基于 DBSCAN 和 KMeans 的行人聚类

自动化钱 001 李艺涵 2206123627

一. 背景

本次作业分别利用 DBSCAN 和 KMeans 算法对 student003 数据集中的行人逐帧进行聚类，并生成相关的 gif 文件进行比较。

DBSCAN 和 K-Means 是两种常见的聚类算法，用于将数据点分组成具有相似特征的簇。尽管它们在方法和假设上有所不同，但都在数据挖掘和机器学习领域广泛应用。

DBSCAN 是一种基于密度的聚类算法，通过将数据点分配给高密度区域形成簇，并通过低密度区域将簇分开。相比于 K-Means，DBSCAN 的一个主要优势在于它能够发现任意形状和大小的簇，而不需要预先指定簇的数量。它对于噪声点具有鲁棒性，并且能够在不同密度的区域中有效地识别簇。但是 DBSCAN 对于如邻域大小和最小样本数等参数的选择比较敏感。

K-Means 是一种基于距离度量的聚类算法，它将数据点划分为预先指定的 K 个簇。K-Means 的优势在于其计算效率较高，易于理解和实现。它适用于具有球状簇形状的数据集，并且对于大型数据集也具有良好的可扩展性。但是 K-Means 需要事先指定簇的数量 K，这对于某些情况下的聚类任务可能是一个挑战。

在比较这两种算法时，需要考虑问题的特点和数据集的性质。如果数据集中存在不规则形状的簇或噪声点，并且无法预先确定簇的数量，那么 DBSCAN 更为合适。另一方面，如果数据集中的簇形状较为规则且簇的数量已知，那么 K-Means 更为合适。此外，对于大型数据集，K-Means 可能更具可扩展性。

二. 工具及相关设置

本次作业使用 python 工具包 sklearn 及 pandas 在 Jupiter Notebook 中完成，运行环境为 anaconda 虚拟环境。所使用的代码及生成的 gif 已上传至 GitHub，可在以下链接中查看：
https://github.com/YihanLi126/Group-Discovery

三. 聚类过程

1. 数据集处理

由于所给数据集列分割空格不规整，重新读写并添加列标题生成新的 txt 文件以便 pandas 读取。随后利用 pandas 读取数据为 csv，如下所示：

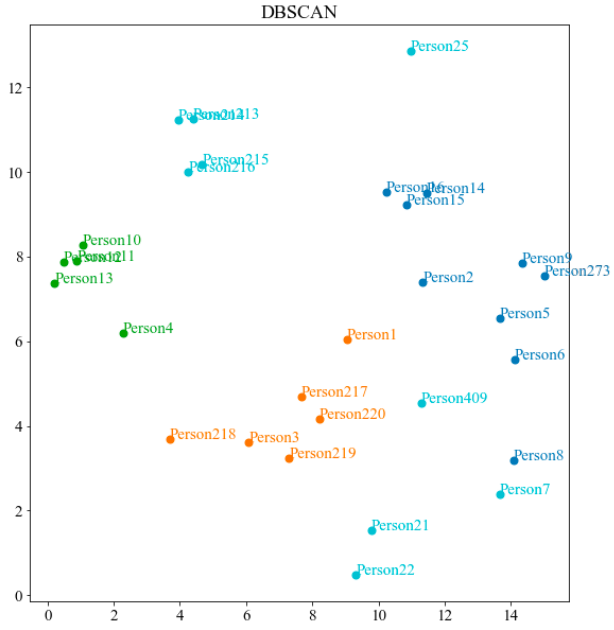
	timestep	ID	X	Y
0	0.0	1.0	9.050000	6.038093
1	0.0	2.0	11.344069	7.398454
2	0.0	3.0	6.082442	3.603763
3	0.0	4.0	2.273023	6.205155
4	0.0	5.0	13.680232	6.539279
...	...	...	...	...
17839	5350.0	299.0	14.368663	5.450513
17840	5350.0	338.0	3.697241	11.511278
17841	5350.0	339.0	3.754487	12.187163
17842	5350.0	375.0	11.913167	11.273096
17843	5350.0	404.0	11.272090	12.968058

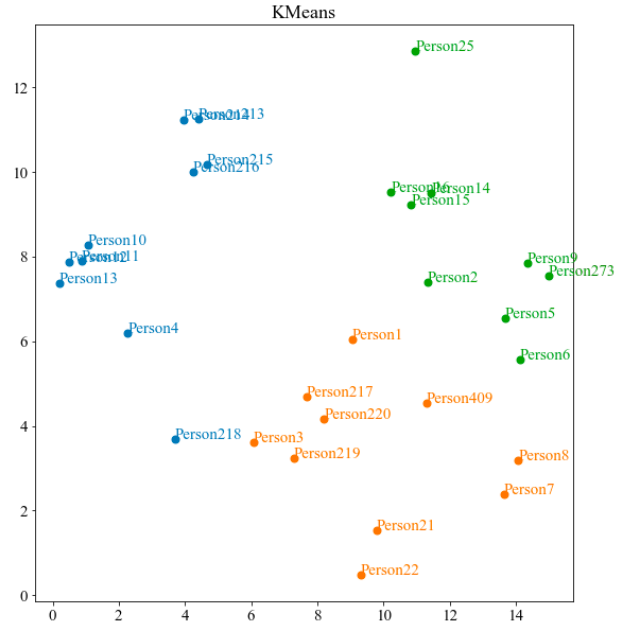
17844 rows × 4 columns

为了便于后续聚类处理，自定义 Person 类将以上数据按照时间戳分别于一个以 python list 为元素的 numpy array 中。Person 类别包含了 ID 以及位置信息，其中位置信息以二元组的形式存储。

2. 设置 DBSCAN 以及 KMeans 聚类方法

设置 DBSCAN ϵ邻域为 2.5，MinPoints 为 5；设置 KMeans 聚类类别为 3 类。提取出每一帧各个 Person 的位置信息分别用两种方法进行聚类，给出第一帧效果图如下：





可以看到，在该情况下，DBSCAN 聚类中出现误差，将距离较远的点聚为一类的（图中浅蓝色点），而 KMeans 较好地完成了聚类，出现了设置的 3 个类别。

3. 生成 gif 文件

将以上操作逐帧重复，利用 matplotlib 生成相应的 gif 文件

四．结果评估

经过多次对于 DBSCAN 参数的调整，观察并证实了 DBSCAN 对于邻域大小和最小样本数这两个参数的高度敏感性；同时在本次任务中 DBSCAN 容易出现将上述将距离较远的点聚为一类的错误。经过对比，发现在本次聚类任务中 KMeans 的效果优于 DBSCAN，分析是由于 DBSCAN 基于密度聚类，将簇定义为由密度可达关系导出的最大的密度相连样本集合，而本次任务中行人密度不均匀、聚类间距差相差较大，加之所选用的参数不一定是最优的，导致 DBSCAN 聚类效果较差。

五．总结

在本次作业中，通过代码搭建 DBSCAN 和 KMeans 算法运行框架并进行测试，我对二者的工作特点以及适用情况有了进一步的认知，有了较大的收获。