



中国研究生创新实践系列大赛  
“华为杯”第十六届中国研究生  
数学建模竞赛

学 校 华东理工大学

---

19102510012

参赛队号

---

1.陈卓

---

队员姓名 2.崔玉旺

---

3.朱洪娟

---

# 中国研究生创新实践系列大赛

## “华为杯”第十六届中国研究生

### 数学建模竞赛

题 目

汽车行驶工况构建

摘 要：

本文基于某城市自身的轻型汽车行驶数据，建立了基于运动学片段的工况数学模型。根据对题目的分析，发现是否能够合理的处理有效数据，以及建立适当的运动学片段，很大程度的影响着汽车行驶工况模型构建同实际情况的差异。

**针对问题一的要求**，需要根据所给的条件对数据进行预处理。首先将日期格式转换成易于处理的时间戳形式，按照连续秒数为累计的十进制运算处理日期，这样为后面的计算提供很大的便利。针对信号缺失的时间不连续数据处理，计算出三个文件的时间间断数分别是：文件一中有 725 个间断，文件二中有 2366 个间断，文件三中有 1080 个间断。对于长时间间断导致该部分运动学片段无法正确反应相关运动特征的数据，选择删去该缺失数据两端速度为零的整条片段；对于短时间间断采用线性插值的方式对数据进行平滑处理。对于加速度大于  $4\text{m/s}^2$ 、减速度大于  $8\text{m/s}^2$  的速度尖点情况以及怠速阶段的毛刺数据，这一段包括从缺失数据时间开始之前处于怠速时刻到缺失数据时间末处于怠速时刻的运动片段，通过计算运动状态的加速度，筛选出异常加减速度的运动片段，同样对其插值去尖峰化处理。对于长时间停车和长时间怠速情况，设置 180s 的阈值，时间过长的数据删除处理。

**针对问题二的要求**，对第一问的预处理后的数据中提取运动学片段。根据运动学片段的定义，即汽车从怠速状态开始至下一个怠速状态开始之间的车速区间，考虑到运动过程的复杂性，结合当前点的加速度、当前点的速度和下一点的速度进行运动学片段的分离。由于存在时间或路程长度过短的短行程，此类片段的数据不能反映出特征指标的大部分信息，所以选择通过将其进行删除。最终从文件一中得到 1455 条运动学片段，从文件二中得到 1690 条运动学片段，从文件三中得到 1289 条运动学片段。

**针对问题三的要求**，对问题二中的提取的运动学片段中计算出对应的特征指标，建立关于特征指标的数据集，选取 14 项运动学特征指标作为特征向量。由于数据量比较大，运动学指标比较多，在文献<sup>[1]</sup>中，发现 PCA 主成分分析对汽车行驶工况数据处理能得到较好的结果，所以选择通过 PCA 主成分分析对样本数据进行降维处理。主要是将所得的样本数据做标准化处理，计算出样本数据的协方差矩阵，再利用主成分分析得到特征指标的贡献度求出对样本数据贡献度大于前 80% 的特征指标。通过得到的主成分因子，对所有样本数据进行基于模拟退火算法的 K 均值聚类分析，大大降低所需聚类次数，得出对汽车行驶工况数据影响最大因素。最后依据聚类出的不同类别的特征指标建立出汽车行驶工况，再通过与实际的汽车运行数据作比较，进行误差分析。

**关键词：**数据清洗，运动学片段特征指标，PCA 主成分分析，K 均值聚类法

# 一. 引言

## 1.1 问题背景

随着我国经济的快速发展，汽车已经成为人们日常出行的普遍选择。近年来，机动车保有量保持快速增长，截止至 2018 年底，全国机动车保有量达到 3.27 亿，其中汽车保有量为 2.29 亿，图 1 给出了我国 2008 年至 2018 年十一年间的汽车保有量的变化情况。伴随着汽车数量的增加，汽车工业成为了支柱产业，为我国创造了大量的经济效益和社会效益，汽车工业的健康可持续发展对我国经济发展至关重要。从环保和节能方面来看，汽车排放造成了严重的环境污染，高油耗问题造成了能源的大量消耗，解决汽车排放和油耗问题迫在眉睫<sup>[2]</sup>。

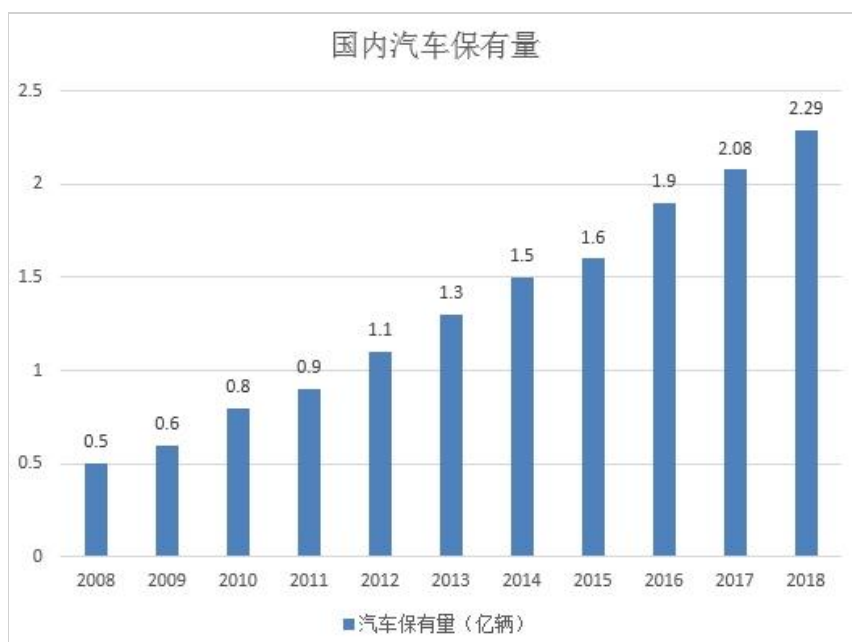


图 1 2008 至 2018 年我国汽车保有量

本文所研究的汽车行驶工况，也叫汽车运转循环是用来表示某一特定车型的车辆在特定的交通环境下行驶过程中的速度和时间曲线，比如市区，市郊或者高速公路等路况下运行，一般时间控制在 1800 秒以内，为了体现汽车在特定道路行驶的运动学特征，是汽车行业以及车辆工程专业相关的一项十分重要的，基础性的技术，通过汽车行驶工况可以测量相关型号车辆的能耗，排放测试等等。

目前世界上主流的汽车行驶工况标准来自于欧洲，美国以及日本。在本世纪初，我国就一直采用欧洲的 NEDC 行驶工况<sup>[3]</sup>，为我国汽车行业提供了很多支持，但是在随着我国的城市建设，高速网络建设，以及汽车数量的飞速增长，国外建立的汽车行驶工况已经和我国的实际路况，基本城市建设有很大区别，已经难以继续如实的反映出我国道路行驶过程中的耗油比和污染物排放，为了能够得到更加适合我国的汽车行驶工况，许多研究机构也做过基于特定城市，比如北京，上海，西安，合肥等的汽车行驶工况<sup>[4]</sup>，根据城市的实际测量数据构建的汽车行驶工况，从而能为我国更加准确调控定制汽车的能源消耗以及汽车的耗油比，尾气排放提供更好的标准。

通过对查阅相关的文献，一般的汽车行驶工况研究流程图如下

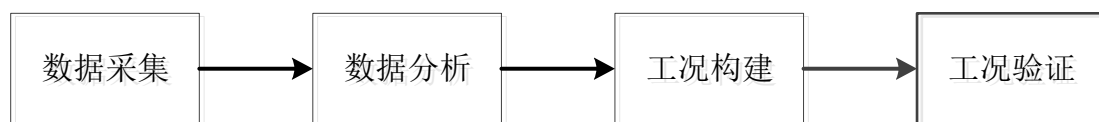


图 2 汽车行驶工况研究流程图

车辆的开发过程离不开对车辆行驶工况的研究和开发，汽车行驶工况是用来表示某一特定车型的车辆在特定的交通环境下行驶过程中的速度和时间曲线。车辆行驶工况是在获取大量的车辆行驶数据的基础上进行的，并对试验数据进行处理所构建的典型工况。充分的车辆行驶工况研究和分析，在制定车辆油耗、排放测试方法和标准等方面起到重要作用，是汽车各项性能指标标定优化的主要基准，也是制定汽车节能减排相关法规<sup>[5]</sup>的依据。目前我国对于车辆的检测普遍采用欧洲工况，由于道路交通更加复杂，车辆情况也发生了很大变化，我国目前使用的欧洲工况已不能很好的反映车辆实际应用时的行驶工况，统一采用欧洲 NEDC 工况（如图 3）进行车辆测试，得到的结果与我国实际车辆使用情况的差异性较大。

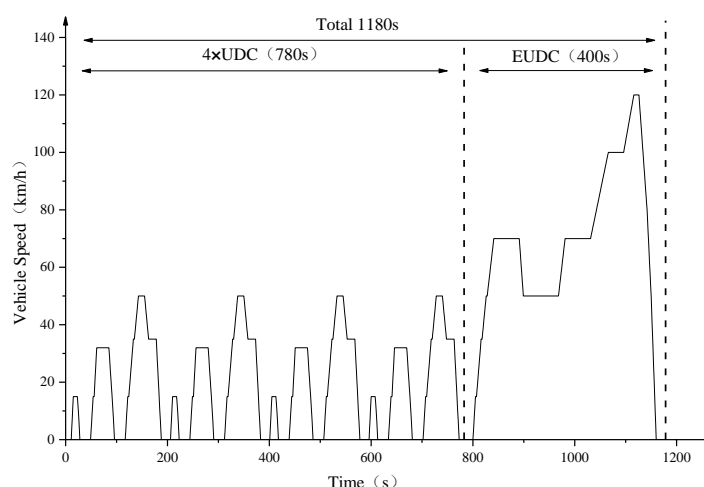


图 3 欧洲 NEDC 工况

因此我国迫切需要对国内车辆实际行驶状况进行调查和分析，针对国内车辆行驶状况构建车辆行驶工况。另外，不同城市和地区的车辆行驶工况都有自己的特点，各地区应该针对各自城市的车辆实际道路行驶状况进行行驶工况的研究和开发，以获取当地真实的车辆行驶数据信息。构建符合当地车辆实际行驶特征工况可以考察某类车辆在某一特定地区的能源消耗量以及为污染物排放提供评价标准和检测依据为发动机及整车设计的动力匹配、经济性能优化提供参考，对于汽车产业的发展、降低排放污染和油耗以及城市交通规划方面具有重大的促进作用<sup>[6-8]</sup>。

1.2 问题描述

为了能够更好的反映出某城市的交通行驶的道路特征，本题通过相同汽车在城市中行驶所收集到的数据来建立该城市的汽车行驶工况，采集频率为 1Hz<sup>[9]</sup>，并给出了三组初始的数据，通过对数据的分析，建立该城市的轻型汽车的行驶工况，并将得到的结果和实际的行驶状态进行对比。

本文考虑的是汽车行驶工况构建的问题。考虑附件中给出同一辆的三个时间段所提供的轻型汽车实际道路行驶采集的数据，利用 excel 对三组数据经纬度进行分析，粗略得出三组数据分别代表市区、郊区、高速三种不同的行驶特征。我们针对不同场景采集的数据，进行运动学分析，得出不同类型道路下的工况及综合各种道路下的工况。三组数据的初始位置如图 4 所示。



图 4 该汽车初始位置定位

首先要求对数据进行预处理，主要的要求包含一下五个部分：

- ① 由于信号丢失造成数据的时间不连续；
- ② 汽车加减速异常，加速度大于正常标准的，减速超过实际情况的运动片段；
- ③ 长期熄火或者不熄火停车的数据；
- ④ 堵车造成的长时间怠速；
- ⑤ 超长时间的怠速状态

其次提取运动学片段，运动学片段是指汽车从怠速状态开始至下一个怠速状态开始之间的车速区间，描述了汽车从加速再到减速的一个完整过程。通过处理后的数据建立一条能够反映数据采集汽车行驶特征的时间长度为 1200-1300 秒汽车行驶工况曲线。

最后计算出汽车行驶工况和处理后的数据的运动指标，使该曲线的汽车运动特征能代表预处理后的数据的相应特征，使两者间的误差最小。考虑创新方法进行特征压缩，使在特征尽可能少的情况下更大地保留信息，并说明构建的汽车行驶工况的合理性。

