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	南京大学
参赛队号	**19102840092**
队员姓名	**1.徐江成**
	2.于海强
	3.高露

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　　基于K均值聚类的汽车行驶工况构建

摘　　要：

汽车行驶工况可以用于确定车辆污染物排放量及燃油消耗量，是汽车工业一项核心技术。目前，我国乘用车行驶工况采用的是欧洲的NEDC，但由于我国的交通状况、行驶状况与道路状况等因素都与欧盟国家有很大不同，因此NEDC工况与我国城市汽车的实际行驶数据偏差较大。另一方面，我国地域辽阔，各个城市的汽车行驶工况也不尽相同。因此，基于各个城市自身的汽车行驶数据制定反映我国实际道路行驶状况的汽车行驶工况，显得越来越重要。

本文根据三组某城市汽车行驶数据，构建了一条时长1201s的汽车行驶工况曲线，较好地反映了所使用的数据源的汽车运动特征。本文首先以多种方式对脏数据进行处理，包括清除和修正等，从三个文件中分别得到了131174，108077，106737条合理的数据。进而将原始数据分别划分为1181，725，649个运动学片段，并计算每个片段的特征参数，得到了一组特征参数矩阵。之后本文对该矩阵使用Min-Max方法进行了归一化处理，并进行了主成分分析，保留了三个主成分，使用K均值聚类算法进行聚类分析，得到了四种类型的运动学片段，每种类型分别有 715、616、482、742 条片段。然后根据各类片段在整个数据集中所占的时间比例，合成了一条时长1201s的行驶工况。最后本文对该结果进行了验证和分析，认为其能较好地体现出该汽车的运动特征。

问题一要求对原始数据进行清洗。对于加速度异常，本文采用了线性插值法进行修正。对于时间不连续的不良数据，当其出现异常持续时间小于30s时，本文采用线性插值法进行了修正，当其出现异常时间超过30s时，本文则将其所属的整个运动学片段内的所有数据删除。对于间歇性怠速或低速行驶超过60s的数据，本文将其速度重置为0km/h，认为这是怠速情况。对于长时间怠速的数据，本文只保留 180s 的怠速数据。最终数据文件 1 剔除的异常记录数为54551条，总记录数为131174条。数据文件2剔除的异常记录数为37748条，总记录数为108077条。数据文件3剔除的异常记录数为58177条，总记录数为106737条。

问题二要求切分运动学片段。本文将自一个怠速状态开始至下一个怠速状态开始的片段视为一个运动学片段，据此进行切分，最终数据文件1得到了1181个运动学片段，数据文件2得到了725个运动学片段，数据文件3得到了649个运动学片段。

问题三要求构建一条时长1200~1300s的，能体现参与数据采集汽车行驶特征的汽车行驶工况曲线。本文对每一个片段，计算了包括平均速度、平均行驶速度、平均加速度等 9 个特征参数。然后使用Min-Max归一化方法对每个参数进行标准化处理，之后为避免信息重叠，进行了主成分分析，保留了三个主成分进行K均值聚类，在聚类前本文使用“手肘

法”确定 K 值为 4，最终得到了四组运动学片段，每组依次有 715、616、482、742 条样本。接着本文根据各类片段在整个数据集中所占的时间比例，合成了一条时长 1201s 的行驶工况。最后使用卡方校验等方式对所得结果进行了校验，认为其能较好地反映出该汽车的形式特征。

关键字：运动学片段，行驶工况，汽车行驶特征，K-Means 聚类

1 问题背景与问题重述

1.1 问题背景

汽车行驶工况又称为车辆测试循环，是描述典型车辆行驶的速度-时间曲线。它可以用于确定车辆污染物排放量及燃油消耗量，体现汽车在道路行驶中的燃料经济性、排放等运动学特征，是汽车工业一项核心技术。目前，我国乘用车行驶工况采用的是欧洲的 NEDC，但由于我国的交通状况、行驶状况与道路状况等因素都与欧盟国家有很大不同，因此 NEDC 工况与我国城市汽车的实际行驶数据偏差较大。另一方面，我国地域辽阔，不同城市之间的交通状况、发展程度与气候条件等都存在着差异，这也导致各个城市的汽车行驶工况也不尽相同[1]。因此，基于各个城市自身的汽车行驶数据制定反映我国实际道路行驶状况的的汽车行驶工况，显得越来越重要。

1.2 问题重述

在上述背景下，题目给出了某城市同一辆测试汽车在不同时间段采集的实际行驶数据，要求利用该数据解决以下几个问题：

问题一：由于数据采集过程中存在着不可避免的实际问题，原始数据中会包含一些不良值。要求通过合理的方法对异常数据进行预处理，并给出处理后的数据记录数。

问题二：我们将车辆从一个怠速开始到下一个怠速开始之间的运动定义为运动学片段。题目要求设计合理的方法，将处理后的数据划分为多个运动学片段，并给出每个数据文件得到的运动学片段数。

问题三：基于上面处理好的数据，采用科学有效的方法构建一条能体现采集数据源行驶特征的汽车行驶工况曲线。同时使用合理的评估体系对行驶工况曲线进行评估，说明其合理性。

2 基本假设与符号说明

2.1 基本假设

1) 假设普通轿车在一般情况下最大的加速度为 4m/s^2，紧急刹车时的最大减速度为 8m/s^2。
2) 假设当汽车车速小于 10km/h 时，就认为是怠速状态。当怠速状态时间超过 180 秒就被认为是异常情况，怠速时间最长可按 180 秒处理。
3) 我们定义汽车加速度大于 0.1m/s^2 的连续过程为加速阶段，汽车加速度小于-0.1m/s^2 的连续过程为减速阶段。
4) 假设采样频率为 1Hz。

2.2 符号说明

表 1 符号说明

符号	说明
V_m	平均速度。即一段时间周期内，汽车速度的算术平均值
V_{mr}	平均行驶速度。即汽车在行驶状态下汽车速度的算术平均值，即不包含汽车怠速状态。
V_{sd}	速度标准差。即一段时间周期内，汽车速度的标准差，包括怠速状态。
T_i	怠速时间比。即一段时间周期内，怠速状态的累计时间长度占该时间周期总时间长度的百分比。
T_a	加速时间比。即一段时间周期内，处在加速状态的累计时间长度占该时间周期总时间长度的百分比。
T_d	减速时间比。即一段时间周期内，处在减速状态的累计时间长度占该时间周期总时间长度的百分比。
A_m	平均加速度。即汽车在加速状态下各单位时间（秒）加速度的算术平均值。
A_d	平均减速度。即汽车在减速状态下各单位时间（秒）减速度的算术平均值。
A_{sd}	加速度标准差。即一段时间周期内，处在加速状态的汽车加速度的标准差。

3 问题一的分析与建模

题目给出了某城市同一辆测试汽车在不同时间段采集的实际行驶数据，覆盖了工作日与节假日、高峰期与非高峰期等多时段的数据，保证了测试数据的可靠性。但由于多种主客观因素影响，原始测试数据中可能包含多种不良数据。问题一就是要求通过多种合理的处理方法对这些异常数据进行识别与预处理。

原始的测试数据包含 3 个数据文件，其中数据文件 1 的原始数据记录数为 **185725** 条，数据文件 2 的原始数据记录数为 **145825** 条，数据文件 3 的原始数据记录数为 **164914** 条。

3.1 加减速度异常的数据处理

从上文的基本假设中我们可知，在一般情况下普通轿车从 0 加速至 100km/h 的时间应大于 7 秒，即最大加速度约为 $4m/s^2$，而轿车紧急刹车的最大减速度约为 $8m/s^2$。在实际道路行驶过程中可能会出现车辆急加速与急刹车的情况，这种情况下往往会导致车辆加速度过大。而超出最大加减速度阈值的不良数据显然会对我们构建行驶工况曲线造成影响，因此需要对这些不良数据进行处理。

我们通过对题目给出的原始采集数据的各条记录进行加速度计算后，发现数据集中也存在这种加减速度异常的数据值。数据文件 1 的异常数据的记录数为 28 条，数据文件 2 的异常数据的记录数为 401 条，数据文件 3 中无加减速度异常的记录。图 1 展示了数据文

件 2 中的一段异常记录，从图中我们可以明显看出测量到的 GPS 速度变化异常，几乎每秒都在进行急加速或急减速，加减速度均超出了正常范围。使用这样的异常数据来构建行驶工况曲线无疑会出现较大的偏差。

