MR定位建模——回归

1953871 邓泉

1. **获取数据**

读取2G\_data.csv的MR数据与2G\_gongcan.csv的工参表数据，进行匹配工参，使得在MR数据的基础上增添了邻接基站的GPS经纬度位置。

|  |
| --- |
| 1. *# 读取数据* 2. sf = pd.read\_csv(readpath + "2G\_data.csv", encoding = 'gb2312') 3. *#匹配工参* 4. sf\_eng = pd.read\_csv(readpath + "2G\_gongcan.csv", encoding = 'gb2312') 5. for i in range(1, 8): 6. sf = sf.merge(sf\_eng, left\_on = ['RNCID\_%d'% i, 'CellID\_%d'% i], 7. right\_on = ['RNCID', 'CellID'], how = 'left', suffixes = ('', '%d'% i)) |

1. **数据集转换**
   1. **相对位置计算 + 异常值处理**

对于每一条MR数据，根据id\_combine= RNCID + CellID从多个邻接基站中标定唯一的主基站，新增两列属性m\_Longitude、m\_Latgitude用于标注主基站的GPS坐标。

计算MR记录的GPS相对坐标，用新增的属性r\_Longitude、r\_Latgitude来记录；将各个邻接基站的GPS坐标转换成以主基站为参考系的相对坐标（替代原值）。

|  |
| --- |
| 1. *# 标定主基站位置+计算相对位置* 2. sf['m\_Longitude'] = '' 3. sf['m\_Latitude'] = '' 4. sf['r\_Longitude'] = '' 5. sf['r\_Latitude'] = '' 6. for i in range(len(sf)): 7. n\_adjacent = sf['邻接基站个数'][i] 8. id\_combine = sf['id\_combine'][i] 9. *# 对于每一条MR数据，根据id\_combine= RNCID\_1+ CellID\_1从多个邻接基站中标定唯一的主基站* 10. for j in range(1, n\_adjacent+1): 11. if sf[ 'RNCID\_%d'% j][i] + sf[ 'CellID\_%d'% j][i] == id\_combine: 12. *# 对于在工参表中无匹配项的主基站，用工参表中的坐标平均值作为其位置* 13. if np.isnan(sf['Longitude%d'% j][i]) | np.isnan(sf['Latitude%d'% j][i]): 14. sf.loc[i, ['m\_Longitude']] = np.mean(sf\_eng['Longitude']) 15. sf.loc[i, ['m\_Latitude']] = np.mean(sf\_eng['Latitude']) 16. sf.loc[i, ['m\_Longitude']] = np.mean(sf\_eng['Longitude']) 17. sf.loc[i, ['m\_Longitude']] = np.mean(sf\_eng['Latitude']) 18. else: 19. sf.loc[i, ['m\_Longitude']] = sf[ 'Longitude%d'% j][i] 20. sf.loc[i, ['m\_Latitude']] = sf[ 'Latitude%d'% j][i] 21. break 22. *# 将手机的坐标转换成以主基站为参考系的相对坐标* 23. sf.loc[i, ['r\_Longitude']] = sf['Longitude'][i] - sf['m\_Longitude'][i] 24. sf.loc[i, ['r\_Latitude']] = sf['Latitude'][i] - sf['m\_Latitude'][i] 25. *# 将各个邻接基站的坐标转换成以主基站为参考系的相对坐标* 26. for j in range(1, n\_adjacent+1): 27. sf.loc[i, ['Longitude%d'% j]] -= sf['m\_Longitude'][i] 28. sf.loc[i, ['Latitude%d'% j]] -= sf['m\_Latitude'][i] |

**缺失值处理：**

* 对于在工参表中无匹配项的主基站，工参匹配时其GPS坐标为缺失值NaN，用工参表中已有的所有基站的GPS坐标平均值代替
* 用一个绝对值足够大的负数-999填充剩余的NaN
  1. **特征选择**

选择各邻接基站的信号及相对坐标信息——SignalLevel,RSSI, Longitude,Latitude作为输入的特征；相对坐标r\_Longitude、r\_Latgitude作为模型的输出。

* 1. **分组**

按照每个MR记录对应的主基站对 MR 记录进行分组，使得每组 MR 记录都有相同的主基站，总的分组个数即为主基站个数。

|  |
| --- |
| 1. group = sf.groupby(by='id\_combine') 2. n\_group = group.count().shape[0] 3. print("分组数量（主基站个数）:", n\_group) *# 总的分组个数即为主基站个数* |

