华南师范大学

《数学建模方法》课程项目
项 目 报 告

项 目 题 目：基于人脸的手机伴随识别方法研究

所 在 学 院：计算机学院

项 目 组 长：张帆

小 组 成 员：傅绍基，吴宇涛，黄子桓

目录

1 摘要

在生活中，人脸数据不能和手机数据同时采集，导致无法进行匹配。因此，为了实现人脸数据和手机数据的**关联碰撞**，本论文提出了使用采集时间和近似采集地点的人脸手机数据匹配方法。通过Python代码，可以使测试数据的正确匹配率达到**24.038%**。

关键词：误差分析，近似匹配，二分算法，哈希

2 引言

（一）研究问题背景及意义

根据场景需要会在一个城市内部署不同数量的手机雷达侦码设备，一般情况下100到300台的规模，有部署在人脸抓拍相机同一位置的，也有单独部署的；同时布置的摄像头的位置可能与手机雷达侦码设备相同，也可能不同，需要通过GPS进行判断。为了使摄像头拍到的人脸与侦码设备侦测到的手机相匹配，即实现人脸数据和手机数据的关联和分析碰撞，本模型提出了一种基于人脸的手机伴随识别方法。

（二）研究内容和目的

本模型主要通过数据采集的近似采集地点和采集时间对人脸数据和手机数据进行关联碰撞。

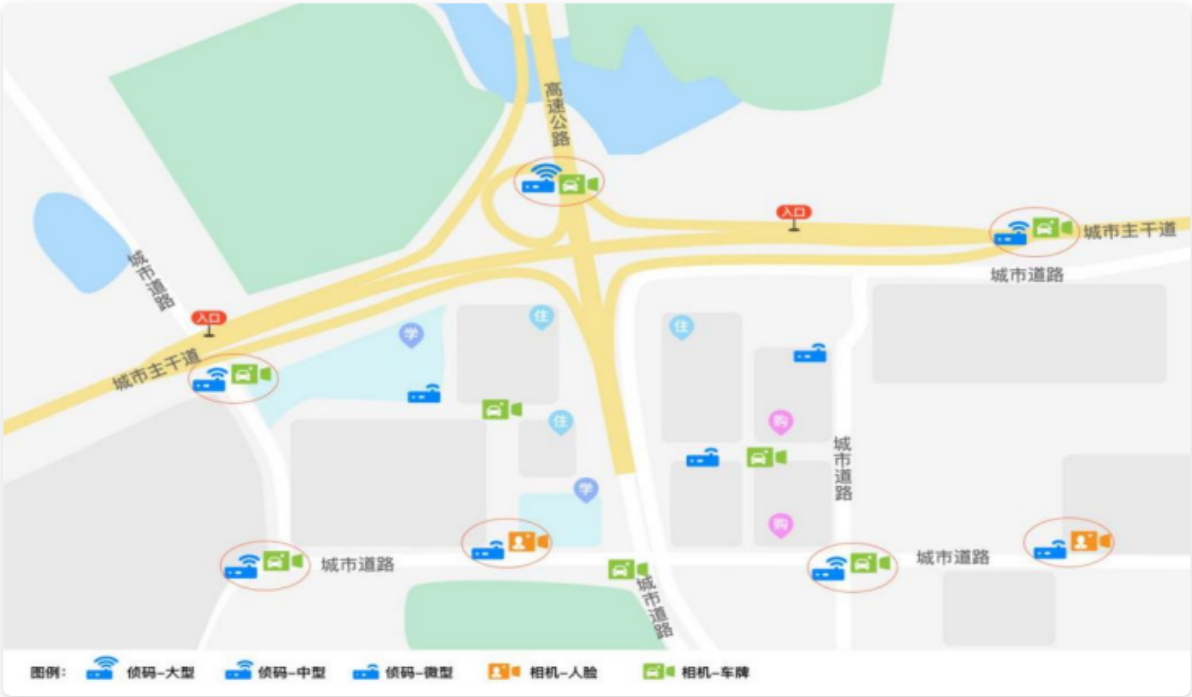

图1

3 模型及方法描述

（一）研究对象介绍

题目给出三个文件：`faceRecord.csv`，`imsiRecord.csv`，`faceImsilinked.csv`。

人物头像表：FID（脸ID）、UPTIME（采集时间）、LON（经度）、LAT（纬度）、PLACEID（近似采集地点）

手机IMSI表：IMSI（手机ID）、UPTIME（采集时间）、LON（经度）、LAT（纬度）、PLACEID（近似采集地点）

人物头像-手机IMSI 关联表：FID（脸ID）、IMSI（手机ID）、label（值为0-1000的数字，表示是同一个人的程度）

其中人物头像表有 `4378808` 行，手机IMSI有 `24402953` 行，最后需要测试的是一个`testfaceids`表。表中有43076个`faceid`，需要模型对前述的三个表进行匹配碰撞，得出 `43076` 个相对应的 `imsi` 号码。

