MLlab3 实验报告

PB19010450 和泳毅

一、 实验要求

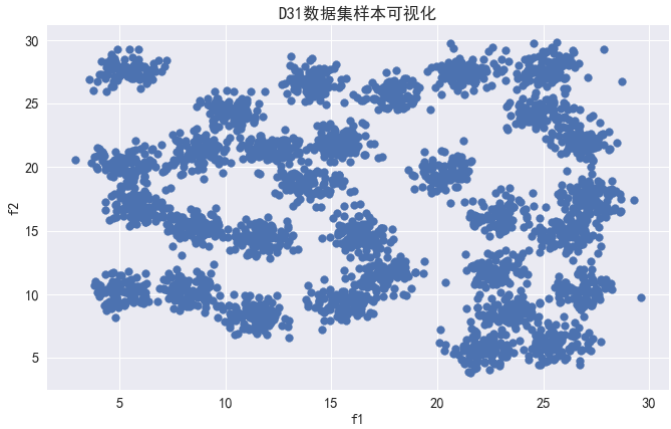
本次实验要求复现论文**Clustering by fast search and find of density peaks**中的聚类模型**Density Peak Clustering（DPC）**，并在给定数据集上进行测试。具体需要完成以下部分：

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

二、数据集介绍

1.D31数据集

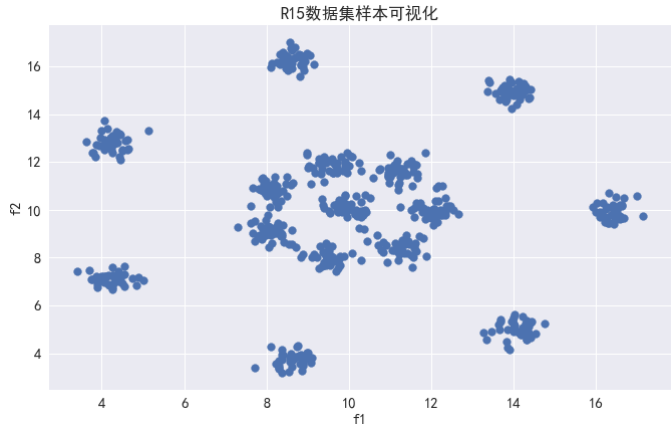
格式为txt文件，共3100个样本，每个样本有两个特征。样本可视化如下：



可以大致看到，该样本可以分为31类，并且可能存在个别离群点。

2.R15数据集

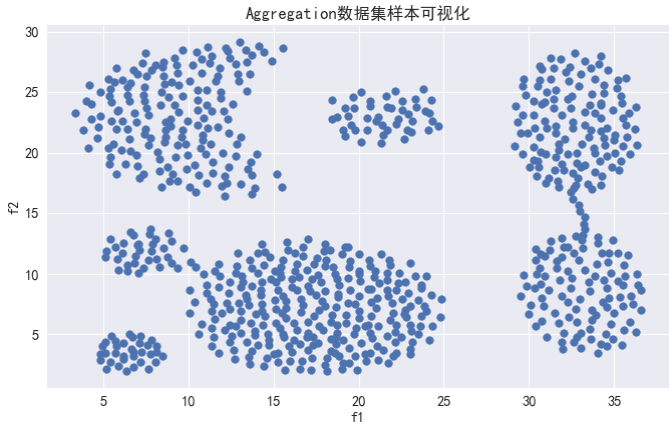
格式为txt文件，共600个样本，每个样本有两个特征。样本可视化如下：



可以大致看到，该样本可以分为15类，并且可能存在个别离群点。

3.Aggregation数据集

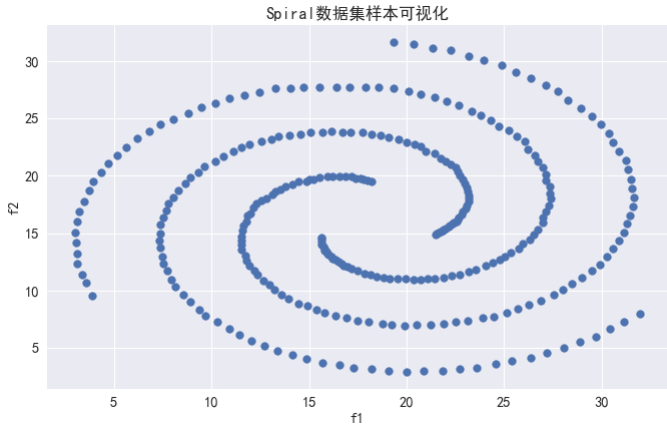
格式为txt文件，共788个样本，每个样本有两个特征。样本可视化如下：



可以大致看到，该样本可以分为7类，并且可能不存在离群点。

4.spiral数据集

格式为txt文件，共312个样本，每个样本有两个特征。样本可视化如下：



可以大致看到，该样本可以分为3类，并且可能不存在离群点。

三、 实验原理

1. 算法思想

经典的聚类算法**K-means**是通过指定聚类中心，再通过迭代的方式更新聚类中心的方式，由于每个点都被指派到距离最近的聚类中心，所以导致其不能检测非球面类别的数据分布。虽然有**DBSCAN**对于任意形状分布的进行聚类，但是必须指定一个密度阈值，从而去除低于此密度阈值的噪音点。

而**DPC** 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，并且聚类中心与其它密度更高的点之间通常都距离较远。

2.算法步骤

（1）计算任意两点之间的距离d_{ij}，并令$d_{ij}=d_{ji}$，$i<j$。

（2）手动确定截断距离d_c。

（3）基于截断核或者高斯核计算局部密度ρ_i。

- 截断核（cut-off kernel）

