**《时空数据处理与组织课程实习》**

**实习报告**

**学 院: 遥感信息工程学院**

**班 级: 2006**

**学 号: 2020302131249**

**姓 名: 马文卓**

**实习地点: 101机房**

**指导教师: 李晓雷**

**2022年 5 月 30 日**

1. **实验目的**

通过实际问题的运用熟悉spark大数据框架下的rdd（rdd的创建、相关操作函数、格式转换等）、dataframe对象（创建、运行机制、操作函数等）的操作，以及sparksql、window等工具的使用。真正意义上的使用大数据平台去处理实际问题，做到学有所用，将知识融会贯通。

1. **实验环境**

Python版本：3.6.13（conda虚拟环境BG）

第三方库：pyspark、findspark、psutil、math等

系统环境：Window10

综上，即在Win10系统下搭建Spark环境，使用Python语言进行编程解决问题。

1. **实验内容和步骤**
2. **实验内容**

本次实习的内容概括为分析车辆的轨迹。

**1.1实验数据**

这里我们采用的GPS轨道数据集是来自微软亚洲研究院的GEOlife项目，该数据收集了2007年4月到2012年8月的一些交通轨道数据，由一个一个点组成。我们实验采用的是第170号数据的第一个点集，编号为20080428112704，具体数据格式如下：

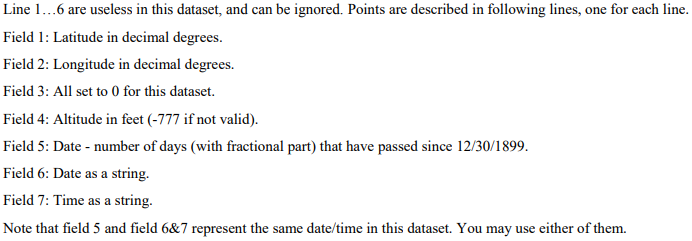


Figure 1:Data Fromat

**1.2具体内容**

（1）车辆速率计算

从原始轨迹数据中获取每个轨迹点的经纬度坐标，通过该轨迹点与前一个点的距离和两个轨迹点的时间差，计算该点的即时速率。根据实际意义，规定起点和终点的即时速率为零。

（2）车辆停留点分析

停留点识别的算法描述如下：

1. 遍历轨迹点，检查速率值，若小于阈值，则将上一个点作为停留开始点，记录其序号。
2. 继续遍历轨迹点，只要轨迹点的速率值仍小于阈值，则将该点作为停留点。

（3）车辆加减速分析

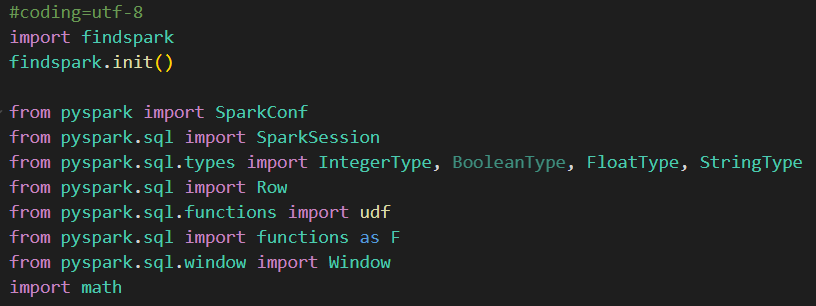
加速和减速都是持续性的过程，而突发的速率变化则可能是数据采集或预处理阶段的误差引起的正常波动，因此加/减速检测方法可以采用寻找至少连续*N*（*N* ≥ 2）个点速率单调变化的片段。

