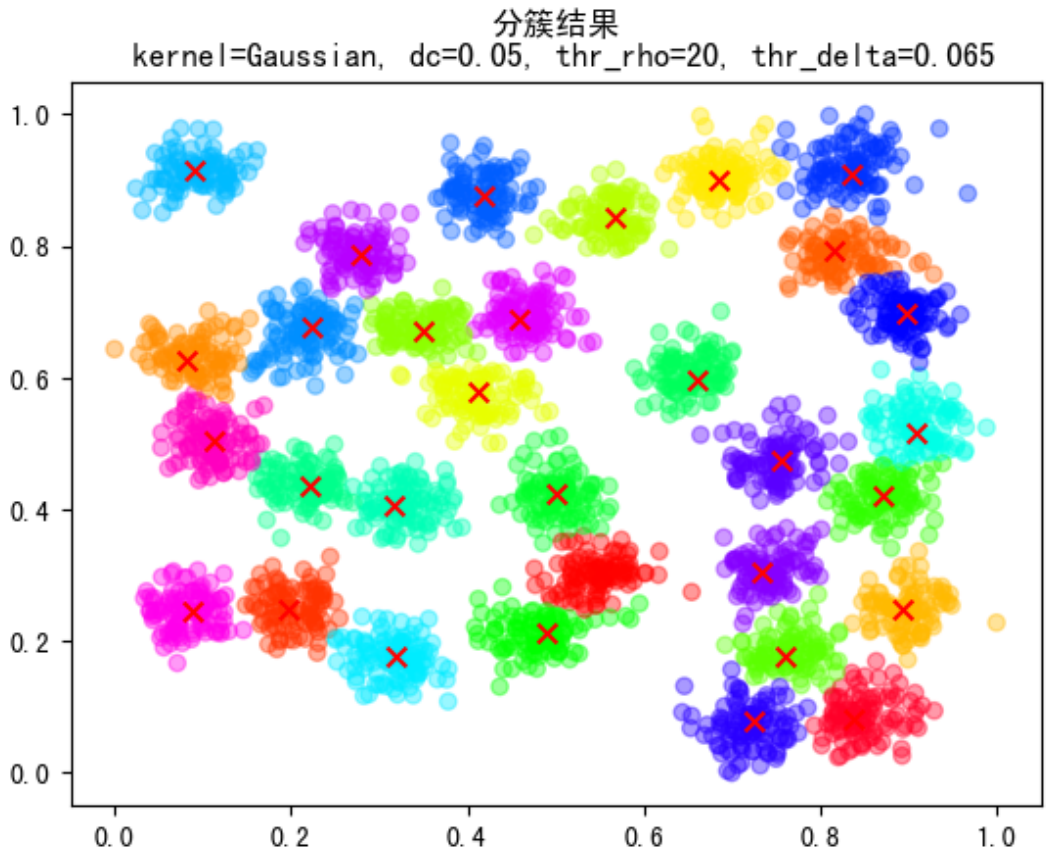


根据决策图，取$thr_rho = 20$，$thr\_delta = 0.065$。得到：

```
聚类中心：
rho: 32.18617452206885  delta: 0.1013195465386932 数据点: [0.83702491 0.08313476]
rho: 35.88562793646806  delta: 0.5527107407793451 数据点: [0.55089201 0.30420969]
rho: 29.62455944438822  delta: 0.09402806937548175 数据点: [0.19701801 0.24993381]
rho: 36.18189460876043  delta: 0.9730981403849877 数据点: [0.81651297 0.79242785]
rho: 29.347707134254847  delta: 0.10586388055748734 数据点: [0.08075498 0.62757038]
rho: 33.462350523072466  delta: 0.15347178044080065 数据点: [0.89425149 0.2484335 ]
rho: 33.11216921042793  delta: 0.1232596926619309 数据点: [0.68602948 0.89974406]
rho: 31.214897680390596  delta: 0.09517206717980058 数据点: [0.41135913 0.5782367 ]
rho: 35.49460933160964  delta: 0.2536691709745524 数据点: [0.56709472 0.84282058]
rho: 34.197144598362975  delta: 0.12587486061764278 数据点: [0.34973714 0.66975554]
rho: 32.62187676068731  delta: 0.09326960600158316 数据点: [0.76083771 0.17721296]
rho: 33.104146245805026  delta: 0.12266444329916193 数据点: [0.87210204 0.42246933]
rho: 32.02068510291562  delta: 0.09300459931904174 数据点: [0.48909765 0.21348513]
rho: 33.635775842273404  delta: 0.1203469010217856 数据点: [0.50073257 0.42441091]
rho: 31.959253726145757  delta: 0.143837450787043 数据点: [0.65974317 0.59659342]