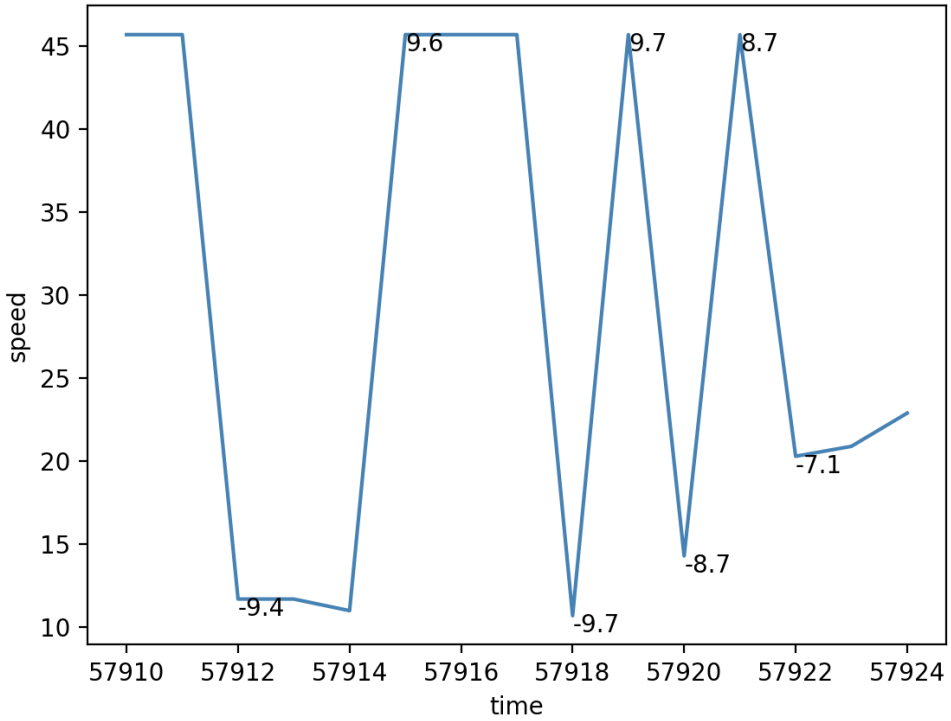


图 1 加减速度异常的速度曲线图

对于这类加减速度异常的原始数据，我们的处理方法是对异常数据记录进行剔除，并使用**线性插值法**对原本的异常数据点进行修正，从而消除离群点数据对于构建行驶工况曲线带来的不利影响。图 2 展示了修正后的数据记录，可以看出通过线性插值法修正后加减速度均已在正常范围内，数据异常问题得以解决。

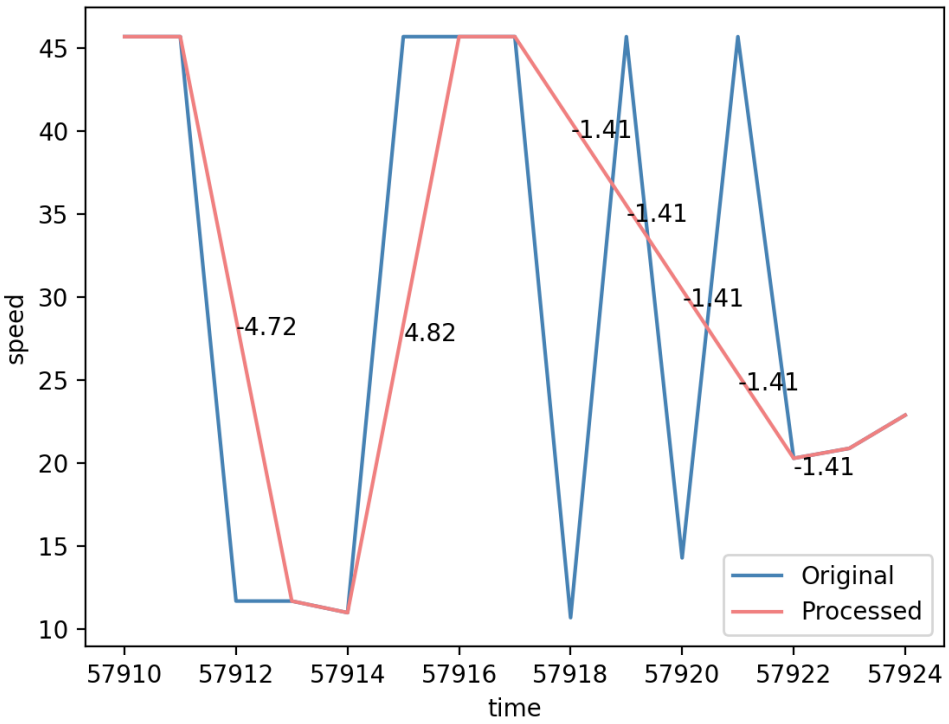


图 2 修正后的速度曲线图

3.2 时间不连续的缺失数据处理

汽车行驶数据通过车载的终端采集设备直接采集，采集过程中可能会出现采集设备异常等问题，导致数据丢失。此外，现实路况也可能给数据采集带来一定的影响，例如在汽车过隧道时，会导致 GPS 信号丢失，无法进行正常的数据采集。这些问题造成了提供的数据中时间不连续，部分数据丢失的问题。

在对缺失数据段进行观察后，我们发现不同的数据段数据缺失的严重程度不同。图 3 展示了两种不同的缺失程度，图 3 a 中仅缺失了 3 秒的数据，而图 3 b 中缺失了超过 60 秒的数据。结合原始数据质量与现实生活的考量，我们设置允许最大不连续时间段的阈值为 30 秒。连续缺失数据未超过阈值的数据段（如图 x a 仅缺失 3 秒的数据段）我们通过**线性插值法**对缺失数据进行填充修正处理，而超过阈值的数据我们则进行删除。在构建行驶工况曲线的过程中，我们选择基于运动学片段进行构建。考虑到缺失数据过多会对整个运动学片段带来影响，因此对于那些超过阈值需要被删除的异常数据，我们选择删除该段缺失数据所在的**整个运动学片段**的数据记录。

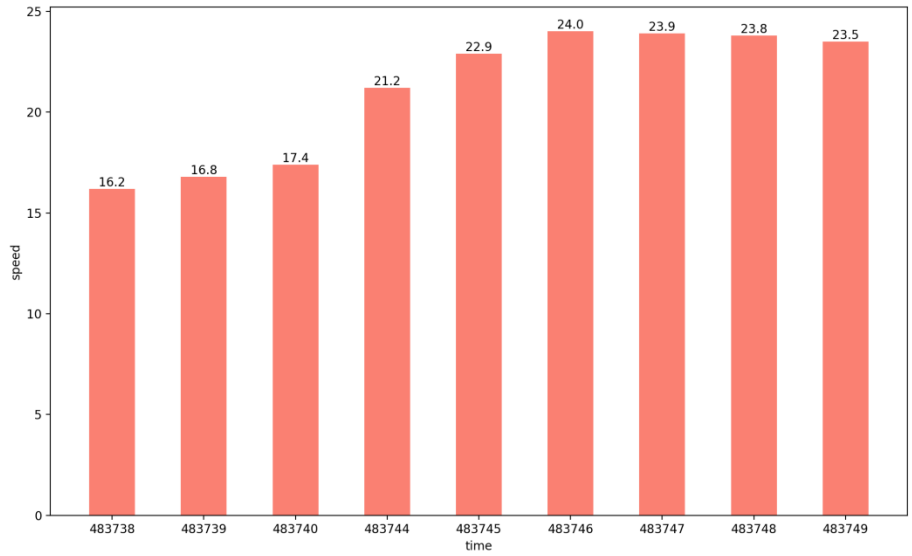


图 3 a 缺失较少的异常数据段

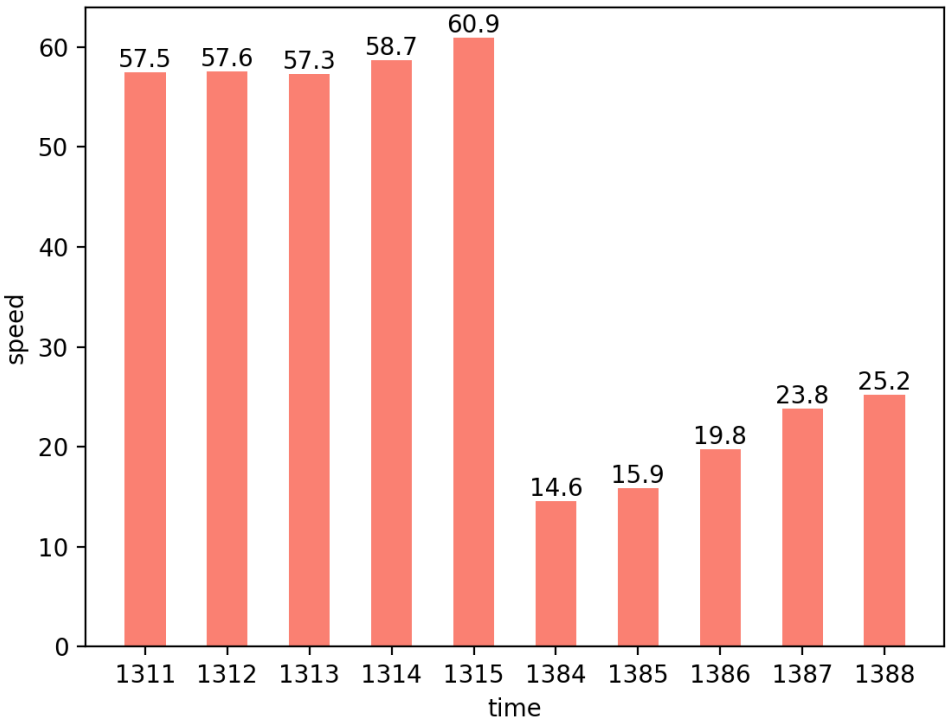


图 3 b 缺失较多的异常数据段

经过对缺失数据进行处理后，数据文件 1 剩余总记录数为 **136810** 条。数据文件 2 剩余总记录数为 **110395** 条。数据文件 3 剩余总记录数为 **108808** 条。