1. **实验步骤**

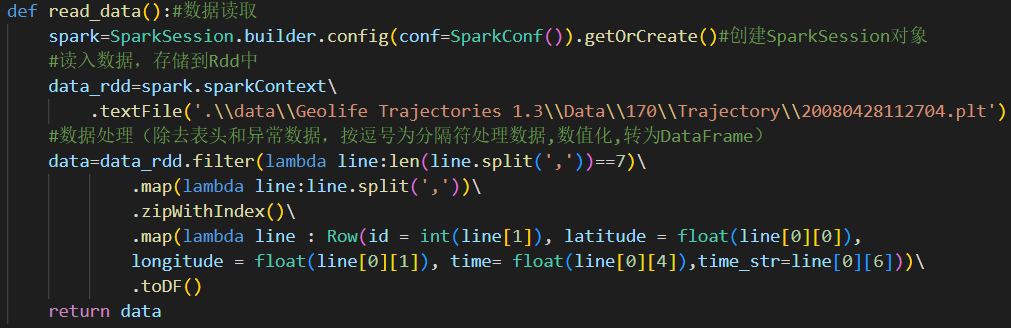
本次实验的主要步骤为：数据读取与预处理->车辆瞬时速度计算->停留点判断->运动状态分析->结果保存。具体实现过程如下：

**2.1数据读取与预处理**

在进行实验之前，首先指定编码格式为utf-8，初始化spark环境，并且导入相关第三方库。具体如下：



本次实验的数据读取和数据预处理封装在函数read\_data中，调用该函数可以返回经过预处理的DataFrame格式数据。首先创建sparkSession对象，使用其textFile函数读取相关数据，得到RDD格式的数据data\_rdd。然后进行预处理：首先使用filter函数过滤表头和异常数据（即限定按逗号分隔后的长度为7），然后以逗号为分隔符进行数据分割，再使用zipWithIndex函数加上索引id，最后将每行数据加上表头生成Row对象（这里我将前四列数据格式转为float类型，方便后面计算），转为DataFrame格式的data。



编写好函数后，在主体框架中调用即可。



**2.2瞬时速度计算**

瞬时速度的计算公式即：速度=距离/时间间隔。因此这个部分我们需要经历三个步骤：距离计算、时间间隔计算、速度计算、对DataFrame进行计算生成speed列。

1. 距离计算：由于我们的初始数据给的是经纬度坐标，因此我们需要从经纬度坐标中计算实际距离。这里我使用的是Haversine公式，公式具体如下：

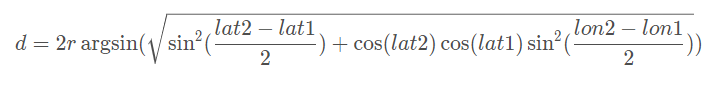
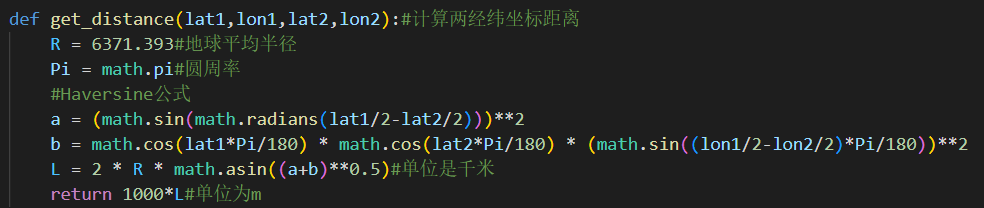


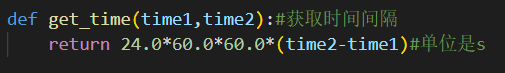
Figure 2:Haversine

按照如上公式，编写获得实际距离的函数get\_distance。这里地球半径取值为6371.393km（越精确越好），最后结果为km单位，转为m返回。

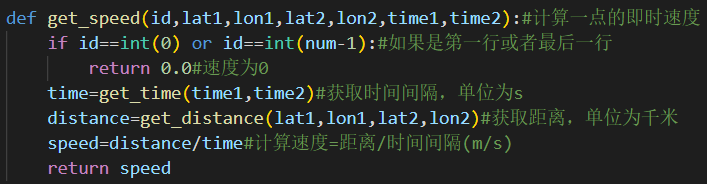


1. 时间间隔计算：由于数据中既以时间戳的形式给出了时间，由以距离1899.12.30日的天数的形式给出时间，因此有了两种计算时间间隔的方法。这里我选择后者计算时间间隔，虽然最后的结果与前者计算有小的差异（小数点后四位开始有差异），但并不影响分析。

