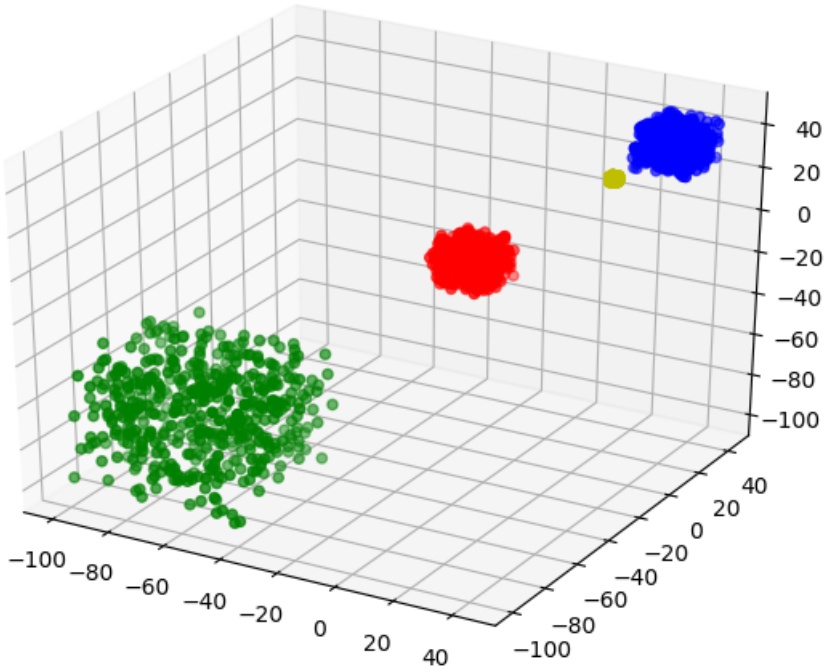
层次聚类实验

1811451 杨航

基本要求(4')：绘制聚类前后样本分布情况

根据试验要求，我使用numpy库中的randomint方法生成了4个随机分布的样本族作为训练样本。共2000个样本，每个族分别有500个样本。其分布如下：



在代码中因为每次生成样本和计算距离的时间较长，我直接将数据集和初始距离计算好了之后保存在文件中直接读入。

(1) 实现 single-linkage 层次聚类算法；

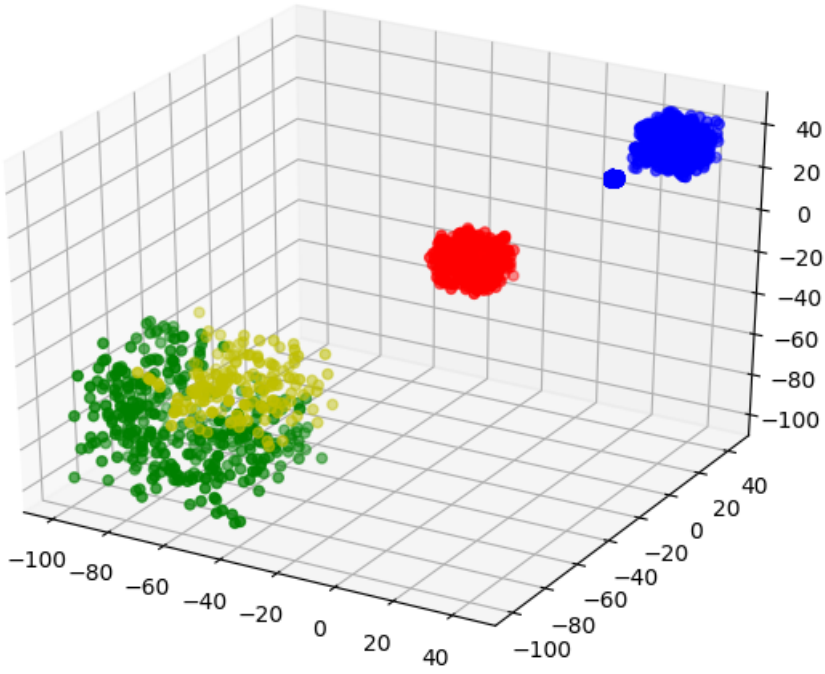
- 最短连接的要领在于，每两个簇之间只考虑距离最短的两个点

  我使用了一个上三角矩阵，节省存储空间和寻找最短连接的时间开销

  伪代码思路：

  1. 遍历每一个样本，找出这个和这个样本距离最小的样本，并把这两个样本的信息和距离保存下来
  2. 再从1 的结果中找出距离最短的，作为这一次遍历中得到的距离最短的两个簇
  3. 把这两个簇合并
  4. 进行下一轮迭代。

