DAM Assignment1-1

吴桐欣 1652677

Task 1

算法流程

1. **计算p值。**对数据集中每一个点，根据`dc`值计算p
2. **计算delta值。**先将数据集根据p（ρ）从大到小排序，再对每一个点计算delta（δ）值。对每个点，只在排在其前面(p更大)的点中寻找离该点距离最近的点。同时在循环中计算第一个点(p最大)到所有其他点的距离，循环结束后取得最大值，作为第一个点的delta值
3. **计算gamma值。**对每一个点计算其gamma（γ）值，gamma = p * delta
4. **计算邻点/参考点。**对每一个点来说，其所属类别由离它最近的密度比它大的点决定，也就是以这个点为参考。由于计算delta值时已经找到了相应点，所以先顺便记录下来，留到后面用
5. **找到peak点。**转置数据集以获得一个gamma的数组。将数组从大到小排序，然后对这个数组求二次差分，并找到差值最大的位置（如果这个位置是第一个，则找次大的位置）。假设这个位置的下标是6，则gamma[6]就是7个peak点中最小的gamma值，作为gamma_limit。遍历数据集，根据gamma_limit找到所有7个peak点（因为数据集按p排序，gamma值大的点p也较大，因此不会浪费太多时间），并给peak点增加一个label维度，赋予各不相同的值。
6. **给所有点分类。**遍历数据集中非peak的点，对于每一个点，先找其参考点是否已经有label维度，若有则将其label值作为自己的label值，否则label值置-1

关键问题

1.如何计算delta值

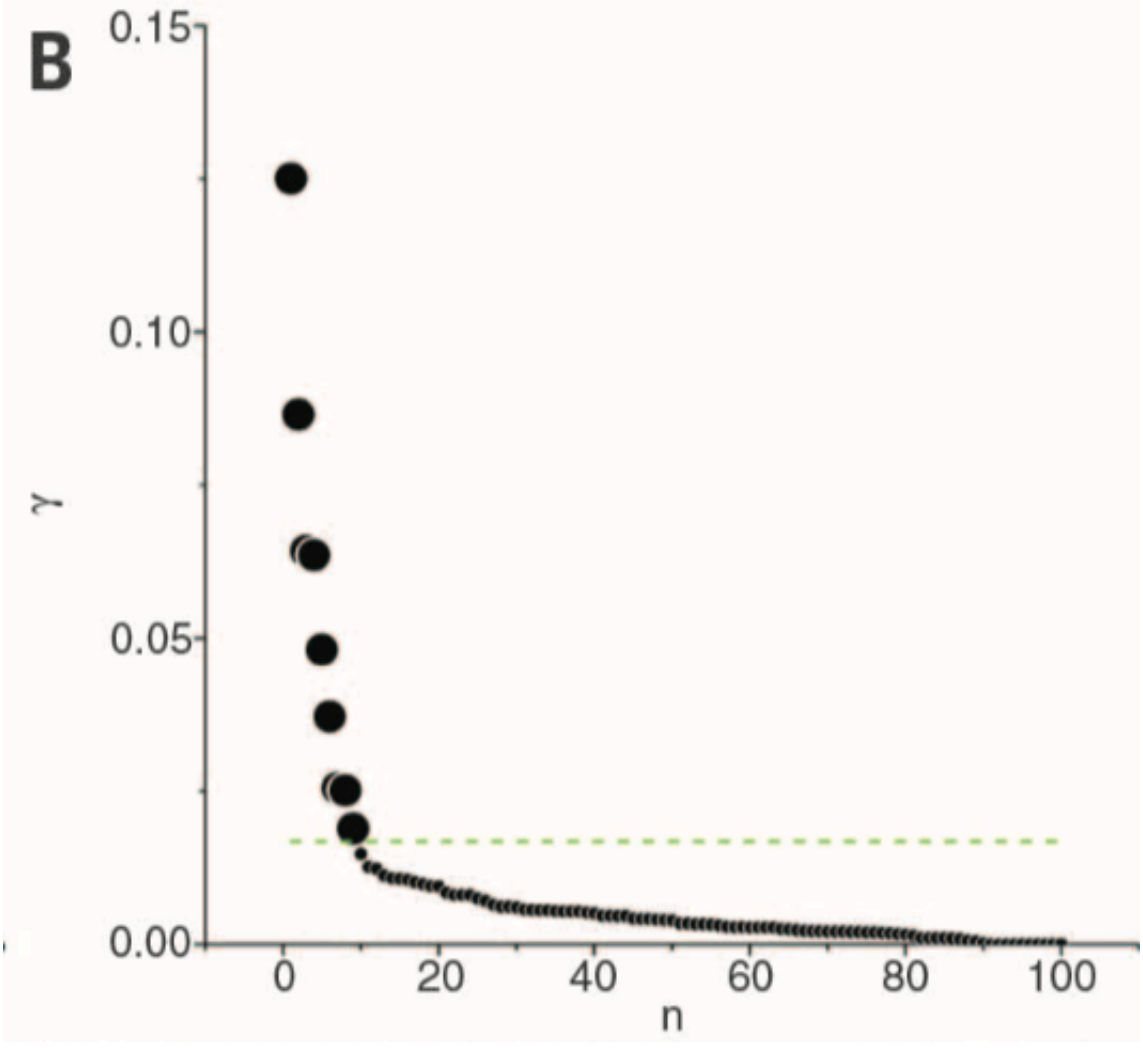
一开始我只是单纯地对每一个点找到离它最近的、密度比它大的点的距离。后来发现这样造成一个问题就是，如果相邻的两个点具有同样的p值，则这两个点可能同时具有很大的delta 值，并同时被选为peak点。