编写get\_time函数返回时间间隔，单位为s。



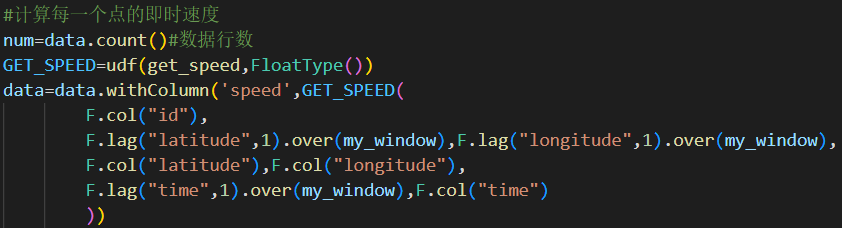
1. 瞬时速度计算：利用前面两个函数获取相邻点的距离和时间间隔，将两者相除，即可得到瞬时速度，单位为m/s。值得注意的是，根据实际意义，我们将第一点和最后一点的速度设为0m/s。



1. 对DataFrame进行计算生成speed列：现在已经可以计算每个点的瞬时速度，可以对data进行计算。使用sparksql中的window函数，可以获取DataFrame格式中每行前（lag函数）后（lead函数）的数据。首先使用Window创建窗口（不分类，按照id排序），具体如下：



然后使用udf（用户定义函数），将get\_speed函数的返回类型限定为浮点型构成自定义函数GET\_SPEED。最后使用withColumn函数增加speed列，其值为GET\_SPEED函数的运算结果。



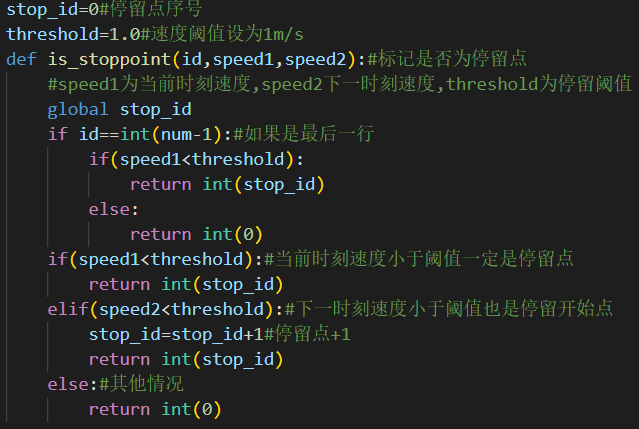
**2.3停留点判断**

停留点判断方法如下：

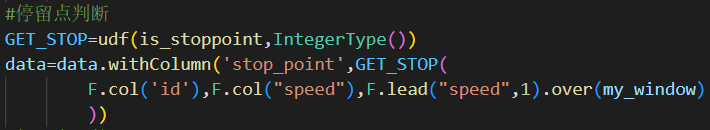
①停留开始点：当前点的速度大于（等于）速度阈值，下一点的速度小于阈值

②停留点：当前点速度小于速度阈值

根据以上停留点判断方法，编写is\_stoppoint函数判断停留点，取速度阈值为1m/s，若是停留点则表上序号，若不是停留点则返回0。

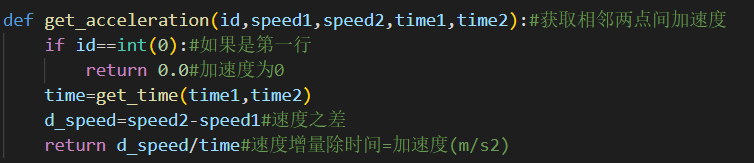


在主程序中和速度求解类似，首先将is\_stoppoint返回类型限定为Integer，包装为udf函数GET\_STOP。使用withColumn函数增加stop\_point列，使用GET\_STOP函数获取其值。

