项目编号 201610486094

武汉大学国家大学生创新创业训练计划项目初期报告

·

商场中移动群组消费基于安卓手机传感器的数据采集及聚类算法研究

院（系）名 称：国际软件学院

专 业 名 称 ： 软件工程

学 生 姓 名 ： 许琳 张泽宇 胡浙捷

胡成 杨耀航

指 导 教 师 ： 朱卫平 副教授

二〇一六年九月

INITIAL REPORT OF PLANNING PROJECT OF INNOVATION AND ENTREPRENEURSHIP TRAINING OF NATIONAL UNDERGRADUATE OF WUHAN UNIVERSITY

The Data Acquisition and Clustering Algorithm Research Based on Android Sensor of Mobile Group Consumption in Mall

College ： Wuhan University

Subject ： Software Engineering

Name ： Xu Lin Zhang Zeyu Hu Zhejie

Hu Cheng Yang Yaohang

Director ： Zhu Weiping Associate Professor

September 2016

郑 重 声 明

本项目组呈交的初期报告，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我们所知，除文中已经注明引用的内容外，本报告的研究成果不包含他人享有著作权的内容。对本报告所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本报告的知识产权归属于培养单位。

项目组签名：_____ 日期：_____

导师签名：_____ 日期：_____

摘　　要

随着社会经济快速的发展，人们的消费方式也越来越多样化，而在公共场合中群组消费占着较大的比重，群组消费的决定相对单独的消费更依赖与群组成员的交流沟通。本文主要以基于移动设备交流的进行的群组性消费活动为研究方向，利用数据收集系统采集商店和人的数据，通过计算机算法识别群组信息，得到群组中最大影响力的领导者以及每位成员角色，从而有针对性的向商场实行个性化推送。

在本项目的初期，我们完成的任务是模拟器开发，数据采集应用与数据采集以及聚类算法设计实现。模拟器完成了对人，商店，商场的建模，在模拟器中可以实现对商场的状况的模拟，从而在不需要实际数据的情况下对聚类算法进行验证。数据采集是使用的安卓手机进行的传感器以及定位的数据采集，采集系统主要包括安卓客户端以及服务器。最后聚类算法实现了基于 DBAD 以及 Cross-correlation 的相似判断，并使用层次聚类进行了分组。对于无交互数据，使用 DBAD 的算法准确度在 65%-85%之间，而 Cross-correlation 准确度在 80%-100%之间。

关键词：模拟器；数据采集；传感器；Redis；Wifi 定位；K-means；DBAD；Cross-correlation

ABSTRACT

With rapid development of social economy, people's consumption is becoming more and more diversified, and in public consumption accounts for a larger proportion in the group, group decision relatively separate consumption more dependent on communication with group members.This article mainly based on mobile communication group of consumer activity as the research direction, using the data store and data collection system, via computer algorithms to identify group information, so as to get the group leader of the biggest influence and role of each member, and targeted to the stores to implement personalized push.

At the beginning of the project, our task is simulator development, data acquisition application and data collecting implement clustering algorithm design. The simulator has completed the modeling of people, stores, and shopping malls. In the simulator we can realize the simulation of the condition of the market, which in the case of don't need the actual data to verify this clustering algorithm.Data acquisition is the use of the android mobile phone positioning sensors and data acquisition, acquisition system mainly includes the android client and server.Finally clustering algorithm based on DBAD and Cross - correlation similarity judgment, a hierarchical clustering and used in group.For no interactive data, using DBAD algorithm accuracy between 65% and 85%, and Cross - correlation between 80% and 100% accuracy.

Key words: simulator；data collection；sensor；Redis；Wifi location；K-means；DBAD；Cross-correlation

目　录

第 1 章　绪论

1.1　研究背景

移动群组消费是指一群人，例如家人，伴侣，同事朋友等，基于移动设备交流的进行的群组性消费活动。群组性消费的决定相对个人单独的消费更依赖与群组成员的交流沟通，往往群组中一个主要人物会决定这个群组最终的走向，因此想要研究群组消费要关注群组内交流行为并且找出群组中影响力最大的人。群组性消费在生活中并不少见譬如大学生社团班级朋友聚餐、情侣闺蜜逛街、家长孩子出游等。据调查，当人在公共场合时，百分之七十的时间是和其他人一起的 [1]，也就是说大部分情况下他们是进行的群组性消费。所谓“众口难调”，往往这时候总会遇到吃什么，玩儿什么，买什么，什么便宜，哪家团购等一系列决策问题。

近年来，越来越多的人使用手机等移动设备进行群组消费。他们使用手机等移动设备扫描，搜索并获得商品售卖信息和预定各种服务和商品。InMobi Insights 团队的的一个调查显示，46%的人已经用他们手机进行过购买，80%的人表示会在未来一年进行此项活动 [2]。在这种趋势下，许多公司也开始对移动群组消费加大市场力度，2014 年全球此类广告投入已达 32.7 亿美元，占互联网广告总数的 1/4 [3]。

