Task 1 实验报告文档

姓名：孙浩然

学号：1652714

论文实现过程简述：

这篇 sci2014 论文一开始说的还算是详细，告诉了我们需要想计算两个值，才能对数据点进行聚类，分别是：

1. Local density
2. Minimum distance from points of higher density / `sigama`

截止到这里，这篇论文还是很清晰易懂的，我很快就实现了这些内容。但是，这个算法后面的步骤都没有详细写在论文里啊！还需要在网上找到了论文附带的一篇 matlab 的代码才知道具体的步骤应该是什么样子的。

数据读取：

Task1的数据不存在预处理的问题，我简单说一下数据读取吧。Task1的数据处理起来很简单，从文件中一行行的读取出来就可以了。

有趣的一个地方是，读取出的每个点都是 `str` 类型的，为了把他们优雅的转化为 `float` 我还尝试过 `map` 函数，但是 `map` 函数的接受的函数指针参数不能为类型强制转换 `float`。

```
with open('./Aggregation.txt', 'r') as file:
  for line in file:
    point = line.strip().split(',')
    POINTS.append([float(i) for i in point])
```

dc的计算：

dc值的计算，论文里面写的不是很清楚，论文里面全部的说法如下：

> *As a rule of thumb, one can choose dc so that the average number of beighbors is around 1 to 2% of the total number of points in the data set*

表意并不是很明确，所以我最终根据网上的 matlab 代码找出了准确的计算方式。首先设 `T` 属于 1-2%，精确地值通过不断地调试参数，再最终确定。

具体的计算方法为：

1. 计算出任意两点之间的距离：`points_distance`
2. 将这些距离放在一个数组中排序：`d_i_j.sort()`
3. 选取第 `T`% 的点作为dc：`d_i_j[round(T*total_number/2*(total_number - 1))]`

局部密度的计算：

这个具体的计算过程我就不详细的写了，论文里面写的相当清楚了。有一点需要注意的是，在计算每个点的局部密度时，需要记录下最大的局部密度值，因为在后面计算 `sigama` 的时候，局部密度值最大的点的计算方法不太一样，在这里需要记录下最大的局部密度值，可以避免后面重复计算。