$$\rho_i=\sum_j \chi(d_{ij}-d_c),\quad \chi(x)=\begin{cases}1, & x<0\\ 0, & x\geq 0\end{cases}$$

- 高斯核（gauss kernel）

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

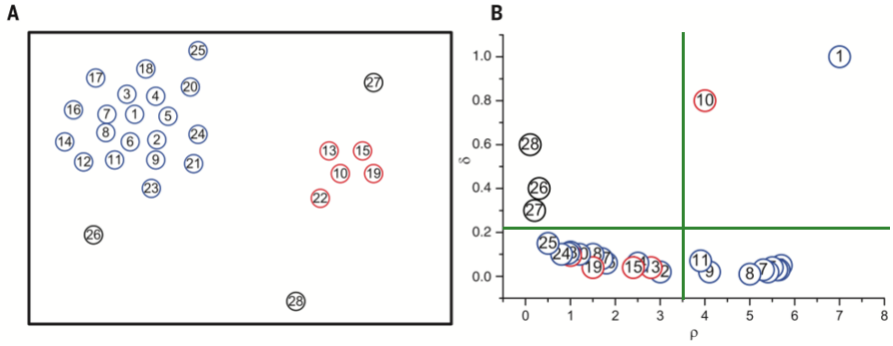
（4）计算基于局部密度的距离。

$$\delta_i=\begin{cases}\max_j d_{ij}, & i\text{是局部密度最大点}\\ \min_{j:\rho_j>\rho_i} d_{ij}, & i\text{非局部密度最大点}\end{cases}$$

（5）根据ρ_i,δ_i画出决策图，手动确定决策边界ρ_v,δ_v。

（6）根据ρ_v,δ_v划分类中心点（Cluster centers）和离群点（OOD points）。

将$\rho_i\geq\rho_v,\delta_i\geq\delta_v$的点划分为类中心点，并赋予不同的类标记；将$\rho_i<\rho_v,\delta_i\geq\delta_v$的点划分为离群点，并赋予统一的标记。



如上图1和10为类中心点，26、27、28为离群点。

（7）根据两点距离d_{ij}、局部密度ρ_i划分剩余样本点。

局部密度由高到低遍历样本点，对剩余样本点划分类别。每个样本点的类别与局部密度比它高且距离最近的点一致。

3.评价指标

DB指数（Davies-Bouldin Index,**DBI**），又称分类适确性指标，计算两个簇C_i、C_j各自的样本间平均距离$\mathrm{avg}(C)$之和除以两个类中心点之间的距离。对同一样本来说，**DBI**越小聚类效果越好。由于**DBI**使用欧氏距离，对环状分布的数据效果很差。

$$\mathrm{avg}(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \leqslant i < j \leqslant |C|} \mathrm{dist}(x_i, x_j)$$

$$d_{\mathrm{cen}}\left(C_i, C_j\right) = \mathrm{dist}(\mu_i, \mu_j), \quad \mu = \frac{1}{|C|} \sum_{1 \leq i \leq |C|} x_i$$