**结果显示，主基站一共有43个。**

****

1. **模型训练与评估**
   1. **构建单个分组的MR定位模型**

* 划分数据集
* 特征工程

尝试了以下无量纲化和特征降维方法的单独及组合使用：

* + - 无量纲化：标准化、归一化；
    - 特征降维：PCA降维、层次聚类降维、过滤低方差特征；

结论是：**单纯的归一化达到了最优的效果，因此采用它。**

* 模型训练
* 误差计算

|  |
| --- |
| 1. def LocateModel(id\_combine, sf, model='Regressor\_DecisionTree'): 2. *# 划分数据集* 3. X = sf[feature\_names] 4. Y = sf[class\_names] 5. from sklearn.model\_selection import train\_test\_split 6. X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=200) 7. *# 特征工程* 8. *# 特征预处理：归一化* 9. from sklearn.preprocessing import MinMaxScaler 10. scaler = MinMaxScaler() 11. X\_train = scaler.fit\_transform(X\_train) 12. X\_test = scaler.transform(X\_test) 13. *# 模型训练* 14. from sklearn.ensemble import RandomForestRegressor 15. estimator = RandomForestRegressor() 16. estimator.fit(X\_train, Y\_train[['r\_Longitude', 'r\_Latitude']]) 17. *# 误差计算* 18. *# 预测的相对经纬度坐标* 19. pred\_rloc = estimator.predict(X\_test) 20. *# 预测的绝对经纬度坐标(还原为原始位置)* 21. pred\_loc = (pred\_rloc + Y\_test[['m\_Longitude', 'm\_Latitude']]).values 22. *# 误差* 23. err = [distance(p,t) for p, t in zip(pred\_loc, Y\_test[['Longitude','Latitude']].values)] 24. return Y\_test, pred\_loc, err |

* 误差分析

|  |
| --- |
| 1. *# 回归误差分析* 2. def ErrorAnalysis(err, savepath): 3. *# 绘制误差分布图，其中 x-轴为排序编号，y-轴为对应的误差* 4. err = sorted(err) 5. fig = plt.scatter(range(len(err)), err, s=25, alpha=.5) 6. plt.rcParams['font.sans-serif']=['SimHei'] 7. plt.rcParams['axes.unicode\_minus'] = False 8. plt.title(model + '\_%d'% id\_combine + ' - 误差分布图', fontsize=14) 9. plt.xlabel('seq', fontsize=14) 10. plt.ylabel('err/m', fontsize=14) 11. plt.xlim(0, ) 12. plt.ylim(0, ) 13. plt.show() 14. fig.figure.savefig(savepath) *# 保存* 16. from sklearn.metrics import r2\_score 17. print('平均绝对误差 MAE : ', np.mean(np.abs(err))) 18. print('均方误差 MSE : ', np.mean(np.square(err))) 19. percentage\_err = np.abs(err - np.mean(err)) / np.mean(err) 20. print('平均绝对百分比误差 MAPE : ', np.mean(percentage\_err)) 21. print('绝对误差中位数 MedAE : ', np.median(err)) 22. print('R2\_socre : ', r2\_score(Y\_test[['Longitude','Latitude']], pred\_loc, multioutput='variance\_weighted')) 23. print('误差标准差 : ', np.std(err)) |