当前应用市场上比较受欢迎的美团，百度糯米，大众点评，以及喵街等只考虑了向用户推送优惠，但不具有即时性，也不具有群组针对性。但是地点是影响移动消费的重要因素[4,5]，群组性消费这一概念也是十分重要，[8]因为单独的个人会因为他们是社会角色一部分进行托管式的群组行为。

此外，在技术层面上，也有许多相关研究。为了分析移动群组消费，首先需要的是一个有效的数据收集系统。典型的要被收集的数据包括在一段时间内人群的轨迹和行为。有多种可用于此的技术，例如 wifi 接入点到用户手机的 wifi 信号可以被收集。每个手机附属的 RFID 标签也可以用来分析。当然移动设备中的传感器也可以用于测定群组移动轨迹和行为[9]。还有就是采集商场中摄像头数据，通过视频流进行识别[10]。此外通过这些数据能否很好的识别出群组关系以及人与人之间交流行为也是实现移动群组消费的关键。[11]提出了移动社交网络环境下的真实社会关系估计，它基于用户 GPS 轨迹挖掘语义化访问地点，然后结合语义化地点以及临近特征估计用户间的社会关系类型，结果表明是可以准确估计家人同事

朋友等社会关系。这些研究均表明了商场移动群组消费系统设计具有它的可行性。

1.2 研究意义

通过对移动群组消费的研究，一方面我们能够对商场中人的群组的交互和消费行为有更加深刻的了解，在行为学方面；另一方面通过移动设备研究群组消费将线上线下人的社会关系结合，促进了基于互联网的移动社交。

移动群组消费是一个新的概念，通过对移动消费者群组的识别和影响力研究。商家能够针对群组消费提出更好的营销策略，减少广告投放随意性，降低商家广告投入成本。

对于群组中消费者本身来说，我们研究的群组性移动消费的推送能够更好的解决“众口难调”的问题，促进用户的理智消费。

1.3 研究方法

通过对相关研究的总结我们的项目主要分为这样几个部分：

⑴调查当前商场中人群信息数据收集方法

⑵设计商场人群及商店模型模拟器软件

⑶基于模式识别聚类设计群组聚类算法并使用模拟数据验证准确度

⑷基于复杂网络知识建立群组影响力模型识别群组人员影响力

⑸将群组聚类算法和影响力算法应用到团购推销中

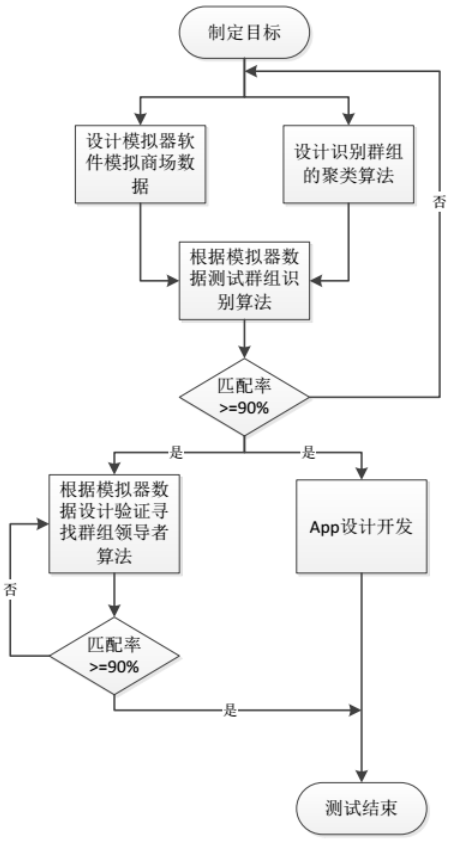

图 **2.1** 项目方案流程

针对于算法我们针对数据的一些特征进行算法设计，然后将实际的数据用于算法的验证。在对算法进行评估之后再根据数据的特征进行进一步的改进，以获得较高的准确率。

1.4 研究内容

初期我们主要进行数据的采集和聚类算法的设计验证，因此主要内容如下：

(1) 设计出手机商场人群数据的方法，开发收集数据应用

(2) 设计聚类算法

(3) 设计商场人群和商店模型开发模拟器软件进行聚类算法设计验证

(4) 采集实际数据，根据实际数据进行聚类算法的修改与评估

第 2 章　模拟器实现

模拟器界面是基于 jbox-2d 的图形界面，分两块分步执行。

2.1 模拟器建模

首先需要说明一下我们的建模

2.1.1 消费群体性质

每个消费群体随机由 1 到 8 人组成，其中有一人是该群体中最具影响力的人（leader），同时每个群体有一定的活动范围，在模拟器中设置为以 leader 为中心，以一定的长度为半径的圆，每位群成员在一般活动时不会超过该范围。群体与群体之间在没有交流的前提下几乎没有相互的影响。