3.3 怠速异常的数据处理

由于汽车行驶工况受交通状况、道路状况、行驶状况等多种主客观因素影响，在数据

采集的过程中往往会出现多种类型的行驶数据。我们的原始数据覆盖了工作日与节假日、高峰期与非高峰期等多时段的数据，这其中也包含了几种怠速异常的不良数据。

一种情况如图 4 所示，数据中存在着一段长时间内行驶速度均为 0km/h 的怠速异常情况，在现实中长期停车熄火等人，或者停车熄火了但仍在采集数据等都会造成这种异常情况的发生。从上文的基本假设中我们可知，怠速时间超过 180 秒被认为是异常情况，怠速时间最长可按 180 秒进行处理。因此对于这类异常我们的处理方法是识别出整段怠速片段，删除多余的异常数据，只保留 180 秒（即最长怠速时间）内的数据。

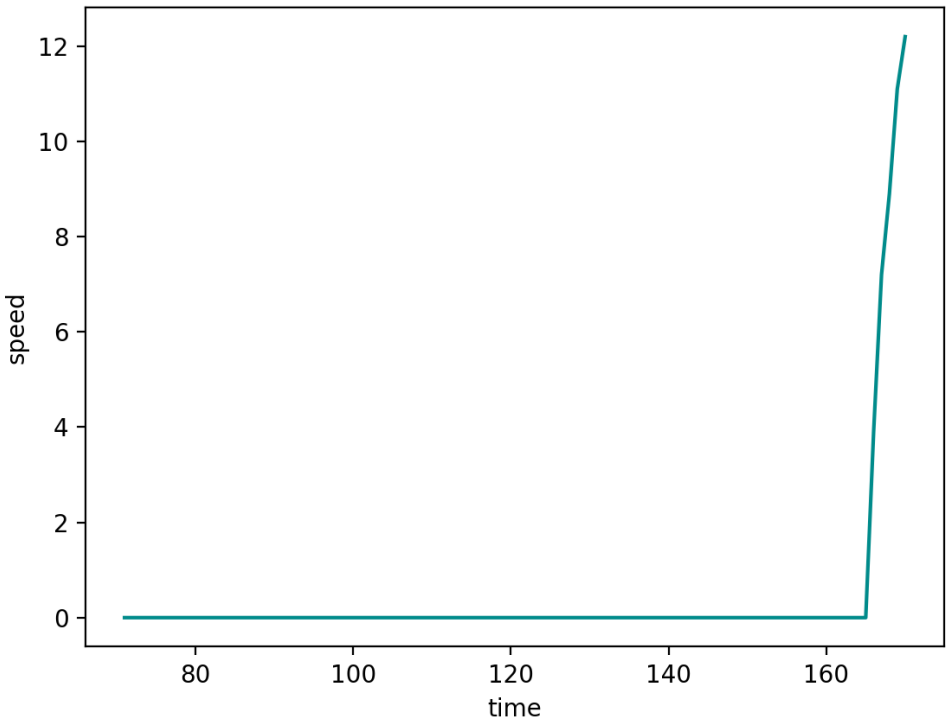


图 4 长期停车的异常数据

另外一种异常情况与实际更加相关。在现实生活中，高峰期或节假日时期的路况一般较为拥堵，极易出现长时间堵车的情况。这时汽车一般处于断断续续行驶且只能低速行驶的情况。这对应于数据中存在着一段长时间行驶速度不为 0 但均小于 10km/h 的异常情况，如图 5 所示。对于这类不良数据，我们的处理方法是将这类数据按照怠速情况进行处理。若连续低速的数据记录超过 60 秒（说明是**长时间**堵车）则将其速度记录全部置为 0km/h，认为这是怠速状况，并按照上述怠速异常的方法进行处理。若连续低速的数据记录未超过 60 秒，则认为是正常情况，保留原始的速度记录。

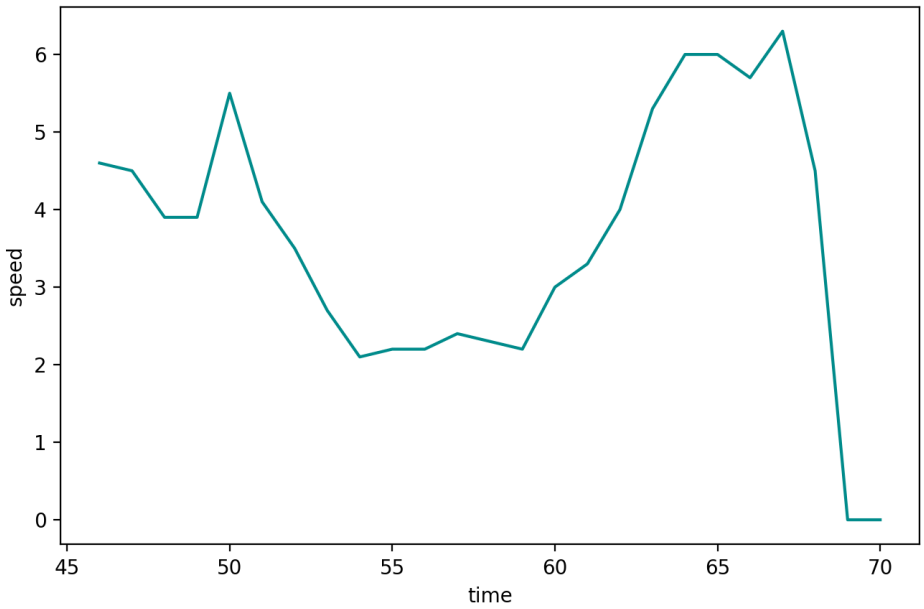


图 5 长时间低速行驶的异常数据

经过上面 3 部分全部预处理之后，数据文件 1 剔除的异常记录数为 54551 条，总记录数为 **131174** 条。数据文件 2 剔除的异常记录数为 37748 条，总记录数为 **108077** 条。数据文件 3 剔除的异常记录数为 58177 条，总记录数为 **106737** 条。

4 问题二的分析建模

在现有的构建行驶工况曲线的方法中，基于运动学片段进行构建是最常用的方法之一。在车辆行驶的过程中，会出现频繁的启动、加速、减速等过程，运动学片段就是指车辆从一个怠速状态开始至下一个怠速状态开始之间的运动过程[1]。如下图所示便是一个运动学片段的定义说明。

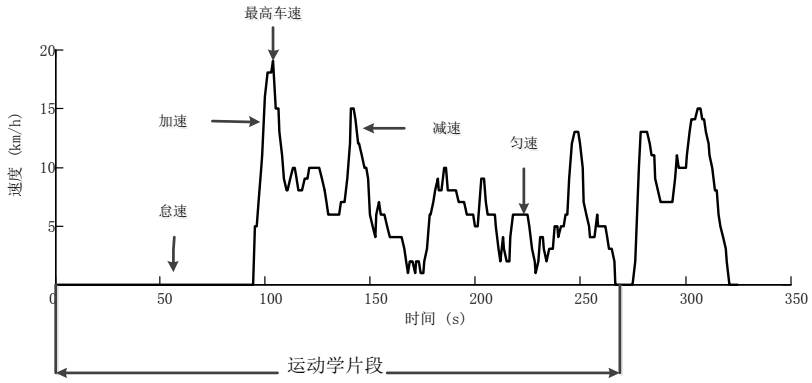


图 6 运动学片段的定义

不同的运动学片段往往对应着不同的交通状况，描述了低速、中速、高速等不同的汽车运行形态，在运动学片段的基础上构建工况是非常合理的[3]，最终我们通过对多种运动动学片段筛选组合来构建行驶工况。