* 地图可视化

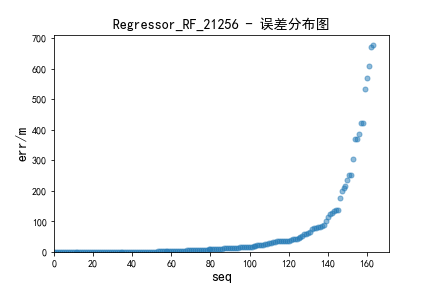
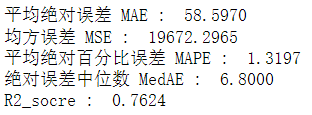
|  |
| --- |
| 1. *# 地图可视化* 2. def MapVisual(pred\_loc, Y\_test, savepath): 3. *# 绘制地图，标注pred\_loc和Y\_test[['Longitude','Latitude']].values* 4. import folium 5. tj\_map = folium.Map(location=[31.286363, 121.214786],zoom\_start=15) 6. *# 添加实际标记：Y\_test[['Longitude','Latitude']].values* 7. for i in Y\_test[['Longitude','Latitude']].values: 8. lat = i[1] 9. lon = i[0] 10. folium.Circle( 11. radius=1, 12. location=[lat, lon], 13. popup='Laurelhurst Park', 14. color='#9AFF9A', *# 绿色* 15. fill=False 16. ).add\_to(tj\_map) 17. *# 添加预测标记：pred\_loc* 18. for i in pred\_loc: 19. lat = i[1] 20. lon = i[0] 21. folium.Circle( 22. radius=1, 23. location=[lat, lon], 24. popup='Laurelhurst Park', 25. color='#FF8C69', *# 橙色* 26. fill=False 27. ).add\_to(tj\_map) 28. tj\_map.save(savepath) 29. return tj\_map |

针对每个分组构建一个对应的MR定位模型：

|  |
| --- |
| 1. model = 'Regressor\_RF' 2. err\_all = [] 3. pred\_loc\_all = [] 4. Y\_test\_all = pd.DataFrame() 5. for i in range(n\_group): 6. id\_combine = list(group)[i][0] *# 该分组的唯一主基站* 7. current = list(group)[i][1] *# 分组的MR数据* 8. *# 针对单个分组训练模型* 9. Y\_test, pred\_loc, err = LocateModel(id\_combine, current, model) 10. *# 误差分析* 11. ErrorAnalysis(err, savepath + model+'/' + model + '\_%d'% id\_combine + ' - 误差分布图') 12. *# 地图可视化* 13. MapVisual(pred\_loc, Y\_test, savepath + model+'/' + model + '\_%d'% id\_combine + ' - tj\_map.html') 14. *# 加入到全局list* 15. pred\_loc\_all.extend(pred\_loc) 16. Y\_test\_all = pd.concat([Y\_test\_all, Y\_test], axis=0) 17. err\_all.extend(err) *# 在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表)* |

**结果示例：**

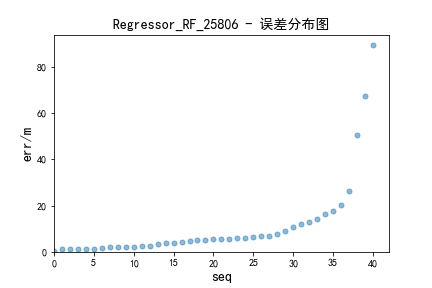
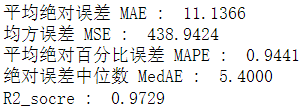
**这是针对id\_combine = 21256和25806的两个分组MR数据拟合的分析结果。**

