交通运输数据技术 作业五

——基于共享单车出行数据的聚类分析

2251140 范程

一、 总述

本次作业主要是基于第 10—12 讲数据聚类分析的内容，也涉及到之前第 8 讲数据质量评价中相似性的部分内容。本次作业的数据来源是台北市共享单车的 OD 数据(通过经纬度可以看出来)，进行聚类的主要流程有：数据预处理、数据特征抽取、数据的相似度量、聚类算法选择与聚类实现、结果评估等。从预处理开始，到算法选择和实现以及最后的结果评估，贯穿了交通数据分析与可视化的主要过程。本次作业我采用了 Python 语言来完成，特征提取部分结合了 R 语言来更好地进行可视化。以下是我对聚类过程的代码的主要分析。

二、 聚类流程分析

（一） 数据预处理

(1) 计算 OD 距离

将文件 obike_1.csv 转换为 obike_1.xlsx 并读取之后，首先需要对数据的量纲进行统一。在 excel 文件中，发现有以下字段：共享单车编号(obike)，出发时间(otime)，出发点纬度(olgt)，出发点经度(olat)，到达点纬度(dlgt)，到达点经度(dlat)以及骑行时间(time, 单位为分钟)。其中需要进行转换是经纬度相关数据。

通过高中自然地理的知识，我们知道地球上两点之间的最短距离实际上是包含两点且圆面通过地心的一条劣弧弧长d。而对于地球上的两点$A(\varphi_1,\lambda_1),B(\varphi_2,\lambda_2)$，在已知其经纬度数据的情况下，可以通过球面余弦定理和 Haversine 公式来进行距离的计算，公式中会出现的R是地球的半径，近似可取$6371km$，φ,λ分别代表经度和纬度。至于函数$haversin(\theta)$，称为半正矢函数，$haversin(\theta) = sin^2\left(\frac{\theta}{2}\right) = \frac{1-\cos(\theta)}{2}$.之后我们就可以解出距离$d$：

$$d = 2Rarcsin\left(\sqrt{sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\,sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right).$$

需要注意的是，经纬度的数据都是弧度制，因此需要关于经纬度的四列数据都进行弧度制转化，代码如下：lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

在完成转化并根据上述原理定义 haversin(lon1, lat1, lon2, lat2)函数之后，我们可以通过 df.apply()函数来求出各条共享单车记录的 OD 点之间距离：

df['distance']=df.apply(lambda row: haversine(row['olgt'],row['olat'],row['dlgt'],row['dlat']), axis=1)，运行后得出距离分布在 0.003km ~ 18.6475km 之内。

(2) 删除异常数据

进一步，我们需要通过计算骑行的平均速度来剔除不合理的数据。通过定义计算骑行时间的函数 get_hour(a)来计算以小时为单位的骑行时间。接着通过

df['time']=df.apply(lambda row: get_hour(row['time']), axis=1)

df['speed']=df['distance']/df['time']

来计算骑行平均速度。考虑到共享单车的速度一般不会超 25km/h，因此再通过 df[df['speed']<=25]来去除这部分异常数据。

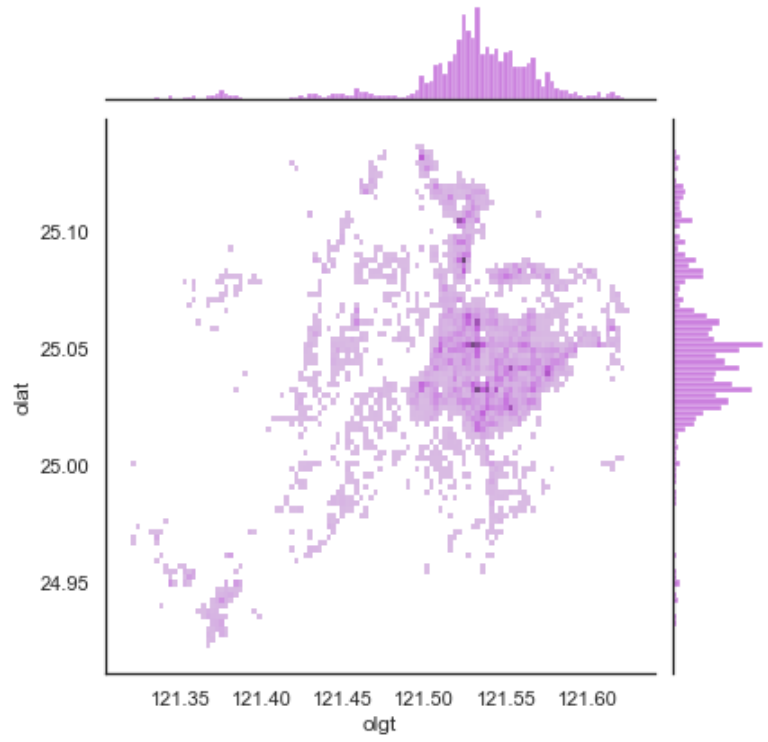
```
In [2]: print('预处理前的数据量为: ',len(df))
预处理前的数据量为:  16386

In [3]: print('预处理后的数据量为: ',len(new_df))
预处理后的数据量为:  16354
```