2.1.2 商场及商店促销

商场由 12 个商店组成，均匀分布在 12 块不同的位置上，每个单位时间每个商店都有百分之 0.017 的概率进行促销活动并持续一段时间，促销活动会引起周围群体的注意从而对其产生吸引，使得整个群体向该商店区域移动。如果在促销时间段内该商店区域的人数达到一定的阈值，则认为该商店促销成功，触发商店聚集事件。但是无论是促销还是聚集，每个群体还是会以一定的概率离开该商店区域，当该商店处于聚集事件时，但区域的人少于阈值，则聚集事件结束，促销事件也随之结束。

2.2 模拟器功能

模拟第一步----模拟商场促销事件及群体的行为并记录相关信息

每个单位时间每个商店都有百分之 0.017 的概率进行促销活动并持续一段时间，这段时间可以自己设置。群体的每个成员（非 leader）运动模式共有三种 1）在活动范围内活动的时候速度是一定的，方向是随机的 2）在活动范围的边缘的时候，速度标量增大，方向指向 leader3）在群体被商店吸引的时候，方向和 leader 移动方向相同，速度和群体移动速度相同，跟着群体一起向商店移动。群体的 leader 运动模式共两种 1）在没有被商店吸引的时候随机方向以一定的速度运动 2）在被商店吸引的时候向着商店以更快的速度移动。

将每个单位时间点的每个群成员的 id（唯一），是否被吸引，被哪个商店吸引等信息记录下来，再将 12 个商店每个单位时间点上商店的 id 和商店是否发生促销及商店是否发生聚集事件等信息记录下来。

模拟第二步—从数据中寻找聚集事件之间的关系

目前设置了三种聚集事件之间的关系 1）start-start，即一个聚集事件的开始在另一个聚集事件的开始之前发生 2）start-end，即一个聚集事件的开始在林一个聚集事件的结束之前发生 3）end-end，即一个聚集事件的结束在另一个聚集事件的结束之前发生。通过对若干次第一步实验所得结果的分析，取平均，来得到三种聚集事件之间关系模式的图像。这里我们采用控制单一变量变化的方法来对一些变量是否对聚集事件产生影响进行研究，我们选择的 5 个变量依次是人群的数目，群体影响范围的半径，促销时间的长短，商店促销的频率，人行走的速度来研究。同时分同步和异步两种种对于事件序列不同的划分来研究其影响。同步即事件发生都处在同一个物理钟下面，异步则是抛弃同一条时间轴，以先后次序来生成事件发生的序列。

第 3 章　数据采集与处理

在考虑如何进行用户的聚类的时候，我们考虑了两个因素，一个是空间上距离，但是由于商场中也许存在比较拥挤的状况，因此仅仅依靠距离是不够的。所以另一个则是同组内用户本身的一些相似性或者跟随性。空间上的距离比较简单实现即可以通过在室内对用户位置的定位来完成。而用户本身的相似性有很多个方面，包括运动方向、加速度等等。想要检测到这种运动的变化，在以往的论文中有使用摄像头数据进行图像跟踪识别也有使用传感器数据进行识别的。摄像头视频识别本身比较复杂，且数据比较难获得，成本较高。使用传感器的同样成本比较大。所以我们初期的实验是使用安卓应用采集手机的传感器数据，还有定位数据并发送到服务器。下面就具体介绍一下数据采集以及处理的系统。

3.1 数据采集安卓端原理

3.1.1 传感器数据原理

图 3.1 展示了安卓手机的坐标系统，指向正常摆放时右侧为 x 正向，前方为 y 轴正向，从屏幕垂直指向外面为 z 轴正向。安卓端传感器数据采集主要使用的传感器是加速度传感器和方向传感器。加速度数据分为 x, y, z 三个方向.xyz 三个值分别为在该方向加速度的负值。方向传感器同样分为三个方向，value[0]即方位角（Azimuth）是指磁北方向与 y 轴的夹角，即手机指向正北的时候 0，东 90，南 180，西 270，可以说是相当于一个指南针作用。value[1]即倾斜角(Pitch) 是指围绕 x 轴的一个转动，取值范围在（-180，180 之间），当 z 轴转向 y 轴的时候为正。value[2]即滚动角(Roll)是指围绕 y 轴的一个转动，取值范围在（-90，90）之间，当设备绕 y 轴顺时针值增大。