其中$\mathrm{dist}(\cdot,\cdot)$用于计算两个样本之间的距离，这里使用欧氏距离；$\mu$代表类$C$的中心点。$\mathrm{avg}(C)$表示类$C$内样本间的平均距离；$d_{\mathrm{cen}}\left(C_i, C_j\right)$表示类$C_i$和$C_j$中心点间的距离。

$$\mathbf{DBI} = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(\frac{\mathrm{avg}(C_i) + \mathrm{avg}(C_j)}{d_{\mathrm{cen}}\left(C_i, C_j\right)} \right)$$

四、核心代码讲解

1.评价

```
from sklearn.metrics import davies_bouldin_score
davies_bouldin_score(data, dpc.label)
```

调库实现DBI指标计算。

2.两点距离与局部密度

```
for i in range(n):
    for j in range(i + 1, n):
        dis[i][j] = dis[j][i] = np.linalg.norm(x[i] - x[j])
    if kernal == "gauss":
        rho[i] = np.exp(-(dis[i] / d_c) ** 2).sum() - 1
    elif kernal == 'cut_off':
        for j in range(n):
            rho[i] += 1 if dis[i][j] < d_c else 0
    else:
        print("This kernal method is not supported.")
        return
```

优化计算时间，用二范数 `np.linalg.norm` 计算欧氏距离d_{ij}，并令$d_{ij}=d_{ji}$。接着根据截断核与高斯核计算方法计算局部密度。用 `1 if dis[i][j] < d_c else 0` 表示示性函数$\chi(d_{ij}-d_c)$。

3.基于局部密度的距离

```
index = np.argsort(-rho)
delta[index[0]] = dis[index[0]].max()
for i in range(1, n):
    delta[index[i]] = dis[index[i]][index[0:i]].min()
```

优化计算时间，将对局部密度的排序转化为映射下标的排序，存储排序后的样本下标。接着根据于局部密度的距离计算方法来计算δ_i， `index[0]` 表示局部密度最大样本下标，`index[0:i]` 表示局部密度比样本i大的样本下标。

4.划分类中心点与离群点

```
j = 1
for i in range(n):
    # 类中心
    if self.rho[i] >= rho_value and self.delta[i] >= delta_value:
        self.center_index.append(i)
        self.label[i] = j
        j += 1
    # 离群点
    elif self.rho[i] < rho_value and self.delta[i] >= delta_value:
        self.label[i] = -1
```

对类中心点依次标记$1,2,3...$，对离群点标记-1。

5.划分剩余点

```
index = self.sorted_rho
for i in range(n):
    if self.label[index[i]] == 0:
        j = np.argmin(self.dis[index[i]][index[:i]])
        self.label[index[i]] = self.label[index[j]]
```

`index` 为局部密度降序排列对应样本下标，按照局部密度从大到小遍历划分剩余点。其中 `np.argmin(self.dis[index[i]][index[:i]])` 表示局部密度比i大且距离最近的样本点对应下标。

五、实验中遇到的问题及解决方案

- **计算复杂度略高**

  刚开始在D31数据集上的运行时间在30s左右，不便于调参。先对计算细节进行代码优化，并以样本下标存储局部密度排序结果。同时将计算距离与密度的部分在类中封装为静态函数，使用 `numba.jit` 进行加速。加速后算法D31上运行时间约2s。