（二）所用工具说明

本模型主要使用 `Python`语言实现， `Python` 在数据清洗与挖掘方面具有得天独厚的优势，其中`numpy`库、`pandas` 库等更是非常方便的工具。

（三）符号说明

符号	含义
T_{1i}	`testfaceids`表里每一个人脸ID在`faceRecord`表出现时获取的时间戳
T_{2j}	每一个时间戳 $[T_{1i+t}, T_{1i-t}]$ 在`imsiRecord`表对应的时间戳

（四）模型描述

对每一个`testfaceids`表里的人脸ID，在`faceRecord`表中搜索相同的人脸ID，得到出现的次数`M`，如果该`M`大于或等于2，获取与该待分析的人脸ID所对应的时间戳 T_{1i} ，基于该时间戳T1i得到时间区间 $[T_{1i-t}，T_{1i+t}]$ ，然后通过搜索`imsiRecord`表获取处于时间区间 $[T_{1i-t}，T_{1i+t}]$ 内的时间戳 T_{2j} 所对应的手机特征码并将处于同一时间区间 $[T_{1i-t}，T_{1i+t}]$ 内的时间戳 T_{2j} 所对应的手机特征码分在同一手机特征码组中，M组该手机特征码组彼此遍历比对，则M组之中出现次数最多的这个手机特征码与该待分析的人脸ID相对应；其中，`i`、`j`均为正整数；`t`为经过参数调优而设置的时间，单位为秒，本模型选用的时间为 $[-128s, 85s]$ ；`M`为等于或大于0的整数。所有条件中，近似采集地点也必须相等。

4 实验结果与分析

建立模型中我们注意到了数据过拟合的问题，有些相同的 `imsi` 号码集合在中间的一组，因为出现次数最多所以直接作为答案与 `faceid` 匹配，而正确的 `imsi` 号码是出现在很多组的，只不过次数没这么多，这就是中间的过拟合问题。为了抑制过拟合问题，模型不再使用出现次数最多的 `imsi` 号码，而是使用出现再最多组的 `imsi` 号码，正确率 `acc` 提高了3个百分点，有效抑制了过拟合问题。

