### 实验已完成部分总结

**关键点**

* 1. 均基于五种压力类型分别进行判断（除了schedule）
  2. 分schedule与非schedule
  3. 判断是纯粹压力区间还是受正面事件影响的压力区间 (done)
  4. 对压力区间的stress series, type, type rate，以及stress的各种measure进行统计
  5. 找到两种压力区间的set，进行基于stress series和type的 两种KNN correlation

**1. 目标: 筛选schedule的学业压力区间及受正面事件影响的学业压力区间,并统计stress/uplift的值**

**程序：ScheduleUp.java**

* 1. **Function: 计算schedule的stresss及uplift情况**

**static void scheduleUplift(String yearNum)**

* + 1. Note: 只是针对给定列表进行计算；没有自己确定是否受正面影响的压力区间这个功能
    2. 输入：

//该入学年度的student列表

String listPath = "E:\\TEST\\POSITIVE\\Schedule\\userList\_"+yearNum+".txt";

//当前student每日的stress和uplift的值及category

String s\_cate\_userPath = "E:\\TEST\\POSITIVE\\CateSta\\"+cur\_user;

String s\_userPath = "E:\\TEST\\POSITIVE\\DaySta\\"+cur\_user;

String u\_cate\_userPath = "E:\\TEST\\POSITIVE\\upCateSta\\"+cur\_user;

String u\_userPath = "E:\\TEST\\POSITIVE\\upDaySta\\"+cur\_user;

* + 1. 输出：（每个teen一行，是其所有符合要求slide的平均值）

String allTeenU = "E:\\TEST\\POSITIVE\\Schedule\\Result\\upliftSlide\_"+yearNum+".txt";

String allTeenS = "E:\\TEST\\POSITIVE\\Schedule\\Result\\stressSlide\_"+yearNum+".txt";

* + 1. 流程：

将列表事件起止时间进行抽取转换

对应到每一个teen的每日统计量（stressor与uplift，包括类型、值）

通过调用下面的calcuSlide函数对每个schedule区间进行计算

对当前teen，取其各个slide的均值，输出

* 1. **Function:计算输入slide的各种stress measure**

**static ArrayList<Double> calcuSlide(int sDate, int eDate, ArrayList<Integer> valueAry, ArrayList<ArrayList<Integer> >cateAry, int L)**

* + 1. 参数：

sDate为开始日期距离1990间隔

eDate为结束日期距离1990间隔

valueAry为该slide的每日压力序列

cateAry为该slide的每日主题词频序列的序列

L为以下两点间距：// int L = daysBetween("1990/1/1", "2016/1/1");

* + 1. 输入：均为参数输入；无文件输入
    2. 输出：acc, avg, RMS, length, max, 主题词频，主题词比例

res.add(accStr);//0

res.add(avgStr);//1

res.add(RMS);//2

res.add(length);//3

res.add(max);//4

for(int i=0; i<KIND; i++)//5-9; 5-10

res.add(accCateAry.get(i));

for(int i=0; i<KIND; i++)//10-14; 11-16;

res.add(ratioCateAry.get(i));

* 1. **Additional Function:**

**ArrayList<ArrayList<Integer> > scheduleToAry(String yearNum)**

* + 1. 参数：入学年份(目前是2010-2013)
    2. 输入：每个年份入学学生所共享的压力事件起止及uplift事件起止

String upList = "E:\\TEST\\POSITIVE\\Schedule\\"+yearNum+"upEvent.txt";

String stressorList = "E:\\TEST\\POSITIVE\\Schedule\\"+yearNum+"sEvent.txt";

* + 1. 输出：四个arraylist，分别存储uplift起，止，stress起，止

res.add(small\_upAry);

res.add(big\_upAry);

res.add(small\_strAry);

res.add(big\_strAry);

* 1. **Additional Function:**

**读取文件夹下所有文件(a1.txt)，并把文件名写到一个.txt文件中**

**static void collectUsers();**

* + 1. 输入：String userPath = "E:\\TEST\\POSITIVE\\Schedule\\OldUserRecord\\2013\\";
    2. 输出： String testPath = "E:\\TEST\\POSITIVE\\Schedule\\userList\_2013.txt";
  1. **Additional Function:**