我们采集数据时主要使用的是加速度的数据以及方向传感器的 value[0]，因为手机向前指的方向与正北方向的夹角已经可以大致确定当前这个人行动的方向。

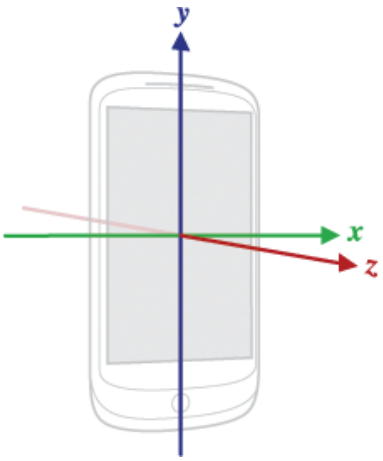

图 **3.1** 安卓手机坐标系

安卓端的设计比较简单，主要是显示传感器数据，以及进行数据传输和停止数据传输。程序使用了 SensorManager 类来注册每个传感器，并采集传感器数据。此外为了准确识别每一部手机，我们同时像服务端传送了手机唯一识别的 id。传送数据的协议是 soap. 安卓手机传感器数据的频率分别有 GAME，UI，NORMAL，为了确保数据的实时性同时保证数据不会过多，我们采取了 UI 频率即 16~17Hz。手机端每收集满 200 条传感器数据即往服务端发送一次请求，从而保证服务端响应而且不会影响数据处理的即时性。

图 **3.2** 安卓端采集 UI

3.1.2 定位数据原理

在安卓客户端，实时获取各个 Access Point 点 RSSI（接收信号强度指示器 Received Signal Strength Indicator）值。如图 3.2 所示 WiFIList 底下的数据

表示的是接入点以及与接入点的 RSSI 值，这个数据可以调用安卓系统的 API 中获得。后期将会使用用户与每个接入点之间的 RSSI 值和训练集进行匹配获得一个定位数据，目前测试的距离误差在 1m 之内。

3.2 数据采集服务器端

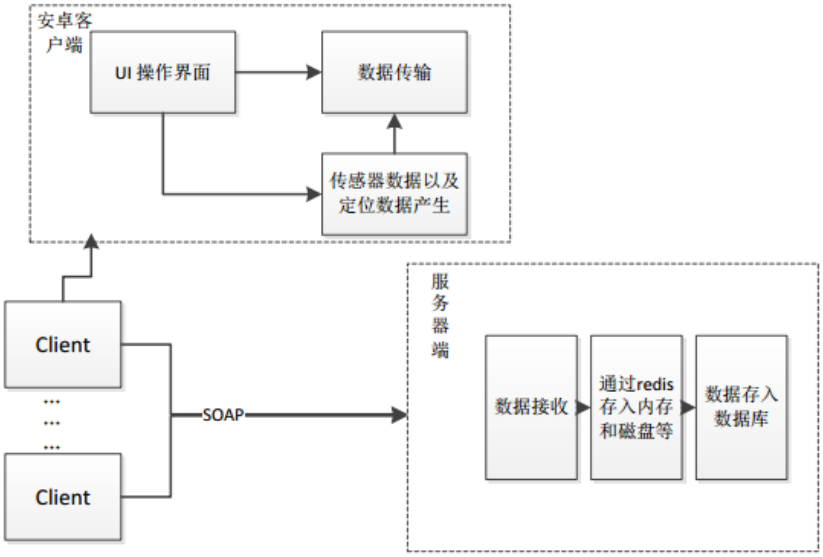

图 **3.3** 数据采集系统图

这个系统目前的后台服务器的责任主要是接受数据，并将数据存入数据库用于后期数据处理。整个后台服务的一个较大的难点在于，数据本身产生速度快且所需空间较大，如果直接在接收的时候就进行数据库的存入，数据库本身有的最大连接数限制一方面会使数据丢失，另一方面也会导致数据库本身的崩溃。所以在想要在运行在 tomcat 上基于 mysql 的服务器完成如此强度的数据存储需要另寻蹊径。在学习操作系统知识的时候，我们曾经学过生产者消费者如果生产者的速度大于消费者，使用缓存会减轻消费者的负担，这个原理同样可以用于这里。我们知道计算机中相对于磁盘存储比较快的就是内存存储。

在客户端数据传过来的时候将数据留在内存中同时另外一边从内存中取出数据存入数据库，起到一个缓存作用。这里我们用到了一个开源的缓存工具—Redis. Redis 是一个开源的使用 ANSI C 语言编写、支持网络、可基于内存亦可内存持久化的日志型、key-value 数据库。它所具有的功能就是能够快速的进行数据的存取，因为它将数据以 key-value 的方式存入的内存，并在一定时候将数据存入

硬盘，因此使用 Redis 可以起到一个临时的数据缓存作用，另外一个程序将数据从 Redis 中读取批量存入数据库中。数据库的结构较为简单，针对每部手机会有其对应的各种传感器表，第一栏为精确到毫秒的时间，后面为传感器的几维数据。

3.3 数据采集过程与处理

完成安卓客户端和服务器端的实现之后，如何采集数据也是为后面算法验证要做的重要的一步。实验设计如下，一组实验中收集了 10 台安卓设备，分别编号 1-10，然后预先设定持手机同学的分组(例如，1234 一组，5678 一组，910 一组)。实验过程中持手机同学需按照预设分组进行运动，模拟在商场中购物吃饭等一系列行为。