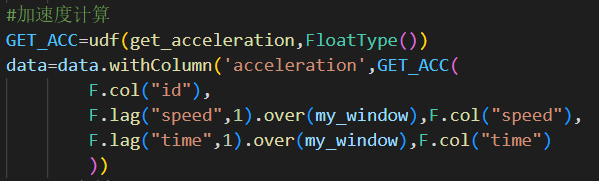


**2.4运动状态分析**

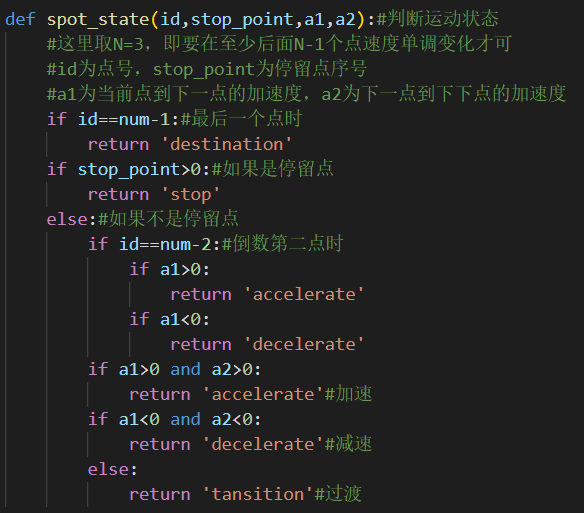
由题目意思可以得知，运动状态的判断需要连续N个区间的速度单调变化，才可认为是加速或者减速。为了判断方便，我们首先求解每个点的加速度（实际上是前一个点到当前点的平均加速度）。其公式为a=（v2-v1）/（t2-t1），根据公式编写get\_acceleration函数，其中设定第一个点的加速度为0，函数返回值单位为m/s2如下：



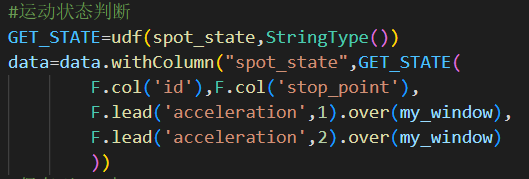
在主程序中，使用上面类似的方法添加acceleration列。



然后编写状态判断函数stop\_state。这里取N=3，即从当前点到下下个点三个点的速度连续变化才可认为当前点为加速或者减速。由于之前计算的加速度为上一点到当前点的平均加速度，因此这里判断当前点的运动状态需要的是下一点的加速度（a1）和下下点的加速度（a2），只要a1、a2>0则加速，a1、a2<0则减速，如果是停留点则标志stop，如果是最后一个点则标志destination，如果连续三点速度变化不单调，则标志位tansition。对于倒数第二个点，由于其后面仅一个点，因此判断两个点的速度是否连续变化即可。具体代码如下：