- **matplotlib作图不便确定决策边界**

  绘制决策图使用plotly库，可以实时查看每个点的坐标。

- **使用与类中心的距离作为标准划分剩余点**

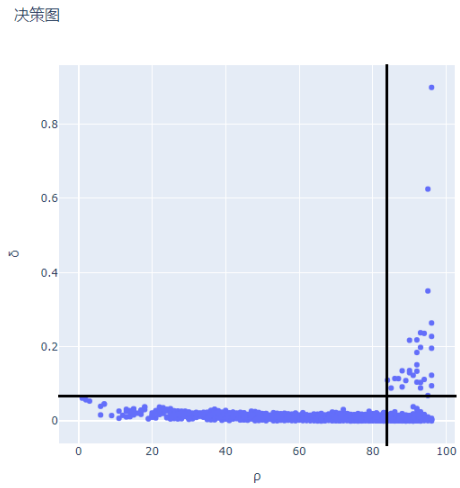
  查看原论文，每个样本点的类别应该与局部密度比它高且距离最近的点一致。

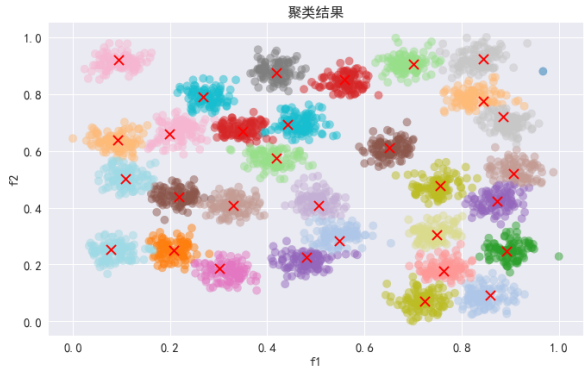
> After the cluster centers have been found, each remaining point is assigned to the same cluster as its nearest neighbor of higher density. The clus-

六、实验结果

1.D31数据集

选取$d_c = 0.06$，根据决策图选取$\rho_v = 84, \delta_v = 0.06$:



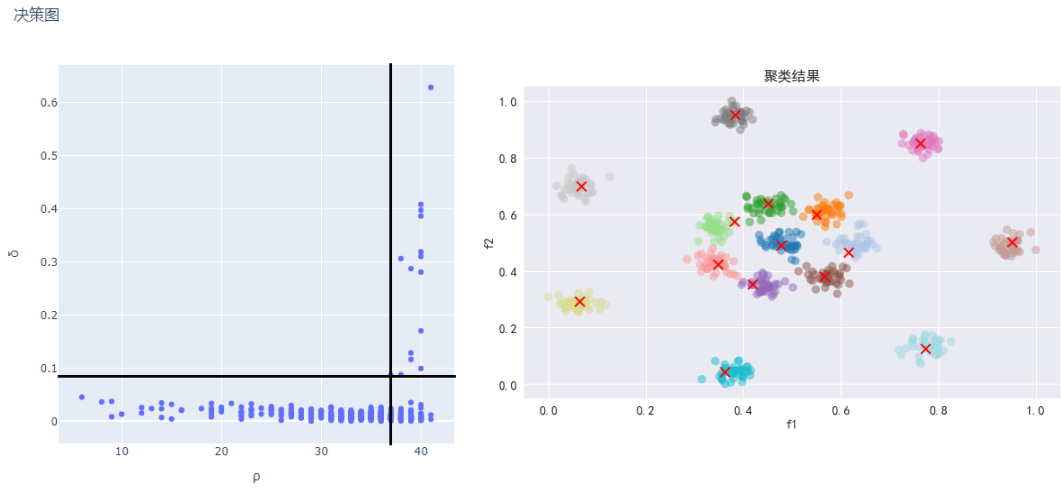


DBI=0.5456326805854976

可以看到样本被完全分为31类，且DBI得分不错。

2.R15数据集

选取$d_c = 0.06$，根据决策图选取$\rho_v = 37, \delta_v = 0.08$：

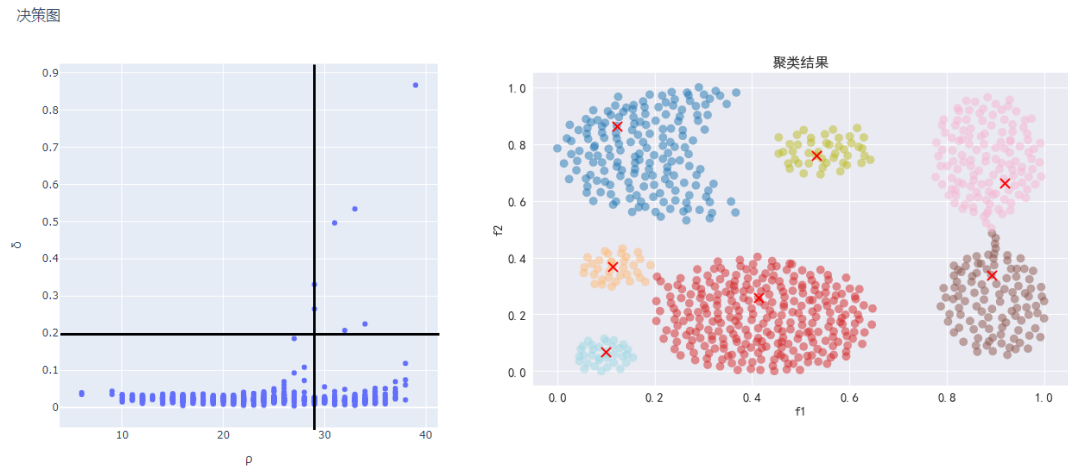


DBI=0.31481596929442923

可以看到样本被完全分为15类，且DBI得分不错。

3.Aggregation数据集

选取$d_c = 0.075$，根据决策图选取$\rho_v = 29, \delta_v = 0.2$：