此外，当消费群组在商场中时，当用户本身没有交互时只是单纯一起行走的时候，他们的运动行为（包括方向，加速度）一般比较一致，但是如果几个朋友一起聊天讨论时他们的运动特征可能略有不同，我们称此为无交互以及有交互情况。这两种情况通常都会出现在现实生活中，所以我们设计的算法需要同时考虑。因此在一组试验中，前 8 分钟我们会进行无交互的群组行为，后 5 分钟会进行有交互行为，使数据起到前后对照的作用。

采集到的数据开始时间和结束时间不同需要对十台设备数据时间进行比较，截取共有的时间段并排序。此外为观察数据是否正常，我们在预处理的过程中会对每秒的数据进行计数，以对各台设备数据有更进一步了解。

第 4 章 算法设计

4.1 基于函数拟合的算法设计及优化

4.1.1 数据输入

我们所记录的数据为三列数据，用英文逗号隔开，第一列保存的是消费者的编号，第二列保存的是时间戳，单位为秒。第三列保存的是消费者在相应时间戳的加速度大小。单位为秒/米²。

将数据输入到程序中的 Python 代码为（使用数据字典以“{user: {ts: [mean, std]}}”的存储结构存储读入的数据）：

代码 **4.1** 处理数据读入的 Python 代码

```
datamap = {}
simmap = {}
simmap1 = {}
#datamap - {user: {ts: [mean, std]}}
fname = 'data'+str(EXPID)+'_'+str(1)+'.csv'
pfile = open(fname, 'r')
pline = pfile.readlines()
lastuser = 0
lastts = 0
tmpvalues = []
def recordInDatamap(user,ts):
        global tmpvalues
        if TYPEFEATURE==1:
                datamap[user][ts][0] = numpy.mean(tmpvalues)
                datamap[user][ts][1] = numpy.std(tmpvalues)
        elif TYPEFEATURE==2:
                vsin = 0.0
                vcos = 0.0
                for j in range(len(tmpvalues)):
                        vsin += math.sin(tmpvalues[j]*math.pi/180)
                        vcos += math.cos(tmpvalues[j]*math.pi/180)
                datamap[user][ts][0] = math.atan2(vsin,vcos)
                datamap[user][ts][1] = math.sqrt(vsin*vsin+vcos*vcos)/len(tmpvalues) #compute R
        elif TYPEFEATURE==3: #for Mclust, datamap[user][ts][1] has no meaning, while datamap[user][ts][0] contains
a Matrix of the results from Mclust
                if len(tmpvalues)>1:
                        MclustModel=r_mclust.Mclust(r_objs.FloatVector(tmpvalues))
                        #equals to alpha<-MclustModel$parameters$pro in R
                        alpha=MclustModel[11][0]
                        #equals to mean<-MclustModel$parameters$mean in R
                        mean=MclustModel[11][1]
                        #equals to std_sigmasq<-MclustModel$parameters$var$sigmasq in R
                        std_sigmasq=MclustModel[11][2][3]
                        if(r.length(mean)[0]>r.length(std_sigmasq)[0]):
                                std_sigmasq=r.rep(std_sigmasq,r.length(mean))
                        datamap[user][ts][0] = r.matrix(r.c(mean,std_sigmasq,alpha),ncol=3)
                        datamap[user][ts][1] = None
                else:
                        del(datamap[user][ts])
        tmpvalues=[]
for line in pline:
        if line=="\n" or line=="\r\n":
                continue
        fields = line.split(',')
        user = int(fields[0])
        ts = int(int(fields[1])/(TSSIZE*LENWINDOW))
        value = float(fields[2])
        if not user in datamap:
```

```
            if tmpvalues!=[]:
                recordInDatamap(lastuser,lastts)
            datamap[user]={}
            datamap[user][ts]=[] #[0.0, 0.0]
            datamap[user][ts].append(0.0)
            datamap[user][ts].append(0.0)
            lastuser=user
            lastts=ts
        elif not ts in datamap[user]:
            recordInDatamap(lastuser,lastts)
            datamap[user][ts]=[] #[0.0, 0.0]
            datamap[user][ts].append(0.0)
            datamap[user][ts].append(0.0)
            lastts=ts
        tmpvalues.append(value)
recordInDatamap(lastuser,lastts)
pfile.close()
```

4.1.2 基于 Mclust 的积分方法

为了和基于 jd-Gaussian 的方法进行对比，我们设计了基于多模式聚类的积分方法，来测试基于 Mclust 的积分方法所得的准确度和基于 jd-Gaussian 的积分方法所得的准确度谁更高。

