第九届湖南省研究生数学建模竞赛承诺书

我们仔细阅读了湖南省高校研究生数学建模竞赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们完全清楚，在竞赛中必须合法合规地使用文献资料和软件工具，不能有任何侵犯知识产权的行为。否则我们将失去评奖资格，并可能受到严肃处理。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们授权湖南省研究生数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

我们参赛选择的题号是（从组委会提供的赛题中选择一项填写）：A

我们的参赛编号（请填写完整参赛编号）：A20241800 1022

所属学校（请填写完整的全名）：国防科技大学

参赛队员（打印后签名）：1. 薛永恒

2. 蔚阳

3. 舒昕洁

指导教师或指导教师组负责人（打印后签名）：

日期：2024 年 7 月 12 日

（请勿改动此页内容和格式。以上内容请仔细核对，如填写错误，论文可能被取消评奖资格。）

第九届湖南省研究生数学建模竞赛

题目：使用智能手机记录人体活动状态

摘要

随着智能手机的普及，越来越多的手机具备评估用户日常活动消耗热量的功能。比如，华为手机的“华为运动健康”软件可以根据用户每天跑步、步行、骑车、爬高等活动状态来计算其当天消耗的热量，因此，本文针对运动状态识别问题，运用 K-Means 聚类、神经网络模型、支持向量机等模型进行计算机求解，解决了如何根据 x、y、z 轴加速度以及角加速度的数据进行运动状态判别以及人员特征（年龄、身高和体重）刻画问题。

对于问题一来说，本文对附件 1 中提供的 3 名实验人员的运动数据进行分析，旨在对每种活动状态的数据进行分类。首先，对原始数据进行了清洗和标准化处理，接着基于运动状态基本特征，筛选出五类；然后再提取了时间域和频率域的特征并进行 PCA 降维，最后通过 K-means 聚类算法对活动状态进行分类。结果表明，基于数据的时-频域特征进行 K-means 聚类再识别和分类活动状态上具有较高的准确性。

对于问题二，本文基于 10 名实验人员的活动数据，建立了一个多层感知机分类器模型，用于识别和分类人员的活动状态。对时序数据切片提取时间域和频率域特征，构建特征向量，对多层感知机分类器进行模型训练，通过十折交叉验证评估模型的准确性。结果显示，模型的整体分类准确率为 98.95%，多数活动状态的 F1 分数接近 1，活动 11（乘坐电梯向上移动）和活动 12（乘坐电梯向下移动）的分类精度相对较低在 93%以上。最终，评估效果最好的模型对附件 3 的数据进行判别，进一步验证了模型的有效性。

对于问题三，进一步分析了传感器数据与实验人员年龄、身高、体重之间的关系，首先计算了 person 4-13 总共 10 个人员加速度数据的均值、标准差、最大值、最小值、众数、高度、过均值点数七个特征值；随后，进行神经网络模型训练，即通过输入不同人员不同测试数据的均值、标准差、最大值、最小值、众数、高度、过均值点数七个特征值数据，训练得到人员身高年龄、身高以及体重；然后，计算 person1-3 的均值、标准差、最大值、最小值、众数、高度、过均值点数七个特征值，并输入到已经训练好的神经网络模型，得到 person1-3 共 3 名人员的年龄、身高和体重，并进行准确度评估。经

计算 person1 的年龄、身高、体重分别为 31，171 和 69，与实际的误差值分别为 19%、7.6%和 8.1%；person2 的年龄、身高、体重分别为 30，171 和 68，与实际的误差值分别为 3.6%、1.1%和 0.8%；person3 的年龄、身高、体重分别为 30，171，66，与实际的误差值分别为 6.4%、6.6%和 11.9%。由此，可见，该模型人员特征预测效果较好。最后，再计算出附件 5 数据的均值、标准差、最大值、最小值、众数、高度、过均值点数七个特征值，代入到已经训练好的神经网络模型，得到预测的年龄、身高以及体重。构建支持向量机模型（SVM），根据附件 5 预测得到的年龄、身高和体重判别其最可能来源于问题 2 中哪一名实验人员；同时，对向量机模型进行较差验证，计算泛化误差，其值为 0.53。

关键词：K-means 聚类算法、分类，神经网络、运动状态识别、支持向量机

目录

1 问题综述

1.1 问题背景

随着国力不断强盛和国家技术水平的不断发展，国民温饱问题已经基本得到解决，但随之而来的是，人们对自己的生活质量和对自己的健康要求逐渐提高。从儿童到老人的各个年龄群体都面临着巨大的生理健康需求，如何利用科技创新来更好的改善和指导人们的身体健康，已经成为国家亟待解决的重大社会需求。基于以上背景，多人群（包括亚健康人群、运动关节受损者、空巢老年人群等）运动安全和日常健康监测、变得非常重要。人们每天有很多运动状态，走路，跑步，坐下，起立等多种运动方式，通过对人体日常运动状态进行监测，可以指导人们制定出健康合理的饮食计划、合理安排每天的运动量，特别对老人，老年痴呆患者，把他们的日常运动数据反馈给医护人员，提高健康生活水平和运动安全有着重要的意义。