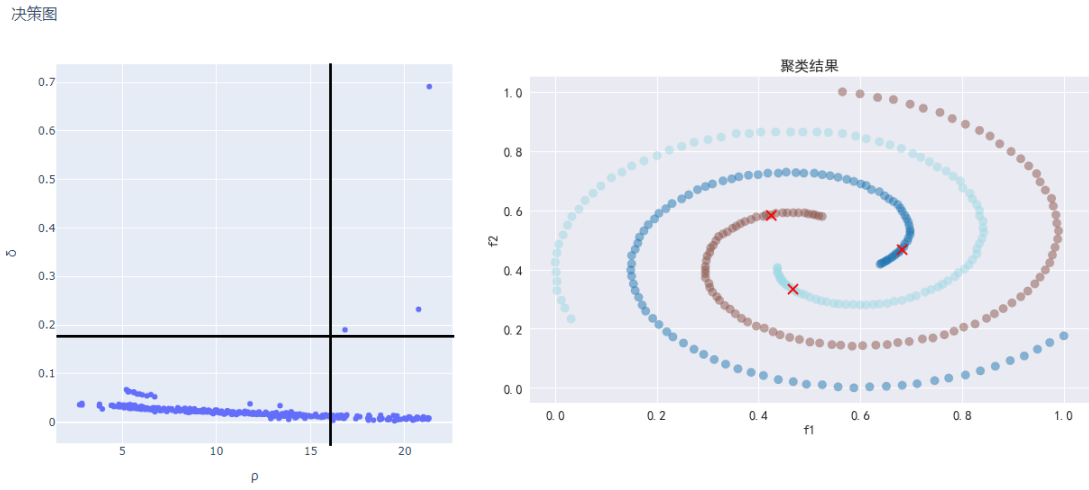


DBI=0.5037235117749277

可以看到样本被完全分为7类，且DBI得分不错。

4.spiral数据集

选取$d_c = 0.1$，采用高斯核，根据决策图选取$\rho_v = 16, \delta_v = 0.19$：

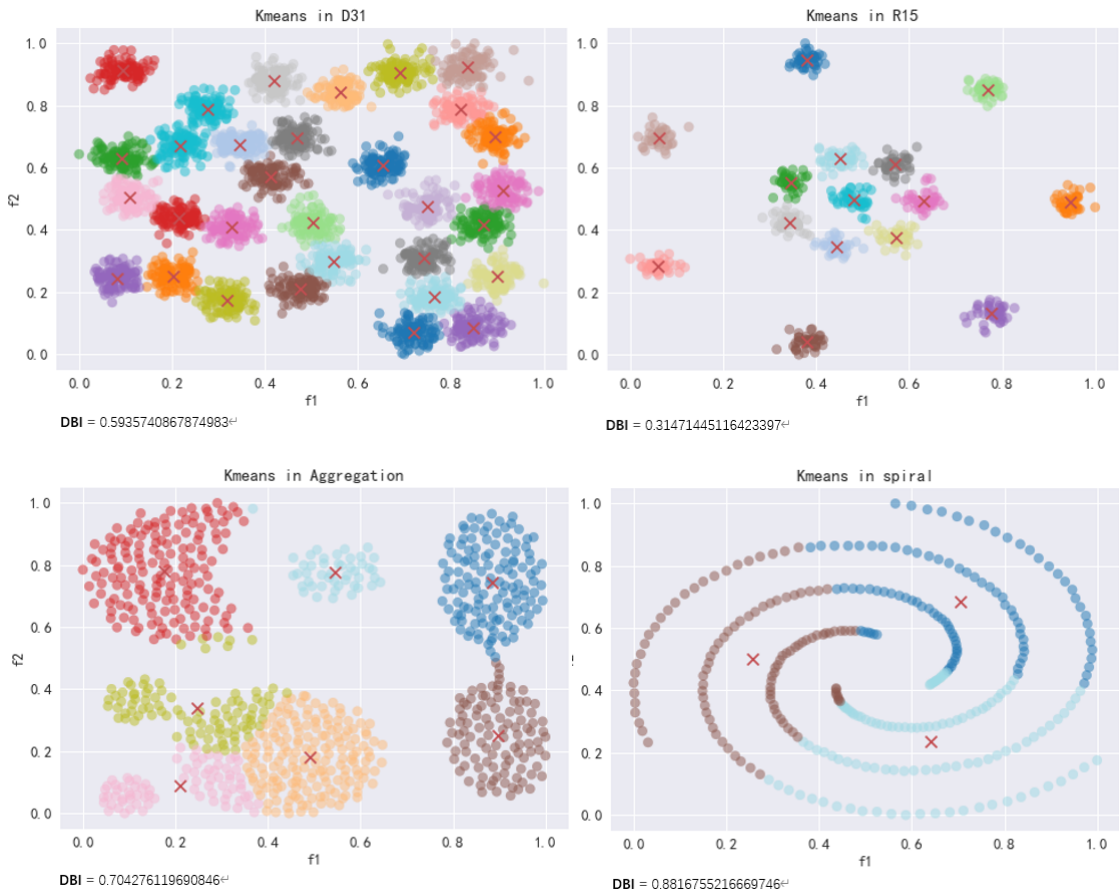


DBI=5.882022552277642

可以看到样本被完全分为3类。

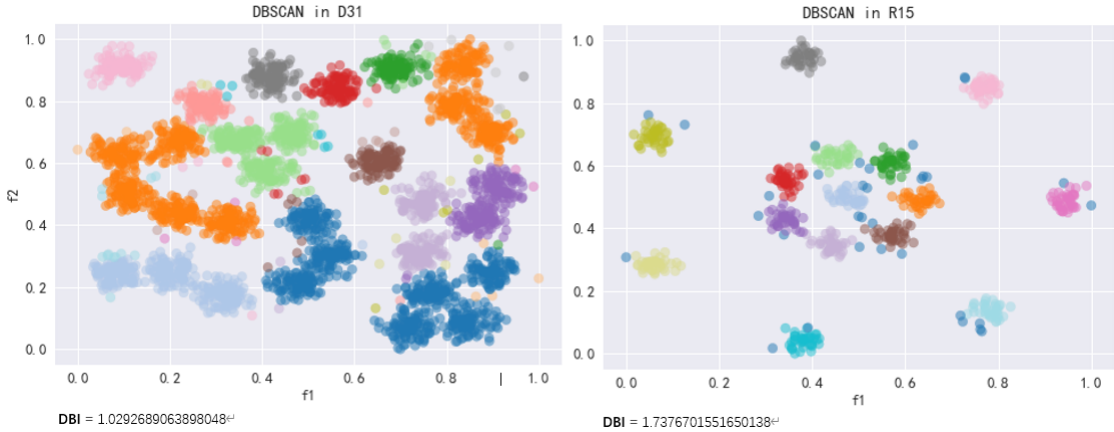
5.模型对比

(1)K-means



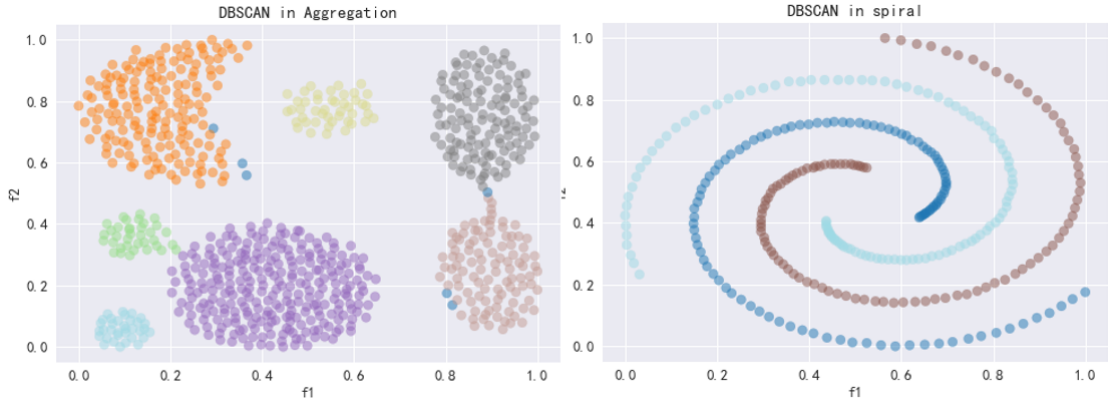