**计算两个日期间隔public static int daysBetween(String s1, String s2)**

* + 1. 格式：SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

1. **目标: 找到压力区间及受正面事件影响的区间，输出区间内容（压力值及topic词频、比例）**

**程序：CorUpliftAtoB.java (20191021)**

* 1. **Function. static void compareStress(int teenNum)** 
     1. 输入：学生数量
     2. 功能：调用findSlide()
        1. **第一次是为124人计算各自的是否有正面事件出现的thresh**

res\_cur.add(up\_avgAllAry.get((**int**) (len\*0.2)));

res\_cur.add(up\_avgAllAry.get((**int**) (len\*0.8)));

res\_cur.add(up\_accAllAry.get((**int**) (len\*0.2)));

res\_cur.add(up\_accAllAry.get((**int**) (len\*0.8)));

res\_cur.add(str\_avgAllAry.get((**int**) (len\*0.2)));

res\_cur.add(str\_avgAllAry.get((**int**) (len\*0.8)));

res\_cur.add(str\_accAllAry.get((**int**) (len\*0.2)));

res\_cur.add(str\_accAllAry.get((**int**) (len\*0.8)));

* + - 1. **后面的1-5循环，是为每一种压力类型，分别找[受正面影响/不受正面影响的区间]**
  1. **Function. static ArrayList<ArrayList<Double> > findSlide (ArrayList<ArrayList<Double> > threshAry, int flag, String typeID)（ 主要作用是输出啊~~~20191015）**

1. 参数：thresh 是每个人判断是否受正面事件影响的阈值；flag代表算thresh还是正式算slide；

输入: 与static ArrayList<ArrayList<Double> > findCorrelation() 相同，及压力类型信息

1. 思路理顺

**Step1. 输入每天的uplift/stress(value,category,post)的数据**

**1) stress day statistic**

-value ()

//【DaySta】[accStress,numStress,Fre,y,m,d,h,id\_start,id\_end]

timeResult1<<valStress<<" "<<numStress<<" "<<fre<<" "

<<vec\_year[i].second<<"-"<<vec\_month[i].second<<"-"<<vec\_day[i].second<<" "

<<vec\_hour[i].second<<" "<<i<<" "<<j-1<<endl;  
//压力累计值; 有压力微博条数; 微博条数; 年月日小时; 开始ID，结束ID；

-category ()

//【CateSta】

//String[] up\_elem = up\_cat\_line.split(" ");

//0 3 0 0 1 1 2 1 1 0 0 2,014 8 16 3 //5+ 3 + 3 + 4 [topic5, rol, http, time]

for(int index=1;index<6;index++)

{

topicResult<<topicSta[index]<<" ";

}

for(int index=1;index<4;index++)

{

topicResult<<rolSta[index]<<" ";//role,act,des

}

for(int index=1;index<4;index++)

{

topicResult<<httpSta[index]<<" ";//http, reply, comment

}

topicResult<<vec\_year[i].second<<"-"<<vec\_month[i].second<<"" <<vec\_day[i].second<<" "<<vec\_hour[i].second<<endl;

**2) uplift day statistic**

-value ()【upDaySta】

-category () 【upCateSta】

-post ()

Stress topic 顺序

"Lexicon/eve\_topic\_school.txt", //0

"Lexicon/eve\_topic\_romantic.txt",//1

"Lexicon/eve\_topic\_peer.txt",//2

"Lexicon/eve\_topic\_healthy.txt",//3

"Lexicon/eve\_topic\_family.txt", //4

Positive topic 顺序

"lex\_uplifts/lexp\_study.txt", //0

"lex\_uplifts/lexp\_romantic.txt",//1

"lex\_uplifts/lexp\_friends.txt",//2

"lex\_uplifts/lexp\_self.txt",//3

"lex\_uplifts/lexp\_family.txt", //4

"lex\_uplifts/lexp\_enter.txt",//entertainment-new lexicon 5 !!!!