传感器技术和微机电系统(Micro Electro Mechanical Systems，MEMS)制造技术的迅猛发展，使得大多数智能手机都具备评估手机使用人日常活动消耗的热量的功能，比如华为手机的“华为运动健康”软件能够根据手机使用人每天跑步、步行、骑车、爬高等活动状态来计算其当天消耗的热量，但如何根据使用人的运动数据，如速度、角速度以及加速度等，进行运动状态识别仍面临着许多问题。目前，已经有很多学者提出了大量基于大量运动状态的数据建立算法模型，来进行运动状态识别，这促使了国民健康管理模式的发展。

1.2 问题提出

智能手机测量人体的活动状态主要依靠内置于其中的运动传感器——加速度计和陀螺仪来实现。加速度计是用于测量手机在三个轴向(X，Y，Z)上的线性加速度变化情况的传感器，当手机发生加速度变化时，加速度计会感应到这种变化并将加速度数据传输给手机处理器进行存储或分析。陀螺仪是用来测量手机绕着三个轴向(X，Y，Z)旋转的角速度的传感器，当携带手机的人发生转向时，陀螺仪会感知到转向的速度和方向，并将这些数据传输给手机处理器进行存储或处理。现要求依据附件中给出的日常活动状态数据，建立模型完成以下问题：

(1) 问题 1：请根据附件 1 提供的活动数据，对每一位实验人员的活动状态的数据进行分类，在论文中将分类结果（编号）填入表 1.1。

表 1.1

分类	Person1	Person2	Person3
第 1 类			
第 2 类			
…			
第 12 类			

(2) 问题 2：请根据附件 2 提供的活动数据（每人 60 组数据）提取 12 类人员活动状态的典型特征，建立人员活动状态的判别模型，并利用你们模型开展以下验证工作：

（a）进一步运用问题 1 的分类模型对该 10 名实验人员数据进行分类（此时，不考虑实验人员的活动状态标签），比较问题 2 中判别模型和问题 1 的分类模型的结果，分析采用分类模型对不同活动类型分类时的分类准确度。

（b）请基于附件 3 中某实验人员 30 次活动的状态数据，运用判别模型，给出该人员的活动状态，在论文中将结果填入表 1.2。

表 1.2

活动类型	判别状态
SY1	
SY2	
…	

(3) 问题 3：请根据附件 4 分析不同人员的同一活动状态是否存在差异？活动状态数据与实验人员的年龄、身高、体重有无关系，能否使用活动传感器数据进行人员画像，进一步，请使用你们的模型判断附件 5 中 10 位实验人员分别最可能来源于问题 2 中哪一名实验人员。在论文中将判别结果填入表 1.3。

表 1.3

活动类型	判别结果
Unknow1	
Unknow2	
Unknow3	
Unknow4	
Unknow5	

1.3 资料条件

各附件的详细说明如下:

• 附件 1 中给出了 3 名实验人员的运动数据，包含每名实验人员每种活动状态的 5 组加速度计和陀螺仪数据，但实验时未记录数据所代表的活动状态；

• 附件 2 中给出了 10 名实验人员的活动数据，包含每位实验人员每种活动状态的 5 组加速度计和陀螺仪数据，但实验时记录了每组数据所代表的实验人员的活动状态；

• 附件 3 中给出了某实验人员 30 次活动的状态数据；

• 附件 4 给出了问题 1 和问题 2 中参与实验的 13 位实验人员的年龄、身高、体重等数据；

• 附件 5 中给出了问题 2 的 10 位实验人员中的 5 位的某次活动数据，数据包含了每人的 12 类活动状态。

2 模型假设与符号说明

2.1 模型基本假设

(1) 假设人员在不同运动状态下以及不同位置条件下的重力加速度分量相同。

2.2 符号说明

本文定义了如下 5 个使用次数较多的符号, 其余符号在使用时注明。

表 2.1 符号说明

符号	含义	单位
mean	均值	/
sid	标准差	/
max	最大值	/
min	最小值	/
$cross\_mean$	过均值点数	/

3 数据预处理

本文的数据预处理方法如图 3.1 所示，主要涉及到重力加速度分解、数据平滑、数据分片、数据清洗五部分组成。需要说明的是，这五种数据预处理方法我们并没有在每一问题都应用，例如问题三我们只采用了重力加速度分解，数据平滑两种