由于数据是由 GPS 采集而来，信号的噪声及缺失和人为因素，会使数据存在冗余与缺失；采取短行程法进行工况建立时，需要提取运动学片段，指的是汽车从怠速状态开始至下一个怠速状态开始之间的车速区间，通常由一个怠速部分和一个运行段构成，且运行段中至少包含一个加速和一个减速状态。运动学片段组成如图 5 所示。

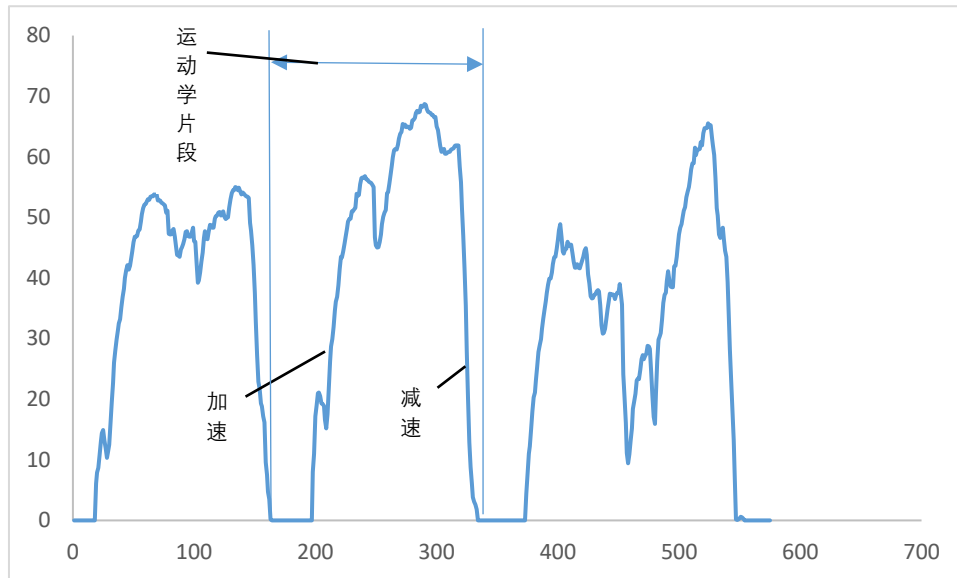


图 5 运动学片段特征指标示意图

根据上述条件及背景，本文主要解决以下问题：

问题一：

对数据进行清洗，使处理后的数据更能反映真实行驶状况。对于数据的时间序列不连续性，缺失的部分进行删除时间点或线性插值增加速度数据；对于由急刹车或急加速引起的加速度绝对值过大和由噪声引起的毛刺进行插值平滑化处理；对于长期停车不熄火、位移不发生改变的时间段进行删除；对于由长期堵车导致断断续续前进的、且最高速度不大于 10km/h 的异常区间按怠速处理，一个怠速状态最长时间为 180s，超过的怠速区间的数据删除。

问题二：

针对对预处理后的数据，选取划分方法，结合筛选条件对车辆行驶进行分割，得出各数据文件最终得到的运动学片段。

问题三：

根据各片段数据，构建一条汽车行驶工况曲线（1200-1300 秒）工况曲线要合理反映汽车的平均速度（km/h）、平均行驶速度（km/h）、平均加速度（ $m/s^2$ ）、平均减速度（ $m/s^2$ ）、怠速时间比（%）、加速时间比（%）、减速时间比（%）、速度标准差（km/h）、加速度标准差（ $m/s^2$ ）等指标，计算出汽车行驶工况与该城市所采集数据源（经处理后的数据）的各指标（运动特征）值，并说明其合理性。

## 二. 模型假设

假设 1：假设汽车在整个测量阶段无重大的性能变化。

假设 2：假设汽车的 GPS 车速信号是没有时间延迟的，其数值代表当前时刻的真实车速。

假设 3：假设汽车行驶的所在的道路保持正常使用，无大面积的封闭。

假设 4：假设在数据测量阶段无恶劣天气，极端状况发生。

### 三. 符号说明

指标编号	符号	符号说明	单位
1	$V_i$	$i$ 时刻对应的速度	$km/h$
2	$\bar{V}$	整个运动学片的平均速度	$km/h$
3	$V_r$	汽车处于非怠速状态时的平均速度	$km/h$
4	$a_{i-1,i}$	$i-1$ 时刻到 $i$ 时刻的加速度	$m/s^2$
5	$\overline{a_+}$	平均加速度	$m/s^2$
6	$\overline{a_-}$	平均减速度	$m/s^2$
7	$a_{\max}$	最大加速度	$m/s^2$
8	$a_{\min}$	最大加速度最大减速度	$m/s^2$
9	$s_+$	加速度标准差	$m/s^2$
10	$s_v$	速度标准差	$m/s^2$
11	$T$	一个运动学片对应的总时间	$s$
12	$T_+$	加速度大于 $0.1m/s^2$ 总时间	$s$
13	$T_-$	减速度小于 $-0.1m/s^2$ 总时间	$s$
14	$T_a$	匀速运动时间	$s$
15	$T_c$	怠速总时间	$s$
16	$B_+$	加速时间占总时间比	%
17	$B_-$	减速时间占总时间比	%
18	$S$	一个运动学片的总路程	$m$

## 四. 数据预处理

数据预处理的目的是为了保证之后数据分析的准确性，降低由于设备或者信号带来的测量误差，设计的数据处理算法应该在满足问题一要求的前提下，尽可能多的保留原始数据的有用信息，这样就能保证处理后的数据具有更加真实可靠，使得到的结果更加可信。

### 4.1 数据预处理要求分析

由于数据采集终端存在一些缺陷及车辆自身的一些原因，会对采集到的试验数据造成一定的误差，包括车速数据的丢失（车速出现冻滞带），出现白噪声及长时间怠速段。误差产生的原因及处理如下：

#### （1）车速数据的丢失：

试验的数据采集终端是基于 GPS 进行开发的，会因为各种因素导致车速数据丢失。考虑到原始数据的庞大和缺失序列较少，我们把对划分运动学片段产生影响的缺失规定为大于 60s。车辆在城市道路内行驶时，GPS 接收信号会不可避免的受到高层建筑、隧道等路段的遮挡，造成定位不成功，车速数据丢失，但此种情况造成的缺失不会太长，大约几秒；考虑车辆在隧道屏蔽信号时间较长，但整体汽车的运行状态不会有大的波动；当汽车熄火且 GPS 停止采集时，将会缺失长时间的速度为 0 的点，此类缺失对速度的连续性不构成影响，所以不予考虑速度，只需把时间速度曲线拼接成连续的即可。

#### （2）白噪声的产生：

由于数据采集终端存在一定的零点偏移，受采样设备自身精度的影响，定位成功的条件下也会出现一定的车速偏差，该误差普遍存在于数据采集设备中<sup>[10]</sup>，表现为车辆怠速和车速较低时，采集到的数据误差较大，会有毛刺，即速度不为 0 且加速度过大，0 至 100km/h 的加速时间小于 7 秒，紧急刹车最大减速度小于  $8 \text{ m/s}^2$ ，需要对车速数据进行线性修正。

#### （3）长时间怠速：

由于车辆与设备的匹配等问题会造成车辆停止后设备不断电，在车辆非工作时间段继续进行数据采集，这就造成了长时间的怠速段。对大于 180s 的怠速段数据进行删除，只保留 180 个怠速数据，选取时间间隔开始前 90s 和时间间隔结束前 90s 的数据，以减小这些不良数据对试验结果的影响。

### 4.2 时间变量的处理

由于数据的时间序列不连续性，缺失的部分进行删除时间点或补充速度值；将原始数据 xlsx 文件转换为 csv 文件，把第一列的时间序列转为时间戳。时间戳是一个能表示一份数据在某个特定时间之前已经存在的、完整的、可验证的数据，是指格林威治时间 1970 年 01 月 01 日 00 时 00 分 00 秒(北京时间 1970 年 01 月 01 日 08 时 00 分 00 秒)起至现在的总毫秒数。原始数据中的时间是一个字符序列，唯一地标识某一刻的时间，格式类似于 2017/11/07 19:08:56.000。我们使用 TIME 包进行格式转换，利用 `strptime()` 函数，将字符串表示的日期时间按要求转换为相应的日期时间。`mktime()` 函数将参数 `strptime()` 函



数所指的 tm 结构数据转换成从公元 1970 年 1 月 1 日 0 时 0 分 0 秒算起至今的 UTC<sup>[1]</sup> 时间所经过的秒数。将时间戳的精度设为秒级，时间的字符格式转换为数据格式后，方便后续对时间间隔的判断。获取的时间戳格式数据和断点个数如下图所示：

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS D:\model\code\data> & C:/Users/cyw/AppData/Local/Programs/Python/Python37/python.exe d:/model/code/data/findpoint.py
生成的第一个时间戳数据: 1513575733.0
断点个数: 725
PS D:\model\code\data>

```

图 6 文件一的间断点统计

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS D:\model\code\data> & C:/Users/cyw/AppData/Local/Programs/Python/Python37/python.exe d:/model/code/data/findpoint.py
生成的第一个时间戳数据: 1509534710.0
断点个数: 2376
时间阈值大于60s需要处理间断点个数: 122 时间阈值不超过60s需要插值间断点个数: 2254
PS D:\model\code\data>

```

图 7 文件二的间断点统计

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS D:\model\code\data> & C:/Users/cyw/AppData/Local/Programs/Python/Python37/python.exe d:/model/code/data/findpoint.py
生成的第一个时间戳数据: 1512128637.0
断点个数: 1098
时间阈值大于60s需要处理间断点个数: 225 时间阈值不超过60s需要插值间断点个数: 873
PS D:\model\code\data>

```

图 8 文件三的间断点统计

考虑到缺失秒数有大有小，大至几个小时，小至几秒，设立阈值 60s 分别考虑大于和小于等于这两种情况，当缺失秒数大于 60 时，将缺失后面的时间全部减去缺失秒数，这样就可以使时间连续；当缺失秒数小于等于 60 时，整体汽车的运行状态不会有大的波动，为减小误差，保留这段时间，插入缺失时间点的速度，使该缺失与前后两端保持平滑。

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS D:\model\code\data> & C:/Users/cyw/AppData/Local/Programs/Python/Python37/python.exe d:/model/code/data/findpoint.py
生成的第一个时间戳数据: 1513575733.0
断点个数: 725
时间阈值大于60s需要处理间断点个数: 351 时间阈值不超过60s需要插值间断点个数: 374
PS D:\model\code\data>

```

图 9 设置阈值 60s 间断点数据

对于由噪声引起的毛刺进行平滑化处理；对于由急刹车或急加速引起的加速度绝对值过大或长期停车不熄火、位移不发生改变的区间进行删除；对于由长期堵车导致断断续续前进的、且最高速度不大于 10m/s 的异常区间按怠速处理，一个怠速状态最长时间为 180s，超过的怠速区间删除，选取时间间隔开始前 90s 和时间间隔结束前 90s 的数据，以减小这些不良数据对试验结果的影响。

本次试验采集的车辆行驶数据信息包括时间、位置信息、速度、油门踏板开度、发动机转速等信息，本文行驶工况构建所需的主要是速度信息，在本地 Excel 文件中对车

辆的行驶数据信息进行整理，保留时间、速度信息，将整理后的行驶数据文件经过预处理，对车速数据突然跳变、不合理加减速等行驶数据信息进行了筛选和消除。车辆行驶的加速度值被限制在  $-8\text{m/s}^2$  和  $4\text{m/s}^2$  之间的合理范围内，

### 4.3 GPS 信号丢失导致的时间不连续

由于速度信息是由 GPS 所收集，当 GPS 信号弱时，时间序列会出现缺失，产生一段时间不连续的数据，有可能导致结果和实际行驶状态之间的误差增大。其次，在保留较短时间的样本中，为了保证时间的连续性，我们通过线性插值来预测缺失数据的车速。

对片段缺失的数量进行统计，三个文件的时间间断数分别为：**文件一中有 725 个间断，文件二中有 2376 个间断，文件三中有 1098 个间断。**这里我们主要考虑了长时间的数据缺失将导致该时间前后的运动学片段不具有可信性，应删除该部分以及其前后的一部分运动状态，但是对于较短时间内的数据缺失，应保留下来，作为运动学样本而言，尽可能多的运动学片段可以提供更多的信息，以及更准确的结果，若是大量删除运动学片段，会导致原始数据的丢失，使得数据可信度降低。