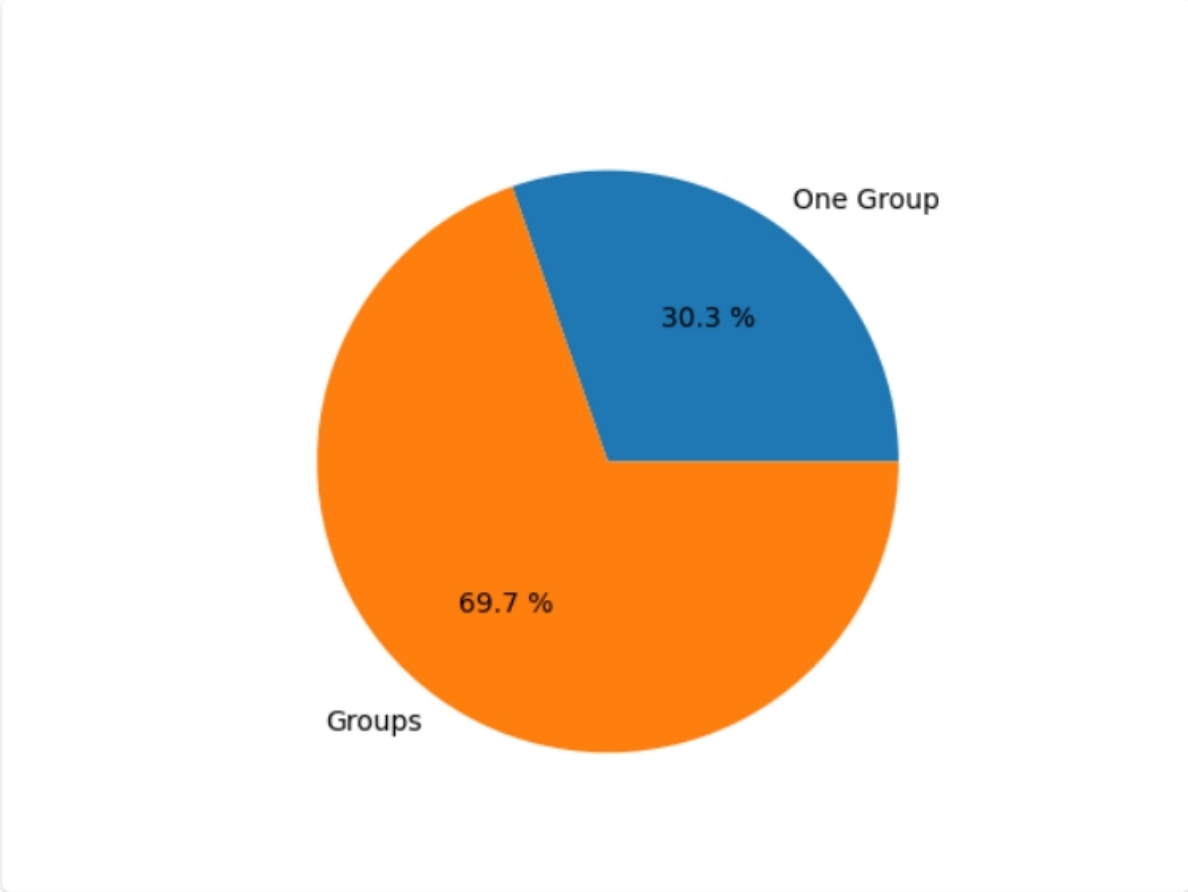

图2

同时，我们发现有43076个测试样例里面有13064个 `faceid` 是只有一组的，即在 `faceRecord` 表里面只出现了一次，因此无法通过上面的算法进行多组比对得出答案。小组目前采用的方法是选择离此 `faceid` 被采集到的时间最近的那个时间的 `imsi` 号码，即为正确答案。

本组在建立模型的时候舍弃了经纬度作为条件，以第1个和第43076个测试样例为例（两个样例均为前述只有一组的情况）：

第1个 `testfaceid` 对应的符合条件的 `imsi` 号码：

```
['460008198046357', '1598884482', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7'],
['460110806925128', '1598884510', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7'],
['460008572764937', '1598884510', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7'],
['460110918061048', '1598884528', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7'],
['460022286336060', '1598884545', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7'],
['460008221737629', '1598884554', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7'],
['460110508385360', '1598884571', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7'],
['460110806869947', '1598884576', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7'],
['460110508599493', '1598884612', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7'],
['460000042114569', '1598884620', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7'],
['460018081911399', '1598884620', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7'],
['460009930325143', '1598884635', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7'],
['460008542736021', '1598884677', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7'],
['460110508391405', '1598884708', '104.72184214004113', '29.344966375887772', '5c505e17498e4a19f0007cc7']
```

第43076个 `testfaceid` 对应的符合条件的 `imsi` 号码：

```
['460028922047414', '1596217075', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460016090205047', '1596217087', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460110918046968', '1596217092', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460110964443289', '1596217093', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460115308399625', '1596217093', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460110508391194', '1596217105', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460022813347423', '1596217110', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460026807337460', '1596217121', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460110821563939', '1596217131', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460025813654511', '1596217145', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460078284209444', '1596217151', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460026813061301', '1596217152', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460028082015737', '1596217154', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460008552745502', '1596217162', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460076410472992', '1596217172', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460115308434694', '1596217179', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460115308434696', '1596217180', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460008158078933', '1596217182', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460110780227804', '1596217200', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460008158052922', '1596217219', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460077298401879', '1596217220', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460009021124103', '1596217224', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460110508374520', '1596217241', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460110964620286', '1596217255', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460028082312213', '1596217268', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460004311848752', '1596217271', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57'],
['460000043992387', '1596217290', '104.718365334428', '29.349161949892', '5cb7fdbb498e744b8edd6d57']
```