**Step2. 找到SI和USI**

* + - 1. 根据值大于当前teen的阈值（每个人单独计算）找到slide（SI和USI）

//原程序中，small对应无明显uplift，Big对应有明显uplift

* + - 1. 根据当然输入的type ID (0,1,2,3,4)，根据rank确定当前slide主要stressor类型
      2. 根据当前输入的uplift ID，确定当前slide主要uplift类型

为5x6的correlation做准备

Note: SI的标准有两个：1是stress够，2是uplift不够

* + - 1. 输出每个teen的<5x6>维度的SI.txt/USI.txt，其中每行都包括：

<stress><uplift><post>，即与文章里的三类measure对应

//先细分看feature分布，不行再画图

1. 输出

**124人所有**

1. **（分类型-stress各种measure）**

String allTeen = "E:\\TEST\\POSITIVE\\Pair\\allTeen"+typeID+".txt";

String allTeenS = "E:\\TEST\\POSITIVE\\Pair\\allTeenSmall"+typeID+".txt";

String allTeenB = "E:\\TEST\\POSITIVE\\Pair\\allTeenBig"+typeID+".txt";

1. **（不分类型-stress各种measure）**

String allPair = "E:\\TEST\\POSITIVE\\Pair\\allPair.txt";

String allPairBig = "E:\\TEST\\POSITIVE\\Pair\\allPairBig.txt";

String allPairSmall = "E:\\TEST\\POSITIVE\\Pair\\allPairSmall.txt";

//bw\_small\_pair.write(AVG1\_s+" "+AVG2\_s+" "+ACC1\_s+" "+ACC2\_s+" "+RMS1\_s+" "+RMS2\_s+" "

+SLOPE1+" "+SLOPE2+" "+PEAK+" "+AVG1\_u+" "+AVG2\_u+" "+ACC1\_u+" "+ACC2\_u+" "

+RMS1\_u+" "+RMS2\_u+" "+LEN1 + " " + LEN2 + " " + LEN + "\r\n");

1. **（主题词频及比例）(20191015此处需要调整: 1)应该是区分small和big；2)区分role; 3)区分emotion; 4) post 等semantic的measures，应该在此处体现; )**

String type1 = "E:\\TEST\\POSITIVE\\Pair\\COMPARE\\type.txt";

// format [5 stress type in phase 1][5 stress type in phase 2][5 stress type all phase]

[6 up type in phase 1][6 up type in phase 2][6 up type in all phase]

String typeRate = "E:\\TEST\\POSITIVE\\Pair\\COMPARE\\type\_rate.txt";

// format [5 stress rate in phase 1][5 stress rate in phase 2][5 stress rate all phase]

[6 up rate in phase 1][6 up rate in phase 2][6 up rate in all phase]

**每个teen各自**

1. NOTE: 以peak为分界，分别记录【区间长度及位置】，并统计了在每个压力区间前后两段内的【压力值及uplift值】
2. String PCombinePath = PPath.replace("smoothUp", "Pair\\intervalPair");

//bw\_pair.write(a\_pos+" "+b\_pos+" "+c\_pos+" "+LEN1+" "+LEN2+" "+LEN+"\r\n");

1. String SCombinePath = PPath.replace("smoothUp", "Pair\\valuePair");

//bw\_pair\_value.write(AVG1\_s+" "+AVG2\_s+" "+ACC1\_s+" "+ACC2\_s+" "+RMS1\_s+" //"+RMS2\_s+" "+SLOPE1+" "+SLOPE2+" "+PEAK+" "+AVG1\_u+" "+AVG2\_u+" "+ACC1\_u+" //"+ACC2\_u+" "+RMS1\_u+" "+RMS2\_u+"\r\n");

新计算for 3 measures (20191016)

avgStress, accStress, rmsStress, maxStress, slide length, avgUp, accUp, rmsUp, maxUp [9]

stressPost = [accstrLevel, strFre, Fre] = [3]

upPost = [accupLevel, upFre, Fre] = [3]

frequency of stress topic [5] + role/act/des[3] + http/reply/comment[3]

stressTopicAll [1]

frequency of uplift topic [6]

upTopicAll [1]