在数据预处理之后，数据量的变化如右图：

之后便可以对有效数据进行特征抽取和聚类等工作。

（二） 特征抽取

这部分的主要任务是要绘制共享单车数据的 O/D 位置分布图。我绘制了出发点 O 的位置分布图。利用 df_d=new_df[['olgt','olat']]来提取出发点的经纬度数据。之后利用 seaborn 库强大的画图功能，调用 seaborn 的 jointplot()函数来联合绘制出发点经度 olgt 和纬度 olat 的直方图和散点图，如下图所示：

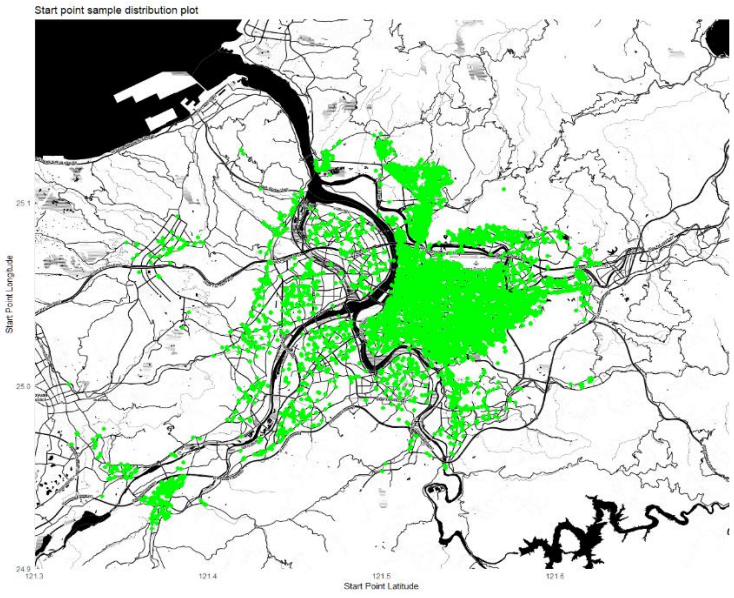


事实上我经过对比发现，在这一部分利用 R 语言的 ggplot2 包，结合 ggmap 在线获取

的底图来绘制效果更好：

```
library(ggplot2)
library(ggmap)
register_stadiamaps(key='ed870b84-66a8-47aa-bb45-dc22569ae3ca') #注册之后的 api
road <- ggmap::get_stadiamap(bbox = c(left = 121.3, bottom = 24.9, right = 121.7, top = 25.2),maptype = c("stamen_toner"), zoom = 13)
p1 <- ggmap(road) + # 用 get_stadiamap()而不是 stamenmap()!
 geom_point(data = data, aes(x = olgt, y = olat),color = 'green', size = 2) +
 theme_minimal() +
 labs(title = 'Start point sample distribution plot') +
 xlab('Start Point Latitude') +
 ylab('Start Point Longitude') # 画 O 点地理位置分布图
p1
```

绘制结果如下：



将以上两幅图相结合，通过 R 语言绘制的图我们可以看出台北市共享单车的出发地点主要集中在淡水河以东的平原地区，通过 Python 绘制的图发现位置集中于（25.02°N~25.07°N, 121.50°E~121.57°E）的区域。该区域是台北的市区范围，拥有密集的捷运系统和商业住宅区、松山机场等需要共享单车来完成出行的地区。在淡水河西部的芦洲乡、二重埔地区和西南靠近新北市的区域也有部分分布。总体来说分布的空间差异较大，与经济发展水平、人口密集程度有关，也与南部分布台湾山脉等地形因素相关。