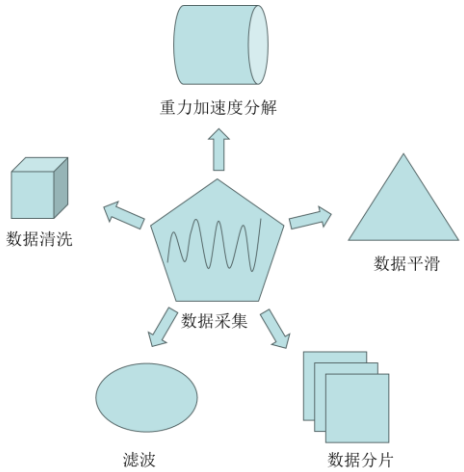


图 3.1 数据预处理流程

3.1 重力加速度分解

加速度传感器在采集数据时会受重力加速度的影响，所以采集来的数据反应的不是真实情况，因此，首先采用一阶低通滤波器求重力加速度分量[1,2]，其计算公式如下：

$$g_j(i) = \alpha g_j(i-1) + (1-\alpha)a_j(i)$$

其中α是滤波参数，j表示三个坐标轴分量，$gj(0)$=$aj(0)$，滤波参数α一般人为设定即可，且满足 0<α<1，实验过程中会根据实验效果调整滤波参数，本文中参照相关文献[3,4]，设定为 0.95。为验证该处理的有效性，选取附件 2 中 person3 运动状态分别为站立、向前跑、跳跃、乘坐电梯向上运动的第一组数据进行验证，得到的结果如图 3.2-3.6 所示。