在经过问题一的预处理后我们得到了一份处理后的数据源，图 7 展示了处理后的部分

数据，图中可以清晰地看出在这段数据中存在着多个运动学片段，如图中 a-b 段即是一个运动学片段。根据运动学片段我们可知，自一个怠速状态开始至下一个怠速状态开始被认为是一个运动学片段，因此我们以速度为 0km/h 的怠速状态为分割点对数据进行分割。当前时间速度值为 0km/h 且前一秒的速度值不为 0km/h 的记录被认为是一个怠速状态的开始点，我们将两个怠速开始点之间的数据片段记录为一个运动学片段，用此方法对三个数据文件进行切分。

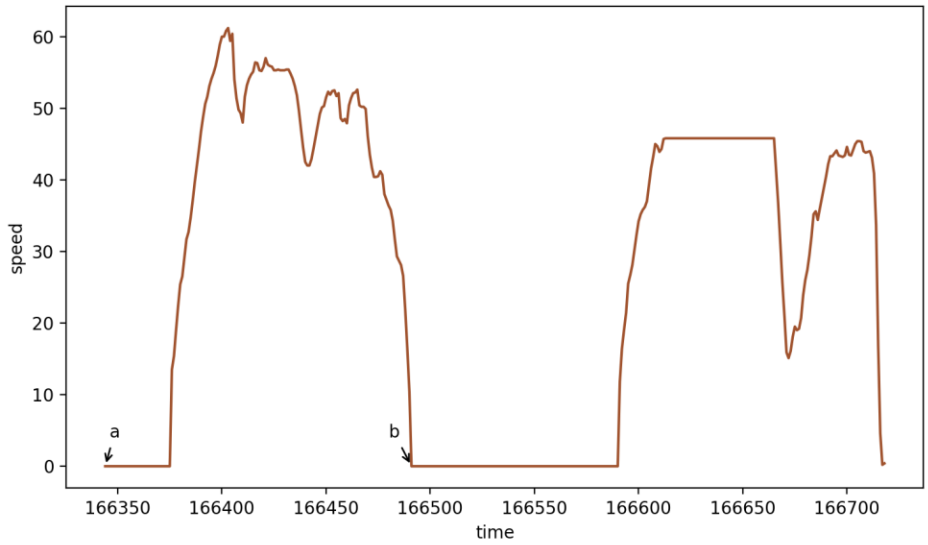


图 7 运动学片段示例

按上述方法划分后，最终数据文件 1 我们得到了 **715** 个运动学片段，数据文件 2 我们得到了 **616** 个运动学片段，数据文件 3 我们得到了 **482** 个运动学片段。

5 问题三的分析建模

5.1 特征参数的选取

对于每一个运动学片段，我们从其中提取 9 种运动学特征参数，并在之后对其进行计算和筛选。详细特征参数见表 2。

表 2 特征参数

编号	特征参数	编号	特征参数
1	平均速度	6	加速时间比
2	平均行驶速度	7	减速时间比
3	平均加速度	8	速度标准差
4	平均减速度	9	加速度标准差
5	怠速时间比		

其中，平均速度为在一个运动学片段内，行驶的距离与时间的比值，而平均行驶速度是不包括怠速阶段在内的平均速度，显然平均速度小于等于平均行驶速度。平均加速度、

平均减速度为该运动学片段内正负加速度各自的平均值。怠速时间比，加速时间比，减速时间比则是各行驶状态在整个运动学片段内所占比例，三者之和小于等于 1。最后，速度标准差，加速度标准差则是用来衡量速度、加速度的离散程度。

5.2 标准化处理与主成分分析

在对每一个运动学片段提取了特征参数后，我们得到了一个 2555 * 9 的特征参数矩阵。

为了避免使用所有特征参数作为分类指标，导致信息重叠，我们对这 2555 * 9 的特征参数矩阵进行了主成分分析[4]。

在进行主成分分析之前，为了消除特征参数间不同量纲的影响，我们对特征参数进行了标准化处理。在进行标准化的算法选择上，我们使用了两种标准化算法。

我们选取的第一种标准化方法是 Min-Max Normalization，将各个特征参数落到了[0,1]区间内，得到的新的特征值序列均值为 0，方差为 1，且无量纲。具体公式如下图所示：

$$y_i = \frac{x_i - \overline{x}}{s}$$

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

我们选择的第二种标准化方法是 Zero-mean Normalization，即标准差标准化，基于原特征序列的均值和标准差来进行序列的标准化，具体公式如下：

$$\mathrm{x}^* = \frac{\mathrm{x} - \mu}{\sigma}$$

对于这两种标准化方法，我们都进行了尝试，并根据后续的降维结果选择了较好的一种进行应用。

降维就是一种对高维度特征数据预处理方法。降维是将高维度的数据保留下最重要的一些特征，去除噪声和不重要的特征，从而实现提升数据处理速度的目的。在实际的生产和应用中，降维在一定的信息损失范围内，可以为我们节省大量的时间和成本。降维也成为应用非常广泛的数据预处理方法。

为了使得数据集更易使用并且降低算法的计算开销。我们对每一种的结果都进行了降维。

降维的算法有很多，比如奇异值分解(SVD)、主成分分析(PCA)、因子分析(FA)、独立成分分析(ICA)，这里我们选择主成分分析来进行数据降维。

主成分分析方法，是一种使用最广泛的数据降维算法。PCA 的主要思想是将 n 维特征映射到 k 维上，这 k 维是全新的正交特征也被称为主成分，是在原有 n 维特征的基础上重新构造出来的 k 维特征。PCA 的工作就是从原始的空间中顺序地找一组相互正交的坐标轴，新的坐标轴的选择与数据本身是密切相关的。其中，第一个新坐标轴选择是原始数据中方差最大的方向，第二个新坐标轴选取是与第一个坐标轴正交的平面中使得方差最大的，第三个轴是与第 1,2 个轴正交的平面中方差最大的。依次类推，可以得到 n 个这样的坐标轴。

通过这种方式获得的新的坐标轴，我们发现，大部分方差都包含在前面 k 个坐标轴中，后面的坐标轴所含的方差几乎为 0。于是，我们可以忽略余下的坐标轴，只保留前面 k 个含有绝大部分方差的坐标轴。事实上，这相当于只保留包含绝大部分方差的维度特征，而忽略包含方差几乎为 0 的特征维度，实现对数据特征的降维处理[5][6]。

第一种标准化数据方法的主成分分析结果如下表 3 所示：

表 3 贡献率

编号	贡献率	编号	贡献率
1	0.55971444	6	0.01751926
2	0.1947529	7	0.01433069
3	0.09489261	8	0.00540254
4	0.07485658	9	0.00159128
5	0.0369397		

第二种标准化数据方法的主成分分析结果如下表 4 所示：

表 4 贡献率

编号	贡献率	编号	贡献率
1	0.46280038	6	0.01777441
2	0.27828961	7	0.01424829
3	0.13251866	8	0.00644189
4	0.06441788	9	0.00247024
5	0.02103864		

鉴于第二种方法区分度略低，我们最终选择了 Min-max 标准化后的结果。

在其累计贡献率约为 80%时应保留的主成分为：

$$m = \min\left(\frac{\sum_{k=1}^{k} \lambda_k}{\sum_{i=1}^{q} \lambda_i} \approx 80\% \right]$$

基于此公式，本次保留前三个主成分进行聚类。

特征参数与主成分相关系数的绝对值大小代表着该特征参数与主成分之间的紧密关系，特征参数与第几主成分相关系数的绝对值最大，表明可以隶属于该主成分，对应关系如表 5 所示。

表中 PC1，PC2，PC3 分别为第一主成分，第二主成分，第三主成分与各特征参数之间的相关系数。

表 5 特征参数与各主成分的相关系数

特征参数	PC1	PC2	PC3
平均速度	0.45345403	-0.02059451	0.22423151
平均行驶速度	0.41606799	0.10269419	0.43614911
平均加速度	-0.04807408	0.55153627	-0.12388956
平均减速度	0.02898712	-0.5608044	0.00400461
怠速时间比	-0.43588477	0.13415529	0.30352172
加速时间比	0.39743191	-0.05215909	-0.29794642

减速时间比	0.32235107	-0.05257394	-0.58653948
速度标准差	0.39635821	0.12743609	0.42650455
加速度标准差	0.0956963	0.57499022	-0.19312517

5.3 聚类分析

5.3.1 K 均值聚类法

我们使用 K 均值聚类算法对之前所获取到的 2555 个运动学片段进行聚类。

k 均值聚类是最著名的划分聚类算法，由于简洁和效率使得他成为所有聚类算法中最广泛使用的。给定一个数据点集合和需要的聚类数目 k， k 均值算法根据某个距离函数反复把数据分入 k 个聚类中。它是一种迭代求解的聚类分析算法，其步骤是随机选取 K 个对象作为初始的聚类中心，然后计算每个对象与各个种子聚类中心之间的距离，把每个对象分配给距离它最近的聚类中心。聚类中心以及分配给它们的对象就代表一个聚类。每分配一个样本，聚类的聚类中心会根据聚类中现有的对象被重新计算。这个过程将不断重复直到满足某个终止条件。终止条件可以是没有（或最小数目）对象被重新分配给不同的聚类，没有（或最小数目）聚类中心再发生变化，误差平方和局部最小[8]。