基于 Mclust 的积分方法的计算方式如以下代码所示：

代码 **4.2** 基于 Mclust 的积分方法（初版）

```
def jdFuncForMclust(matrixi,matrixj):
    num_rows1=r.nrow(matrixi)[0]
    num_rows2=r.nrow(matrixj)[0]
    def integrandForMclust(floatSexpVector):
        resultList=[]
        for ele in floatSexpVector:
            h1=0
            for i in range(1,num_rows1+1):
                temp1=matrixi.rx(i,1)[0]
                temp2=matrixi.rx(i,2)[0]
                temp3=matrixi.rx(i,3)[0]
                value=1/r.sqrt(2*r.pi[0]*temp2)[0]
                exp_content=-(ele-temp1)*(ele-temp1)/(2*temp2)
                h1=h1+temp3*value*r.exp(exp_content)[0]
            h2=0
            for i in range(1,num_rows2+1):
                temp1=matrixj.rx(i,1)[0]
                temp2=matrixj.rx(i,2)[0]
                temp3=matrixj.rx(i,3)[0]
                value=1/r.sqrt(2*r.pi[0]*temp2)[0]
                exp_content=-(ele-temp1)*(ele-temp1)/(2*temp2)
                h2=h2+temp3*value*r.exp(exp_content)[0]
            resultList.append(r.abs(h1-h2)[0])
        return r_interface.FloatSexpVector(resultList)
    return r.integrate(integrandForMclust,-numpy.inf,numpy.inf)[0][0]
```

但是这段代码所得到的精确度并不如基于 jd-Gaussian 的积分方法。最大精确度要低 30%。因此我们模仿了 jd-Gaussian 方法的部分思想，改进了我们的 Mclust 积分代码：

代码 **4.3** 基于 Mclust 的积分方法（改进版）

```
def jdFuncForMclust(matrixi,matrixj):
    num_rows1=r.nrow(matrixi)[0]
    num_rows2=r.nrow(matrixj)[0]
    num = int(math.ceil(100*math.sqrt(3)))
    integration=0.0
    for step in range(num):
        x = step/10.0
        yi=0.0
        yj=0.0
        for i in range(1,num_rows1+1):
            meani=matrixi.rx(i,1)[0]
            stdi=matrixi.rx(i,2)[0]
            alphai=matrixi.rx(i,3)[0]
            yi = yi+alphai/(math.sqrt(2*math.pi)*stdi)*math.exp(-0.5*math.pow((x-meani)/stdi,2))
        for j in range(1,num_rows2+1):
            meanj=matrixj.rx(j,1)[0]
            stdj=matrixj.rx(j,2)[0]
            alphaj=matrixj.rx(j,3)[0]
            yj = yj+alphaj/(math.sqrt(2*math.pi)*stdj)*math.exp(-0.5*math.pow((x-meanj)/stdj,2))
        try:
            integration += (yi-yj)*math.log(yi/yj)
        except:
            pass
    integration *= 0.1
    return integration
```

这段改进版的代码的确提高了我们最后所得到的准确度。相比于代码 4.2 的初版，改进版得到的准确度提高了 20%，但是相比于 jd-Gaussian 积分方法还是会底 9%。不过对于基于 Mclust 的方法，能让所得的准确度达到这个水平，已经相当不易了。

4.1.3 基于 Mclust 的积分方法的 30 倍速度优化

4.2 节中的两段未经过速度优化的代码使用的大多是 rpy2 模块的方法。rpy2 模块的方法是用 Python 底层调用 R 语言控制台，速度自然会慢。因此代码 4.2 和代码 4.3 各运行一次，都要花上 12 小时左右的时间。这种速度我们是无法忍受的。

经过分析，我们发现，代码 4.2 和代码 4.3 都要做大量的循环处理，每次循环都要使用很多的 rpy2 模块内的方法。而且循环每隔一定的次数，都会再次使用之前使用过的数据。因此要优化代码的速度，我们从这两个痛点入手进行速度优化。我们把循环中使用到的 rpy2 模块方法替换成等价的且更高效的 Python 内置方法，这样速度就会有很大程度上的提高。

代码 4.2 经过优化后的 Python 代码如下：

代码 **4.4** 代码 4.2（Mclust 积分方法初版）经速度优化后的代码

```
def jdFuncForMclust(matrixi,matrixj):
    num_rows1=r.nrow(matrixi)[0]
    num_rows2=r.nrow(matrixj)[0]
    num = int(math.ceil(100*math.sqrt(3)))
    integration=0.0
    meani_list=[]
    stdi_list=[]
    alphai_list=[]
```

```
    meanj_list=[]
    stdj_list=[]
    alphaj_list=[]
    for i in range(1,num_rows1+1):
        meani_list.append(matrixi.rx(i,1)[0])
        stdi_list.append(matrixi.rx(i,2)[0])
        alphai_list.append(matrixi.rx(i,3)[0])
    for j in range(1,num_rows2+1):
        meanj_list.append(matrixj.rx(j,1)[0])
        stdj_list.append(matrixj.rx(j,2)[0])
        alphaj_list.append(matrixj.rx(j,3)[0])
    for step in range(num):
        x = step/10.0
        yi=0.0
        yj=0.0
        for i in range(num_rows1):
            meani=meani_list[i]
            stdi=stdi_list[i]
            alphai=alphai_list[i]
            yi = yi+alphai/(math.sqrt(2*math.pi)*stdi)*math.exp(-0.5*math.pow((x-meani)/stdi,2))
        for j in range(num_rows2):
            meanj=meanj_list[j]
            stdj=stdj_list[j]
            alphaj=alphaj_list[j]
            yj = yj+alphaj/(math.sqrt(2*math.pi)*stdj)*math.exp(-0.5*math.pow((x-meanj)/stdj,2))
        try:
            integration += (yi-yj)*math.log(yi/yj)
        except:
            pass
    integration *= 0.1
    return integration
```