从图中可以明显看出，重力加速度消除后，五组站立数的三个轴加速度值都在 0 上下浮动，与实际事实相符。这说明，使用一阶低通滤波器消除重力加速度的影响效果明显。

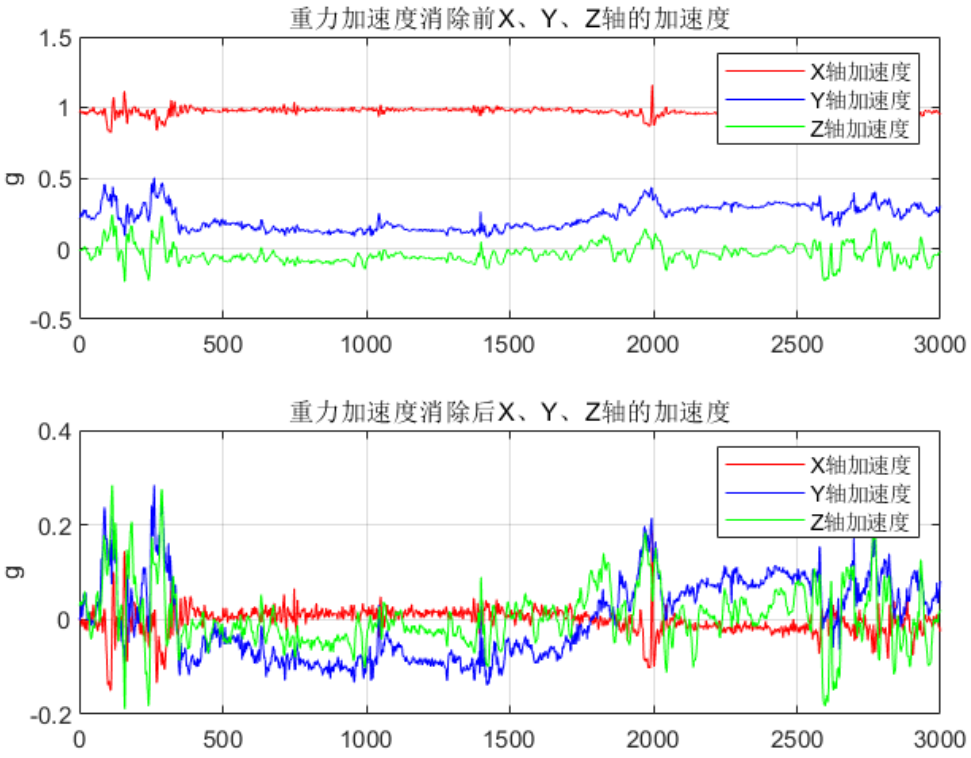


图 3.2 站立状态下第一组数据重力加速度消除前后对比

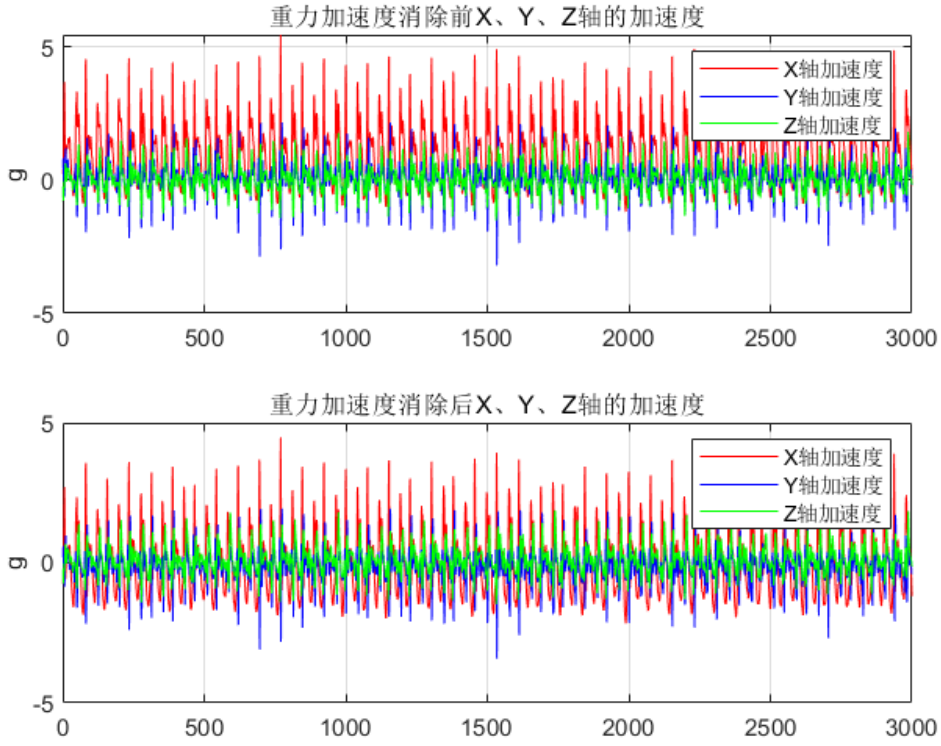


图 3.3 向前跑第一组数据重力加速度消除前后对比

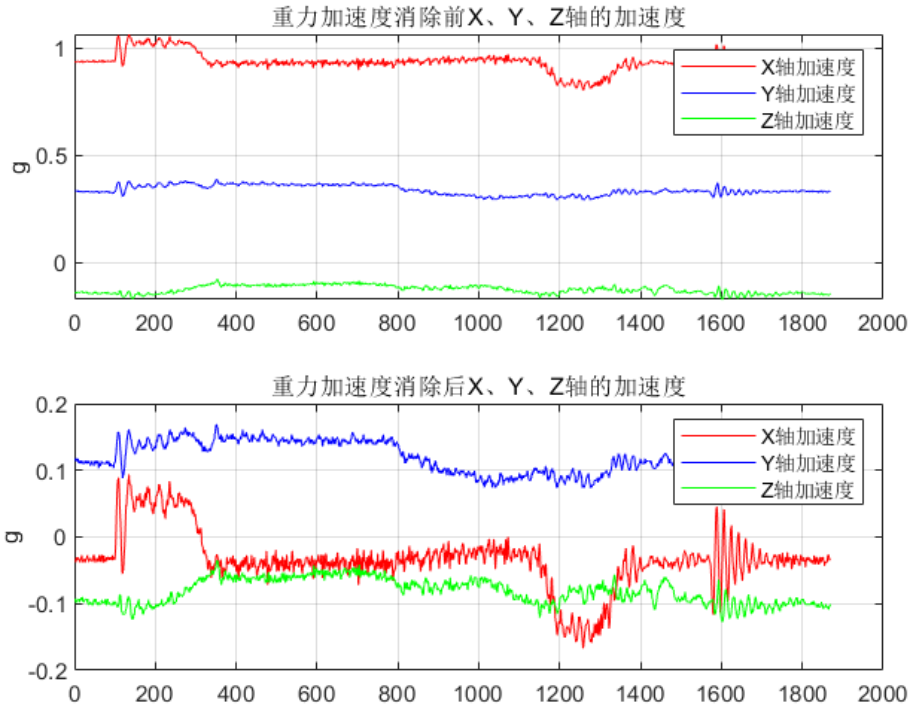


图 3.4 乘坐电梯向上第一组数据重力加速度消除前后对比

3.2 数据平滑

采用两次 3 点均值平滑，即连续三个原始采样点的均值作为一个新的采样点。假设某一段原始采样点的 X 轴上分量为$(x_1^0, x_2^0, \ldots, x_n^0)$，经过一次 3 点均值平滑后，得到的数据为$(x_1^1, x_2^1, \ldots, x_{n-2}^1)$，再经过一次 3 点均值平滑后，得到的数据为$(x_1^2, x_2^2, \ldots, x_{n-4}^2)$，每进行一次均值平滑，这段数据会减少两个采样点，经过两次 3 点均值平滑后，原来的采样点将减少 4 个。下面的公式是计算的 3 点均值平滑的计算过程：

$$x_i^1 = \frac{x_i^0 + x_{i+1}^0 + x_{i+2}^0}{3} \; x_i^2 = \frac{x_i^1 + x_{i+1}^1 + x_{i+2}^1}{3} \tag{3-1}$$

$$x_i^2 = \frac{x_i^1 + x_{i+1}^1 + x_{i+2}^1}{3} = \frac{1}{3}\left(\frac{x_i^0 + x_{i+1}^0 + x_{i+2}^0}{3} + \frac{x_{i+1}^0 + x_{i+2}^0 + x_{i+3}^0}{3} + \frac{x_{i+2}^0 + x_{i+3}^0 + x_{i+4}^0}{3}\right) \tag{3-2}$$

Savitzky-Golay 算法原理：

假设有一组数据点$yy = \{y_1, y_2, \ldots, y_N\}$，首先，定义一个窗口$W_i$包含$y_i$,及其周围的点；其次，在窗口$W_i$上使用最小二乘法拟合一个 m 阶多项式$p(x)$，其中 m 通常是(窗口大小−1)/2；然后，计算多项式$p(x)$在$x = x_i$处的值,即$\hat{y}_i = p(x_i)$；最后将$\hat{y}_i$赋值给原始数据点y_i的对应位置,得到平滑后的数据点。