由于 K 均值聚类算法需要指定 K 的值，因此我们首先需要决策出合理的 K 值。在决策 K 值时我们所采取的方法是“手肘法”。

手肘法的核心思想是：随着聚类数 k 的增大，样本划分会更加精细，每个簇的聚合程度会逐渐提高，那么误差平方和 SSE（sum of the squared errors，误差平方和）自然会逐渐变小。并且，当 k 小于真实聚类数时，由于 k 的增大会大幅增加每个簇的聚合程度，故 SSE 的下降幅度会很大，而当 k 到达真实聚类数时，再增加 k 所得到的聚合程度回报会迅速变小，所以 SSE 的下降幅度会骤减，然后随着 k 值的继续增大而趋于平缓，也就是说 SSE 和 k 的关系图是一个手肘的形状，而这个肘部对应的 k 值就是数据的真实聚类数。这也是该方法被称为手肘法的原因。计算 SSE 的具体公式如下：

$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2$$

其中，Ci 是第 i 个簇，p 是 Ci 中的样本点，mi 是 Ci 的质心（Ci 中所有样本的均值），SSE 是所有样本的聚类误差，代表了聚类效果的好坏。

我们假定最合理的聚类结果产生的类别数量不会太多，因此我们计算了 K 从 1 到 8 时的 SSE 分布曲线，如图 8 所示。

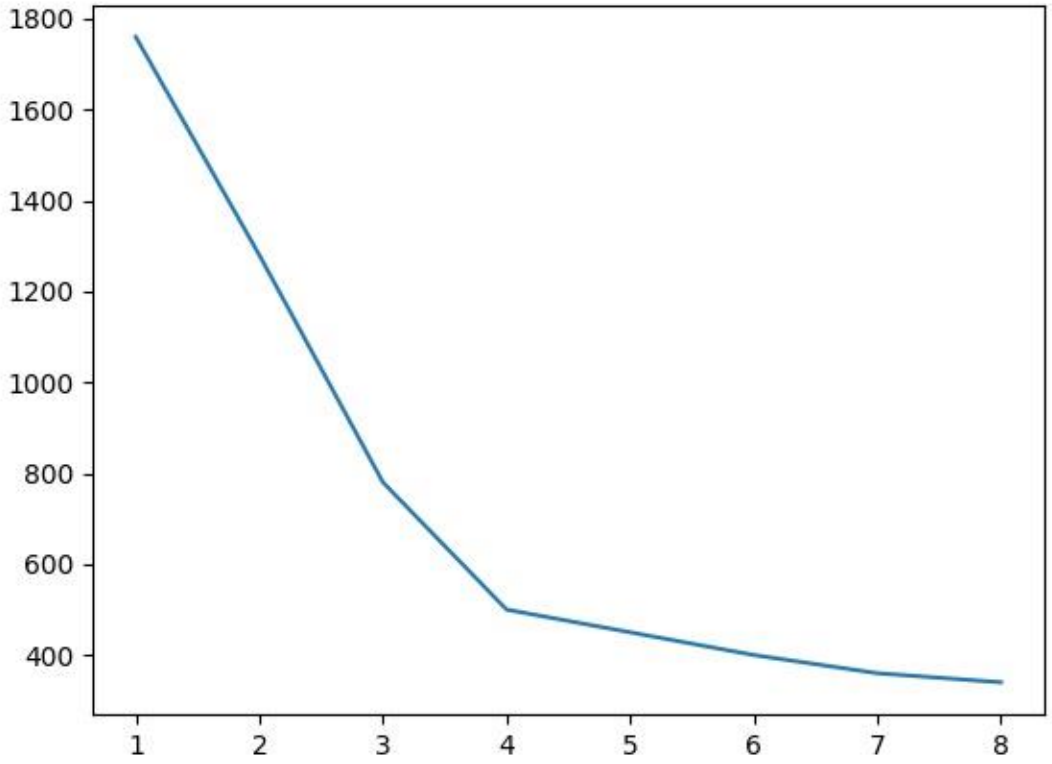


图 8 SSE-K 分布曲线

图中横轴为每次所选去的 K 值，纵轴为每个 K 值所对应的 SSE 值。最终选取了 4 作为本次 K 均值聚类算法的 K 值。

在进行聚类之前，我们还需要确定用于评估“两个样本间距离”的指标。我们选取了最常用的四种计算距离的公式进行调研，分别是欧氏距离、曼哈顿距离、夹角余弦距离和相关距离[7]。

欧氏距离是这 4 种方法中最简单直观，也是最常使用的一种方法，其具体公式如下：

$$d_{12} = \sqrt{\sum_{k=1}^{n} (x_{1k} - x_{2k})^2}$$

曼哈顿距离为标准坐标系上的绝对轴距综合，其结果是两个坐标差的绝对值，是一种“城市区块距离”，其计算公式如下：

$$\mathrm{d}_{12} = \sum_{k=1}^{n} |\mathrm{x}_{1k} - x_{2k}|$$

夹角余弦距离是通过从原点出发，并指向两个位置坐标所产生的两条向量间的夹角来测距，其计算公式如下：

$$\cos(\theta) = \frac{a \cdot b}{|a||b|}$$

相关距离依据相关系数来计算距离。相关系数是衡量给定随机变量 X 与 Y 的相关性的一种方法，取值范围是[-1,1]。相关系数和相关距离的计算公式分别如下所示：

$$\rho_{XY} = \frac{\mathrm{Cov}(X,Y)}{\sqrt{D(X)}\sqrt{D(Y)}} = \frac{E((X-EX)(Y-EY))}{\sqrt{D(X)}\sqrt{D(Y)}}$$
$$D_{xy} = 1 - \rho_{XY}$$

经过调研，我们认为相关距离和夹角余弦距离更适合用于文本数据聚类的计算。而曼哈顿距离与欧式距离较为接近，但通常使用欧氏距离聚类质量更高，且各类聚类算法实现中均默认实现了欧氏距离。因此我们选择了使用欧式距离作为评估样本间相似性的指标。

5.3.2 基于 K 均值的聚类分析

最后，我们为 K 均值聚类算法编写了程序进行聚类分析。K-means 的优点是相对高效并且通常终止在局部最优，但可用全局最优技术改进。(模拟退火和遗传算法)