（三） 相似度量与聚类

进行聚类时需要通过样本点的分布来选择聚类算法来进行聚类。我选择了 K-means 算法，结合两种相似度度量标准：欧式距离和曼哈顿距离来进行聚类，并比较结果的不同。

K-means 聚类的原理对需要聚类的样本 X = np.array(df_d[['olgt','olat']])，

随机选择 k 个聚类中心作为质心：

```
n_clusters = 3，
np.random.seed(0)
centers = X[np.random.choice(range(len(X)), n_clusters, replace = False)]
```

计算每个样本点到质心的欧氏距离或 Manhattan 距离：

```
while True:
    distances = np.linalg.norm(X[:,np.newaxis] - centers, axis = -1)  # 这里是欧氏距离
    # distances = np.sum(np.abs(X[:,np.newaxis] - centers), axis = -1) # Manhattan 距离
```

根据距离划分样本点的归类：labels = np.argmin(distances, axis=1)

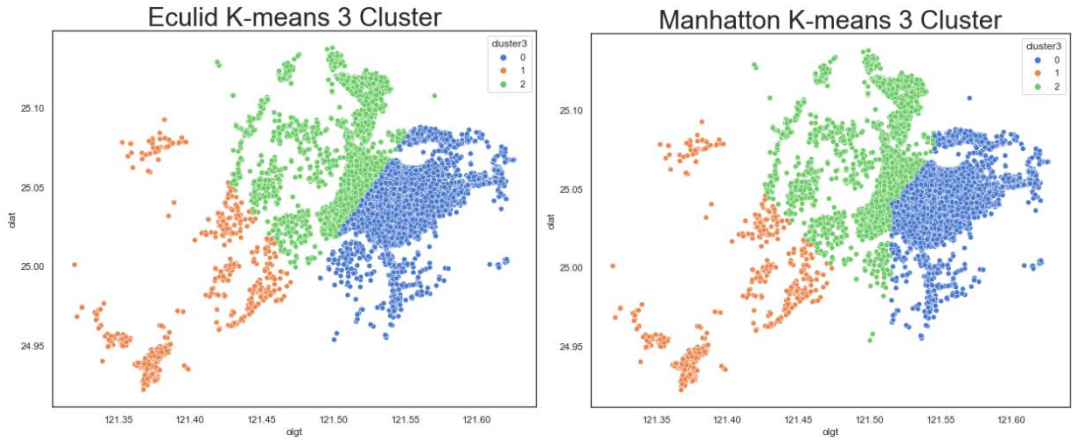
再在每一类中更新质点的位置：

```
new_centers = np.array([X[labels == i].mean(axis=0) for i in range(n_clusters)])
```

迭代上述操作，直至质心位置不再变化为止：

```
if np.all(centers == new_centers):
    break
centers = new_centers.copy()
```

以下是聚类的结果：



通过两种不同的距离选择完成的 K-means 聚类，我们不难发现，在大部分结果的归属方面，两种距离都有较大的相似性。这部分样本点主要是类内靠近质心的部分。而对于位

于两个类便捷处的样本点，两种距离展现出了不同的结果。如图中 label 为 2 的绿色样本所示，其在南部区域的聚类结果显示出，Manhattan 距离的 K-means 聚类结果范围要大于欧氏距离的结果。可能是由于在距离的计算上，这部分点恰好与绿色部分的质心处在相似的经度或者纬度位置上，（即 Manhattan 距离可能只有一个纬度）从而导致了 Manhattan 距离要小于欧氏距离。但是虽然结果相似，但是具体算法的选择还需要根据当地道路、街区分布的实际情况来进行。