选取附件 2 中 person3 运动状态分别为站立、向前跑、跳跃、乘坐电梯向上运动的第一组数据进行重力加速度消除并平滑处理，其前后的对比效果如图 3.5-3.8 所示。由图

可知，在使用两次三点均值、Savitzky-Golay 算法[5]平滑后，绝大多数的噪音被过滤掉了，而整体的波形信号却保留下来了，重力加速度消除与平滑为下一步提取进行特征提取具有重要意义。

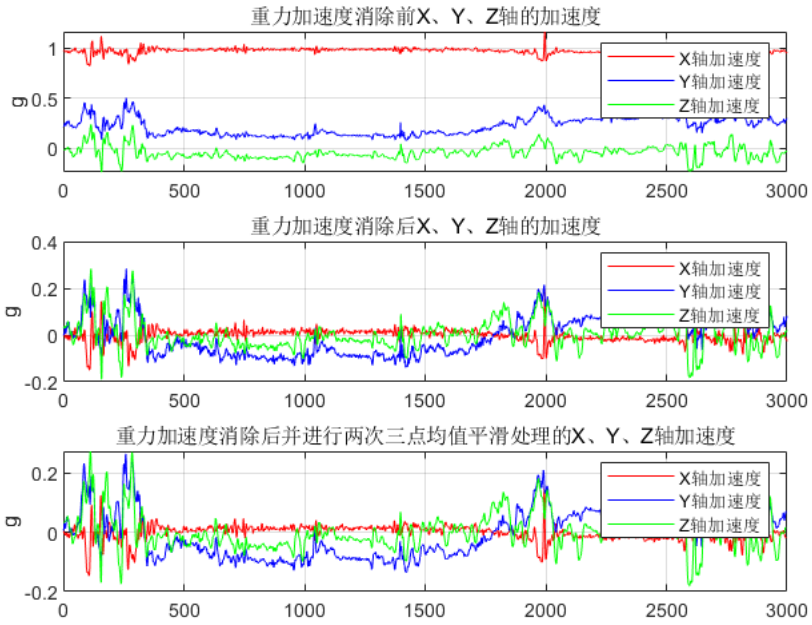


图 3.5 站立状态第一组数据重力加速度消除并进行两次三点均值平滑处理的前后对比图

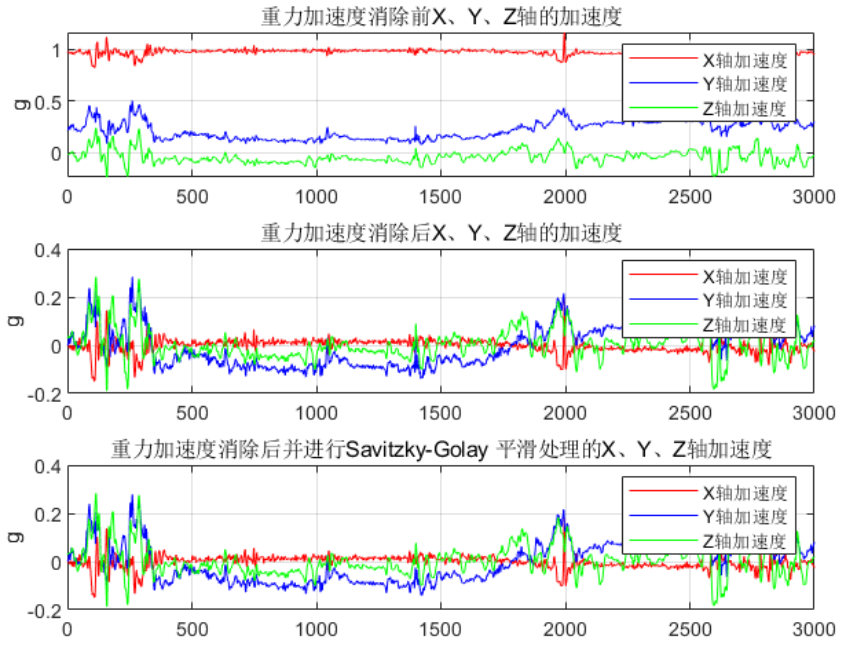


图 3.6 站立状态第一组数据重力加速度消除并进行 Savitzky-Golay 平滑处理的前后对比图

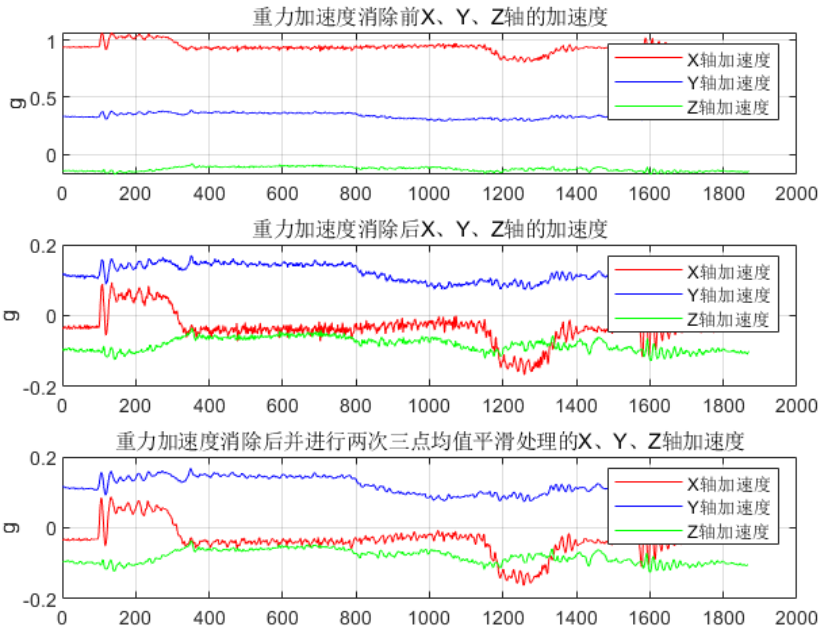


图 3.7 乘坐电梯向上运动第一组数据重力加速度消除并进行两次三点均值平滑处理的前后对比图

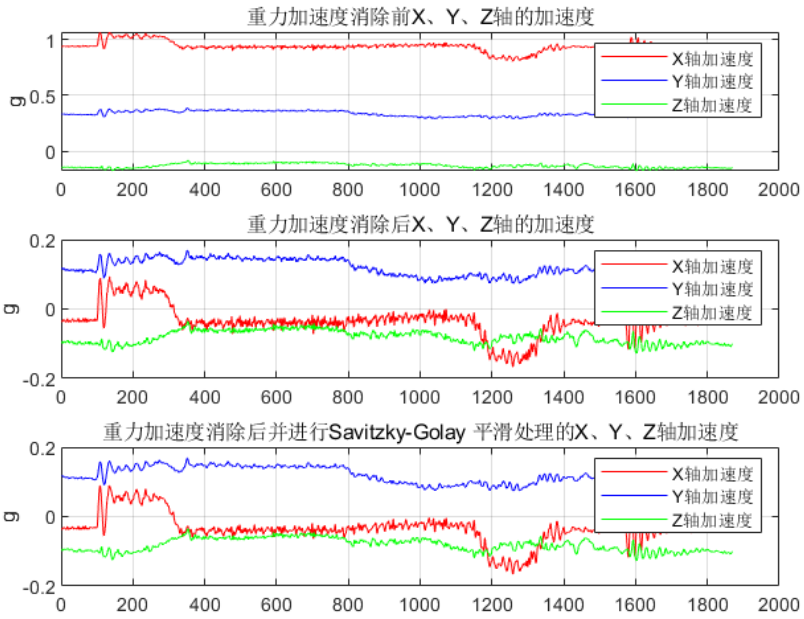


图 3.8 乘坐电梯向上运动第一组数据重力加速度消除并进行 Savitzky-Golay 平滑处理的前后对比图

分别计算经两次三点均值平滑和 Savitzky-Golay 平滑处理的均方误差（MSE）和平均绝对误差（MAE）,计算结果分别如表 1 和表 2 所示。均方误差（MSE）和平均绝对误差（MAE）的计算公式分别为：

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3-3}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3-4}$$