rho: 35.73865771770489  delta: 0.3544990740342196 数据点: [0.2206326  0.43720766]
rho: 32.20289211981844  delta: 0.08063424229721056 数据点: [0.31672843 0.40631895]
rho: 31.16048599665672  delta: 0.0821691850125294 数据点: [0.90847195 0.5158415 ]
rho: 31.579818541592658  delta: 0.16678324320385593 数据点: [0.31827976 0.1771247 ]
rho: 33.61240705836801  delta: 0.22325600764405562 数据点: [0.08971818 0.91448239]
rho: 33.44939504034086  delta: 0.10985487170688306 数据点: [0.22313195 0.67584503]
rho: 29.494457193994826  delta: 0.13014809064052546 数据点: [0.41911575 0.87494484]
rho: 29.190142721019594  delta: 0.09774697151079895 数据点: [0.83633543 0.91042273]
rho: 34.00519541921837  delta: 0.11557300213496802 数据点: [0.8986469  0.69729062]
rho: 35.664878095164376  delta: 0.283122664001958 数据点: [0.72464018 0.07783956]
rho: 34.003301837067134  delta: 0.16541891885008048 数据点: [0.75523571 0.47542141]
rho: 34.954868032657984  delta: 0.17564478423298022 数据点: [0.73386193 0.30394493]
rho: 34.971190089622  delta: 0.29275846389006543 数据点: [0.27872102 0.78669138]
rho: 33.502795585244286  delta: 0.1056640895167202 数据点: [0.45953633 0.69005383]
rho: 31.669745706799816  delta: 0.2200030204051637 数据点: [0.0890287  0.24552114]
rho: 33.506401873413154  delta: 0.11757549621868388 数据点: [0.11178144 0.5044568 ]
异常点：
rho:  []  delta:  [] 数据点: []
```