优化后的代码的速度提升了 30 倍左右。实测平均每次运行只需 10-20 分钟的时间，达到可接受的范围。

代码 4.2（Mclust 积分方法改进版）也同样做了类似的速度优化，优化后所得到的代码如下：

代码 **4.5** 代码 4.3（Mclust 积分方法改进版）经速度优化后的代码

```
def jdFuncForMclust(matrixi,matrixj):
    num_rows1=r.nrow(matrixi)[0]
    num_rows2=r.nrow(matrixj)[0]
    meani_list=[]
    stdi_list=[]
    alphai_list=[]
    meanj_list=[]
    stdj_list=[]
    alphaj_list=[]
    for i in range(1,num_rows1+1):
        meani_list.append(matrixi.rx(i,1)[0])
        stdi_list.append(matrixi.rx(i,2)[0])
        alphai_list.append(matrixi.rx(i,3)[0])
    for j in range(1,num_rows2+1):
        meanj_list.append(matrixj.rx(j,1)[0])
        stdj_list.append(matrixj.rx(j,2)[0])
        alphaj_list.append(matrixj.rx(j,3)[0])
    def integrandForMclust(floatSexpVector):
        nonlocal meani_list
        nonlocal stdi_list
        nonlocal alphai_list
        nonlocal meanj_list
        nonlocal stdj_list
        nonlocal alphaj_list
        resultList=[]
        for ele in floatSexpVector:
            h1=0
            for i in range(num_rows1):
                meani=meani_list[i]
                stdi=stdi_list[i]
                alphai=alphai_list[i]
```

```
                value=1/numpy.sqrt(2*numpy.pi*stdi)
                exp_content=-numpy.square(ele-meani)/(2*stdi)
                h1=h1+alphai*value*numpy.exp(exp_content)
            h2=0
            for j in range(num_rows2):
                meanj=meanj_list[j]
                stdj=stdj_list[j]
                alphaj=alphaj_list[j]
                value=1/numpy.sqrt(2*numpy.pi*stdj)
                exp_content=-numpy.square(ele-meanj)/(2*stdj)
                h2=h2+alphaj*value*numpy.exp(exp_content)
            resultList.append(numpy.abs(h1-h2))
        return r_interface.FloatSexpVector(resultList)
    return r.integrate(integrandForMclust,-numpy.inf,numpy.inf)[0][0]
```

经过实际测试，本段代码的一次运行速度也在 20 分钟以内，速度优化所得的优势得以显现。相比于未优化时的代码 4.2 和代码 4.3，代码 4.4 和代码 4.5 的积分算法的速度提升了 30 倍。这个速度提升效率相当可观。

通过以上的分析，我们决定首要使用 DBADP1.py 中的基于 jd-Gaussian 的积分方法。但是我们所编写的最新的基于 Mclust 的积分函数 jdFuncForMclust 仍添加到了 DBADP1.py 中。它作为一个新的积分方法，被我们当作唯一的一个可备选的积分方法，用来应对可能出现的复杂情况。

4.2 基于 Cross-correlation 的聚类算法设计

4.2.1 算法原理

基于 correlation 的聚类，同样将时间序列分窗口，两两计算每个对应窗口的 cross correlation 获得相似度。这里计算 correlation，我们分别使用了原始数据和差值（mean(t-T,t)-mean(t,t+T)），以减少噪音的干扰。最后将所有窗口相似值平均从而得到整体的相似值。得到相似值或者差异值矩阵之后我们可以使用各种聚类算法对数据进行 10 台设备进行聚类，使用的聚类算法是层次聚类。

代码 **4.6** 计算两个窗口数据之间的 cross-correlation

```
def calwincorr(X, Y,delay):
    mx = 0.0
    my = 0.0
    sx = 0
    sy = 0
    for i in range(len(X)):
        mx += X[i]
    for j in range(len(Y)):
        my += Y[j]
    mx /= len(X)
    my /= len(Y)
    for i in range(len(X)):
        sx += (X[i] - mx) * (X[i] - mx)
    for j in range(len(Y)):
        sy += (Y[j] - my) * (Y[j] - my)
    denom = math.sqrt(sx * sy)
#     for delay in range(maxdelay, maxdelay + 1):
    sxy = 0
    for i in range(len(X)):
```

```
            j = i + delay
            if j < 0 or j >= len(X) or j >= len(Y):
                continue
            else:
                sxy += (X[i] - mx) * (Y[j] - my)
    r = sxy / denom
    return r
```