由上述例子可以看出，这种只有一组情况 `testfaceid` 本身无法通过组与组之间的比对找到出现次数最多的 `imsi` 号码，同时组内所有符合条件的 `imsi` 号码经纬度都是相同的， 因此我们在建立模型的时候舍弃了经纬度。针对这种情况，模型选取的是距离此 `testfaceid` 采集时间最近的那个 `imsi` 号码。

5 小结与展望

模型正确率`Acc`只有`24.038%`，可能是模型所选用的时间区间不是最优区间；同时，对13064个只有一组情况的`testfaceid`没有进行有效处理。

我们注意到还有一个`faceImsilinked`我们没有使用，里面有73378个不相关的`faceid`和`imsi`号码，我们下一步想要做的就是在碰到只有一组的情况时，由`faceImsilinked`表中去掉不需要的`imsi`号码，同时记录已经被其他组使用过的`imsi`号码，减少误选的可能性。

6 参考文献

[1]李新,翟玉美,肖洪祥.Android手机中人脸检测算法的改进及实现[J].桂林理工大学学报,2014,34(01):175-179.

[2]赵豪越,王静一,陈思琪,黄建昌.基于智能手机的人脸识别技术研究[J].信息与电脑(理论版),2018(20):106-108.

7 附录

`fanzgg.py`(python程序源代码)

```
import math
import collections
import numpy as np
import pandas as pd
from functools import reduce
import os, csv, sys, time, random
from pandas import Series, DataFrame


def Local_time(Unix_time):  # 将Unix时间转换成北京时间
    beijing_local = time.localtime(Unix_time)
    dt = time.strftime("%Y-%m-%d %H:%M:%S", beijing_local)
    return dt


def Area(lng, lat):  # lat已知纬度，lng已知经度
    r = float(6371)  # 地球半径千米
    dis = float(0.2)  # km距离
    dlng = float(2 * math.asin(math.sin(dis / (2 * r)) / math.cos(lat *
math.pi / 180)))
    dlng = dlng * 180 / math.pi  # 角度转为弧度
    dlat = float(dis / r)
```

```
    dlat = dlat * 180 / math.pi
    left = lng - dlng
    right = lng + dlng
    top = lat + dlat
    bottom = lat - dlat
    return left, right, top, bottom  # 东南西北四个方向的范围


def lower_bound(nums, target):  #返回nums中第一个>=target的值得位置，如果nums中都比target小，则返回len(nums)
    low, high = 0, len(nums) - 1
    pos = len(nums)
    while low < high:
        mid = (low + high) // 2
        if nums[mid] < target:
            low = mid + 1
        else:
            high = mid
    if nums[low] >= target:
        pos = low
    return pos


def upper_bound(nums, target):  #返回nums中第一个>target的值得位置，如果nums中都不比target大，则返回len(nums)
    low, high = 0, len(nums) - 1
    pos = len(nums)
    while low < high:
        mid = (low + high) // 2
        if nums[mid] <= target:
            low = mid + 1
        else:
            high = mid
            pos = high
    if nums[low] > target:
        pos = low
    return pos

time_index = []  #时间索引列表
loc_index = []  #地点索引列表

#读入FanzimsiRecord第二列，返回时间索引列表time_index
pwdl = pd.read_csv('FanzimsiRecord.csv', sep=',', names=['key1', 'key2', 'key3', 'key4', 'key5'])
test1 = list(pwdl.key2)
for index in test1:
    time_index.append(int(index))
#删除
del pwdl
del test1[:]
del test1

#读入FanzfaceRecord第二列，返回时间索引列表loc_index
pwdl = pd.read_csv('xiaoheshang.csv', sep=',', names=['key1'])
test2 = list(pwdl.key1)
for index in test2:
    loc_index.append(int(index))
#删除
```

```
del pwdl
del test2[:]
del test2

print("阶段1完成！")

cnt = 1  #计数器
face_find = []
ans_imsi = []
printcsv = []

with open('fanzimsiRecord.csv', 'r') as u:
    reader2 = list(csv.reader(u))
    print("阶段2完成！")

with open('huanjie.csv', 'r') as f:  #记录选择的faceID的全部信息
    reader1 = list(csv.reader(f))
    print("阶段3完成！")

with open('testfaceids.csv', 'r') as p:  #记录选择的faceID的全部信息
    reader3 = list(csv.reader(p))
    print("阶段4完成！")

with open('testfaceids.csv', 'r') as l:  #记录选择的faceID的全部信息
    cunzai = list(csv.reader(l))
    print("阶段5完成！")

for num in range(1, len(reader3)):
    face_find.clear()
    ans_imsi.clear()

    test_id = reader3[num][1]