对比表 1 和表 2 中的均方误差（MSE）和平均绝对误差（MAE），在同一运动状态下的同一方向轴的条件下，若表 1 的 MSE 大于表 2，则颜色填充为黄色；若表 1 的 MAE 大于表 2，则颜色填充为蓝色。因此，由表 1 和表 2 可以判断出两次三点均值平滑处理的效果优于 Savitzky-Golay 平滑处理，故本文中，选取两次三点均值算法进行经重力加速度消除后的数据光滑处理。

表 1 两次三均值平滑处理

运动状态	方向轴	均方误差 (MSE)	平均绝对误差 (MAE)
站立第一组数据	X 轴	0.000178	0.008126
	Y 轴	0.000190	0.008946
	Z 轴	0.000247	0.010109
乘坐电梯向上运动第一组数据	X 轴	0.000108	0.007297
	Y 轴	0.000017	0.003014
	Z 轴	0.000021	0.003424

表 2 Savitzky-Golay 平滑处理

运动状态	方向轴	均方误差 (MSE)	平均绝对误差 (MAE)
站立第一组数据	X 轴	0.000007	0.001743
	Y 轴	0.000015	0.002243
	Z 轴	0.000005	0.001561
乘坐电梯向上运动第一组数据	X 轴	0.000007	0.002024
	Y 轴	0.000003	0.001314
	Z 轴	0.000002	0.001246

3.3 数据分片

时序数据的主要特点是其在时间上连续，具有上下文关系。在时序数据的分析过程中需要对数据的片段进行分析，为此在本问题的研究中我们使用了数据切片方法，原理如图 3.13 所示。在进行窗口分割时如何确定分割窗口是一个关键性问题。例如持续时间比较短的活动，比如坐下，如果窗口太长或太短都不能有效的识别。其实，在运动状态识别中的很多分类错误都是由于分割窗口的大小选择不当而造成的。如果窗口太短，有可能不能覆盖一个动作的跨度，如果窗口太长，有可能会重叠两个不相关的活动。