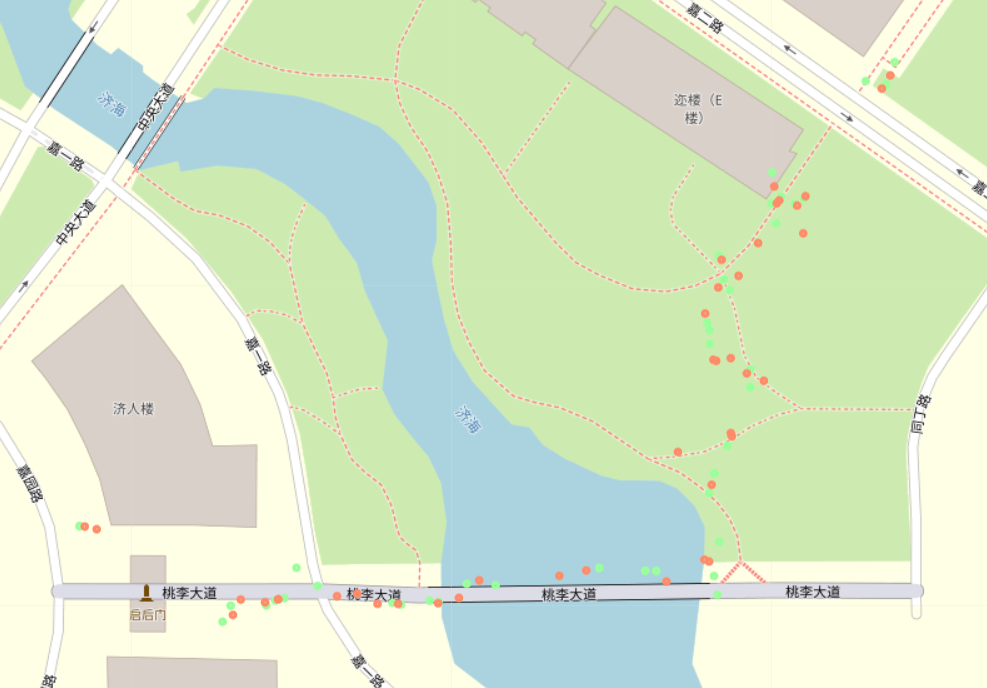
 

****

[..\result data\regression\Regressor\_RF\Regressor\_RF\_21256 - tj\_map.html](../result%20data/regression/Regressor_RF/Regressor_RF_21256%20-%20tj_map.html)

**图例. id\_combine = 21256**

****

****

[..\result data\regression\Regressor\_RF\Regressor\_RF\_25806 - tj\_map.html](../result%20data/regression/Regressor_RF/Regressor_RF_25806%20-%20tj_map.html)

**图例. id\_combine = 25806**

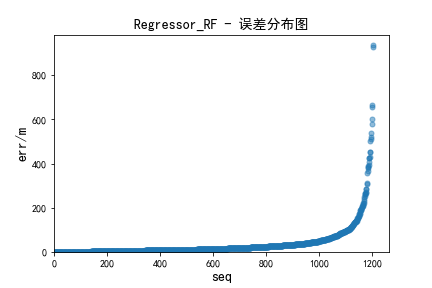
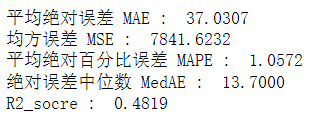
**可以看出，分组25806的数据量明显少于分组21256，其误差指标也小了很多。**

* 1. **误差分析（全局）**

综合所有分组的数据，进行全局的误差分析。

|  |
| --- |
| 1. ErrorAnalysis(err\_all, savepath + model+'/' + model + ' - 误差分布图') |

**结果如下：**

* 1. **地图可视化预测效果（全局）**

综合所有分组的数据，进行全局的地图可视化分析。

|  |
| --- |
| 1. MapVisual(pred\_loc\_all, Y\_test\_all, savepath + model+'/' + model + ' - tj\_map.html') |

**结果如下：**

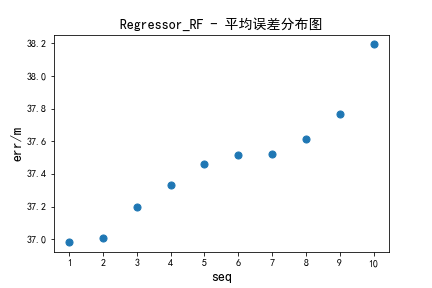
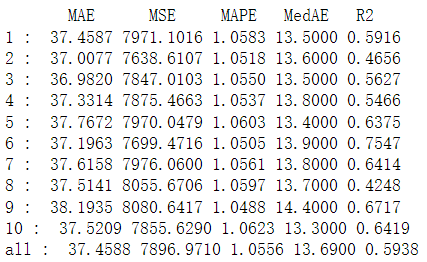
****

[**..\result data\regression\Regressor\_RF\Regressor\_RF - tj\_map.html**](../result%20data/regression/Regressor_RF/Regressor_RF%20-%20tj_map.html)