解决方案是，将点按照p值从大到小排序，即使是相同p值的点也会有一个先后顺序。对每一个点，只在排在它前面的点(即，大于或等于该点p值的点)中找到离它最近的点。如此一来，如果相邻的两个点具有同样的p值，则只有一个点具有很大的delta值，其余点将有很小的delta值，也就只有一个点会被选为peak点。

2.如何找到peak点

我希望算法能够自动找到所有peak点。根据decision graph，p值和delta值都要很大才行，因此我选择将二者相乘，能将两个参数合二为一进行排序。

> 论文中也给了一个图: A hint for choosing the number of centers is provided by the plot of $\gamma=\rho\delta$ sorted in decreasing order (Fig. 4B).



除peak点外，其他点的gamma值都相对小，因此通过求差分找到gamma值急剧变小的位置。起初我只求了一次差分，但发现在peak点之间的差值远大于其余点之间的差值，所以求二次差分能够较好地定位到图中的"拐点"。找到拐点后，大于拐点处gamma值的点都是peak点。

3.如何确定dc

> 论文中： 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.

即，所有点的"邻居"个数(p值)的平均数约为整个数据集的1%～2% 本数据集的大小是788，则平均数约为7.88～15.76 首先，我选取了1～10这个区间

```
dc: 1, average neighbours: 4.284263959390863
dc: 2, average neighbours: 18.055837563451778
dc: 3, average neighbours: 36.8756345177665
```

再选取1.1～1.9这个区间，发现1.3～1.9都是符合要求的

```
dc: 1.1, average neighbours: 5.479695431472082
dc: 1.2, average neighbours: 6.5786802030456855
dc: 1.3, average neighbours: 7.728426395939087
dc: 1.4, average neighbours: 8.954314720812183
dc: 1.5, average neighbours: 10.32994923857868
dc: 1.6, average neighbours: 11.736040609137056
dc: 1.7, average neighbours: 13.233502538071066
dc: 1.8, average neighbours: 14.718274111675127
dc: 1.9, average neighbours: 16.28680203045685
```