代码 **4.7** 根据最后算出的相似矩阵进行层次聚类

```
def hcluster(dist_matrix,cluster,cnum):

    while len(dist_matrix)>cnum:
        max=-999
        #print(cluster)
        #print(dist_matrix)
        for ci in dist_matrix:
            for cj in dist_matrix:
                if ci==cj:
                    continue
                #print(str(ci)+' '+str(cj))
                if max<dist_matrix[ci][cj]:
                    max=dist_matrix[ci][cj]
                    i=ci
                    j=cj
        s=i
        l=j
        if i>j:
            s=j
            l=i
        #print(str(s)+' '+str(l))
        cluster[s]=[cluster[s],(cluster[l])]
        #print(cluster[s])
        del cluster[l]
        for k in dist_matrix:
            if dist_matrix[s][k]>dist_matrix[l][k]:
                dist_matrix[s][k]=dist_matrix[l][k]
        for r in dist_matrix:
            del dist_matrix[r][l]
        del dist_matrix[l]
    #print(cluster)
    return cluster
```

4.2.2 算法评估

得到聚类结果之后，我们使用(TP+TN)/Total（True Positive，True Negative）来计算准确度的一个数值，即正确分为一组的个数加上正确没有分为一组的个数除以整个分组的个数。

第 5 章　结论和展望

在项目初期，我们初步完成了以下几个目标：

(1)进行了人和商店建模，开发完成了模拟器，并申请了软件著作权。

(2)完成了数据采集系统，包括客户端和服务端。

(3)设计实验进行了多次手机传感器数据采集。

(4)设计了基于传感器数据的两种聚类算法.

(5)使用传感器数据进行了聚类算法的设计和验证，在两种算法上分别获得了65%-85%和 80%-100%的准确率。从准确度来说第二种方法较第一种要更加理想。但是，从算法过程中我们可以发现，第一种方法计算时只需要算出窗口的函数分布参数，然后再比较不同设备各个对应窗口之间的参数的 Jeffery's divergence 即可，而 cross-correlation 则需要每次计算窗口里面所有的数据。当服务器是分布式的时候，后者的数据传输的代价就比较大。所以在接下来的工作我们还需要对算法进一步改进。

接下来我们的工作将继续深入，包括对聚类算法的继续改进，例如将室内定位技术加入群组聚类，对群组进行影响力建模，以及完整的安卓端开发，这将是难点重点。当然做这些算法更多的是希望可以有实际的用处，所以我们也会在开发出安卓应用后进行推广。

参考文献

[1] Moussaid M.;Perozo N.;Garnier S.;Helbing D. Theraulaz, G. The Walking Behaviour of Pedestrian Social Groups and Its Impact on Crowd Dynamics[J].PLoS ONE，2010，5，1-7.

[2] InMobi Insights Team. Global Mobile Media Consumption[EB/OL]. http://info.inmobi.com/rs/inmobi/images/Global_Mobile_Media_Consumption_Wave_2_Whitepaper.pdf, 2013.

[3] eMarketer. Mobile Ad Spend in China Hits $7 Billion This Year[EB/OL]. http://www.emarketer.com,2014.Ghose, A.

[4] Goldfarb A.; Han S.P. How Is the Mobile Internet Different?[J]. Search Costs and Local Activities. Information Systems Research, 2013, 24, 613－631.

[5] Molitor D.; Reichhart P.; Spann M.; Ghose A. Measuring The Effectiveness of Location-based Advertising, A Randomized Field Experiment[EB/OL].http://www.fox.temple.edu/cms/wp-content/uploads/2015/01/mksc-crowd.pdf, 2014.

[6] Luo, X.; Andrews, M.; Fang, Z.; Phang, C.W. Mobile Targeting[J]. Management Science, 2013, 60, 1738－1754.

[7] 阿里 O2O 再出新招 “喵街”年内欲完成 500 家商城上线[EB/OL],http://news.xinhuanet.com/tech/2015-05/06/c_127769048.htm.

[8] Fisher, R.J. Group-Derived Consumption: the Role of Similarity and Attractiveness in Identification With a Favorite Sports Team[M]. Advances in Consumer Research, 1998, 25, 283－288.

[9] 张希伟,戴海鹏,徐力杰,陈贵海.无线传感器网络中移动协助的数据收集策略[J].软件学报，2013,24(2):198-214

[10] 李娜，方卫宁. 基于视频流的地铁人群目标识别[J]-北京交通大学学报（自然科学版）,2006(1).

[11] 吕明琪，王琦晖，胡克用. 移动社交网络环境下的真实社会关系估计[J].计算机应用与软件,2015(1).