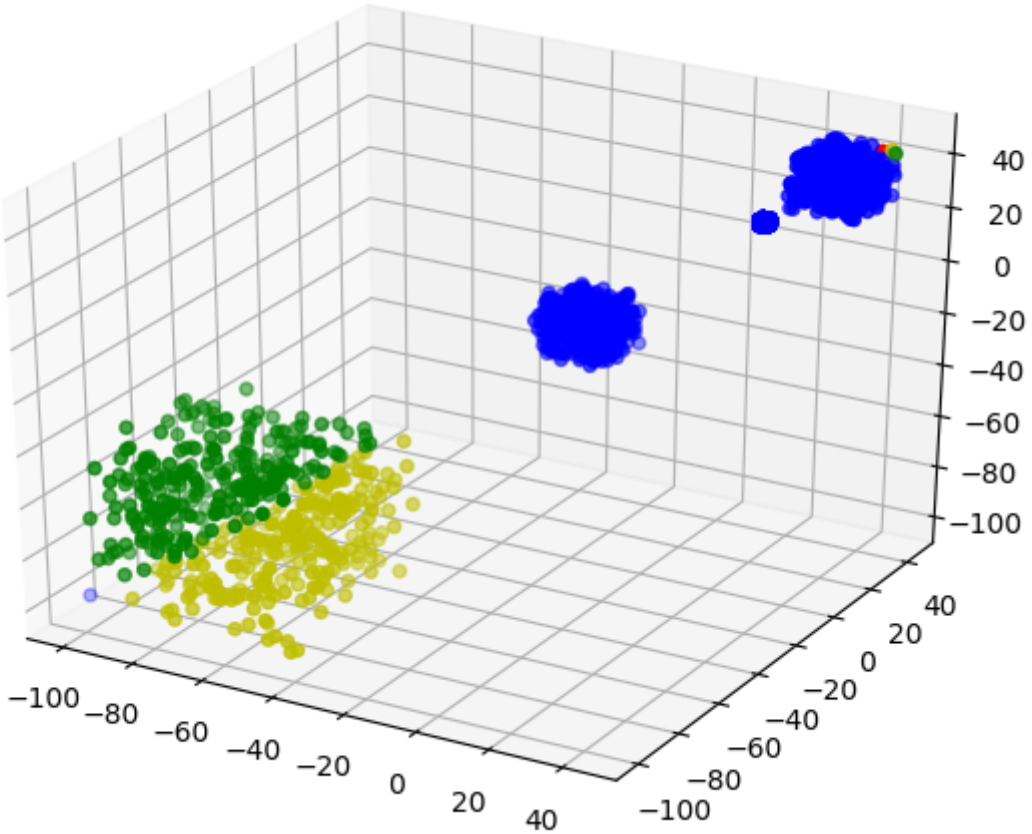
结果：



因为数据集分布的原因，比较小的哪一团样本因为和旁边的样本距离较近，所以被分在了一起；比较大的那一团样本又因为比较分散，而被聚集成了两类。

(2) 实现 complete-linkage 层次聚类算法。

全连接的算法里面，核心是找到所有两两聚类之间的最大距离，再从中找到最近的两个，并把他们合并。其他和单链接一样。



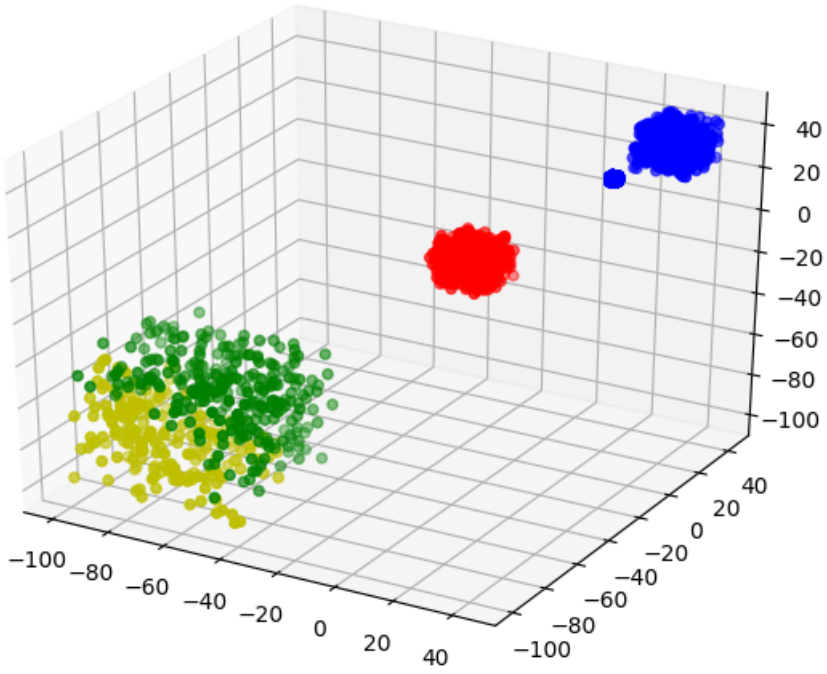
因为数据分布的特点，全连接聚类的算法不是很理想。

中级要求(1')：实现 average-linkage 层次聚类算法，绘制样本分布图。

平均距离算法的思路和上面类似，区别在于，每一次要计算两个样本簇中每个点两两之间的距离再求平均，

- 优点是：理论上可以更加客观的描述两个簇的距离。
- 但是缺点是计算量增大，足足跑了我一个下午的时间。

结果如下：



提高要求(1')：对比上述三种算法，给出结论。

最短距离算法：

- Single Linkage的计算方法是将两个组合数据点中距离最近的两个数据点间的距离作为这两个组合数据点的距离。这种方法容易受到极端值的影响。两个很相似的组合数据点可能由于其中的某个极端的数据点距离较近而组合在一起。
- 这种算法很容易受到噪音的影响。
- 计算量最小，速度最快。

全连接算法：

- Complete Linkage的计算方法与Single Linkage相反，将两个组合数据点中距离最远的两个数据点间的距离作为这两个组合数据点的距离。Complete Linkage的问题也与Single Linkage相反，两个不相似的组合数据点可能由于其中的极端值距离较远而无法组合在一起。
- 但是这种算法在一定程度上革除了噪音的影响。
- 需要对比的量增加，速度变慢，但是没有均值法慢。
- 但是在我的数据集里面，表现得效果最不理想。

平均值算法

- Average Linkage的计算方法是计算两个组合数据点中的每个数据点与其他所有数据点的距离。将所有距离的均值作为两个组合数据点间的距离。这种方法计算量比较大，但结果比前两种方法更合理。
- 缺点就是太慢了，跑一次三四个小时。
- 但是在我的数据分布中，它的表现并没有比最短距离算法好很多。可能在更正常的数据集上面会更加体现出其优势。