K-means可以较好的分类D31和R15，在Aggregation上表现不理想，在不采用核方法的情况下无法良好分类spiral。

(2) DBSCAN



DBI = 1.0292689063898048

DBI = 1.7376701551650138

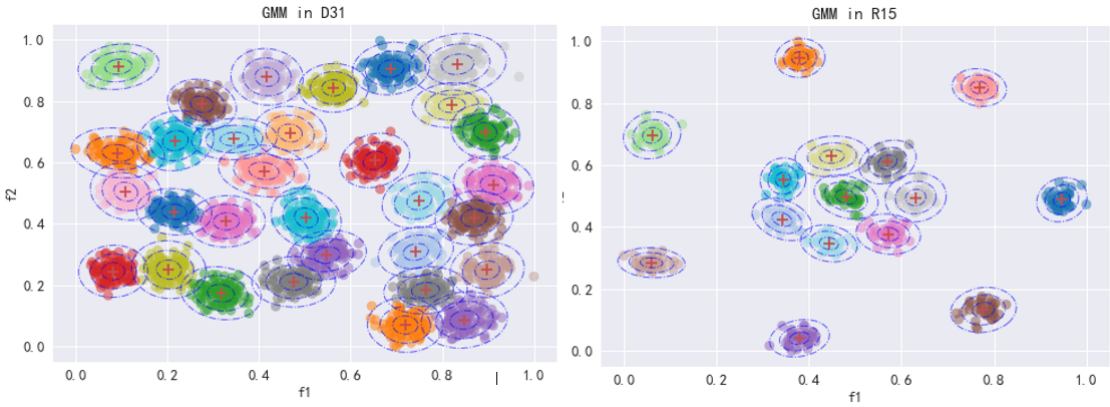


DBI = 1.101060801544115

DBI = 5.871563489504275

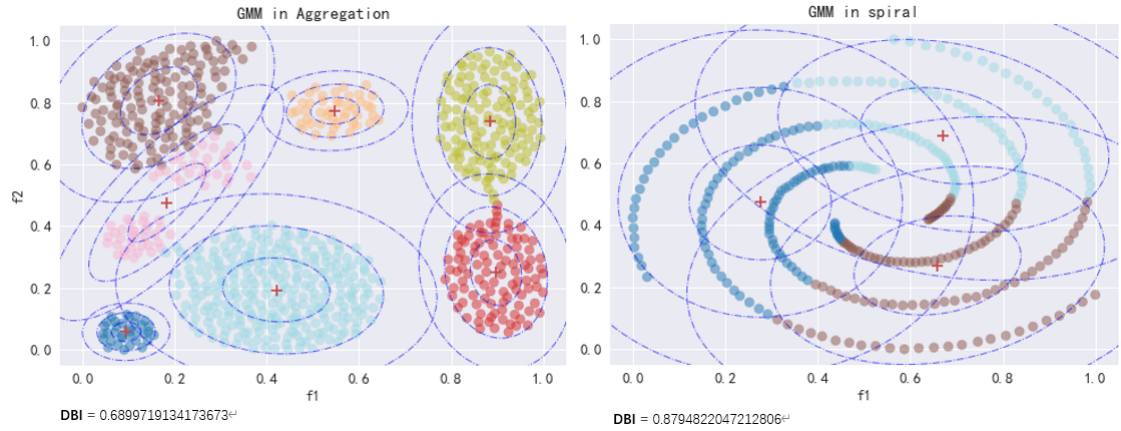
DBSCAN在D31和R15上的表现都不是很好，但可以正确分类Aggregation，在不采用核方法的情况下可以完全分类spiral。