* 1. **交叉验证**

|  |
| --- |
| 1. *# 重复 10 次训练集/测试集的选择和误差计算* 2. print('       MAE      MSE     MAPE   MedAE   R2') 3. MAE, MSE, MAPE, MedAE, R2\_score = [], [], [], [], [] 4. *# 10-fold cross validation* 5. for j in range(10): 6. *# 对每一个分组构建定位模型* 7. model = 'Regressor\_RF' 8. err\_all = [] 9. pred\_loc\_all = [] 10. Y\_test\_all = pd.DataFrame() 11. for i in range(n\_group): 12. id\_combine = list(group)[i][0] *# 该分组的唯一主基站* 13. current = list(group)[i][1] *# 分组的MR数据* 14. *# 针对单个分组训练模型* 15. Y\_test, pred\_loc, err = LocateModel(id\_combine, current, model) 16. *# 加入到全局list* 17. pred\_loc\_all.extend(pred\_loc) 18. Y\_test\_all = pd.concat([Y\_test\_all, Y\_test], axis=0) 19. err\_all.extend(err) *# 在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表)* 20. MAE.append(np.mean(np.abs(err))) 21. MSE.append(np.mean(np.square(err))) 22. percentage\_err = np.abs(err - np.mean(err)) / np.mean(err) 23. MAPE.append(np.mean(percentage\_err)) 24. MedAE.append(np.median(err)) 25. from sklearn.metrics import r2\_score 26. R2\_score.append(r2\_score(Y\_test[['Longitude','Latitude']], pred\_loc, multioutput='variance\_weighted')) 27. print('%d : '% (j+1), '%.4f'% np.mean(np.abs(err)), '%.4f'% np.mean(np.square(err)), '%.4f'% np.mean(percentage\_err), 28. '%.4f'% np.median(err), '%.4f'% r2\_score(Y\_test[['Longitude','Latitude']], pred\_loc, multioutput='variance\_weighted')) 30. print('all : ', '%.4f'% np.mean(MAE), '%.4f'% np.mean(MSE), '%.4f'% np.mean(MAPE), 31. '%.4f'% np.mean(MedAE), '%.4f'% np.mean(R2\_score)) 32. *# 绘制平均误差概率分布图，其中 x-轴为排序编号，y-轴为对应的平均误差 Average error distribution plot* 33. MAE = sorted(MAE) 34. fig = plt.scatter(range(1, len(MAE)+1), MAE, s=50) 35. plt.rcParams['font.sans-serif']=['SimHei'] 36. plt.rcParams['axes.unicode\_minus'] = False 37. plt.title(model + ' - 平均误差分布图', fontsize=14) 38. plt.xticks(np.linspace(1,10,10)) 39. plt.xlabel('seq', fontsize=14) 40. plt.ylabel('err/m', fontsize=14) 41. plt.show() 42. fig.figure.savefig(savepath + model+'/' + model + ' - 平均误差分布图') *# 保存* |

**交叉验证结果如下：**

****

* 1. **对比：基于分类模型的定位方法**

将本方法与基于分类模型的定位方法（详见’1 分类.docx’）中表现最优的随机森林模型进行对比分析。

**对比分析：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **MAE** | **MSE** | **MAPE** | **MedAE** | **R2** |
| **决策树（分类）** | 48.9774 | 18564.1883 | 1.3240 | 10.6500 | 0.9134 |
| **高斯朴素贝叶斯（分类）** | 315.6243 | 205359.2955 | 0.8853 | 186.7000 | 0.0250 |
| **K近邻（分类）** | 165.7530 | 114012.0393 | 1.3112 | 20.7000 | 0.4581 |
| **随机森林（分类）** | 26.9132 | 8961.2203 | 1.1512 | 9.1900 | 0.9588 |
| **随机森林（回归）** | 33.4484 | 6681.9465 | 1.0560 | 12.2600 | 0.5157 |

**MAE**：具体而言，随机森林回归模型相比随机森林分类模型MAE增加了约10m（近40%），说明预测值误差（距离：米）更大。

**MSE**：随机森林回归模型的MSE相对略小，考虑到其MAE更高，则从一定程度上说明其数据变化程度更小，效果更稳定。

**MAPE**：两者MAPE相近，均小于10，说明预测精度都比较高。

**MedAE**：该指标相对MAE来讲减弱了异常值的影响，随机森林回归模型的MedAE更大，与MAE反映的评价结果一致。

**R2**：决定系数R2用于度量因变量的变异中可由自变量解释部分所占的比例，以此来判断统计模型的解释力。随机森林分类模型的R2更接近1，远高于随机森林回归模型，说明其解释力更强。

（评价指标：标蓝色 —— 随机森林分类模型表现更优；

标绿色 —— 随机森林回归模型表现更优；

标黄色 —— 两者相当）

此外，基于随机森林回归模型的定位方法通过主基站进行了分组，结果显示，各分组之间的预测精度存在不可忽视的差距，需要进一步考量。

综合来看，**基于随机森林分类模型的定位方法**在绝大多数情况下是更好的模型。