### 4.4 汽车加减速异常的数据、长时间停车和低速运动处理

对于此类数据，我们通过在填充和删除相关数据的基础之上计算出每个时刻的加减速速度，和要求的值作比较，筛选出不符合条件的数据，删除整块运动学片段。

#### 4.4.1. 长时间停车。

长时间停车表示汽车处于静止状态，汽车有可能处于长时间怠速状态，或者发动机停机，发动机停机将导致转速为零，参考三组数据的发动机参数后，得到数据最小值大于 100，说明没有发动机停机状态。根据这一结果，我们得到汽车只可能处于长时间怠速状态，如果在连续的某一段的时间里车辆的速度为零，汽车可能处于道路行驶的过程中，车辆速度为零的可能是在等红绿灯，交通堵塞等情况，但是此类情况汽车的车速不可能长时间一直为零，故当汽车超过 180s 速度为零，我们认为汽车处于停车状态。

#### 4.4.2. 长时间堵车，或间断的低速（速度低于 $10\text{km/h}$ ）行驶

按照怠速处理，考虑到城市中行驶当发生交通堵塞，等红绿灯等情况时会出现车辆持续间断低速，该段的加速度和平均速度很小，对我们数据特征影响不大，所以将此类情况按照怠速处理。根据怠速的定义，是指汽车停止运动，但发动机保持最低转速运转的连续过程，其中最低转速我们通过所给的 Excel 表格中的车速低于  $10\text{km/h}$  数据所对应的发动机转数进行分析，在发动机转速介于  $600\text{r/min}$  至  $800\text{r/min}$  时，车辆是处于怠速状态，对于大于  $800\text{r/min}$  的运动状态视为汽车开始做加速运动，而对于转速较低的数据，视为汽车刚发动时数据。

其中，对于时间大于 180 秒的怠速状态，我们选择删除怠速运动的中间时刻的点来缩减总的怠速时间到 180 秒，这样做是为了保护怠速开始和结束时刻的数据，因为加速刚开始和减速结束阶段，会有速度小于  $10\text{km/h}$  的时刻，但这些点是用来描述加减速度的。

数据预处理流程图如图 10 所示：

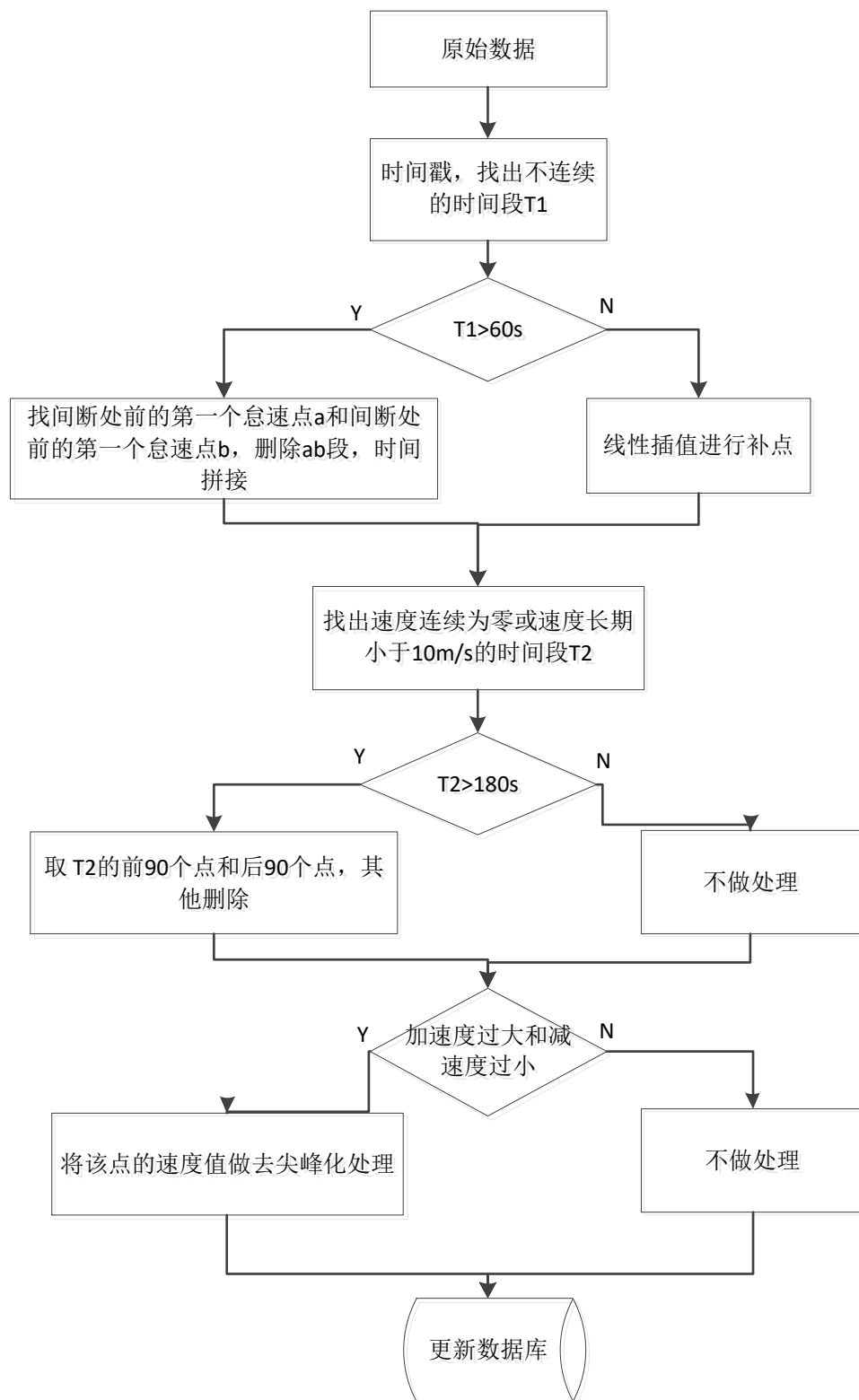


图 10 数据预处理流程

## 五. 运动学片段提取

### 5.1. 运动学片段分析

运动学片段要求所选取的速度-时间片段满足汽车从怠速状态开始至下一个怠速状态开始之间的车速区间，它能够完整的表现出从怠速开始加速，匀速，减速等一系列运动状态，这样的—个片段中能够提取出 18 个特征指标信息：平均速度（km/h）、平均行驶速度（km/h）、平均加速度（ $\text{m/s}^2$ ）、平均减速度（ $\text{m/s}^2$ ）、怠速时间比（%）、加速时间比（%）、减速时间比（%）、速度标准差（km/h）、加速度标准差（ $\text{m/s}^2$ ）等指标，在之后建立模型中，可以通过每一个运动学片段中提取相应的运动学特征指标，建立对应的特征值数据矩阵，就可以进行数据的分析工作。

### 5.2. 运动学片段划分

第一问中在处理好的数据中进行运动学片段的划分的算法，首先算法必须满足运动学片段的划分要求，在文中并未明确给出其划分的要求和条件，但在许多文献中提出了对于时间过短或者行驶路程过短的运动学片段过多，将会造成运动学片段提取的特征指标偏差过大，所以使用文献<sup>[12]</sup>中所给的要求提取运动学片段：

1. 运动学片段持续时间大于 20 秒。
2. 运动学片段的行驶总路程大于 10 米。
3. 车辆加速度在  $0.1\text{m/s}^2 < a < 4\text{m/s}^2$  范围内，减速度在  $-8\text{m/s}^2 < a < -0.1\text{m/s}^2$  范围内。

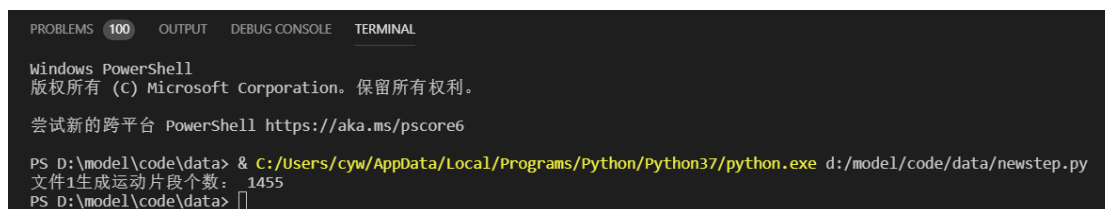
对于不符合要求的片段，将其直接从运动学片段样本中删除。

根据以上的条件，我们得出三个文件的运动学片段，每个文件运动学片段数量如下表所示：

表 1 三个文件运动学片段数量

文件编号	文件一	文件二	文件三
运动学片段数量	1455	1690	1289

使用 python 对数据运行的结果记录图：



```
PROBLEMS (100) OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。
尝试新的跨平台 PowerShell https://aka.ms/pscore6
PS D:\model\code\data> & C:/Users/cyw/AppData/Local/Programs/Python/Python37/python.exe d:/model/code/data/newstep.py
文件1生成运动片段个数: 1455
PS D:\model\code\data>
```

图 11 文件 1 运动学片段数量

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

停止点: 144572
停止点: 144600
生成运动片段个数: 1690
PS D:\model\code\data>

```

图 12 文件 2 运动学片段数量