聚类结果如图 9 所示：

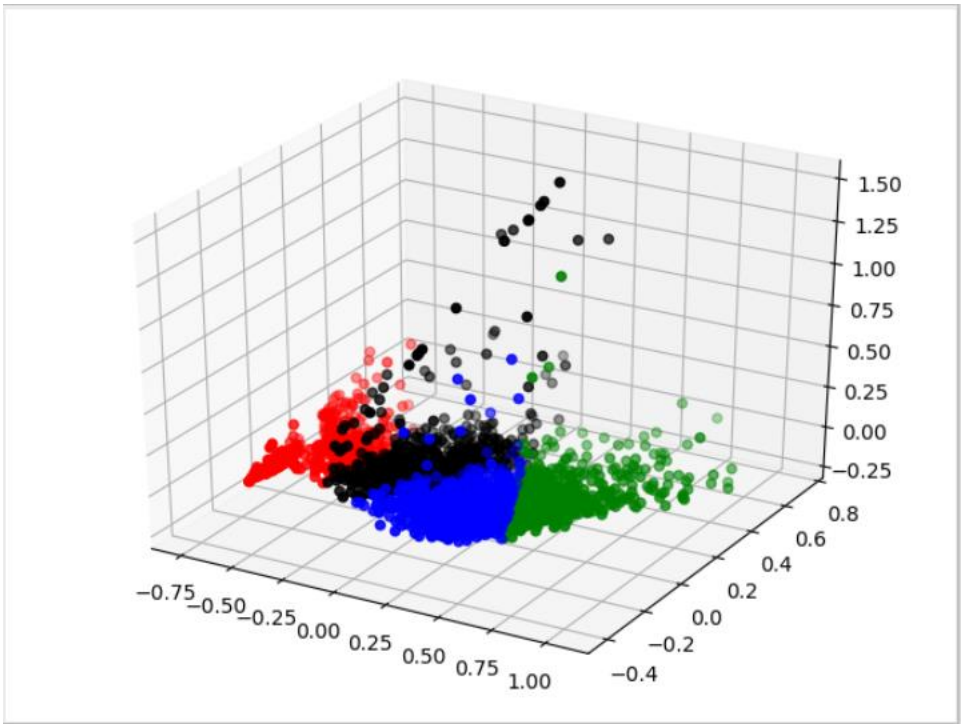


图 9 聚类结果图

对于每一类的运动学片段，我们计算了其特征参数，结果如表 4 所示：

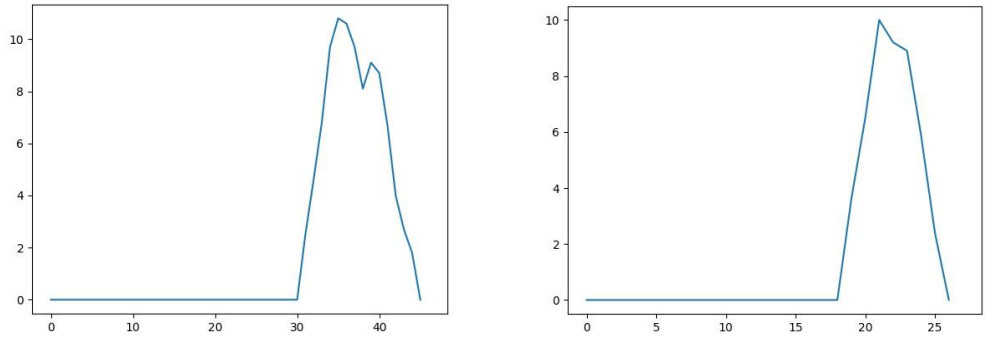
表 6 每一类的特征参数

特征值(单位)	簇 1	簇 2	簇 3	簇 4
Vm（km/h）	19.80	11.22	39.66	1.66

Vmr（km/h）	25.80	25.34	45.95	7.62
Vsd（km/h）	12.03	13.05	19.34	3.03
Ti（%）	16	50	12	72
Ta（%）	43	27	47	17
Td（%）	38	24	35	18
Am（m/s^2）	2.01	4.16	1.59	3.04
Ad（m/s^2）	-2.19	-4.49	-2.33	-3.02
Asd（m/s^2）	2.89	4.29	3.20	1.69

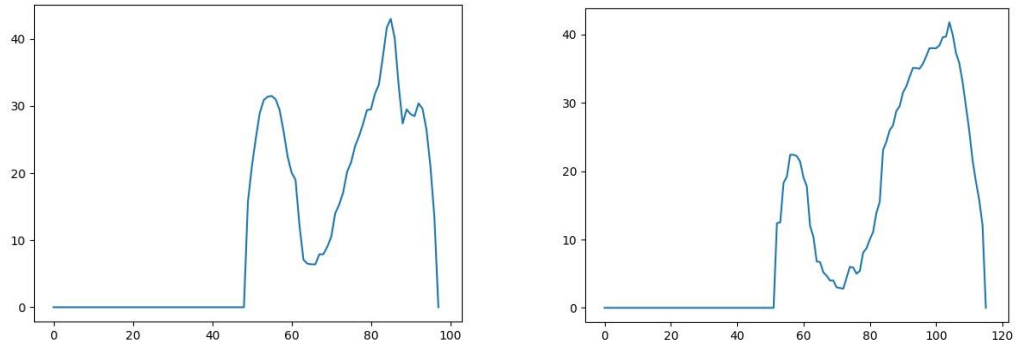
从表中数据我们可以发现每一类数据的特征不尽相同，可以代表不同类型的行驶状况。

并且对其中每一个类，选取具有代表性的科学运动片段，用 matplotlib 对这些典型运动片段作图，进一步观察他们的特征。



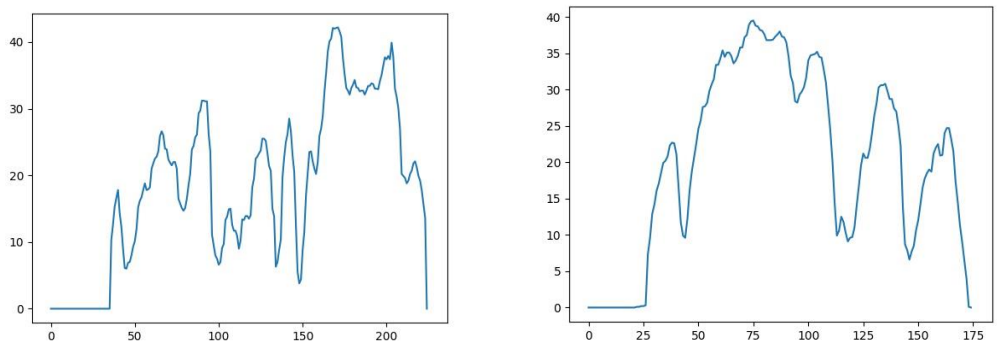
第一类典型运动片段

可以发现这类科学运动片段的普遍特点是怠速时间占比较大，运动时速度变化较少（先加速后减速）。对应现实中这主要反映了较为拥堵的交通状况，车辆平均速度较低，怠速时间比占比很高。



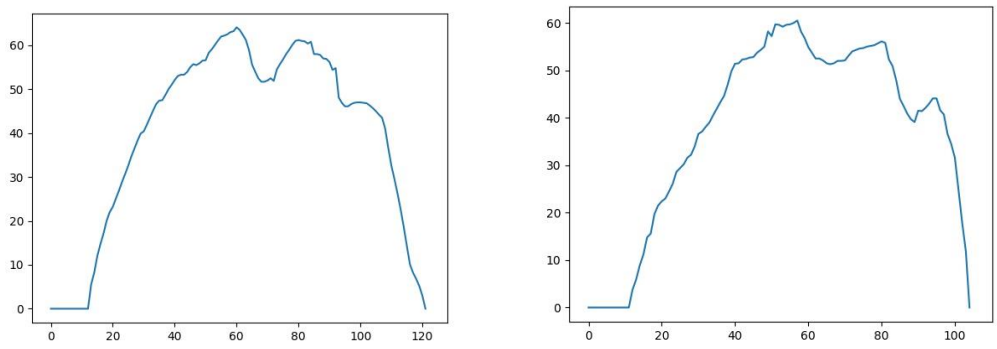
第二类典型运动片段

可以发现这类科学运动片段的普遍特点是怠速时间占比仍然较大，但是相比第一类科学运动片段，运动时速度变化较丰富（先加速后减速，再加速再减速）。对应现实中这可能反映了经常遇到红绿灯等存在停车时间的路段。



第三类典型运动片段

可以发现这类科学运动片段的普遍特点是怠速时间占比较小，运动时速度变非常丰富，这一般反应了交通状况适中的路段。



第四类典型运动片段

可以发现这类科学运动片段的普遍特点是怠速时间占比非常小，运动时速度变化比较少（先加速后减速，速度响度平稳）。可以看到第四类平均行驶速度很高，怠速时间比占比较低且加速时段占比较高，这可能对应了现实中交通状况较好的路段。

5.4 工况的合成

聚类的最终目的是为了构建出与原始数据相吻合的工况曲线，本次我们需要拟合出时间长度在 1200~1500s 的运动学片段。在 5.3 中我们计算得到了每一类中的典型运动片段，我们根据各类在整个数据集中所占的时间比例，来确定各类片段在最终工况合成中所占的时间和个数。具体公式如下：

$$k_i \quad \left. \begin{array}{l} \sum_{i=1}^{n_j} T_{ij} \\ = \dfrac{t_i n_j}{\sum_{j=1}^{n_j} T_{ij}} \end{array} \right\}$$

其中：T_i 为第 i 类在工况合成中所占的时间，N_i 为第 i 类片段中片段的个数，T_{ij} 为第 i 类中第 j 个片段所持续的时间，T_{all} 为样本中所有片段持续的总时间，T_{cycle} 为工况合成所