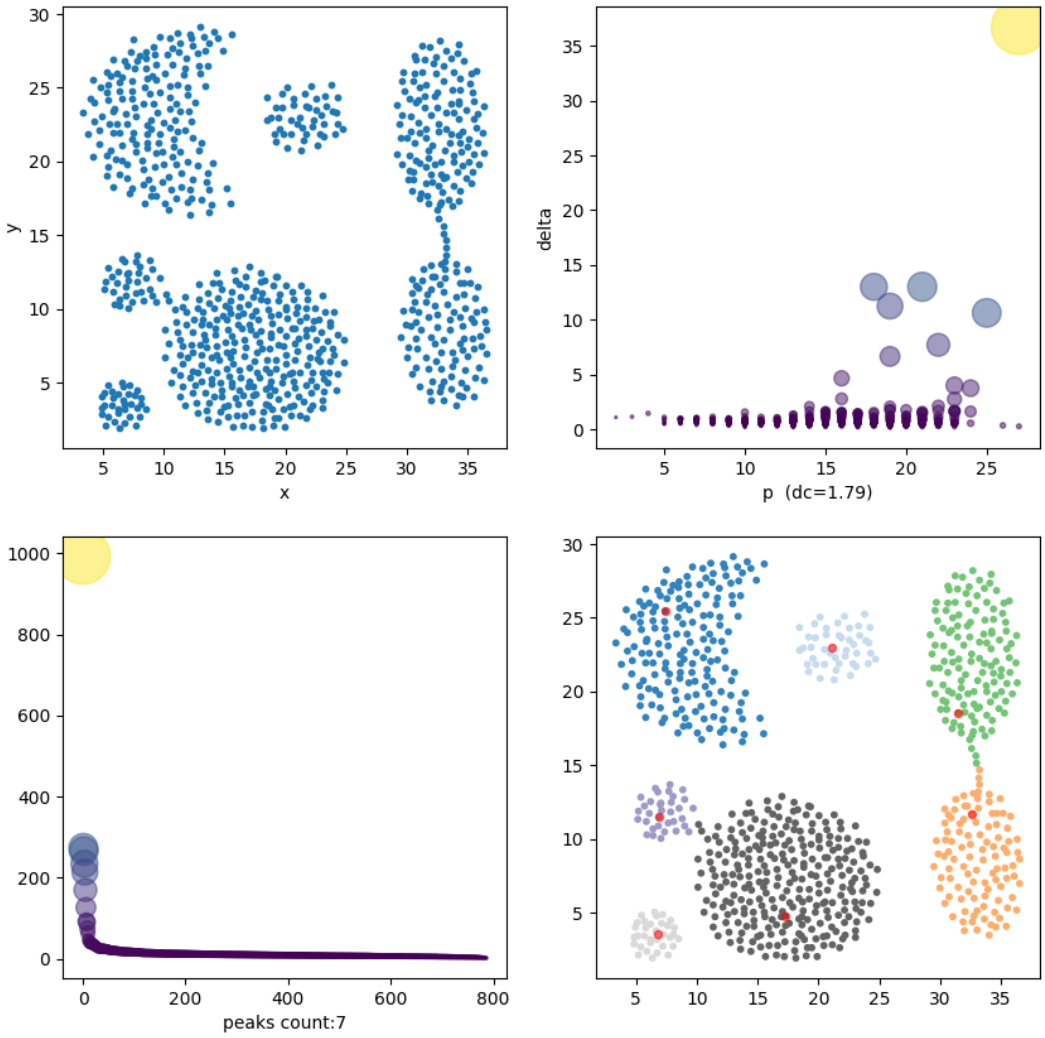
接下来则根据聚类结果比较谁的效果更好，发现dc=1.8时，能够找到7个peak点，并准确地对所有点进行分类。

dc	clusters	silhouette_score	calinski_harabaz_score
1.8	7	0.4932100128893331	1202.1167911276136
2.2	3	0.5236146973162428	1041.3034685761345