共: name+27列

**输出格式for matlab画图：**

filename,

sAVG,sACC,sRMS,sPEAK,LEN,uAVG,uACC,uRMS,uPEAK, [9]

sPOST,upPost,POST,[3]

sTOPIC[5]

ROL,HTTP,sTopicAll,[3]

upTopic[6]

upTopicAll[1]

* 1. **Function: 找到两个集合，计算correlation**

**static void correlationStress(int teenNum, int K)**

* + 1. **参数：其中teenNum是用户数量，K是KNN中的邻居数量定义**
    2. **输入：无外部文件输入；均为参数输入**
    3. **功能：调用findCorrelation()**
       1. **第一次是为124人计算各自的是否有正面事件出现的thresh**

res\_cur.add(up\_avgAllAry.get((**int**) (len\*0.2)));

res\_cur.add(up\_avgAllAry.get((**int**) (len\*0.8)));

res\_cur.add(up\_accAllAry.get((**int**) (len\*0.2)));

res\_cur.add(up\_accAllAry.get((**int**) (len\*0.8)));

res\_cur.add(str\_avgAllAry.get((**int**) (len\*0.2)));

res\_cur.add(str\_avgAllAry.get((**int**) (len\*0.8)));

res\_cur.add(str\_accAllAry.get((**int**) (len\*0.2)));

res\_cur.add(str\_accAllAry.get((**int**) (len\*0.8)));

* + - 1. **后面的1-5循环，是为每一种压力类型，分别找[受正面影响/不受正面影响的区间]**
  1. **static ArrayList<ArrayList<Double> > findCorrelation( ArrayList<ArrayList<Double> > threshAry, int flag, String typeID, int K)**
     1. **参数：**

threshAry是teenNum大小的，每个用户的八种thresh (经上一轮循环得到)

flag代表：1是正经寻找当前type的两类区间；0是只为计算每个用户的thresh

typeID: {0,1,2,3,4}，代表当前压力源类型

K: 代表为knn定义的邻居数量

* + 1. **输入：**

String dirParseIn = "E:\\TEST\\POSITIVE\\smoothUp\\"

String posiCatPath = PPath.replace("smoothUp", "upCateSta");

String SPath = PPath.replace("smoothUp", "smoothStr");

String strCatPath = PPath.replace("smoothUp", "CateSta");

* + 1. **输出：**
       1. **所有区间（124人所有，按各类型，ALL/BIG/SMALL）**

**//small: 受到Obvious Uplift;**

**//big: 未受到Obvious Uplift**

String allPair = "E:\\TEST\\POSITIVE\\Pair\\correlation\\T"+typeID+"\\allPair.txt";

String allPairBig = "E:\\TEST\\POSITIVE\\Pair\\correlation\\T"+typeID+"\\allPairBig.txt";

String allPairSmall = "E:\\TEST\\POSITIVE\\Pair\\correlation\\T"+typeID+"\\allPairSmall.txt";

//bw\_all\_pair.write(s.trim() + "\r\n");

// 0.0188 0.0951 0.1993 0.2896 0.3763 0.5255 0.7025 0.8497 0.92 0.9014 0.8165 0.7069 0.6251

//**NOTE**: 每个区间序列长度可能不同

* + - 1. **每个人(按类型)**

**//1)每个区间压力序列值**

String PCombinePath = PPath.replace("smoothUp", "Pair\\correlation\\T"+typeID+"\\valueBig");

String SCombinePath = PPath.replace("smoothUp", "Pair\\correlation\\T"+typeID+"\\valueSmall");

//bw\_small\_pair.write(s.trim() + "\r\n");

//0 0.1391 0.1595 0.2797 0.4545 0.6061 0.6974 0.7268 0.7137 0.6782 0.6067 0.4889 0.3322

**//NOTE**: 每个区间序列长度可能不同..

**//2)每个区间主题词频率+比例值**

String LCombinePath = PPath.replace("smoothUp", "Pair\\correlation\\T"+typeID+ "\\stressorValueBig");