需的时间，K_i 为第 i 类片段在拟合的工况中的片段个数。

依据前一阶段得到的聚类结果，我们拟合出了时间长度为 1201s 的用车行驶工况，行驶工况的速度-时间曲线如图 10 所示。

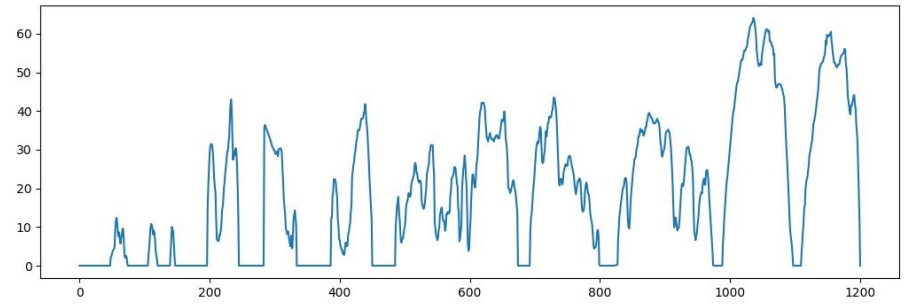


图 10 行驶工况速度-时间曲线

行驶工况的加速度-时间曲线如图 11 所示。

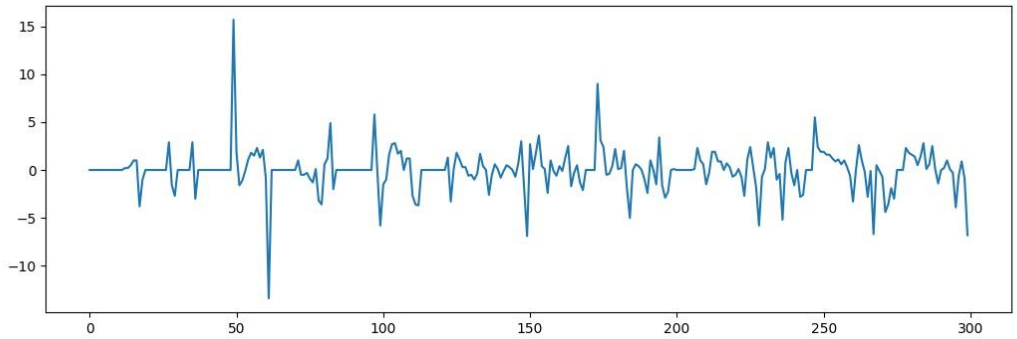


图 11 行驶工况加速度-时间曲线

6 模型评估

6.1 指标计算

对于 5.4 中合成的工况，我们计算其各项特征参数值，其结果如表 7 所示。

表 7 工况曲线特征参数值

特征值(单位)	拟合的工况曲线的特征参数值
Vm（km/h）	19.05
Vmr（km/h）	26.83
Vsd（km/h）	12.80
Ti（%）	29
Ta（%）	38
Td（%）	30
Am（m/s^2）	2.59
Ad（m/s^2）	-2.99

Asd（m/s^2）	2.44

6.2 合理性分析

6.2.1 特征参数值对比

为了对拟合的行驶工况曲线进行评估，我们计算了处理后的数据源的各项特征参数，并于拟合的行驶工况曲线进行对比，结果如表 8 所示。

表 8 原始数据与行驶工况的比较

特征值(单位)	拟合的工况	原始数据源	误差/%
Vm（km/h）	19.05	17.21	10.7
Vmr（km/h）	26.83	28.10	4.5
Vsd（km/h）	12.80	11.04	15.9
Ti（%）	32	39	17.9
Ta（%）	35	32	9.3
Td（%）	30	28	7.1
Am（m/s^2）	2.59	2.75	5.8
Ad（m/s^2）	-2.99	-3.01	0.6
Asd（m/s^2）	2.44	2.94	17.0

由表 6 的数据我们可以发现，拟合的行驶工况曲线各项特征值与原始数据源的平均误差约为 9.8%，误差值小于 10%，说明我们构建的汽车行驶工况代表性较好，与原始数据吻合度较高。

6.2.2 卡方检验

卡方检验是一种用途很广的计数资料的假设检验方法，它属于非参数检验的范畴，主要是比较两个及两个以上样本率以及两个分类变量的关联性分析。其根本思想就是在于比较理论频数和实际频数的吻合程度或拟合优度问题[10]。

卡方检验的计算公式为：

$$x^2 = \sum \frac{(A-T)^2}{T}$$

其中 A 为实际值，T 为理论值。卡方 x^2 用于衡量实际值与理论值的差异程度，它包含了以下两个信息：实际值与理论值偏差的绝对大小，以及差异程度与理论值的相对大小。

我们使用卡方检验对拟合的工况曲线进行验证，假设曲线与各行驶阶段的分布无差异，$\alpha = 0.05$。

计算后我们得到卡方值为 2.2247，小于根据自由度查询到的 3.84，所以我们可以认为拟合的行驶工况曲线与原始数据源的各行驶阶段的分布无差异。同时得到 p-value 为 0.9733，说明该曲线与数据源吻合度较高，较为精确。

参考文献

[1] 刘希玲, 丁焰. 我国城市汽车行驶工况调查研究[J]. 环境科学学报, 2000, 20(1):23-27.

[2] 石琴, 郑与波, 姜平. 基于运动学片段的城市道路行驶工况的研究[J]. 汽车工程, 2011, 33(3):256-261.

[3] 李孟良, 朱西产, 张建伟, 等. 典型城市车辆行驶工况构成的研究[J]. 汽车工程, 2005, 27(5):557-560.

[4] 高建平，孙中博，丁伟，郗建国. 车辆行驶工况的开发和精度研究[J]. 《浙江大学学报:工学版》 , 2017 , 51 (10) :171-179

[5] 白伟华. 浅谈主成分分析[J]. 数码世界, 2017(7).

[6] Liquan，主成分分析 PCA（Principal Component Analysis）在 sklearn 中的应用及部分源码分析，https://www.cnblogs.com/lochan/p/7001907.html，2019/09/22

[7] 林滨. K_Means 聚类的多种距离计算方法的文本实验比较[J]. 《福建工程学院学报》, 2016 ,14 (1) :80-85

[8] AmirHosein Fadaei, Seyed Hossein Khasteh. Enhanced K-means re-clustering over dynamic networks[J]. Expert Systems with Applications, 2019, Volume 132:126-140

[9] 汪晶，邹学玉，喻维明，孙咏. 分布式 MVC-Kmeans 算法设计与实现[J]. 《长江大学学报(自然科学版) 》，2019，16(6): 113-119

[10] Snowdroptulip, 统计学——卡方检验和卡方分布，https://blog.csdn.net/snowdroptulip/article/details/78770088，2019/09/22

附录

问题一部分代码

%% 第一部分

```
# read csv
d = pd.read_csv('original_data/file1.csv', usecols=['时间', 'GPS 车速'])
print("Before data processing,data shape is:")
print(d.shape)

# data processing
print("Data processing...")
# 1 data transform --replace datatime to time series
d['时间'] = d['时间'].apply(lambda x: int(
    (str_to_datetime(x.replace(".000.", "")) - datetime.datetime(2017, 12, 1, 19, 43,
56)).total_seconds()))
%% 第二部分
# 2 data processing
# 重复值清洗

# 2.0 长期低速度行驶(超过 20 秒)
i = 0
while i < d.shape[0] - 1:
    if d['GPS 车速'][i] < 10:
        forward = i
        while d['GPS 车速'][forward] < 10 and forward < d.shape[0] - 1:
            forward += 1
        if (forward - i) > 20:
            d['GPS 车速'][i:forward] = 0
            i = forward + 1
            continue
    i += 1

i = 0
while i < d.shape[0] - 1:
    # 2.1 时间不连续删除整个运动片段
    if d['时间'][i + 1] - d['时间'][i] > 3:
        forward = i + 1  # 运动片段尾指针
        backward = i  # 运动片段头指针
        while d['GPS 车速'][forward] >= 10 and forward < d.shape[0] - 1:
            # 非怠速
            forward += 1
        while (not (d['GPS 车速'][backward - 1] >= 10 and d['GPS 车速'][backward] <= 10))
```

```
and backward > 1:
                # 非怠速
                backward -= 1
            to_drop.append([backward, forward])
            i = forward + 1
            # print("#1      " + str(i))
            continue
        # 2.2 加速度异常删除整个运动片段
        if d['GPS 车速'][i] > 100:
            len = i
            while d['GPS 车速'][len] > 10 and len > 1:
                len -= 1
            if i - len < 7:
                forward = i + 1   # 运动片段尾指针
                backward = i   # 运动片段头指针
                while d['GPS 车速'][forward] >= 10 and forward < d.shape[0] - 1:
                    forward += 1
                while (not (d['GPS 车速'][backward - 1] >= 10 and d['GPS 车速'][backward] <=
10)) and backward > 1:
                    backward -= 1
                to_drop.append([backward, forward])
                i = forward + 1
                # print("#2      " + str(i))
                continue
        # 2.3 长期怠速(停车超过 180 秒)
        if d['GPS 车速'][i] < 10:
            forward = i
            while d['GPS 车速'][forward] < 10 and forward < d.shape[0] - 1:
                forward += 1
            if (forward - i) > 180:
                # print("长期停车/怠速异常" + str(i))
                # 删剩 180s
                to_drop.append([i, forward - 180])
                i = forward - 180 + 1
                # print("#3      " + str(i))
                continue
        i += 1

for x in to_drop:
        d = d.drop(d.index[x[0]:x[1]])

        %% 第三部分
# write csv
print("Writing processed data...")
```

```
d.to_csv('processed_data/file1.csv', index=False)
```

问题二部分代码

```
%% 第一部分
# read csv
d = pd.read_csv('processed_data/file1.csv')
print(d.shape)
%% 第二部分
i = 0
spl = []
while i < d.shape[0] - 1:
    if d['GPS 车速'][i] != 0 and d['GPS 车速'][i + 1] == 0:
        # end of a fragment
        spl.append(i)
    i += 1

print(spl)
label = 0
for k in range(len(spl)):
    new = d[label:spl[k] + 2]
    new = new.reset_index(drop=True)  # 重新进行 index 生成
    new.to_csv('processed_data/file1_fragments/fragment' + str(k + 1) + '.csv', index=False)
    label = spl[k] + 1
```

