北京郵電大學

Beijing University of Posts and Telecommunications

《人工智能实验报告》

实验一：站点聚类

题　　目：	站点聚类实验
学　　院：	计算机学院
小组成员：	娄司宇
成员学号：	2019211452
小组成员：	田文阳
成员学号：	2019211438
小组成员：	姚昊岩
成员学号：	2019211446

站点聚类实验

1 运行环境

1.1 硬件信息

- CPU：Intel i7-9750H
- 内存：16G

1.2 操作系统及软件

- 系统：Ubuntu 20.04.3 LTS
- 编程语言：Python

2 操作指令

- 安装`scikit-learn`,`numpy`,`pandas`,`geohash`

```
pip install scikit-learn
pip install numpy, pandas
pip install python-geohash
```

- 运行程序

```
python process.py
```

3 数据预处理与算法分析

3.1 数据预处理

本次实验对站点进行聚类，主要需要的数据属性是经纬度坐标与吃水量，而吃水量有许多失真项(都为0)，所以首先要对这部分数据进行填充。我们采取的填充方法也比较简单，即遇到吃水为0时，以该数据前面出现的最近的非0数据来填充。

当船航行到锚点停泊或在LNG站点运输时，船应该处于静止状态，因此在聚类前，我们筛选出所有速度小于等于1的记录来作为输入。

3.2 算法分析

3.2.1 地图分割

因为整个数据共近800W条数据，受限于物理内存的限制并不能直接进行聚类分析；因此首先采取`geohash`的方法将地图划分为1024个部分，然后可以对每块先分别聚类，再对第一次聚类结果进行聚类，防止在分块边界的点被分割掉。

3.2.2 聚类算法

因为给出的数据集中含有很多噪点，若选用基础的`KMEANS`算法，将会在很大程度上影响准确率，因此我们选取以密度作为聚类标准的`DBSCAN`算法；与之类似的还有`OPTICS`算法，但是`OPTICS`算法运行速度太慢，难以应对上万条数据的处理，因此权衡之后我们选取`DBSCAN`算法来进行聚类分析。

因为`DBSCAN`算法需要计算邻接矩阵，因此需要$O(n^2)$的内存空间，因此需要耗费大量内存。为了减小内存使用，我们对每个块内数据进行处理时，也要进行抽样。若一个块内数据小于等于36000条数据则直接进行聚类，否则随机选取36000条数据进行处理。

第一次聚类出结果后，分别对LNG站点结果和锚点结果进行第二次聚类，得到的结果作为聚类结果，并计算出每个类的中心点。

3.2.3 分类方法

对聚类出的结果还要进行分析，判断出哪些是普通的锚点，哪些是LNG站点，LNG站点又要判断是出站还是入站。判断站点主要依据的就是吃水量的变化。因为都是LNG船舶，所以同一艘船吃水量发生波动的点应该就是发生了油量变化，我们认为该船此时在LNG站点，而吃水量增加即出口油，认为该点是出站；吃水量减少即进口油，认为该点是入站。

先要判断吃水量发生变化的点属于哪个类中，遍历已分类的中心点到该点的距离，若是小于1km即认为在一个类中，反之将其认为噪点。

我们对吃水量发生变化的点所在类打标签，若是吃水量增加就+1，吃水量减少就-1，最终根据标签进行分类。当标签值大于3时认为是出站，小于-3时认为是入站，其余的认为是锚点。

4 代码运行结果

4.1 运行截图

```
➜ LNG-clustering git:(main) X /bin/python3 /home/llsgyn/myrepo/2021Fall/LNG-clustering/process.py
        mmsi   timestamp  status  speed        lon        lat  draught              geohash
1  205421000  1620717669       0      9 -92.665520  24.534849       90  5669001245836436570
2  205421000  1620719003       0     10 -92.669563  24.539316       90  5669001259327562389
3  205421000  1620721234       0      9 -92.676865  24.547518       90  5669001420553470116
4  205421000  1620721254       0      9 -92.676918  24.547583       90  5669001421247791905
5  205421000  1620723580       0     10 -92.683929  24.557182       90  5669001115044131374
total time: 509.9952511000156 seconds....
```