（四） 结果评估

对于 K-means 聚类结果的评估是针对不同 k 的取值来进行的。我采用了轮廓系数和 Calinski-Harabaz 指数来进行评价。

(1) 轮廓系数

轮廓系数是评估聚类效果的指标，取值范围为$[-1,1]$，越接近 1 表示聚类效果越好。其计算原理如下：

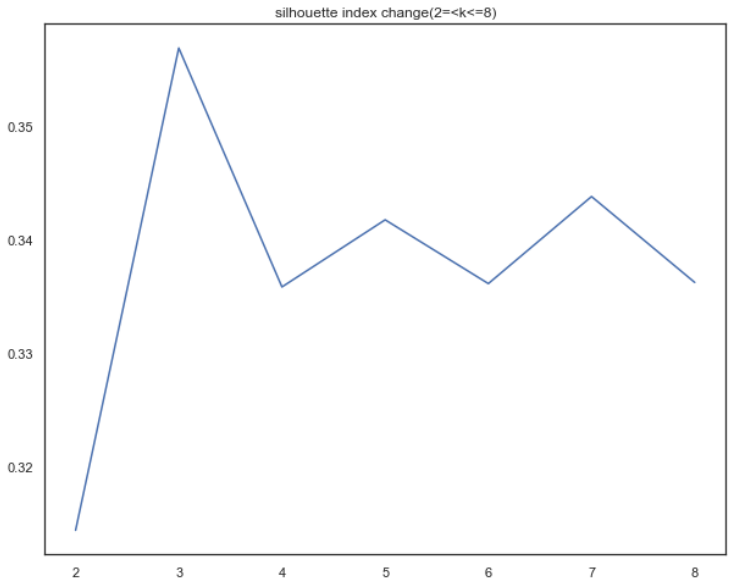
$$S = \frac{1}{N}\sum_{i=1}^{N}\frac{b(i)-a(i)}{\max\left[a(i),b(i)\right]}.$$

其中$a(i)$表示第i个簇内样本点之间距离的平均值，$b(i)$表示第i个簇内样本点到其他簇内样本点距离的平均值。

根据上述原理，在不调用 sklearn.metrics 类中的 silhouette_score()函数的情况下，我编写了如下函数来求轮廓系数：

```
def my_silhouette_score(X, labels):
    n_samples, _ = X.shape
    n_labels = len(np.unique(labels))
    a = np.zeros(n_samples)
    b = np.zeros(n_samples)
    for i in range(n_samples):
        a[i] = np.mean(cdist(X[labels == labels[i]], [X[i]]))  #计算 a[i]
    for i in range(n_samples): # 这里算的是 b[i]
        b[i] = np.min([np.mean(cdist(X[labels == k], [X[i]])) for k in np.unique(labels) if k != labels[i]])
        score = np.mean((b - a) / np.maximum(a, b))
  return score
```

在调用时，我把 k 的范围设置为 2~8，并通过 matplotlib.pyplot 类来绘制不同 k 下的轮廓系数，如下图所示：



其中 k 和轮廓系数的对应关系如下：

```
In [4]: k_list1
Out[4]: [2, 3, 4, 5, 6, 7, 8]

In [5]: silhouette_index
Out[5]:
[0.3143696616131188,
 0.35692733865425175,
 0.3358443484126796,
 0.341767942959414,
 0.33613968716627685,
 0.34382296928553274,
 0.3362434232941209]
```

可以看出$k = 3$时，聚类效果最好。实际上虽然轮廓系数越接近 1 越好，但是在现有数据状况下，$k = 3$时聚类结果已经可以达到不错的效果。

(2) Calinski-Harabaz 指数

Calinski-Harabaz 指数通过计算类中各点与类中心的距离平方和来度量类内样本点聚类结果的紧密度，通过计算各类中心点与数据集中心点距离平方和来度量数据集的分离度，再由两个指数相除得到，指数越大，说明聚类效果越好。具体的计算公式如下：

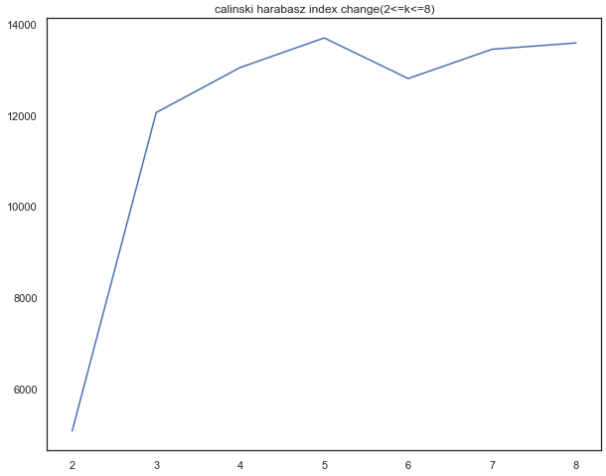
$$CH = \frac{tr(B_k)}{tr(W_k)} * \frac{m-k}{k-1}, \text{其中} \begin{cases} W_k = \sum_{i=1}^{k} \sum_{x \in C_i} (x - C_i)(x - C_i)^T \\ B_k = \sum_i n_i (C_i - c)(C_i - c)^T \end{cases},$$