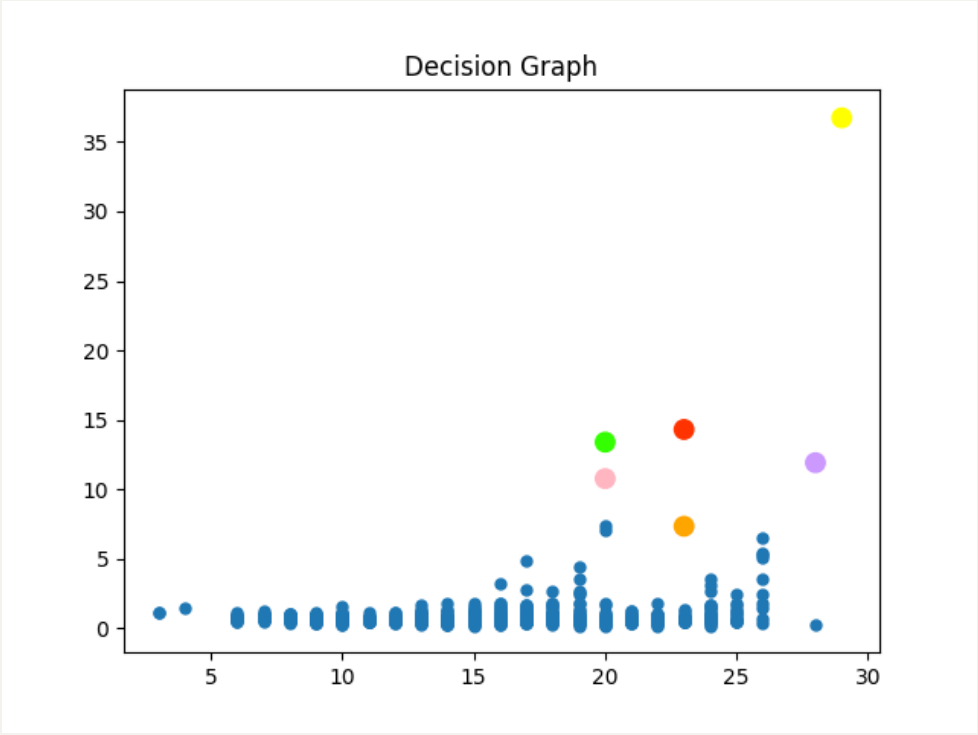
计算每个点最相近的点：

这句话我的表述可能不太妥当，换句话说：就是对一个点 i，所有局部密度比点 i 大的点中，距离点 i 最近的点。这一步我觉得可以算的上整个算法的核心了，正是因为这个数组，在选取每个 cluster 的中心点后，每个点才能找到属于自己的那一个 cluster。不过这个计算方法倒也挺简单的。记这个数组为 `min_distance_points` 。

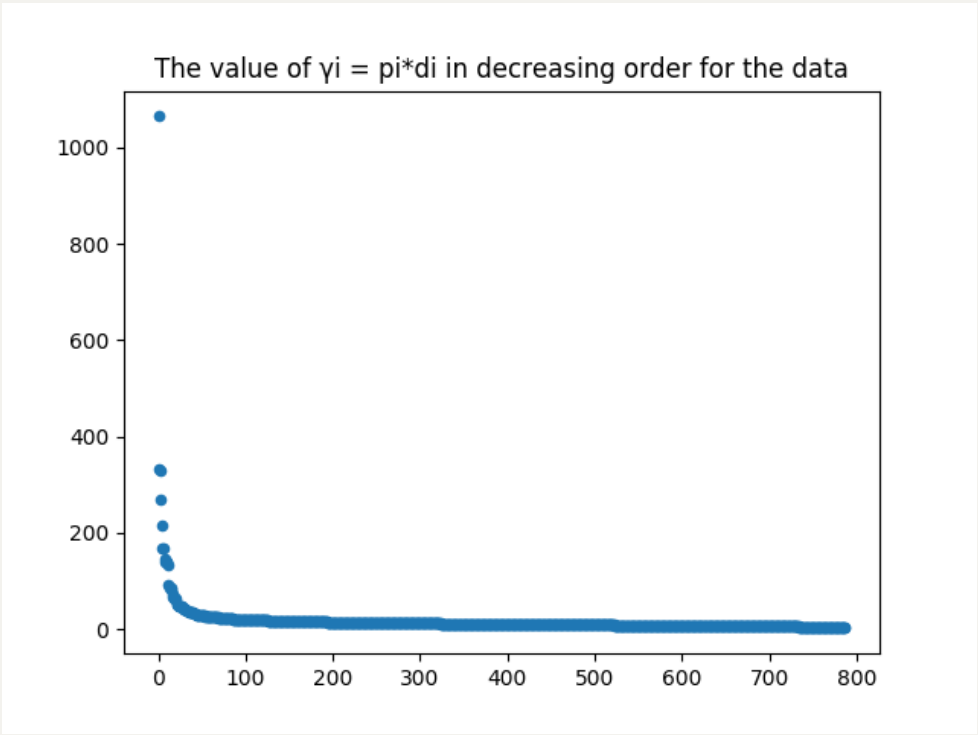
sigema 与 gamma 的计算：

sigema 的计算我就不说了，完全按论文来就可以了。

gamma 在论文中算是一个 `optional` 的辅助变量，是在 cluster 的中心点**无法用肉眼简单分辨出的时候**，用来判断中心点的。在这里，我觉得这篇论文就有点问题，这个过程还需要人为的干预，不是非常的科学。原本中心点应该是在下图的右侧靠上附近的几个点，但是由于这些点不太明显，我们无法直接看出，所以需要再计算一个 gamma。



gamma = local_density * sigema，可以绘制出下图，帮助进一步判断 cluster 的中心点：



在不进行人为干预的情况下，我们可以选择 gamma 最大的几个值最为中心点。

聚类：

这是task1的最后一个步骤，反而却比较简单，再上一个步骤中，我们已经选出了所有的 cluster 的中心点，接下来就是借助上文中 `计算每个点最相近的点` 这一小结中产生的数组 `min_distance_points` ，找出每一个点所属的 cluster 的即可。