```

PROBLEMS  100  OUTPUT  DEBUG CONSOLE  TERMINAL

文件3生成运动片段个数: 1289
PS D:\model\code\data>

```

图 13 文件 3 运动学片段数量

运动学片段提取流程图:

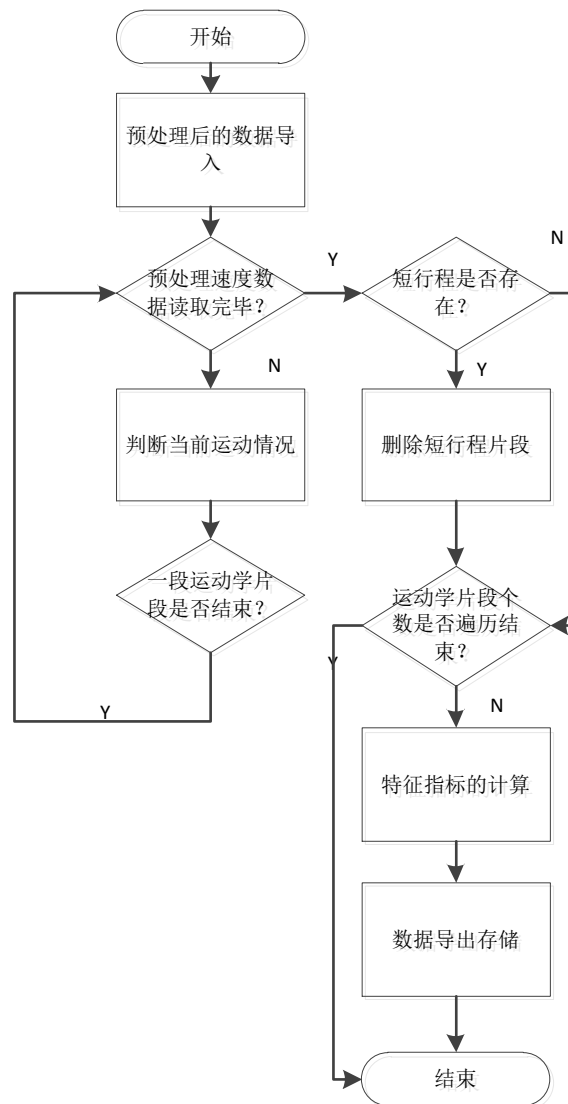


图 14 运动学片段提取流程图

随机选择的一条运动学片段速度时间图:

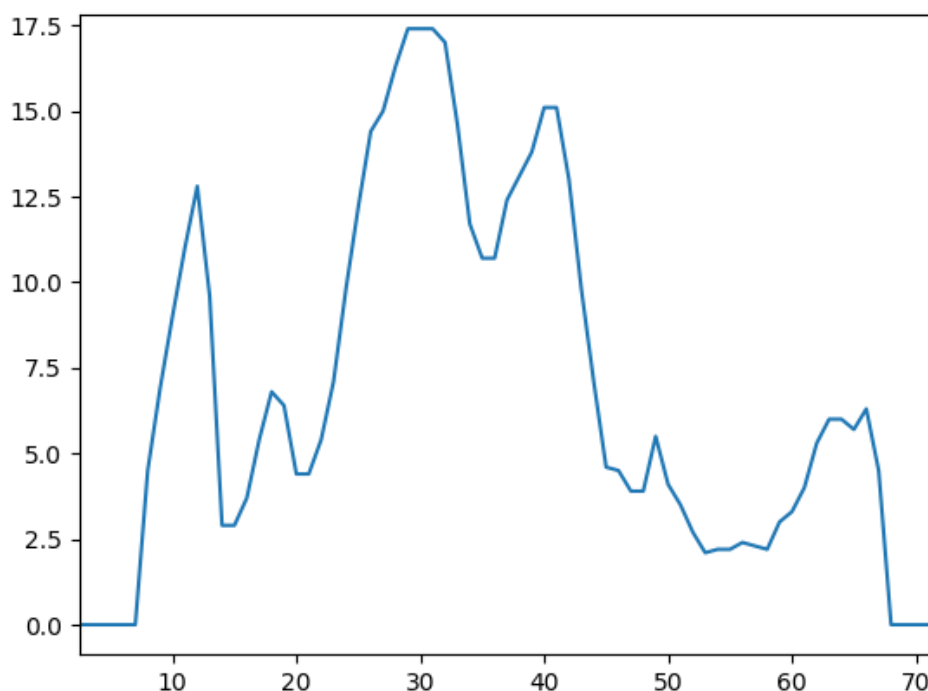


图 15 典型运动学片段示意图

## 六. 汽车行驶工况的构建

在前面的两章中，我们完成了对数据的预处理，以及运动学片段的选取。在这一章中，需要确定所得到的运动学片段特征指标构成的数据矩阵，然后对数据通过 PCA 主成分分析方法，提取贡献率排行前几名主成分因子。利用主成分因子的数量为类数做基于模拟退火算法的 K-means 聚类分析，得到具有不同运动特性的类别，最后从各个类的运动学片段中分别选取具有代表性的运动学片段，组成时间长度为 1200-1300 秒的汽车行驶工况，构成典型汽车运动工况。

### 6.1 特征指标选取

根据所给的数据信息中，包含了车辆的 GPS 车速，X，Y，Z 轴加速度，汽车定位经纬度，发动机转速，扭矩百分比，油门踏板，空燃比，发动机负荷，进气流量。其中最准确的数据是 GPS 车速，相对于其他车辆性能数据有一定参考性，但不够准确。因为汽车的发动机运行具有一定的偶然性，没有准确的规律，其数据不可靠，所以选取 GPS 车速为指标建立特征指标数据。

通过速度和时间关系，我们可以计算出以下的特征指标

1.速度，平均速度，平均行驶速度

速度即第*i*时刻对应的速度为 $V_i$  由表中数据给出，由于记录仪的频率为 1Hz，即每隔一

秒记录一次数据，根据运动学公式 $a = \frac{\Delta V}{\Delta T}$  可知，

第*i-1*到第*i*时刻的加速度为

$$a_{i-1,i} = \frac{V_i - V_{i-1}}{t_i - t_{i-1}} * \frac{1000}{3600} = \frac{V_i - V_{i-1}}{3.6}$$

平均速度

$$\bar{V} = \frac{\sum_{i=1}^n V_i}{T}$$

平均行驶速度

$$V_r = \frac{\sum V_i}{T - T_c} (V_i \text{ 为非怠速状态的速度})$$

## 2.加速度，最大加速度，平均加速度，最大减速度，平均减速度

*i-1*时刻到*i*时刻的加速度 $a_{i-1,i} = \frac{V_i - V_{i-1}}{t_i - t_{i-1}} * \frac{1000}{3600} = \frac{V_i - V_{i-1}}{3.6}$

最大加速度

$$a_{\max} = \max\{a_{i-1,i}\} (i = 1, 2, \dots, n)$$

最大减速度

$$a_{\min} = \min\{a_{i-1,i}\} (i = 1, 2, \dots, n)$$

平均加速度

$$\bar{a}_+ = \frac{\sum a_{i,i-1}}{T_+} (a_{i,i-1} > 0.1m/s^2)$$

平均减速度

$$\bar{a}_- = \frac{\sum a_{i,i-1}}{T_-} (a_{i,i-1} < -0.1m/s^2)$$

## 3.速度标准差，加速度标准差

速度标准差

$$s_v = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (V_i - \bar{V})^2}$$

加速度标准差

$$s_+ = \sqrt{\frac{1}{k-1} \sum_{i=1}^k (a_{i,i-1} - \bar{a})^2}$$

4.总时间，加速时间，减速时间，加速时间比，减速时间比，匀速运动时间  
总时间*T*，为该段运动学片段总时间

怠速时间 $T_c$ ，速度低于 10km/h，且加速度绝对值低于  $0.1m/s^2$  的运动状态的总时间，

加速时间 $T_+$ ，为加速度大于  $0.1m/s^2$  的总时间

加速时间比  $B_+ = \frac{T_+}{T}$

减速时间  $T_-$ ，为加速度小于  $-0.1m/s^2$  的总时间

减速时间比  $B_- = \frac{T_-}{T}$

匀速运动时间  $T_a$  为汽车加速度的绝对值小于  $0.1m/s^2$  非怠速的连续过程，即满足此类条

件的运动时刻总和 
$$\begin{cases} -0.1m/s^2 < a_{i-1,i} < 0.1m/s^2 \\ V_i > 10km/h \end{cases}$$

由于不同类的数据之间的数值差异较大，会对距离造成不同的影响，为了解决这一问题，需要先对各个类之间的数据做标准化处理，

## 6.2 特征值数据的标准化处理

一般对于不同类之间的数据分布不同，直接对不同类之间进行熟知的大小比较不具有可行性，通常会通过标准差标准化和极差标准化来标准化数据<sup>[3]</sup>

一．标准差标准化

设数据集的数据编号为  $x_{ij}$ ，则求出对应的修正样本标准差  $s_j = \sqrt{\frac{1}{n-1} \sum_{k=1}^n (x_{kj} - \bar{x}_j)^2}$ ，其

中  $\bar{x}_j$  表示  $j$  样本的平均值  $\bar{x}_j = \frac{1}{n} \sum_{j=1}^n x_{kj}$

则标准化后的数据

$$x'_{kj} = \frac{x_{kj} - \bar{x}_j}{s_j},$$

这样标准化后的数据均是不带有量纲的，而且其样本的均值等于 0，其修正样本方差为 1。

## 6.3 主成分分析

主成分分析适用于多元统计分析的重要工具，在实际的生活中，对一件事情的描述，可以找到十几个或是几十个相关的指标，这些指标中往往是具有联系的，但并不是每一项指标的具有很强的相关性，实际上我们只需最主要的几个关键性的指标就可以描述出样本的绝大多数信息，而实现这一思想的方法就是主成分分析。

以下是主成分分析的相关数据的具体计算流程图



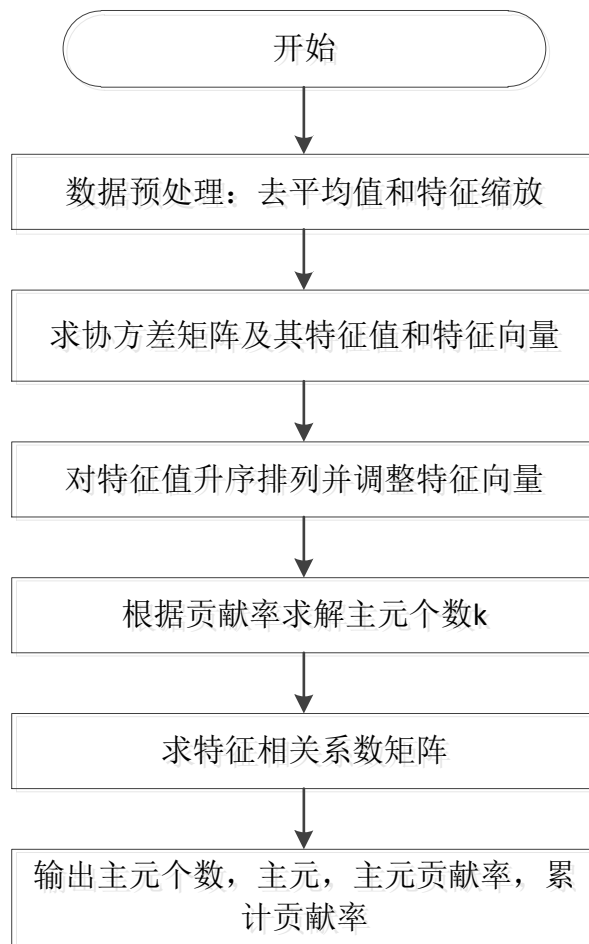


图 16 PCA 主成分分析流程

在对样本做完主成分分析后，我们得到所有的特征指标的贡献率。通常会选择累计贡献率大于 8% 的主成分，对其贡献率从大到小排序，找到前贡献率和为 80% 的前几个特征值标。由表 2 可知，主成分一的累计贡献率为 51.46% < 80%，主成分 2 累计贡献率为 70.33% < 80%，主成分 3 的累计贡献率为 80.33% > 80%，所以我们通过主成分分析成功将原来 18 维的数据将为 3 维，并且通过 python 得到一下的运动学片段在三个主成分下的得分表 2，以及图 2 主成分得分贡献比柱状图：

文件 1 中三个主成分对应的特征由高到低依次排列为平均速度，最小减速度，平均加速度：

表 2 文件 1 部分运动学片段对应的主成分得分

运动学片段序号	主成分 1	主成分 2	主成分 3
1	-1.4490	0.2933	-2.5035
2	-1.3159	-0.0027	-2.3133
3	-1.3236	0.2513	-2.4819
4	-1.3525	-0.0617	-3.4506
5	-1.3540	0.1818	-3.2284
6	-0.6110	1.2980	-5.4523

表 3 文件 1 主成分的贡献率与累计贡献率表

主成分序号	贡献率	累计贡献率
1	51.47%	51.47%
2	18.87%	70.33%
3	10.00%	80.33%
4	5.64%	85.97%
5	4.53%	90.50%
6	2.64%	93.14%