共31个cluster，没有异常点。

- 聚类结果图

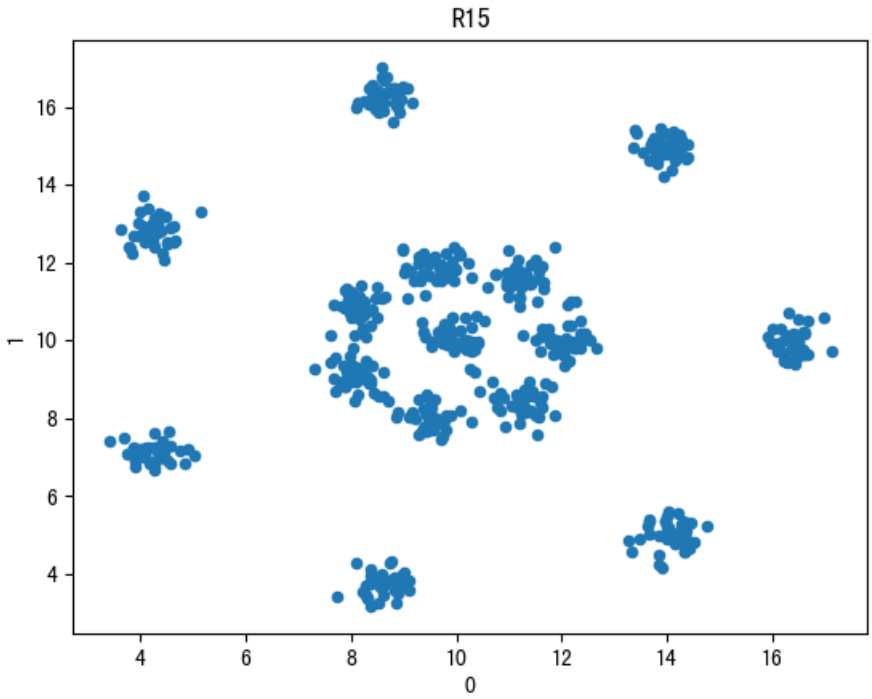


- DBI

```
DBI:  0.5520423239381427
```

3. 数据集 R15.txt

- 原始数据

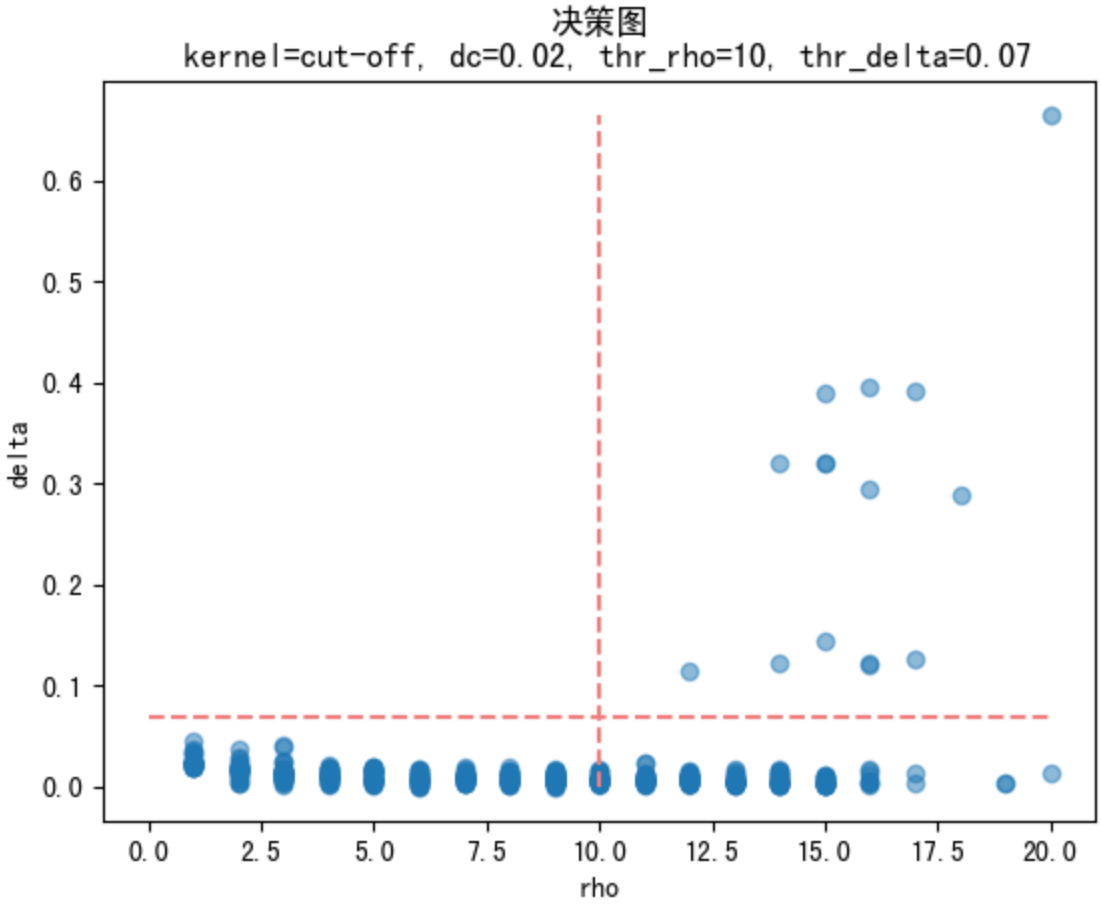


- 调参

kernel	dc	thr_rho	thr_delta	DBI
cut-off	0.02	10	0.07	0.3147
Gaussian	0.02	4	0.07	0.3147
cut-off	0.025	12	0.07	0.3147
Gaussian	0.025	5	0.07	0.3147

取`d`中`i<j`的部分，并对其进行升序排序，取前1%~2%位置的数，1%处为0.0181，2%处为0.0268。几种参数决策图都较好，DBI相同。任意选一种参数$kernel = cut-off, dc = 0.02$。