按照 `local_density` 从大到小的顺序，遍历每一个点，针对点 i 按照以下规律进行递归：

如果点 i 不属于任何 cluster，那么点 i 就属于点 `min_distance_points[i]` 所属的 cluster

实验结果与分析：

整个论文的实现，有两个参数是可以进行不断调整的：

1. 分类数目
2. T值 (1-2%)

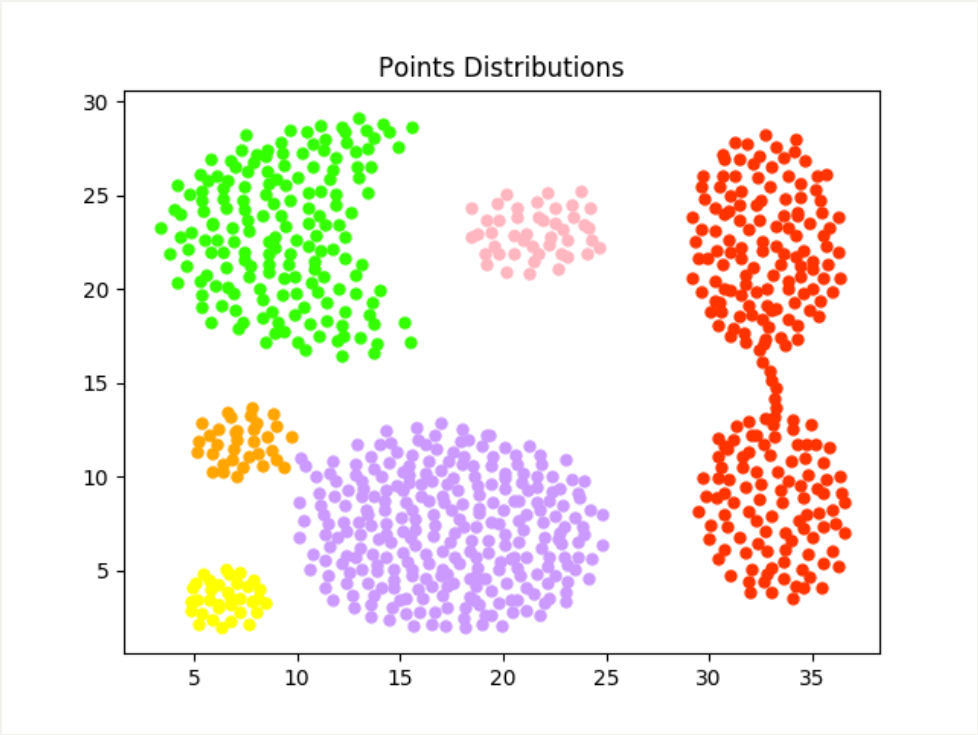
实验结果的情况随着这两个值的不断变化而变化。虽然经过长久的尝试调试参数，最终我都没有取出来非常符合期待的聚类结果，但是我个人觉得已经是较为合理的结果了。

在这两个值中，T值具体代表什么我一点都不清楚，所以我只能对他不断地变化，希望能得到一个更好的结果；但是分类数目我们可以明显从图中看出，分类数目应该在7左右，可惜的是，我没能成功的把它分成合理的7份。