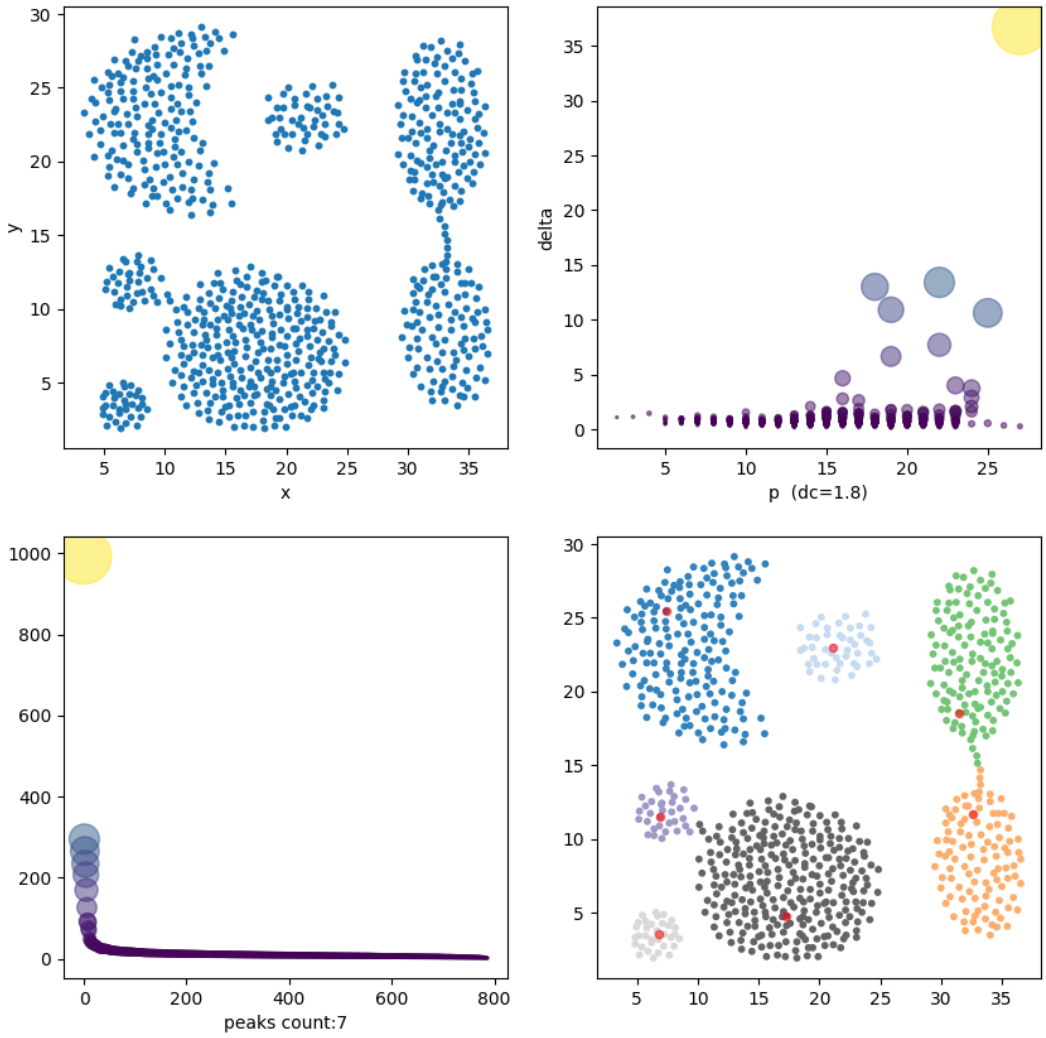
再细化一下，选取1.25～1.35这个区间，发现1.79、1.8的聚类效果最好，而且二者分类结果是一致的。

dc	clusters	silhouette_score	calinski_harabaz_score
1.79	7	0.4932100128893331	1202.1167911276136
1.8	7	0.4932100128893331	1202.1167911276136

dc=1.79



dc=1.8



4.如何进行分类

将数据集按照p值从大到小排序，依次计算参考点、分类。peak点最先确定label值。每个点都会以密度比自己大的点为参考，由于密度大的点先被分类，所以基本不会出现有点没分类的情况。

代码优化

1.减少遍历

优化前 每一点用一个数组存储，数组下标及对应值含义如下(neighbour是参考点的下标)：

```
0-x, 1-y, 2-p, 3-delta, 4-gamma, 5-label
```

p值、delta值、gamma值、neighbour值都是按顺序一次次计算的，也就是执行了好几次对数据集的遍历。计算delta值和neighbour值的代码几乎是一样的，但是一前一后分别进行了计算。 **优化后**

```
0-x, 1-y, 2-p, 3-delta, 4-gamma, 5-neighbour, 6-label
```

计算delta值的循环中，得到delta值的同时，可以得到gamma值和neighbour值。记录neighbour的值，在后面分类的时候用。

2.减轻循环负担

优化前 计算delta值时，对所有点两两配对计算距离，外层循环和内层循环都是对整个数据集进行遍历，是O(n^2)的时间复杂度 **优化后** 在p值计算后，将整个数据集进行排序。内层循环不必再遍历整个数据集，只需要遍历一部分，是O(1/2(n^2))的时间复杂度

3.减少排序

优化前 对数据集进行了三次排序：(1)计算p值后按p值排序；(2)选取peak点时按gamma值排序；(3)进行分类时按p值排序。 **优化后** 主要是在记录了neighbour值后，如果进行三次排序，前面记录的neighbour值就无效了。所以选peak点时不再对数据集排序，而是单把gamma值拿出来作为一个数组进行排序。

4.计算点与点之间的距离

优化前

此算法使用最简单的欧式距离，使用一个计算两点间距离的函数，用两层循环遍历数据集。

优化后

调用sklearn.metrics中提供的`pairwise_distances(X)`，即可快速计算得到数据集中任意两个数据点之间的欧式距离，用一个距离矩阵存储。

-