N是样本总数，c是第i个簇中的样本点，C_i是第i个样本的质心，n_i是第i个簇中样本点的数量，m为样本数，k是指定的簇的数量，B_k是类间协方差矩阵，W_k是类内协方差矩阵，tr为矩阵的迹（参考《概率论与数理统计》和知乎以及 CSDN）。

根据上述原理，可以通过以下过程来计算 Calinski-Harabaz 指数：

```
def my_calinski_harabasz(X, labels):
    n_samples, _ = X.shape
    n_labels = len(np.unique(labels))
    extra_cov, intra_cov = 0., 0.
    mean = np.mean(X, axis=0)
    for k in range(n_labels):
        cluster_k = X[labels == k]
        mean_k = np.mean(cluster_k, axis=0)
        extra_cov += len(cluster_k) * np.sum((mean_k - mean)**2)  #类间协方差
        intra_cov += np.sum((cluster_k - mean_k)**2)  # 类内协方差
    score = (1. if intra_cov == 0. else extra_cov * (n_samples - n_labels) /(intra_cov *
(n_labels - 1.)))     # 计算 Calinski-Harabasz index
    return score
```

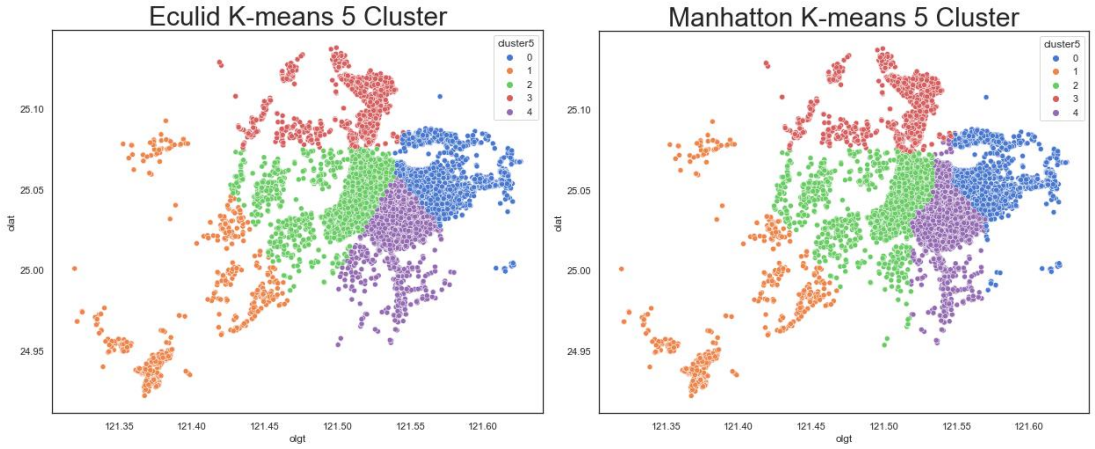
同样地，绘制 Calinski-Harabasz 指数随 k 的变化图：



以下是k和对应的 Calinski-Harabasz 指数：

```
In [6]: k_list2
Out[6]: [2, 3, 4, 5, 6, 7, 8]

In [7]: ch_index
Out[7]:
[5072.4458346543215,
 12061.04949652142,
 13049.628179412433,
 13699.061339650412,
 12808.065381634553,
 13450.622425776799,
 13589.252526810846]
```

从图中可以看出，$k = 5$时聚类的效果理应最好，因此我再次绘制了 k = 5 时，两种距离下 K-means 聚类的可视化结果：



在簇的数量增加之后，原本图中偏西南部，靠近新北市的样本点被归到了新的一类中，东北角的聚类情况也有所改变。我们可以发现，$k = 5$时候的聚类对于离群点和分散的点的效果更优。

三、 总结与反思

本次作业的进行初期我才用了 R 语言，但由于在进行聚类时，计算距离的数据集和质心之间维度不同的问题一直难以解决，故采用 python 进行。我充分感受到 python 在绘图时丰富的库和类，matplotlib 和 seaborn 等都具有强大的功能。但 R 语言也有其特殊的优势，在可视化时可以加载在线地图，因此我在可视化共享单车出发点数据时都进行了使用。

同时，在聚类算法的选择上，我们应该根据需要聚类的对象及其所在的地理位置、区位特征来综合考虑，如本次作业中可以借助高德地图来参考台北市相关地理事物的分布。在聚类结果的评估上可以采用不止一种系数来评价，从而选择最优的结果。