String LSCombinePath = PPath.replace("smoothUp", "Pair\\correlation\\T"+typeID+ "\\stressorValueSmall");

//stressor\_bw\_small.write(SMALL\_NUM+word+"\r\n");

// 0 1 1 2 0 1 0.2 0.2 0.4 0.0 0.2 [SMALL\_NUM+5type+5ratio]

* 1. **实现KNN方法**

**static Double knnMethod (ArrayList<ArrayList<Double> >strBigAry, ArrayList<ArrayList<Double>> strSmallAry, int K){**

* + 1. 参数：两个set的arraylist，其中每个list元素是一段series，是一个比较的基本单元；K是标记值，表示找到第几个邻居
    2. 输入：即全部参数输入；无文件输入
    3. 返回：是否>1.96的值

Note: ratio（即在同一集合的比例）在末尾的中间结果，未输出

#### Matlab 程序说明

1. TEEN\_Predict.m
   1. 根据AICBIC进行arima(p,0,q)中p和q的调整，取得最优值
   2. 对原bigValue（即压力区间连城的，根据压力类型）的结果进行归一化
   3. 输出normalize和predict值，及MSE,RMSE,MAPE,MAD结果
      1. 个人的，和124平均的分别输出
      2. 按照T0-T5五个类型分别进行预测
2. AdjustYesOrNo.m
   1. 设置四种情况 cor1&&cor2, cor1||cor2, cor1, cor2
   2. 当满足每一种情况，进行调节thresh，即 thresh \*avgHis添加到预测结果上
   3. 程序逻辑
      1. Topic = 0:4

//read in avgHis, avgLen, corStress, corStressor for each Teen

//output:

['E:\TEST\POSITIVE\Pair\correlation\adjust\','metricT',num2str(TOPIC),'.txt'];

* + - 1. Pos = 1:length(fileList)
         1. I=1:1:4 //conditions

Thresh = 0:0.01:1

1. corAdjust.m
   1. 主要功能：调整correlation\_stress\*a和correlation\_stressor\*b
      1. 粒度

for a=0:0.01:1

for b=0:0.01:1

end

end

* + 1. 注：该程序不包括预测部分，只是对预测结果和normalize结果进行对比调整注：每个人单独调整，根据MSE, RMSE, MAPE, MAD（取单个算最小值）

注：124人记录124个最小值的和

* 1. 问题
     1. 感觉没啥意义啊（因为是在预测结果上进行的调整，应该是在新数据才对，或者在原train的数据上，但是又找不到train的数据预测得到的值）

1. testSurface.m

matlab画图测试

1. testPlot.m

matlab画图测试

1. SVARIMA.m

标准seasonal 预测（matlab自带passenger数据）

1. testARIMA.m

标准arima预测，及AICBIC调参（matlab模拟生成数据）

1. TEEN\_SVARIMA.m
   1. 简单地对teen数据进行了seasonal预测的测试
   2. Seasonal间隔根据lenHis得到
   3. 问题：不怎么准啊。。。

检查灰度图

POST!!!

//1)

bw\_measure.write(df.format((AVG1\_s+AVG2\_s)/2)+" "+df.format((ACC1\_s+ACC2\_s)/2)+" "

+df.format((RMS1\_s+RMS2\_s)/2)+" "+df.format(PEAK)+" "+df.format(LEN)+" "

+df.format((AVG1\_u+AVG2\_u)/(-2))+" "+df.format((ACC1\_u+ACC2\_u)/(-2))+" "

+df.format((RMS1\_u+RMS2\_u)/2)+" "+df.format(PEAK\_UP\*(-1))+" ");

//2) stressPost = [accstrLevel, strFre, Fre] = [3]

for(int k=0; k<3; k++){

double tmp = cur\_postAry.get(k);

bw\_measure.write(tmp+" ");

}

//3) upPost = [accupLevel, upFre, Fre] = [3]

for(int k=0; k<3; k++){

double tmp = cur\_up\_postAry.get(k);

bw\_measure.write(tmp+" ");

}