T=0.02，分类数目=6

经过尝试，T=0.02的时候，聚类的结果较好。

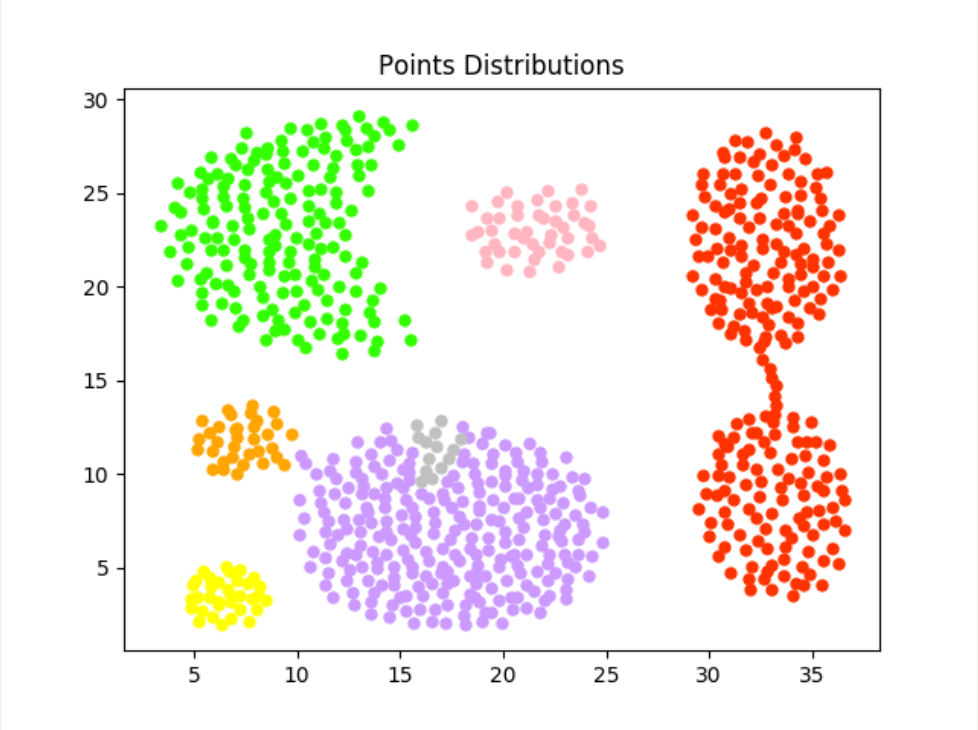
而在调试分类数目的过程中，我发现当分类数目为6时，得到了一个结果相对很好的方案，可以在下图中明显的看到效果。除了右侧红色的点，其他的点都很好的分成5类。此时的轮廓系数为0.42，也是我暂时得到的最大的轮廓系数。



T=0.02，分类数目=7

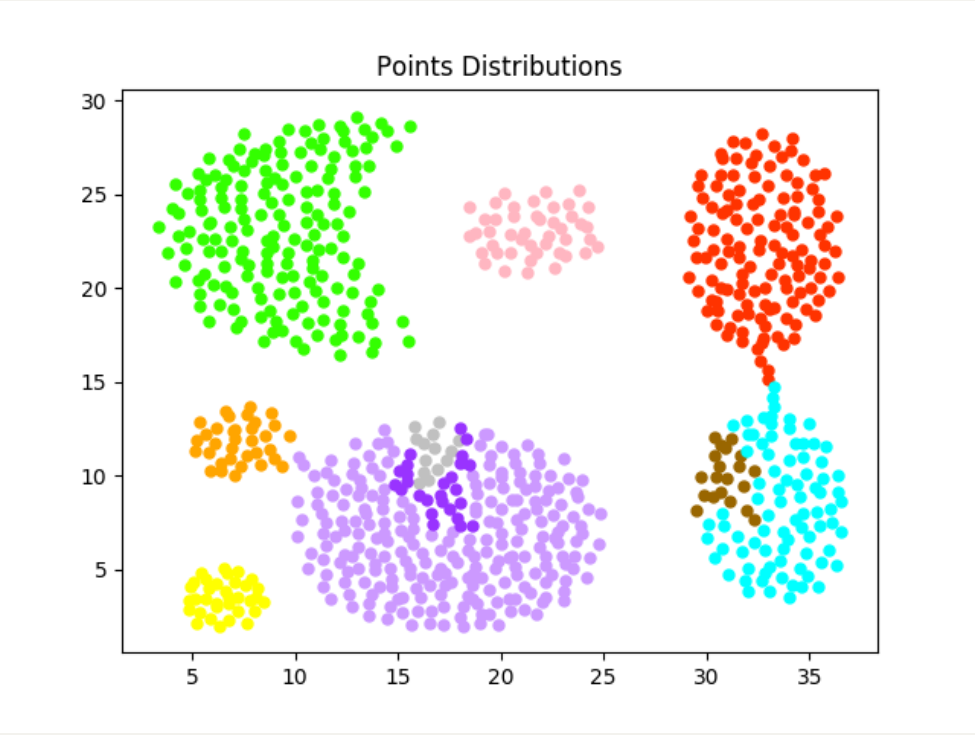
我本以为当分类数目为7时，右侧的红色的点能够相应的继续细分为两个cluster。但是很可惜的没有。