- 最终聚类结果
  - 决策图



根据决策图，取$thr_rho = 6$，$thr\_delta = 0.07$。得到：

聚类中心：

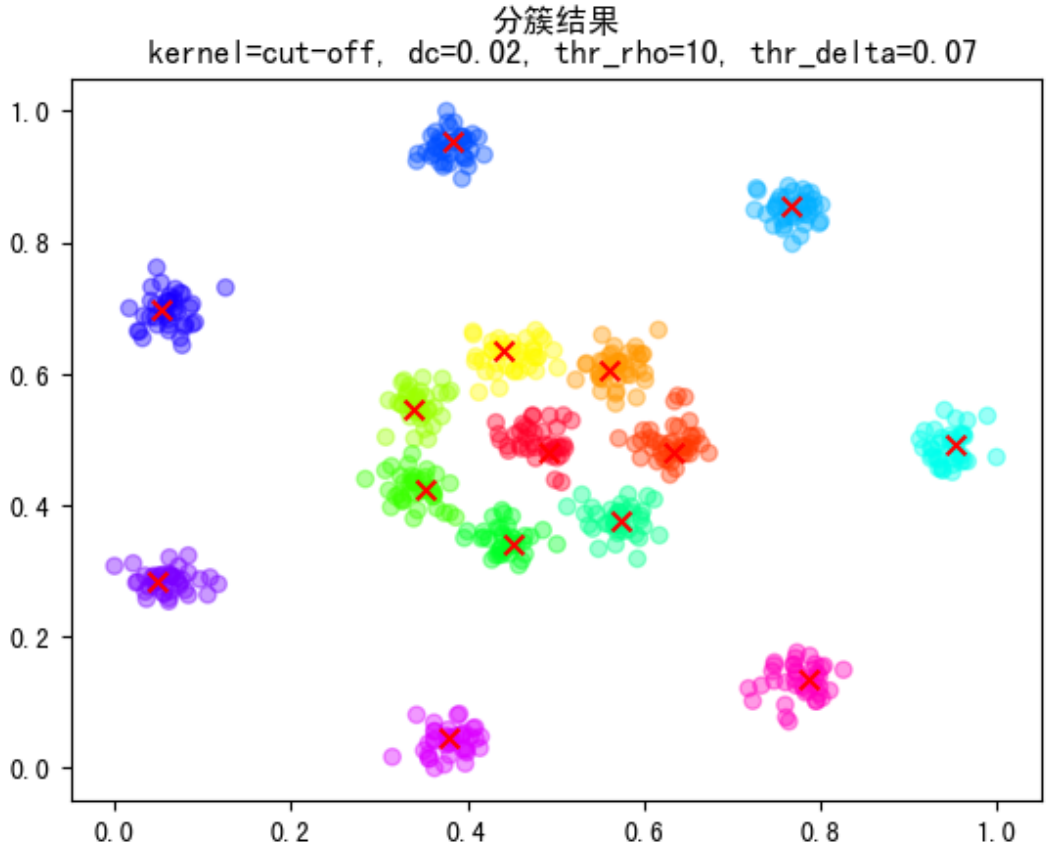
```
rho: 14.0  delta: 0.12177780997729562 数据点: [0.49205655 0.48012144]
rho: 16.0  delta: 0.12092269519559948 数据点: [0.63358111 0.48200087]
rho: 15.0  delta: 0.14381387710499613 数据点: [0.55983093 0.6054648 ]
rho: 12.0  delta: 0.11502250644864004 数据点: [0.44075208 0.63481278]
rho: 20.0  delta: 0.6653716587267169 数据点: [0.33872613 0.54705797]
rho: 16.0  delta: 0.12347933573368522 数据点: [0.35155225 0.42359404]
rho: 17.0  delta: 0.12751417632522521 数据点: [0.45168343 0.33945352]
rho: 18.0  delta: 0.28989479712204025 数据点: [0.57353156 0.37704207]
rho: 15.0  delta: 0.3197928726738887 数据点: [0.95321382 0.49212086]
rho: 16.0  delta: 0.3951930713563003 数据点: [0.76621484 0.85427208]
rho: 15.0  delta: 0.3892852530777385 数据点: [0.38332605 0.95243603]
rho: 15.0  delta: 0.3208435847926432 数据点: [0.05334499 0.69799046]
rho: 17.0  delta: 0.3917069867822488 数据点: [0.04868095 0.28379355]
rho: 16.0  delta: 0.2954540062306965 数据点: [0.37880775 0.04698569]
rho: 14.0  delta: 0.32020457287844545 数据点: [0.78720303 0.13633078]
```

异常点：

```
rho:  []  delta:  [] 数据点: []
```

共15个cluster，没有异常点。

- 聚类结果图



- DBI

DBI： 0.31471445116423397