    start_face = lower_bound(loc_index, int(test_id))
    end_face = lower_bound(loc_index, int(test_id) + 1)
    for i in range(start_face, end_face):
        face_find.append(reader1[i])

    for i in range(len(face_find)):
        ans_imsi.append([])

    for i in range(len(face_find)):
        flag = face_find[i][1]
        start_time = int(face_find[i][1]) - 128
        end_time = int(face_find[i][1]) + 85
        start_index = lower_bound(time_index, start_time)
        end_index = lower_bound(time_index, end_time)
        #print("第" + str(i + 1) + "组的时间索引是：", start_index, end_index)
        if (end_index != 24402954):
            for j in range(start_index, end_index + 1):
                if (face_find[i][2] == reader2[j][2]):
                    if (reader2[j][0] not in ans_imsi[i]):
                        if(len(face_find) != 1):
                            ans_imsi[i].append(reader2[j][0])
                        else:
                            if (reader2[j][0] not in cunzai):
                                ans_imsi[i].append(reader2[j][0])
        else:
```

```
            for j in range(start_index, 24402954):
                if (face_find[i][2] == reader2[j][2]):
                    if (reader2[j][0] not in ans_imsi[i]):
                        if (len(face_find) != 1):
                            ans_imsi[i].append(reader2[j][0])
                        else:
                            if (reader2[j][0] not in cunzai):
                                ans_imsi[i].append(reader2[j][0])

    real_ans = 0
    fpx = reduce(lambda x, y: x.extend(y) or x, ans_imsi)
    pp = collections.Counter(fpx)

    if (len(pp.keys()) != len(fpx)):
        for t in pp.keys():
            if (pp[t] == max(pp.values())):
                real_ans = t
                break
    else:
        result = sorted(fpx)
        real_ans = result[0]


    print(cnt, test_id, real_ans)
    printcsv.append(real_ans)
    cnt += 1

print("阶段6完成！")

with open('benbenjiejie.csv', 'w', newline='') as n:
    www= []
    writer = csv.writer(n)
    headline = ['NO', 'IMSIID']
    writer.writerow(headline)
    for i in range(len(printcsv)):
        www.append(i + 1)
        www.append(printcsv[i])
        writer.writerow(www)
        www.clear()
    n.close()
```

0.24038.csv	(提供了50行的表格数据)
NO	IMSIID
1	460000042114569
2	460021813422406
3	460004311833940
4	460015727108214
5	460008552727824
6	460011308642360
7	460028082109448
8	460110856864487
9	460110856874186
10	460007210439389
11	460018081993127
12	460110806886814
13	460008542749200
14	460012890086881
15	460029841552123
16	460022813441231
17	460009021140448
18	460008582715107
19	460000234876709
20	460110856906843
21	460026807317620
22	460020583546244
23	460022286270810
24	460110508313151
25	460008522731172
26	460018815982193
27	460009021222995
28	460004321830876
29	460014635909171

0.24038.csv	(提供了50行的表格数据)
30	460029841541600
31	460008572733774
32	460028813240786
33	460008542730847
34	460021826254052
35	460023081428602
36	460008221749959
37	460028082305550
38	460016009015344
39	460014655912989
40	460110821607944
41	460026807311594
42	460026807320760
43	460000034973409
44	460018815981783
45	460011928511167
46	460027280348675
47	460008157076836
48	460110508270948
49	460110508398043
50	460110700011077