7	2.21%	95.35%
8	1.67%	97.02%
9	1.29%	98.31%
10	0.66%	98.97
11	0.55%	99.53%
12	0.43%	100.00%
13	0.04%	100.00%
14	0.00%	100.00%

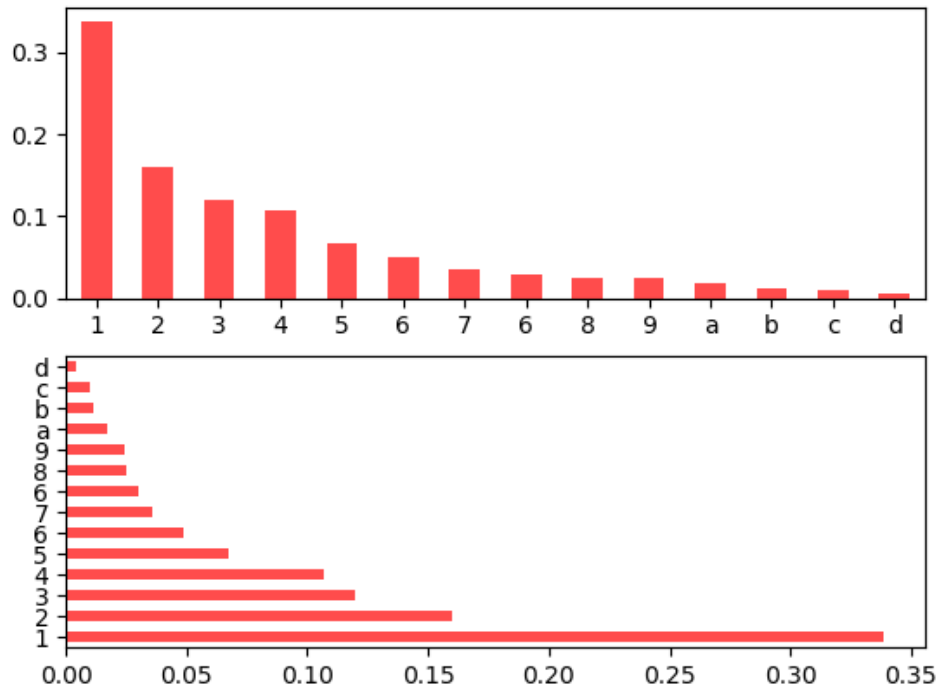


图 17 文件 1 主成分得分贡献比柱状图

文件 2 中三个主成分对应的特征由高到低依次排列为平均加速度，平均减速度，怠速时间比例：

表 3 文件 2 部分运动学片段对应的主成分得分

运动学片段序号	主成分 1	主成分 2	主成分 3
1	-0.0247	-0.6227	-1.6279
2	1.5686	0.6852	5.6784
3	1.8937	1.2266	2.1228
4	1.3506	0.1765	1.3780
5	0.9860	-0.5901	0.5982
6	1.4050	0.2849	1.4892

表 4 文件 2 主成分的贡献率与累计贡献率表

主成分序号	贡献率	累计贡献率
1	43.71%	43.71%
2	17.63%	61.34%
3	12.42%	73.76%
4	8.99%	82.74%
5	5.21%	87.95%
6	3.41%	91.37%
7	2.78%	94.15%
8	1.90%	96.05%
9	1.28%	97.33%
10	0.90%	98.23%
11	0.73%	98.96%
12	0.54%	99.50%
13	0.34%	99.84%
14	0.16%	100.00%

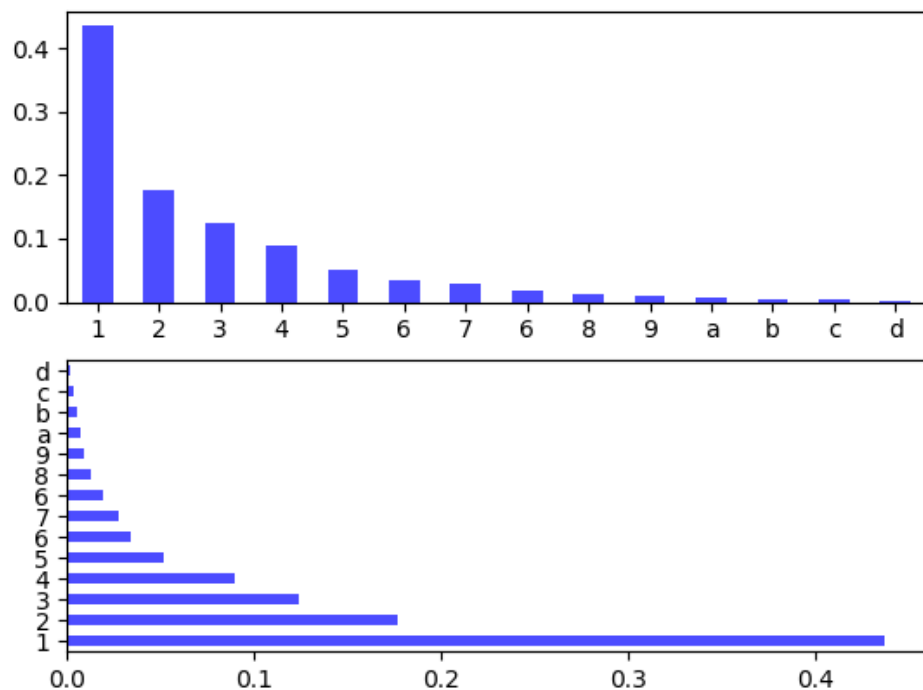


图 18 文件 2 主成分得分贡献比柱状图

文件 3 中三个主成分对应的特征由高到低依次排列为平均加速度、平均减速度、速度标准差：

表 5 文件 3 部分运动学片段对应的主成分得分

运动学片段序号	主成分 1	主成分 2	主成分 3
1	-0.0247	-0.6227	-1.6279
2	1.5686	0.6852	5.6784
3	1.8937	1.2266	2.1228
4	1.3506	0.1765	1.3780
5	0.9860	-0.5901	0.5981
6	1.4049	0.2849	1.4892

表 6 文件 3 主成分的贡献率与累计贡献率表

主成分序号	贡献率	累计贡献率
1	32.20%	32.20%
2	17.91%	50.11%
3	13.14%	63.25%
4	10.24%	73.49%
5	6.21%	79.71%
6	4.39%	84.10%

7	3.46%	87.56%
8	2.98%	90.54%
9	2.56%	93.10%
10	1.92%	95.01%
11	1.77%	96.79%
12	1.23%	98.02%
13	1.03%	99.05%
14	0.95%	100.00%

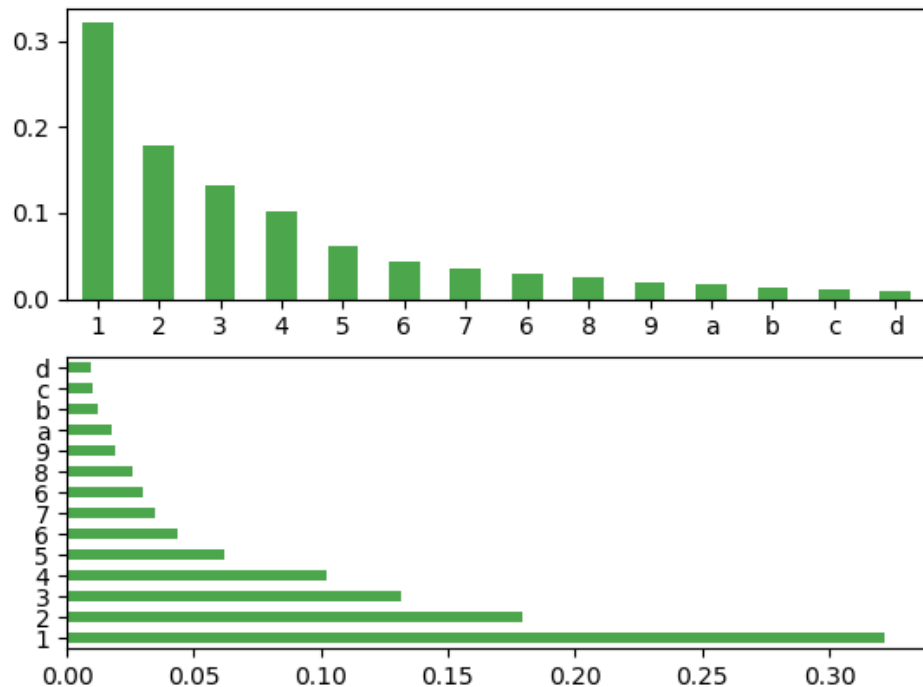


图 19 文件 3 主成分得分贡献比柱状图

#### 6.4. K 均值聚类

本节首先对上节主成分分析所获得的新的运动学片段总体样本进行聚类分析,其目的是使得各组类内的短行程其交通特征相似高,从而分析不同类型道路的运动学片段,最终得到相应类型道路上的行驶工况。

聚类分析指将物理或抽象对象的集合分组为由类似的对象组成的多个类的分析过程,聚类分析是按照相似性将变量或者样品进行分类。相似性的度量方式包括距离样品之间和相似系数变量之间,聚类分析就是基于相似程度的大小,使得关系密切的变量样品聚集到一个小的分类单元,然后逐步扩大,使得关系疏远的聚合到一个大的分类单位,直到所有的样品或变量都聚集完毕。聚类分析的聚类方法主要有系统聚类法,层次聚类法,动态聚类

法，使用系统聚类法需要计算每一个样本之间的距离矩阵，由于本问题的样本数据大于100条，会导致计算量过大，运算时间过长，内存占比过高，计算机无法运行，而动态聚类法并不需要产生这样的距离矩阵，在处理大量的样本时，其运算速度和内存占比较少，算法比系统聚类更优越<sup>[5]</sup>。

选用常用的动态聚类法 K 均值聚类，考虑初始类心的选取具有很大的随机性，在此我们提出了一种创新性的想法。在初始类心选择中，我们先用模拟退火算法求取质心。模拟退火算法(Simulated Annealing, SA)的思想借鉴于固体的退火原理，当固体的温度很高的时候，内能比较大，固体的内部粒子处于快速无序运动，当温度慢慢降低的过程中，固体的内能减小，粒子的慢慢趋于有序，最终，当固体处于常温时，内能达到最小，此时，粒子最为稳定。模拟退火算法便是基于这样的原理设计而成，这样在一定程度上增加了寻找到全局最优解的可能性。

作为衡量样本之间的距离通常有欧式距离，闵氏距离和马氏距离，这里选择欧式距离作为聚类的距离。欧式距离，设样本中的样本数据为  $x_i, x_j$ ，则样本欧式距离表示为

$$d(x_i, x_j) = \left[ (x_i - x_j)^T (x_i - x_j) \right]^{\frac{1}{2}}$$

其算法流程大致为：

1) 确定聚类因子，根据上一步中主成分分析的结果，选择其中的运动学特征值，作为反映道路交通特征的参数作为聚类因子。

2) 确定初始的聚集点，和期望的分类数量，通过样本距离最远的逐渐生成初始的  $k$  个聚点， $x_1^0, x_2^0, \dots, x_k^0$

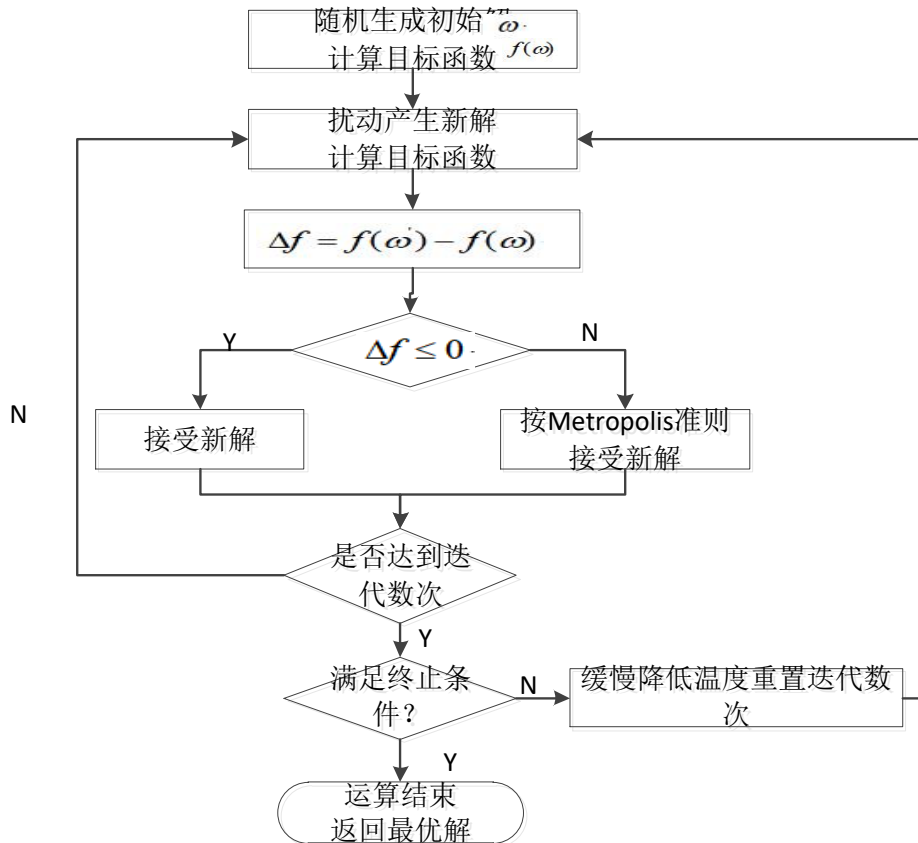


图 20 模拟退火算法流程

3) 设初始点的集合为  $L^0 = \{x_1^0, x_2^0, \dots, x_k^0\}$ ，计算出样本中所有数据和初始点距离，并且按以下方式得到第一次分类

$G_i^0$  表示第 0 次分类距  $x_i^0$  最近的样本， $G_i^0 = \{x: d(x, x_i^0) \leq d(x, x_j^0), j=1, 2, \dots, n\}$ ，则对所有的初始聚点得到初始分类  $G^0 = \{G_1^0, G_2^0, \dots, G_k^0\}$ 。

4) 根据初始分类计算新的据点，设新的据点集合为  $L^1 = \{x_1^1, x_2^1, \dots, x_k^1\}$ ，其中

$$x_i^1 = \frac{1}{n_i^0} \sum_{x_j \in G_i^0} x_j \quad (n_i^0 \text{ 表示 } G_i^0 \text{ 中的样本数量})$$

根据得到的新聚点，再次重复上一步操作得到新的分类，由于以上的计算会使各个类别之间的距离越来越远，所以以此计算下去，得到的分类将趋于稳定，直到之后两步的分类相同时，可以终止操作，输出分类结果，聚类的流程图如下

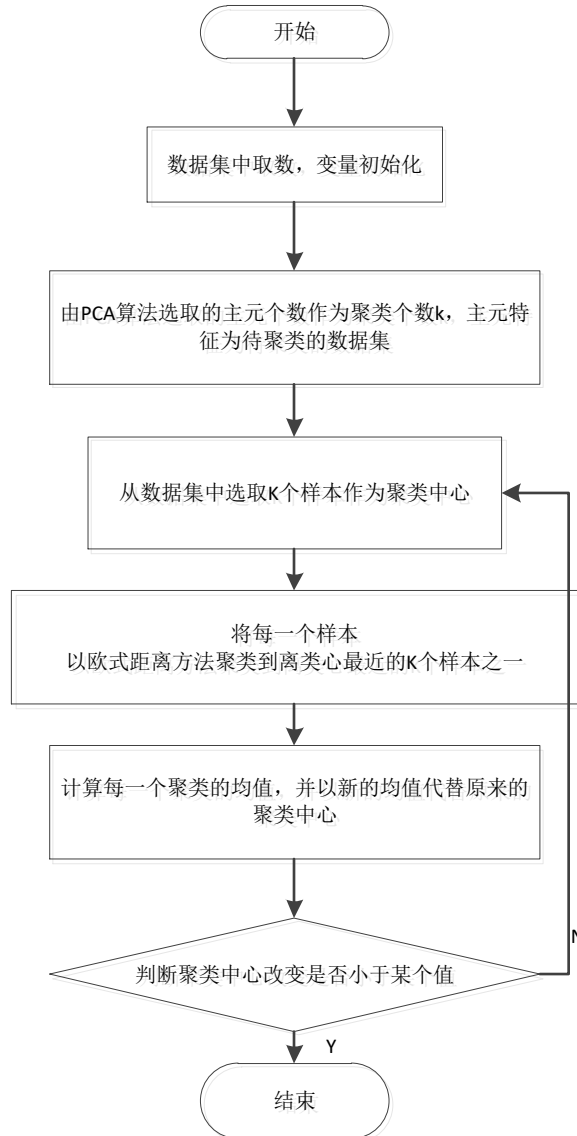
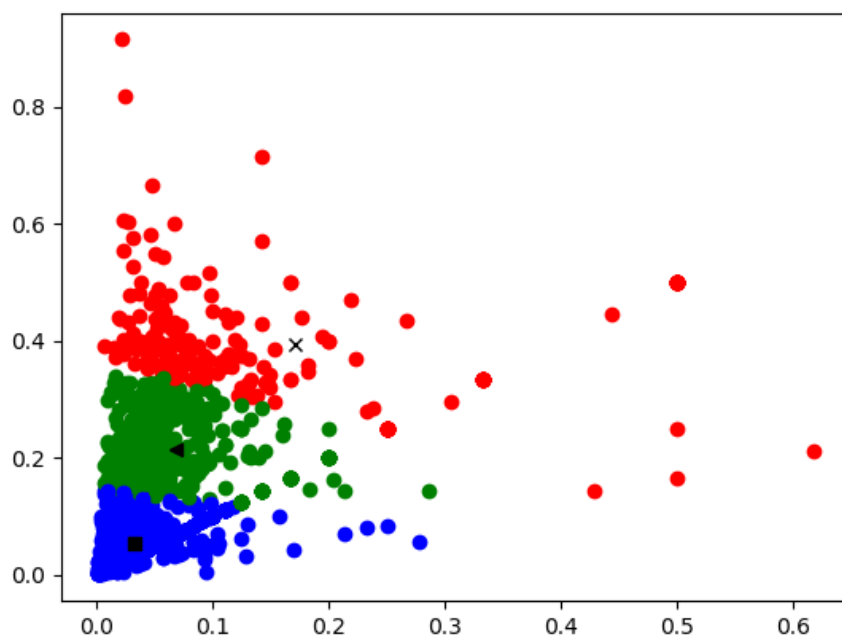


图 21 K 均值聚类分析流程图

以下是对三份文件分别做的聚类点图，我们将所有运动学片段分为了三类，图 22、图 23、图 24 分别是文件一、二、三中数据的聚类结果和生成的聚类次数以及聚类中心点，整理如下：

表 8 聚类中心结果

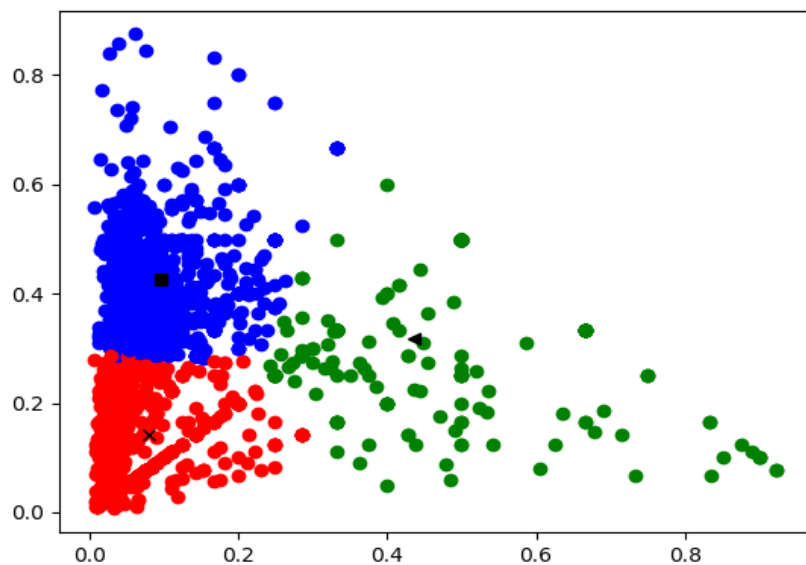
	文件一	文件二	文件三
聚类次数	19	15	14
聚类中心一	(0.17065239, 0.39474891)	(0.08048612, 0.14304311)	(0.08678176, 0.26344413)
聚类中心二	(0.03346877, 0.05330474)	(0.09588839, 0.42593666)	(0.36577933, 0.33940257)
聚类中心三	(0.06889778, 0.21549975)	(0.43523084, 0.31734229)	(0.03555118, 0.05389578)