问题三部分代码

```
%% 第一部分 汽车运动特征评估体系
def calculate(filePath):
d = pd.read_csv(filePath)

# 运动时长
total_time = d.shape[0]

# 平均速度
ave_speed = format(np.mean(d['GPS 车速']), '.2f')

# 平均行驶速度
ave_runtime_speed = format(np.mean(d[d['GPS 车速'] >= 10]['GPS 车速']), '.2f')

acceleration_time = 0
deceleration_time = 0
acceleration_sum = 0
deceleration_sum = 0
```

```
    for i in range(d.shape[0] - 1):
        if d['GPS 车速'][i] < d['GPS 车速'][i + 1]:
            acceleration_time += 1
            acceleration_sum += d['GPS 车速'][i + 1] - d['GPS 车速'][i]
        elif d['GPS 车速'][i] > d['GPS 车速'][i + 1]:
            deceleration_time += 1
            deceleration_sum += d['GPS 车速'][i + 1] - d['GPS 车速'][i]

    # 平均加速度
    if acceleration_time == 0:
        ave_acceleration = 0.00
    else:
        ave_acceleration = format(acceleration_sum / acceleration_time, '.2f')

    # 平均减速度
    if deceleration_time == 0:
        ave_deceleration = 0.00
    else:
        ave_deceleration = format(deceleration_sum / deceleration_time, '.2f')

    # 怠速时间比
    slow_rate = format(d[d['GPS 车速'] == 0].shape[0] / d.shape[0], '.2f')

    # 加速时间比
    acceleration_rate = format(acceleration_time / (d.shape[0] - 1), '.2f')

    # 减速时间比
    deceleration_rate = format(deceleration_time / (d.shape[0] - 1), '.2f')

    # 速度标准差
    std_speed = format(np.std(d['GPS 车速'], ddof=1), '.2f')

    acceleration = []
    for i in range(d.shape[0] - 1):
        acceleration.append(d['GPS 车速'][i + 1] - d['GPS 车速'][i])
    # 加速度标准差
    std_acceleration = format(np.std(acceleration, ddof=1), '.2f')

    line = [filePath, ave_speed, ave_runtime_speed, ave_acceleration, ave_deceleration,
slow_rate,
            acceleration_rate, deceleration_rate, std_speed, std_acceleration]
    out = open('processed_data/fragments_set/all_file.csv', 'a', newline='')
    csv_write = csv.writer(out, dialect='excel')
    csv_write.writerow(line)
```

```
    %% 第二部分 PCA 主成分分析
    data = pd.read_csv(inputfile)  # 读入数据
# 数据标准化
# 1、min-max 标准化（Min-Max Normalization）
def min_max(data):
    return (data - data.min()) / (data.max() - data.min())
# 2、Z-score 标准化方法(主成分太多了，没有用这种方法)
def Z_score(data):
    return (data - data.mean()) / (data.std())

index, cal = data["filePath"], data
data_norm = min_max(cal)
# data_norm["filePath"] = index
print(data_norm.head())

from sklearn.decomposition import PCA
pca = PCA()  # 保留所有成分
pca.fit(data_norm)
print(pca.components_)  # 返回模型的各个特征向量
print('每个成分各自方差百分比 :')
print(pca.explained_variance_ratio_)  # 返回各个成分各自的方差百分比(也称贡献率）

pca = PCA(3)  # 选取累计贡献率大于 80%的主成分（1 个主成分）
pca.fit(data_norm)
low_d = pca.transform(data_norm)  # 降低维度
pca = pd.DataFrame(low_d, index)
pd.concat([pca, cal], axis=1).to_csv(outputfile)

    %% 第三部分 K-Means 聚类
    loan_data = pd.read_csv('processed_data/fragments_set/all_file_pca.csv')

# 设置要进行聚类的字段
loan = np.array(loan_data[['0', '1', '2']])
# 设置类别为 3
clf = KMeans(n_clusters=4)
# 将数据代入到聚类模型中
clf = clf.fit(loan)

loan_data['label'] = clf.labels_

# 提取不同类别的数据
loan_data0 = loan_data.loc[loan_data["label"] == 0]
loan_data1 = loan_data.loc[loan_data["label"] == 1]
loan_data2 = loan_data.loc[loan_data["label"] == 2]
```

```
loan_data3 = loan_data.loc[loan_data["label"] == 3]

    %% 第四部分 选取每个簇的代表样本
    # 计算最接近簇心的科学运动片段
# 欧几里得距离
def get_sample(filePath, cluster_center):
    loan_data = pd.read_csv(filePath)

    loan_data['distant'] = ((loan_data['0'] - cluster_center[0]) ** 2 + (
            loan_data['1'] - cluster_center[1]) ** 2 + (
                                        loan_data['2'] - cluster_center[2]) ** 2) ** 0.5
    return loan_data[loan_data['distant'] == loan_data['distant'].min()]

simple0 = get_sample(filePath0, cluster_center0)
simple1 = get_sample(filePath1, cluster_center1)
simple2 = get_sample(filePath2, cluster_center2)
simple3 = get_sample(filePath3, cluster_center3)


print(simple0)    # processed_data/file1_fragments/fragment594.csv
print(simple1)    # processed_data/file1_fragments/fragment1167.csv
print(simple2)    # processed_data/file1_fragments/fragment793.csv
print(simple3)    # processed_data/file1_fragments/fragment101.csv

    # matplotlib 作图
def draw_simple(filePath, cluster):
    loan_data = pd.read_csv(filePath)
    X = loan_data["时间"]-min(loan_data["时间"])
    Y = loan_data["GPS 车速"]
    plt.plot(X, Y)
    plt.plot(max(X+1),0)
    plt.savefig('picture/k=4/cluster' + str(cluster) + '.jpg')
    plt.show()

    %% 第五部分 计算每个簇的评估体系指标
    def cal(data, list0):
    ave_speed = []
    ave_runtime_speed = []
    ave_acceleration = []
    ave_deceleration = []
    slow_rate = []
    acceleration_rate = []
    deceleration_rate = []
    std_speed = []
```

```
    std_acceleration = []
    for i in range(data.shape[0]):
        filePath = data["filePath"][i]
        if filePath in list0:
            ave_speed.append(data["平均速度"][i])
            ave_runtime_speed.append(data["平均行驶速度"][i])
            ave_acceleration.append(data["平均加速度"][i])
            ave_deceleration.append(data["平均减速度"][i])
            slow_rate.append(data["怠速时间比"][i])
            acceleration_rate.append(data["加速时间比"][i])
            deceleration_rate.append(data["减速时间比"][i])
            std_speed.append(data["速度标准差"][i])
            std_acceleration.append(data["加速度标准差"][i])

    %% 第六部分 构建汽车行驶工况曲线
    # 按照样本比例选取的每个簇代表样本
    cluster1_1 = pd.read_csv("processed_data/file2_fragments/fragment123.csv")
cluster1_2 = pd.read_csv("processed_data/file1_fragments/fragment747.csv")
cluster1_3 = pd.read_csv("processed_data/file1_fragments/fragment444.csv")
cluster2_1 = pd.read_csv("processed_data/file3_fragments/fragment4.csv")
cluster2_2 = pd.read_csv("processed_data/file3_fragments/fragment32.csv")
cluster2_3 = pd.read_csv("processed_data/file2_fragments/fragment366.csv")
cluster3_1 = pd.read_csv("processed_data/file1_fragments/fragment923.csv")
cluster3_2 = pd.read_csv("processed_data/file2_fragments/fragment370.csv")
cluster4_1 = pd.read_csv("processed_data/file1_fragments/fragment118.csv")
cluster4_2 = pd.read_csv("processed_data/file1_fragments/fragment119.csv")
cluster4_3 = pd.read_csv("processed_data/file1_fragments/fragment412.csv")
cluster1 = pd.concat([cluster1_1, cluster1_2, cluster1_3], axis=0)
cluster2 = pd.concat([cluster2_1, cluster2_2, cluster2_3], axis=0)
cluster3 = pd.concat([cluster3_1, cluster3_2], axis=0)
cluster4 = pd.concat([cluster4_1, cluster4_2, cluster4_3], axis=0)
total = pd.concat([cluster4, cluster2, cluster1, cluster3], axis=0)
print(total)
total = total.reset_index()
total.to_csv("clusterALL/final.csv")
plt.figure(figsize=(12.5, 4))
Y = total["GPS 车速"]
plt.plot(Y)

plt.savefig('picture/k=4/fileALL/combined.jpg')
plt.show()

%% 第七部分 评估验证模型
kf_data = np.array([cluster1, cluster2, cluster3, cluster4, total, final])
```

```
kf = chi2_contingency(kf_datad)
print('chisq-statistic=%.4f,  p-value=%.4f,  df=%i  expected_frep=%s'%kf)
```