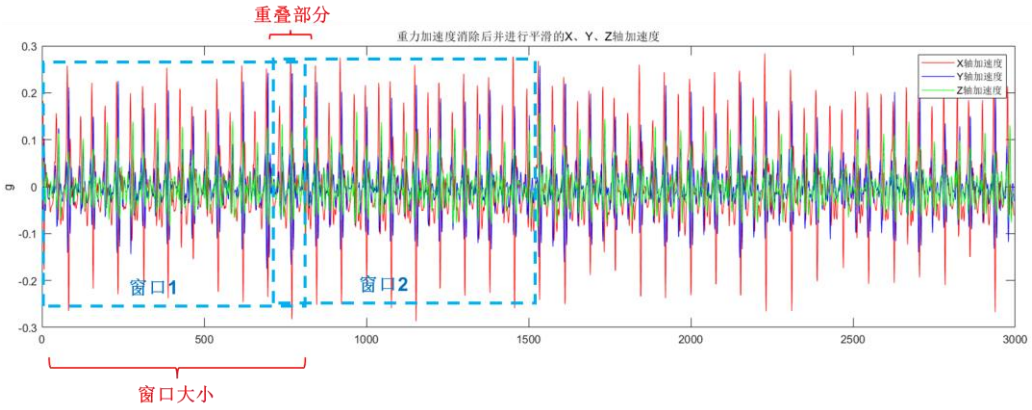


图 3.9 近周期变化的加速度分窗处理

信号周期的计算可以通过信号的自相关函数计算得到，以附件 2 中 Person10 的 a1t1.xlsx 文件为例，对其 x 轴加速度信号进行自相关计算，结果如图 3.10 所示：

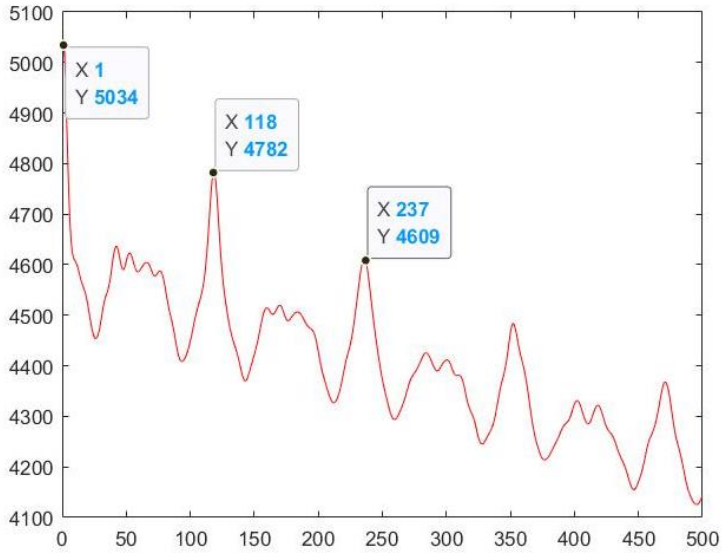


图 3.10 周期自相关分析图

图中，前三个峰值之间的间隔约为 120。在本题的解题过程中，切分窗口大小保守的选择为 150 个样本点以确保周期信号的完整，切分步长选择为时间窗口长度的 50%为 75 个样本点。在窗口对数据进行连续切片的过程中，舍弃最后一个完整数据切片之后的小于窗口长度的数据，由于监测数据量较多，且每个数据文件中都包含大量完整的信号周期数据，丢弃少量截尾样本不会产生显著的影响。

3.4 数据清洗

箱型图法（Box-plot）是一种被广泛应用的异常点去除方法，其具有良好的泛化性，对采集数据的分布形式无约束条件。箱型图如图 3.11 所示。

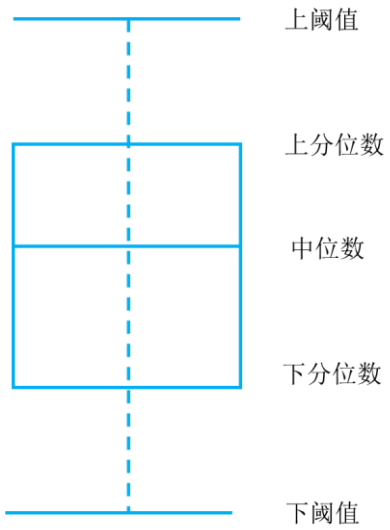


图 3.11 箱型图示意图

将每一类数据由大到小排列，取每类数据的最大值、最小值、上/下四分位值和中位值，设定阈值对异常点进行筛选。数据集的阈值如公式所示：

$$Limit_{up} = Max[Q_L + 1.5(Q_U - Q_L)] \tag{3-5}$$

$$Limit_{down} = Min[Q_L - 1.5(Q_U - Q_L)] \tag{3-6}$$

式中的上四分位值是数据由大到小排列的前 25%对应值，下四分位值是数据由大到小排列的后 25%对应值。这里将$Limit_{up}$作为每类数据的上阈值，$Limit_{down}$作为每类数据的下阈值，其中Q_L为采样数据最小值，Q_U为采样数据最大值。箱线图法用于处理设定范围内的异常值，对各类数据都具有较好的规范效果。