```
[0.06878932 0.21532018]
**** 第19次迭代 ****
聚类类心为: [[0.17065239 0.39474891]
[0.03346877 0.05330474]
[0.06889778 0.21549975]]
Congratulations, cluster complete!
□
```

图 22 文件一中 K-means 聚类结果



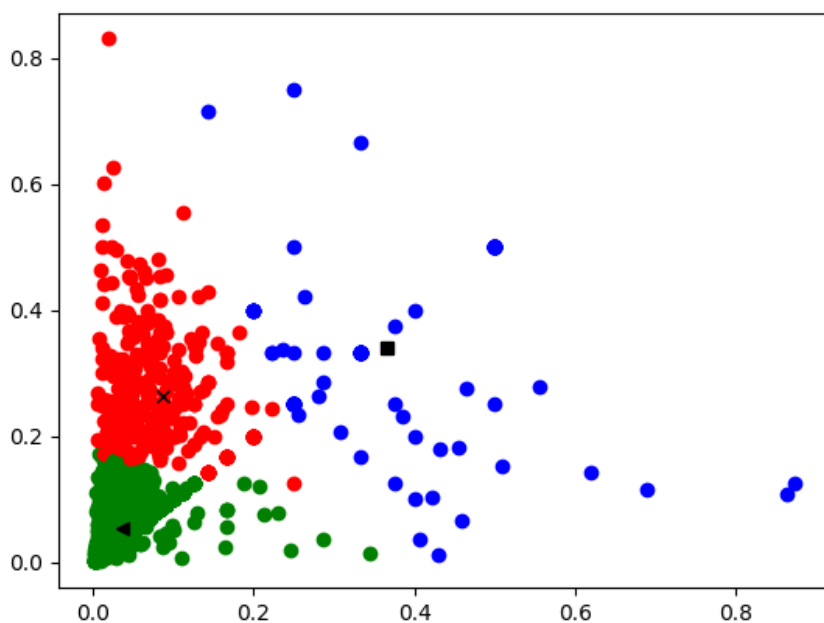


```

聚类类心为: [[0.08028355 0.14190443]
[0.0959439 0.4250575 ]
[0.43523084 0.31734229]]
***** 第11次迭代 *****
聚类类心为: [[0.08048612 0.14304311]
[0.09588839 0.42593666]
[0.43523084 0.31734229]]
Congratulations,cluster complete!
聚类次数: 11
PS D:\model\code\data> 

```

图 23 文件二中 K-means 聚类结果



```

聚类类心为: [[0.086565 0.26319442]
[0.36577933 0.33940257]
[0.03558809 0.05374654]]
**** 第14次迭代 ****
聚类类心为: [[0.08678176 0.26344413]
[0.36577933 0.33940257]
[0.03555118 0.05389578]]
Congratulations,cluster complete!
聚类次数: 14
PS D:\model\code\data>

```

图 24 文件三中 K-means 聚类结果

由以上数据可得，通过模拟退火算法的 K-means 聚类算法能够快速有效的收敛至类心。通过类心的确定，可得每条工况中与类心生成曲线的误差值，选取误差最小为最接近实际工况的运动学状态片段。

## 6.5 构建汽车行驶工况及评价指标的选取

通过样本的聚类结果，能够更加准确的反映在不同交通路况情况下的车辆行驶特征指标，在建立汽车行驶工况模型时，也需要考虑汽车在交通繁忙路段，交通道路分布密集路段，和高速公路等不同路况的特征指标。

用随机挑选的多个短行程组成一个行驶工况，由一个个短行程连续产生，直到链接起来的行驶工况的行驶时间和行程长度达到预期的工况要求。如果这一工况的主要特征值与总体数据的该特征值的误差在以内，并且总的运行时间保持不变，那么构建的这一工况就能代表该地区该类型道路的循环工况。否则重新随机挑选个短行程，组成新的汽车行驶工况图，工况计算误差等，直到满足上述条件为止。

对于确定汽车行驶工况构建中的短行程个数，我们用除以该类短行程的平均运行时间，大致估计出所需的运动学片段数量，并且挑选出具有典型特征的运动学片段组成汽车运行工况，下图 6.5 即为所得的汽车行驶工况图。

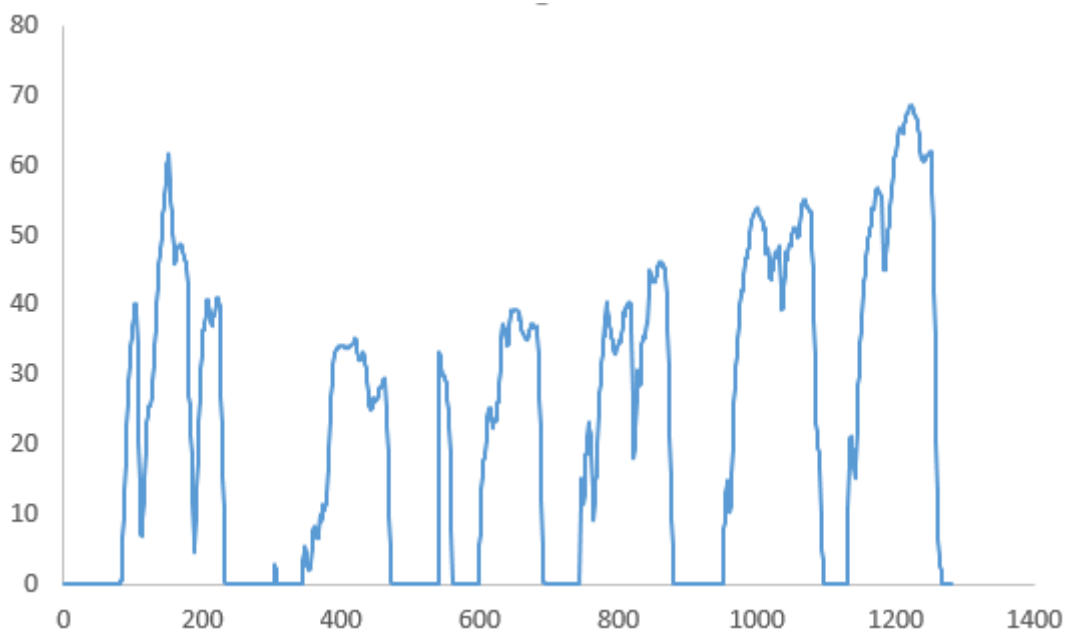


图 25 汽车行驶工况图

计算出的汽车行驶工况中的加速时间比为 31.32%，怠速时间比为 42.5%，平均速度为

22.03km/h，这是由于市区的交通路况较为拥堵，加速时间比，和怠速时间比较高，而平均速度较低与其他路况，与实际情况较为符合。

## 七. 模型评价及改进

### 7.1 模型评价

考虑到数据处理的好坏将影响到后续的数据分析和统计学建模，所以在原始数据缺失的情况下对较长时间的缺失，将和这段时间相关的行驶状态的信息从数据中删除，避免了因为数据缺失造成的样本数据带来的影响，对于较小时间的缺失，我们选择缺失时间前后的几个数据点通过拉格朗日插值的办法不全空缺的时间速度值。因此可以保留更多的运动学片段的样本数据，保证了之后的主成分分析和聚类分析的准确性。

由于统计学中的主成分分析和聚类分析都具有成熟的理论体系，而关键在于特征指标的选择，以及分类过程初始聚点在样本中的分布是否充分均匀。为此我们从时间，速度，加速度，不同运动状态时间占总片段时间占比等所有的特征指标为主成分因子中提取了三个贡献率最高的主成分因子作为指标。

### 7.2 模型不足

本文在数据缺失的处理过程中，对时间数据缺失段较小的位置做了线性插值，但是这样未必能够适合汽车运动的数据变化特征，可能会影响数据分析结果，其次在

## 参考文献

- [1] 马志雄, 李孟良, 张富兴, 朱西产, 主成分分析法在车辆实际行驶工况开发中的应用[U].武汉理工大学学报, 26(4), 32-34, 2004。
- [2] 宋宝珍, 西安市实际道路车辆行驶工况的研究[D].西安: 长安大学, 2009。
- [3] 刘剑平, 朱坤平, 陆元鸿, 应用数理统计.上海: 华东理工大学出版社, 188-192, 2012。
- [4] 野晨晨, 沈阳市乘用车城市道路行驶工况研究[D].吉林: 吉林大学, 2018。
- [5] 马志雄, 朱西产, 李孟良, 乔维高, 张富兴, 动态聚类法在车辆实际行驶工况开发中的应用[J].武汉: 武汉理工大学学报, 27(11), 69-71, 2005。
- [6] 张建伟, 李孟良, 艾国和, 张富兴, 朱西产, 车辆行驶工况与特征的研究[J], 汽车工程学报, 27(2), 220-226, 2005。
- [7] 宋怡帆, 基于聚类和 Python 语言的深圳市城市道路车辆行驶工况构建[D], 西安: 长安大学, 2018。
- [8] 张洁, 城市公共汽车行驶工况构建与研究[D], 西安: 长安大学, 2011。
- [9] 吕红梅, 基于复杂路面的汽车运行工况域的动态调节[D], 重庆: 重庆理工大学, 2011。
- [10] 周虹, 基于自适应粒子群的 k-中心聚类算法研究[D], 长沙: 长沙理工大学, 2012。
- [11] 潘登, 混合动力汽车城市循环工况构建及运行工况多尺度预测[D], 北京: 北京理工大学, 2015。
- [12] 吴其伟, 吕吕林, 锁国涛, 城市公交车发动机循环工况的试验研究与建立[J];内燃机工程, 03, 73-76, 2006。

## 附录

数据预处理文件 `lextra.py`:

```
import xlrd
import xlwt
import numpy as np
import time
file_name = "text2.xlsx"
file = xlrd.open_workbook(file_name)

# 输出 Excel 中表的个数
#print(file.nsheets)

# 读取某张表
sheet = file.sheet_by_name("sheet1")
# 获取表的行数
nrows = sheet.nrows
# 获取表的列数
ncols = sheet.ncols
#print("nrows: %d, ncols: %d" % (nrows, ncols))
str_data = sheet.col_values(0)
str_speed = sheet.col_values(1)
# 获取第一行第一列的数据
ex_data=[0]*(nrows+2)
ex_speed=[0]*(nrows+2)
K=145827#K=185726
year=np.zeros((K,1))
mon=np.zeros((K,1))
day=np.zeros((K,1))
hour=np.zeros((K,1))
mins=np.zeros((K,1))
sec=np.zeros((K,1))
for x in range(1,len(str_speed)-nrows+K):
    ex_speed[x]=str_speed[x]

for x in range(1,len(str_data)-nrows+K):
    ex_data[x]=str_data[x]
    ex_data[x]=ex_data[x][0:19]
    #temp_ex_data[x]=ex_data[x]
    i = time.strptime(ex_data[x], '%Y/%m/%d %H:%M:%S')
    year[x-1]=np.array(i.tm_year)
    mon[x-1]=np.array(i.tm_mon)
    day[x-1]=np.array(i.tm_mday)
    hour[x-1]=np.array(i.tm_hour)
```

```

mins[x-1]=np.array(i.tm_min)
sec[x-1]=np.array(i.tm_sec)
over_data=0
error=0
start_data=0
start_delet_order=[0]*(712)
sdo=0
over_delet_order=[0]*(712)
odo=0
count=1
time_error=0
new_time_error=0
baderror=0
h=0
time_error=np.zeros((712))
for i in range(K):
    if i<K-1:
        if sec[i+1]-sec[i]==1:
            count+=1
        else:
            if sec[i+1]-sec[i]==-59 and mins[i+1]-mins[i]==1:
                count+=1
            else:
                if mins[i+1]-mins[i]==-59 and hour[i+1]-hour[i]==1:
                    count+=1
                else:
                    if hour[i+1]-hour[i]==-23 and day[i+1]-day[i]==1:
                        count+=1
                    else:
                        if day[i+1]-day[i]==-31 or day[i+1]-day[i]==-
30 and mon[i+1]-mon[i]==1:
                            count+=1
                        else:
                            print('连续时间数=',count)
                            baderror+=1
                            print('断点总数=',baderror)
                            #print('断点开始点=',ex_data[i+1])
                            #print('断点结束点=',ex_data[i+2])
                            time_error[h]=(day[i+1]*86400+hour[i+1]*3600+mins[i+1]
]*60+sec[i+1])-(day[i]*86400+hour[i]*3600+mins[i]*60+sec[i])
                            #new_time_error=time_error+new_time_error
                            if time_error[h]>60:
                                #new_time_error=time_error+new_time_error
                                for k in range(i,0,-1):

```

```

        #start_time_delet_order[sd0]=k
        if ex_speed[k]<=10:
            start_data=ex_data[k]
            start_delet_order[sdo]=k
            sdo+=1
            print ('start_data=',start_data)
            #print ('start_delet_order=',start_delet_
order)

            break
        for k in range (i+1,len(ex_data)):
            if ex_speed[k]<=10:
                over_data=ex_data[k]
                over_delet_order[odo]=k
                odo+=1
                print ('over_data=',over_data)
                #print ('over_delet_order=',over_delet_or
der)

                break
            h+=1

        #time_error=0
        #error=over_delet_order-start_delet_order
        #print ('error=',error)
        #print ('temp_ex_data=',temp_ex_data)
        #print ('len(ex_data)=',len(ex_data))

        #count=0
        #print ('start_delet_order=',start_delet_order)
        #print ('len_start_delet_order=',len(start_delet_order))
        #print ('over_delet_order=',over_delet_order)

        #print ('new_time_error=',new_time_error)
        print ('len_over_delet_order=',len(over_delet_order))
        ee=0
        ee_sum=0
        for i in range (len(over_delet_order)):
            ee=over_delet_order[i]-start_delet_order[i]
            ee_sum=ee_sum+ee
            for j in range(len(over_delet_order)):
                over_delet_order[j]=over_delet_order[j]-ee
                start_delet_order[j]=start_delet_order[j]-ee
            for j in range (start_delet_order[i]-1,start_delet_order[i]+1):
                if j==start_delet_order[i]:
                    for k in range(1,ee+1):

```

```

        ex_data=np.delete(ex_data,j+k,0)
        ex_speed=np.delete(ex_speed,j+k,0)
    #print ('ex_data=',len(ex_data))
print ('new_ex_data=',len(ex_data))
print ('ee_sum=',ee_sum)
print ('ex_data=',ex_data)
print ('ex_speed=',ex_speed.shape)

#np.savetxt('data_2.csv', ex_data, fmt='%s')
#np.savetxt('foo.csv',uni,delimiter=',', fmt = '%s')
#np.savetxt('speed_2.csv', ex_speed, delimiter = ',')

```

数据预处理文件 2 extra\_model.py:

```

import xlrd
import xlwt
import numpy as np
模块
import time
def init_data(file_name,K):
    #file_name = "text1.xlsx"
    file = xlrd.open_workbook(file_name)

    # 输出 Excel 中表的个数
    #print(file.nsheets)

    # 读取某张表
    sheet = file.sheet_by_name("sheet1")
    # 获取表的行数
    nrows = sheet.nrows
    # 获取表的列数
    ncols = sheet.ncols
    #print("nrows: %d, ncols: %d" % (nrows, ncols))
    str_data = sheet.col_values(0)
    str_speed = sheet.col_values(1)
    # 获取第一行第一列的数据
    ex_data=[0]*(nrows+2)
    ex_speed=[0]*(nrows+2)
    #K=185726#K=185726
    year=np.zeros((K,1))
    mon=np.zeros((K,1))
    day=np.zeros((K,1))
    hour=np.zeros((K,1))
    mins=np.zeros((K,1))
    sec=np.zeros((K,1))
    for x in range(1,len(str_speed)-nrows+K):

```

#导入 xlrd



```

ex_speed[x]=str_speed[x]

for x in range(1,len(str_data)-nrows+K):
    ex_data[x]=str_data[x]
    ex_data[x]=ex_data[x][0:19]
    #temp_ex_data[x]=ex_data[x]
    i = time.strptime(ex_data[x], '%Y/%m/%d %H:%M:%S')
    year[x-1]=np.array(i.tm_year)
    mon[x-1]=np.array(i.tm_mon)
    day[x-1]=np.array(i.tm_mday)
    hour[x-1]=np.array(i.tm_hour)
    mins[x-1]=np.array(i.tm_min)
    sec[x-1]=np.array(i.tm_sec)
return ex_speed,ex_data,year,mon,day,hour,mins,sec
def ex_infor(ex_speed,ex_data,year,mon,day,hour,mins,sec,K):
    over_data=0
    start_data=0
    start_delet_order=[0]*(3000)
    sdo=0
    over_delet_order=[0]*(3000)
    odo=0
    count=1
    baderror=0
    for i in range(K):
        if i<K-1:
            if sec[i+1]-sec[i]==1:
                count+=1
            else:
                if sec[i+1]-sec[i]==-59 and mins[i+1]-mins[i]==1:
                    count+=1
                else:
                    if mins[i+1]-mins[i]==-59 and hour[i+1]-hour[i]==1:
                        count+=1
                    else:
                        if hour[i+1]-hour[i]==-23 and day[i+1]-day[i]==1:
                            count+=1
                        else:
                            if day[i+1]-day[i]==-31 or day[i+1]-day[i]==-
30 and mon[i+1]-mon[i]==1:
                                count+=1
                            else:
                                print('连续时间数=',count)
                                baderror+=1
                                print('断点总数=',baderror)

```

```

        #print('断点开始点=',ex_data[i+1])
        #print('断点结束点=',ex_data[i+2])
        if day[i+1]-day[i]<=1 and mins[i+1]-mins[i]!=1:
            for k in range(i,0,-1):
                if ex_speed[k]>=10:
                    start_data=ex_data[k]
                    start_delet_order[sdo]=k
                    sdo+=1
                    print ('start_data=',start_data)
                    #print ('start_delet_order=',start_de
let_order)

                break
            for k in range (i+1,len(ex_data)):
                if ex_speed[k]>=10:
                    over_data=ex_data[k]
                    over_delet_order[odo]=k
                    odo+=1
                    print ('over_data=',over_data)
                    #print ('over_delet_order=',over_dele
t_order)

                break
            return start_delet_order,over_delet_order
def infor_delet(start_delet_order,over_delet_order,ex_data,ex_speed):
    ee=0
    ee_sum=0
    for i in range (len(over_delet_order)):
        ee=over_delet_order[i]-start_delet_order[i]
        ee_sum=ee_sum+ee
        for j in range(len(over_delet_order)):
            over_delet_order[j]=over_delet_order[j]-ee
            start_delet_order[j]=start_delet_order[j]-ee
        for j in range (start_delet_order[i]-1,start_delet_order[i]+1):
            if j==start_delet_order[i]:
                for k in range(1,ee+1):
                    ex_data=np.delete(ex_data,j+k,0)
                    ex_speed=np.delete(ex_speed,j+k,0)
            #print ('ex_data=',len(ex_data))
        print ('new_ex_data=',len(ex_data))
        print ('ee_sum=',ee_sum)
        print ('ex_data=',ex_data)
        print ('ex_speed=',ex_speed.shape)
        return ex_data,ex_speed
file_name = "text3.xlsx"
K=164915#K=185726#K=145826

```

```

ex_speed,ex_data,year,mon,day,hour,mins,sec=init_data(file_name,K)
start_delet_order,over_delet_order=ex_infor(ex_speed,ex_data,year,mon,day,hour,mins,sec,K)
#del_data,del_speed=infor_delet(start_delet_order,over_delet_order,ex_data,ex_speed)
#np.savetxt('data.csv', del_data, fmt='%s')
#np.savetxt('speed.csv', del_speed, delimiter = ',')

```

运动学片段提取 newstep.py:

```

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

```

#要求的数据

```
T=[0]*2000 #运行时间
```

#data 为读入的数据

#获取数据

#加表头

```

data = pd.read_csv('D:/model/code/data/xxx2.csv')
data=np.array(data)
vLength=len(data[:,0])
bingo=np.zeros((2000,10000))
everyT=[0]*vLength

```

flag=1 #一个运动学片段里有多少个速度

index=temp=0 #生成数组序列标志

shortTime=1

shortTim=[] #标志位判断短行程时间

shortDistance=data[:,0][0]

shortDis=[] #标志位判断短行程距离

for i in range(vLength):

if(i>0):

#加速度从当前时刻算

accelerate=(data[:,0][i]-data[:,0][i-1])/3.6 #第 i+1 点的加速度

if(accelerate!=0 and data[:,0][i]==0 and data[:,0][i+1]==0):

#

停止点的判断条件,三个与操作

# print('出现',accelerate,i)

#停止点当前时刻 a!=0,下一时刻 v=0,表明下一时刻运动就结束了

#数据读入第 index 行

# print(flag+1,temp)

for j in range(flag-temp+2):

#运动学片段

bingo[index,:][j]=data[temp+j]

```

        shortTim.append(shortTime+1)
        shortDis.append(shortDistance)
        shortTime=0
        shortDistance=0
        #运行时间
        everyT[index]=i-temp
        #第几个片段
        index=index+1
        #起始点
        temp=flag+1
        flag=flag+1
    else:
        shortTime=shortTime+1
        shortDistance=shortDistance+data[:,0][i]
        flag=flag+1
# print(index)
#短行程判断
short=0
for i in range(index):
    if(shortTim[i]<20 or shortDis[i]<10):
        short=short+1
        bingo=np.delete(bingo,i,axis=0)
index=index-short

print('文件 2 生成运动片段个数: ',index)


#指标计算
#=====
# 指标个数为 index
#=====
#要求的数据
Twait=[0]*index    #怠速
Tacc=[0]*index    #加速
Tdec=[0]*index    #减速
Tave=[0]*index    #匀速
distance=[0]*index#路程
everyTemp=0
T=everyT[0:index]    #时间段
Speed=np.zeros((index,vLength))
sumaccSpeed=[0]*index    #平均加速度
sumdecSpeed=[0]*index    #平均减速度
accStd=[0]*index    #加速度标准差
speedStd=[0]*index    #速度标准差

```

```

aveSpeed=[0]*index #平均速度
TwaitPercent=[0]*index #怠速比例
TaccPercent=[0]*index #加速比例
TdecPercent=[0]*index #减速比例
TavePercent=[0]*index #匀速比例

for i in range(index):
    distance[i]=sum(bingo[i])
    timewait=1
    timeacc=1
    timeave=1
    timedec=1
    sumacc=1
    sumdec=1
    for j in range(T[i]):
        acc=(bingo[i][j]-bingo[i][j-1])/3.6
        Speed[i][j]=acc
        if (acc==0 and bingo[i][j]==0):
            timewait=timewait+1
        if (acc>0 and bingo[i][j]!=0):
            timeacc=timeacc+1
            sumacc=sumacc+acc
        if (acc<0 and bingo[i][j]!=0):
            timedec=timedec+1
            sumdec=sumdec+acc
        if (acc==0 and bingo[i][j]!=0):
            timeave=timeave+1
    Twait[i]=timewait
    Tacc[i]=timeacc
    Tave[i]=timeave
    Tdec[i]=timedec
    sumaccSpeed[i]=sumacc/timeacc
    sumdecSpeed[i]=sumdec/timedec

maxSpeed=[0]*index
maxaccSpeed=[0]*index
mindecSpeed=[0]*index
for i in range(index):
    maxSpeed[i]=max(bingo[i,:])
    maxaccSpeed[i]=max(Speed[i,:])
    mindecSpeed[i]=min(Speed[i,:])
    accStd[i]=np.std(Speed[i,:][slice(T[i])],ddof=1)
    speedStd[i]=np.std(bingo[i,:][slice(T[i])],ddof=1)
    aveSpeed[i]=distance[i]/T[i]

```

```

    TwaitPercent[i]=Twait[i]/T[i]
    TaccPercent[i]=Tacc[i]/T[i]
    TdecPercent[i]=Tdec[i]/T[i]
    TavePercent[i]=Tave[i]/T[i]

#矩阵的拼接
feature=np.vstack((TavePercent,TdecPercent,TaccPercent,T,distance,maxSpeed,aveSpeed,maxaccSpeed,mindecSpeed,sumaccSpeed,sumdecSpeed,speedStd,accStd,TwaitPercent))
feature2pca=np.transpose(feature)

# print('运行时间 T: ',T,'路程',distance,'最大速度: ',maxSpeed,'平均速度: ',aveSpeed,
#       '最大加速度: ',maxaccSpeed,'最小减速度: ',mindecSpeed,'平均加速度 ',sumaccSpeed,'平均减速度',sumdecSpeed,
#       '速度标准差: ',speedStd,'加速度标准差: ',accStd,
#       '总速时间比例: ',TwaitPercent,'加速时间比例: ',TaccPercent,'减速时间比例: ',TavePercent,'匀速时间比例: ',TdecPercent)
# print('特征矩阵: ',feature)

# 绘图
plt.plot(range(0,10000),bingo[100])
plt.show()
np.savetxt('feature2pca.csv', feature2pca, delimiter = ',')
#匀速时间比例，减速时间比例，总速时间比例
PCA 主元分析 PCA.py:
#-*-coding:utf-8-*-
import numpy as np
import pandas as pd
from scipy import linalg,stats
import matplotlib.pyplot as plt
import seaborn as sns
#sns.set()
from matplotlib.font_manager import FontProperties
from pylab import *
#fname = "/home/w/soft/anaconda2/envs/lstm/lib/python3.6/site-packages/matplotlib/mpl-data/fonts/ttf/SimHei.ttf"
#myfont = FontProperties(fname=fname)

#data = pd.read_csv('file1.csv')

def pca():
    Xtrain= pd.read_csv('feature2pca.csv')
    # Xtest = xx_test
    #print('Xtrain=',Xtrain)

```

```

#print('Xtrain.shape=',Xtrain.shape)
# 标准化处理
X_mean = np.mean(Xtrain, axis=0) #按求 Xtrain 平均值
X_std = np.std(Xtrain, axis=0) #求标准差
#print('X_std=',X_std)
X_row,X_col = Xtrain.shape #求 Xtrain 行、列数
Xtrain = (Xtrain-np.tile(X_mean,(X_row,1)))/np.tile(X_std,(X_row,1))##将平均
值扩展为与 xtrain 相同形状的，横向拉伸
# print('Xtrain=',Xtrain)
# 求协方差矩阵
sigmaXtrain = np.cov(Xtrain, rowvar=False)#rowvar=0，说明传入的数据一列代表一个
变量，一行代表一个样本
#对协方差矩阵进行特征分解，lamda 为特征值构成的，T 的列为单位特征向量，且与 lamda 中
的特征值一一对应：
lamda,T = linalg.eig(sigmaXtrain)

#取对角元素(结果为一列向量)，即 lamda 值，并上下反转使其从大到小排列，主元个数初值为
1，若累计贡献率小于 90%则增加主元个数
# D = flipud(diag(lamda))
T = T[:,lamda.argsort()] #将 T 的列按照 lamda 升序排列，从小到大，返回的是索引值

print ('lamda.argsort()=' ,lamda.argsort())
lamda.sort() #将 lamda 按升序排列
D = -np.sort(-np.real(lamda)) #提取实数部分，按降序排列
num_pc = 1
if (sum(D[0:num_pc])/sum(D))<0.85:
    num_pc += 1
num_pc+=1

#取与 lamda 相对应的特征向量
P = T[:,X_col-num_pc:X_col]
#print ('主元的特征向量',P)
# print ('主元的特征向量 shape',P)
print('主元个数=',num_pc)
gong=D[0:X_col+1]/sum(D)
fig,axes=plt.subplots(2,1)
data=pd.Series(gong,index=list('1234567689abcd'))
data.plot.barh(ax=axes[1],color='blue',alpha=0.7)
data.plot.bar(ax=axes[0],color='blue',alpha=0.7,rot=0)
fig.savefig('p2.png')
# print('第一个主元是', lamda.argsort(1))
Z=np.dot(Xtrain,P)
print('贡献率: ',gong)
print('得分: ',Z[0:6])

```

```
pass
```

```
if __name__=='__main__':  
    # data()  
    pca()
```

### K-means 聚类文件 Kmeans.py:

```
import numpy as np  
import matplotlib.pyplot as plt  
  
# 加载数据  
def loadDataSet(fileName):  
    data = np.loadtxt(fileName,delimiter='\t')  
    return data  
  
# 欧氏距离计算  
def distEclud(x,y):  
    return np.sqrt(np.sum((x-y)**2)) # 计算欧氏距离  
  
# 为给定数据集构建一个包含 K 个随机质心的集合  
def randCent(dataSet,k):  
    m,n = dataSet.shape  
    centroids = np.zeros((k,n))  
    for i in range(k):  
        index = int(np.random.uniform(0,m))  
        centroids[i,:] = dataSet[index,:]  
    return centroids  
  
# k 均值聚类  
def KMeans(dataSet,k):  
    temp=0  
    m = np.shape(dataSet)[0] #行的数目  
    # 第一列存样本属于哪一簇  
    # 第二列存样本的到簇的中心点的误差  
    clusterAssment = np.mat(np.zeros((m,2)))  
    clusterChange = True  
  
    # 第 1 步 初始化 centroids  
    centroids = randCent(dataSet,k)  
    while clusterChange:  
        temp=temp+1
```



```

print('***** 第%d 次迭代 *****' % temp)
print('聚类类心为: ',centroids)
clusterChange = False

# 遍历所有的样本（行数）
for i in range(m):
    minDist = 100000.0
    minIndex = -1

    # 遍历所有的质心
    #第 2 步 找出最近的质心
    for j in range(k):
        # 计算该样本到质心的欧式距离
        distance = distEclud(centroids[j,:],dataSet[i,:])
        if distance < minDist:
            minDist = distance
            minIndex = j
    # 第 3 步：更新每一行样本所属的簇
    if clusterAssment[i,0] != minIndex:
        clusterChange = True
        clusterAssment[i,:] = minIndex,minDist**2
#第 4 步：更新质心
for j in range(k):
    pointsInCluster = dataSet[np.nonzero(clusterAssment[:,0].A == j)[0]]
# 获取簇类所有的点
    centroids[j,:] = np.mean(pointsInCluster,axis=0) # 对矩阵的行求均值

print("Congratulations,cluster complete!")
return centroids,clusterAssment,temp

def showCluster(dataSet,k,centroids,clusterAssment):
    m,n = dataSet.shape
    if n != 2:
        print("数据不是二维的")
        return 1

    mark = ['or', 'ob', 'og', 'ok', '^r', '+r', 'sr', 'dr', '<r', 'pr']
    if k > len(mark):
        print("k 值太大了")
        return 1

    # 绘制所有的样本
    for i in range(m):
        markIndex = int(clusterAssment[i,0])

```

```

plt.plot(dataSet[i,0],dataSet[i,1],mark[markIndex])

mark = ['xk', 'sk', '<k', 'Dk', '^b', '+b', 'sb', 'db', '<b', 'pb']
# 绘制质心
for i in range(k):
    plt.plot(centroids[i,0],centroids[i,1],mark[i])

plt.show()
data = np.loadtxt(open("D:/model/code/data/feature1pca.csv"),delimiter=",")
dataSet=data[:,0:2]
print(dataSet)
k = 3
centroids,clusterAssment,temp = KMeans(dataSet,k)

showCluster(dataSet,k,centroids,clusterAssment)
print('聚类次数: ',temp)

```

#### 运动学片段匹配 match.py:

```

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#要求的数据
T=[0]*2000 #运行时间
#data 为读入的数据
#获取数据
#加表头
data = pd.read_csv('D:/model/code/data/data2.csv')
data=np.array(data)
def zuiyou(bingo,centroids):
    min1=0

    best1=np.zeros((len(bingo),1))
    for i in range (len(bingo)):
        a=centroids[0][1]-hanshu1(bingo[i])

        b=centroids[1][1]-hanshu2(bingo[i])

        c=centroids[2][1]-hanshu3(bingo[i])

        d=(0.5*a+0.3*b+0.2*c)
        if (d<min1):
            min1=d
            e=bingo[i]

```

```
        print ('最优解=',bingo[i])

    return e
pbest=zuiyou(bingo,centroids)
plt.plot(range(0,1300),pbest)
plt.show()
```