从上图可以看到，运行总时间在500s左右（上方输出是文件头信息）

4.2 性能分析

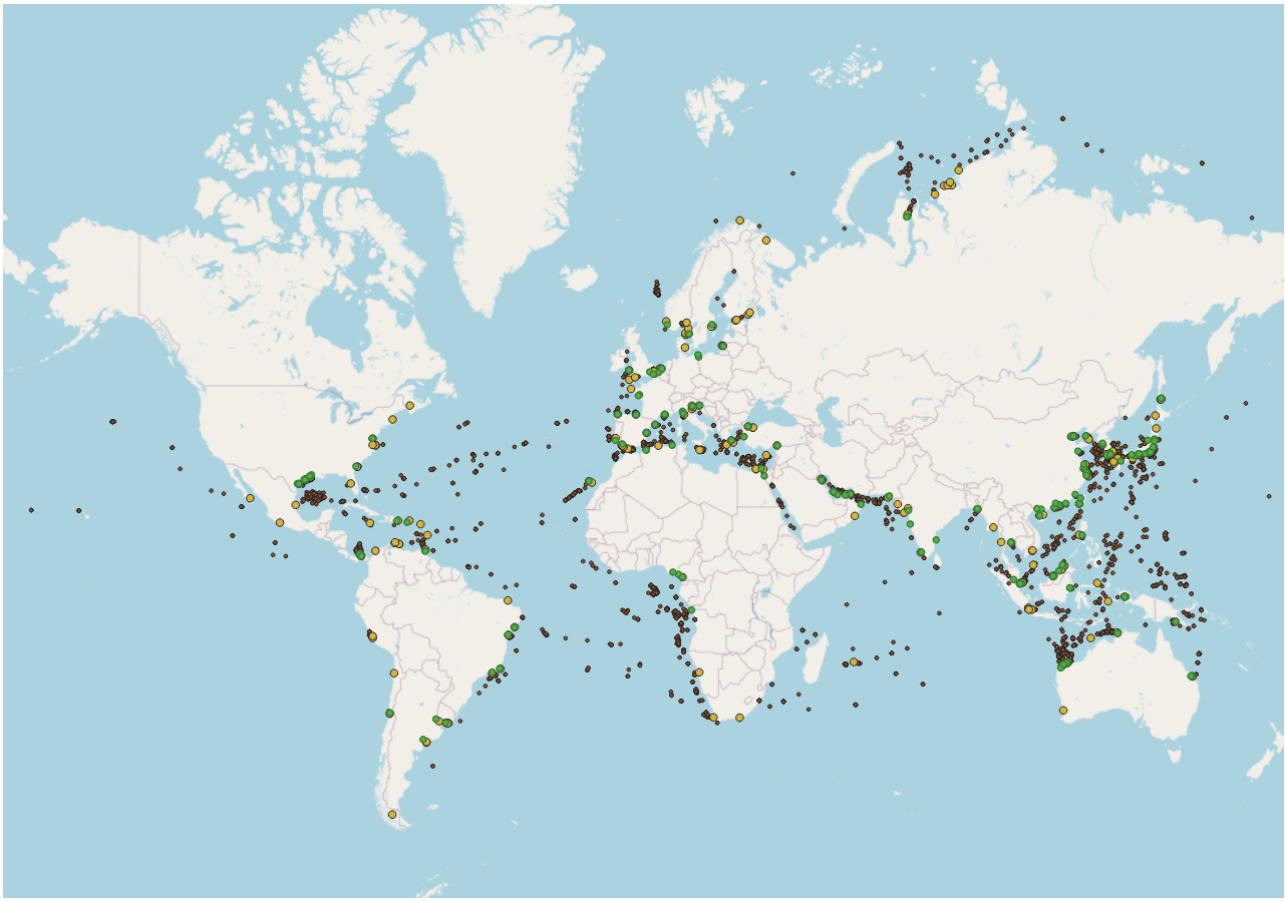
本次实验内存性能耗费在于`DBSCAN`算法计算聚类模型时，而`scikit-learn`提供了`n_jobs`参数来进行并行计算，官方文档对其描述如下：

> The number of parallel jobs to run. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors.

因此当`n_jobs=12`时，即可实现多线程最大化利用CPU进行并行计算。

而时间性能主要在于计算`geohash`值，`pandas`的`apply`函数并不支持多线程，因此在此处会花费大量时间。

4.3 结果分析



我们对聚类结果命中率可以进行一个简单的测试，假设给定数据集中所有吃水发生变化的点都是LNG站点，然后比较这些点中有多少在我们聚类结果中。经过我们简单测试，我们算法的命中率在85%左右，毕竟有些吃水发生变化的点也并不是LNG站点，因此90%的结果已经足够令我们满意。

除此之外，我们还在谷歌地图中查询我们的聚类结果，发现大部分聚类中心点的确是在LNG站点附近。

具体结果如上图所示，绿色点代表LNG站点，黄色点代表锚点，棕色小点是原始数据1%绘制出。最终结果是172左右个LNG站点，200个左右的锚点。为了追求LNG站点的准确度，我们将`DBSCAN`算法的`eps`参数设置的相对较小，因此会聚类出较多锚点。

5 遇到问题及解决

5.1 数据划分

最初进行聚类时打算随机抽样一部分数据直接进行聚类，但是受限于内存因素，直接抽样仅能计算对10W条数据进行聚类。我们认为这样计算出的结果与实际值可能偏差较大，因此采取`geohash`对所有数据进行划分，将一个复杂问题拆解为相对简单的多个子问题，然后进行聚类后对结果再次聚合。

5.2 物理内存限制

因为`dbscan`算法要耗费大量内存空间，我们在实验之初一直遇到因为内存不够而导致系统将程序强制终止的情况。为了解决这个问题，我们尝试过更换与`DBSCAN`相似的`OPTICS`算法，但是`OPTICS`算法的问题在于计算速度过慢，我们最终还是选取`DBSCAN`算法。为了解决内存不够的问题，采取了两个办法。第一：购买内存条，内存不够就增加。第二：减少数据输入，每次抽样最多只选取36000条数据进行处理，此时刚好可以在16G运行内存的极限状态下运行代码。

6 实验总结

实验总共花费时间大概在5天左右，花费了大量时间在程序运行与参数调整方面。本次实验应该是上大学以来做的最具有探究性质的实验了。相比与其他实验给出既定解决方案，只需要我们自己实现代码，本次实验只有输入数据与实验目的，其余部分都是需要我们自己一步步摸索的。没做实验时的轻视，开始时的迷茫，炸内存时的痛苦，出结果时的喜悦，这次实验体会到了很多。

这次实验也学到了很多知识，对聚类算法的了解，`KMEANS`，`DBSCAN`等算法原理及应用。除了这些之外，最重要的是初步的了解到了如何进行科学研究，发现问题，如何设计算法解决，设计出算法后如何实现，这些都是很宝贵的经历。