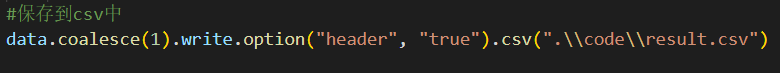


类似的，使用udf包装为GET\_STATE函数。然后使用withColumn函数添加spot\_state列。



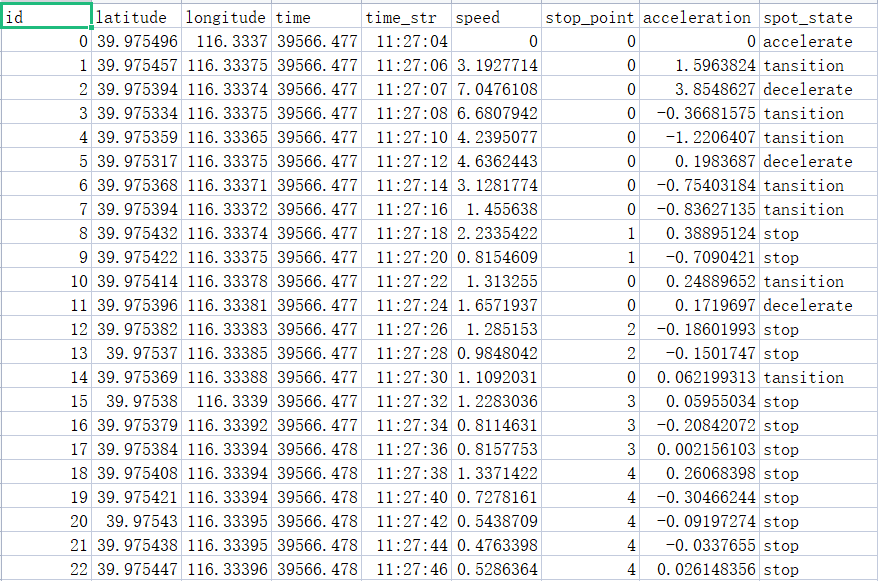
**2.5结果保存**

由于这里的数据为spark dataframe结构，并不是pandas dataframe，所以并不能使用to\_csv函数保存。鉴于spark分布式框架体系，我们使用如下方式进行保存：



1. **实验成果**

本次实验成果为一个result.csv文件（文件可见附录）。其格式为id（点号）、latitude（纬度）、longitude（经度）、time（时间）、time\_str（时间戳）、speed（速度）、stop\_point（停留点标志）、acceleration（加速度）、spot\_state（运动状态）



1. **问题及解决**

本次实验遇到了一些问题，具体如下：

1. **格式问题**：由于使用了窗口函数，导致操作对象全部为column对象，导致出现了很多问题。
2. 例如在调用计算速度函数时，由于第一行数据没有前一行，如果在函数中仍然调用前一行数据则会报错。解决方法就是在加入判断，对第一行进行特殊处理。
3. 在计算距离函数中使用sin、cos等一系列的数学公式时，我开始调用的是pyspark.sql的functions中的函数，但由于其操作对象是column而报错（当时就是说操作对象不为column）。解决方法是调用math中的相关函数。
4. 最开始使用withColumn增加列时，会出现“返回类型不匹配”的错误，解决方法是利用udf将运算函数限定类型，再在withColumn中调用。如下：



2.**特殊数值处理**：比如在计算速度的时候要设置第一行和最后一行的速度为0m/s。当时因为dataframe格式没有办法单独修改某个值而花费了大量的时间。最后使用zipWithIndex函数增加了序号列，根据序号进行判断即可特殊设置。（但是这里的问题和上面类型的问题一起遇到，导致花费了大量时间解决，但也收获颇丰）。

1. **实习体会**

通过本次实验，主要是复习巩固了spark的基础知识，将所学内容运用到实际当中。

对于本次实习的学习方式我是比较适应的，由浅入深，循序渐进，在一个一个的小练习中掌握基础知识，在最后的大实验中融会贯通。对于spark的RDD编程，我认为其更加离散化，虽然也主要是逐行操作，但是通过一些自定义的函数也可以实现数据的自由。而spark的DatFrame则更加结构化严谨化，其主要是行操作甚至窗口操作。至于sparksql则是一个实用额的数据增删改查的工具，将spark和sqsl语句结合从而提供更加多样化的功能。sparkML主要是关于机器学习领域，其流水线思的操作步骤，大大简化了机器学习编程的难度。

本次实验主要是分了车辆轨迹的速度、停留点、运动状态等。事实上在事件问题中我们会碰到各种细节，为了使编程模型更加贴近现实，我们必须处理好这些细节，这也有助于我们提高思维能力和编程习惯。尽管在实验中遇到很多困难，但是在老师的耐心帮助下和自己的细心探索之下，完美的完成了实验任务。

本次实验让我初步认识了大数据处理框架，初步熟悉了大数据处理流程，为以后的学习工作奠定了良好的基础。最后感谢老师的教导，致此。