(3)GMM



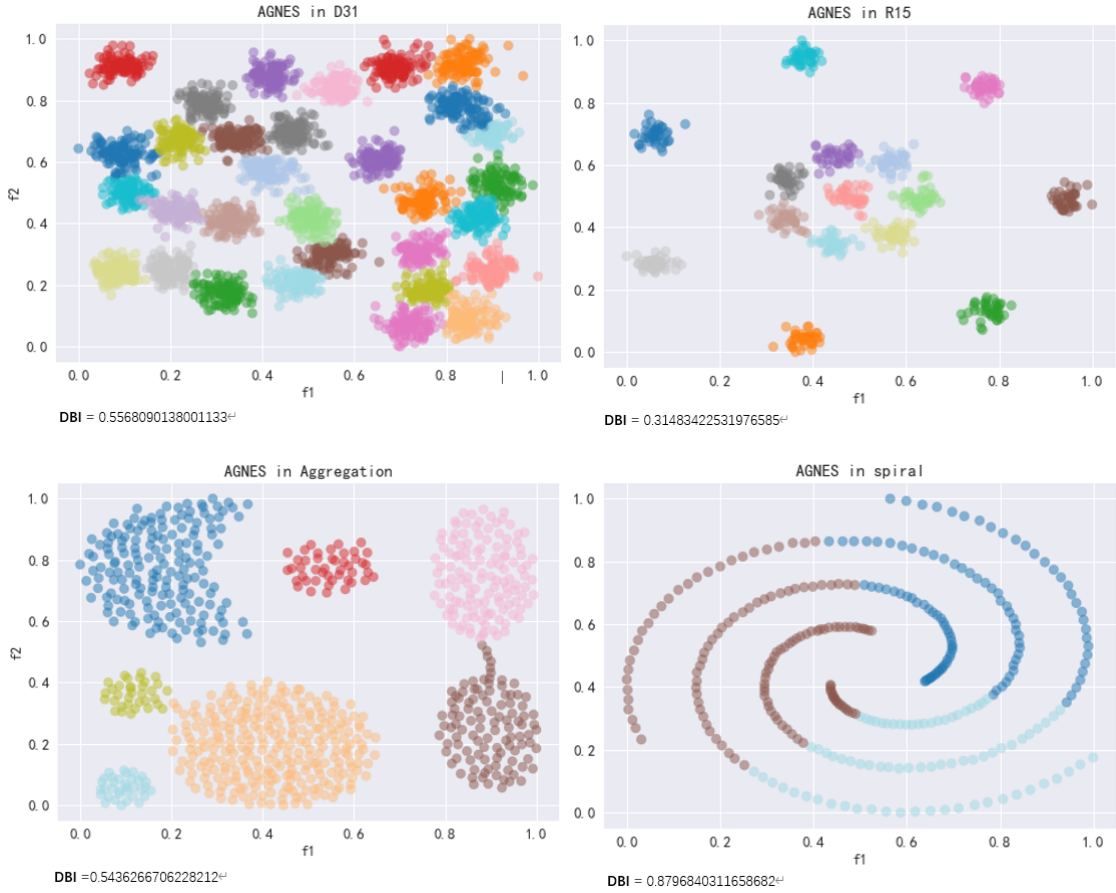
DBI = 0.5478456532987915

DBI = 0.31471445116423385



GMM和Kmeans表现相似，可以较好的分类D31和R15，在Aggregation上表现一般，在不采用核方法的情况下无法良好分类spiral。

（4）AGNES



AGNES可以较好的分类D31、R15和Aggregation，但在不采用核方法的情况下无法良好分类spiral。

七、结论

DPC聚类算法集合了K-means和DBSCAN的算法思想，即可以正确分类适用于Kmeans与GMM等需要计算类中心的球形/椭球形数据，也可以正确分类适用于DBSCAN等与密度相关的非球形数据（如螺旋形）。同时DPC算法时间复杂度为$O(n^2+n\log(n)+n/2)$，比一般的K-means算法的复杂度低。所以，DPC算法是一个能在多种类型数据集上都有良好表现的简单的聚类算法。