我仔细考虑了这个现象出现的原因，问题出现在**中心点的选择上**，论文的作者在论文中选择中心点时，没有完全把这个过程交给程序去做，而是通过人眼选择出了中心点。通过下图可以看出，程序选择的第7个中心点（灰色）本应该出现在红色的点中，却错误的出现在了粉色的点中，如果我们通过人工干预，跳过这个潜在的中心点，应该可以进一步扩大轮廓系数。



T=0.02，分类数目=10

下面这个图的轮廓系数和上图的几乎相同，虽然在这个图里面出现了很多不合理的聚类(深紫色、棕色和灰色)，但是值得注意的是，浅蓝色的点终于把红色的点分开了，如果可以人工干预这个程序的话，我会将分类数目选为7，然后将第七个中心点人为的设定为浅蓝色的中心点，应该就可以得到较好的聚类结果。



论文实现过程亮点与难点：

距离计算：

task1中的距离定义没什么好说的，一共就是二维坐标，两点之间距离公式，勾股定理计算就完了。但是值得注意的是，本论文实现的过程中，多次出现了距离计算的需求。经过思考，虽然每次都把每个点到任意点的距离，只会增加时间复杂度的常数。不过，我们这个数据集非常小，只有788个点，所以我在实现过程中，干脆把任意两点间的距离记录在了 `points_distance` 数组中，其中有效值形成了一个上三角矩阵，为了调用方便，无效值使用 -1 填充。用空间换时间。

将每个点分入 相应的cluster：

这个步骤其实理解起来很容易，但是在代码实现的过程中，会遇到一些问题，如果不注意，很难看出错误来。由于这个过程是从密度高的点一步步向外面辐射，一个问题是，如果两个点挨得足够近，互相是距离最近的点怎么办？那么这个地柜求解就可能出现无限递归。

解决的方法倒也容易，虽然不是很好想。在计算距离每个点最近的点时，我们规定，只有局部密度大于它的点，才能成为距离它最近的点，这样就有效的避免了循环递归的情况。

分类数目的选取：

分类数目的选取是本论文实现最困难的地方之一，将所有的点画出到坐标系中后，可以明显的看出应该分为7类左右，所以可以在参数调试的过程中不断地尝试，但是不禁让人好奇，如果是更多维度的数据呢？那么应该如何判断数据的分类数目？不停的调整参数，直到轮廓系数最大嘛？

T的选取：

这个 T的值，我一点头绪都没有，只知道它的大致范围，网上也没有找到较好的解释，问了问几个同学，都是在没有任何道理的调整尝试，我觉得这样非常不科学，但是也没有别很好的方法，我分别在 分类数目=6, 7时，尝试了 T = 0.01/0.015/0.02，发现其中0.02的效果最好，于是选择了0.02。

中心点的选取：

我个人觉得，这个论文最大的问题就在于中心点的选取上，它没有一个完善的寻找中心点的算法，或者说，它的这个密度聚类算法的核心不在于寻找中心点。中心点的确定可能又是一个新的算法了。同分类数目的选取一样，在二维的平面上，还可以通过肉眼观察，辅助选取中心点，但是如果是多维的数据呢？那么中心点的确定就只能全部交给程序来完成。虽然论文的作者提供了 gamma 这个参数来辅助选取中心点，但是通过上文中的实验结果也可以看出，这个选取中心点的方法也不是那么完美，还有一些缺陷。