3.5 4 阶低通巴特沃斯滤波器数据滤波

巴特沃斯滤波器[6]是一种非常平滑的滤波器，它使用无尖峰的巴特沃斯多项式作为频率响应的逼近。这种滤波器在通过和阻带之间提供最平滑的过渡，但可能不如其他类型的滤波器（如切比雪夫滤波器）具有陡峭的截止特性。

对于一个 4 阶（n=4）低通巴特沃斯滤波器，其传递函数通常具有如下形式：

$$H(s) = \frac{1}{(1+\frac{s}{\omega_c})^2(1+\frac{s}{\omega_c \cdot \sqrt{2}})^2} \tag{3-7}$$

$H(s)$是拉普拉斯变换域中的传递函数。

ω_c是截止频率，表示滤波器在该频率下增益下降到-3dB（即大约为最大值的 70.7%）

在时间域中，这个传递函数对应于一个递归滤波器，其系数由巴特沃斯多项式确定。对于离散时间信号，可以使用双线性变换将连续时间的巴特沃斯滤波器转换为离散时间的。

以 Person1 的 SY01 样本中的 X 加速度为例，经 4 阶（n=4）低通巴特沃斯滤波器前后的对比如图图 3.12 所示，可见通过使用具有 4Hz 截止频率的 4 阶低通巴特沃斯滤波器进行滤波，数据噪声明显减少，同时并未丢失图线中的波峰波谷信息。

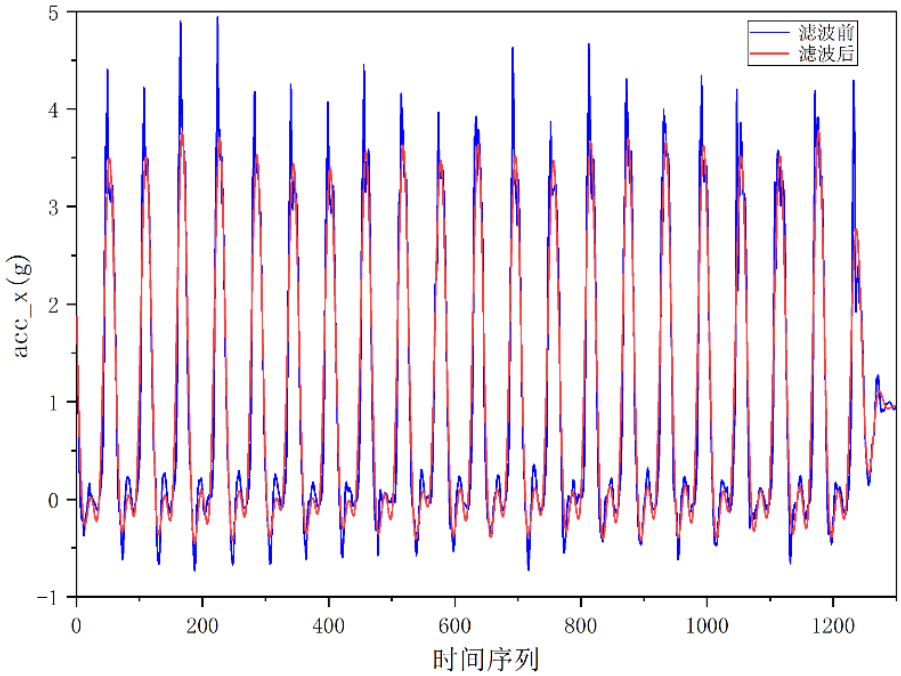


图 3.12 4 阶低通巴特沃斯滤波器进行滤波前后对比图

4 问题与数据分析

4.1 问题分析

4.1.1 问题一分析

问题一要求我们对附件 1 中有 3 名实验人员的运动数据（未记录数据所代表的活动状态）进行运动状态分类。首先，我们先对数据进行初步分析，对于站立状态、乘坐电梯等数据，其 X 轴加速度存在明显的变化特征，故我们先对附件 1 数据预划分出五类，即站立、躺下、坐下、乘坐电梯向上以及乘坐电梯向下五类。然后，在对剩余数据尽心时域、频域特征提取后，进行特征数据降维，最后在进行 K-means 聚类成七类。

4.1.2 问题二分析

问题二要求我们请根据附件 2 提供的活动数据提取 12 类人员活动状态的典型特征，建立人员活动状态的判别模型，并利用模型开展以下验证工作：

（a）进一步运用问题 1 的分类模型对该 10 名实验人员数据进行分类（此时，不考虑实验人员的活动状态标签），比较问题 2 中判别模型和问题 1 的分类模型的结果，分析采用分类模型对不同活动类型分类时的分类准确度。

（b）请基于附件 3 中某实验人员 30 次活动的状态数据，运用判别模型，给出该人员的活动状态。

结合对所测得数据的分析以及相关研究的调研，提取采集的多轴数据时域和频域特征，构建时序数据片段的时域和频域特征描述向量。之后，构建多层感知机神经网络，使用时频域描述特征对神经网络模型进行训练，得到运动状态数据的判别模型。最后，对判别模型性能进行评估验证。

4.1.3 问题三分析

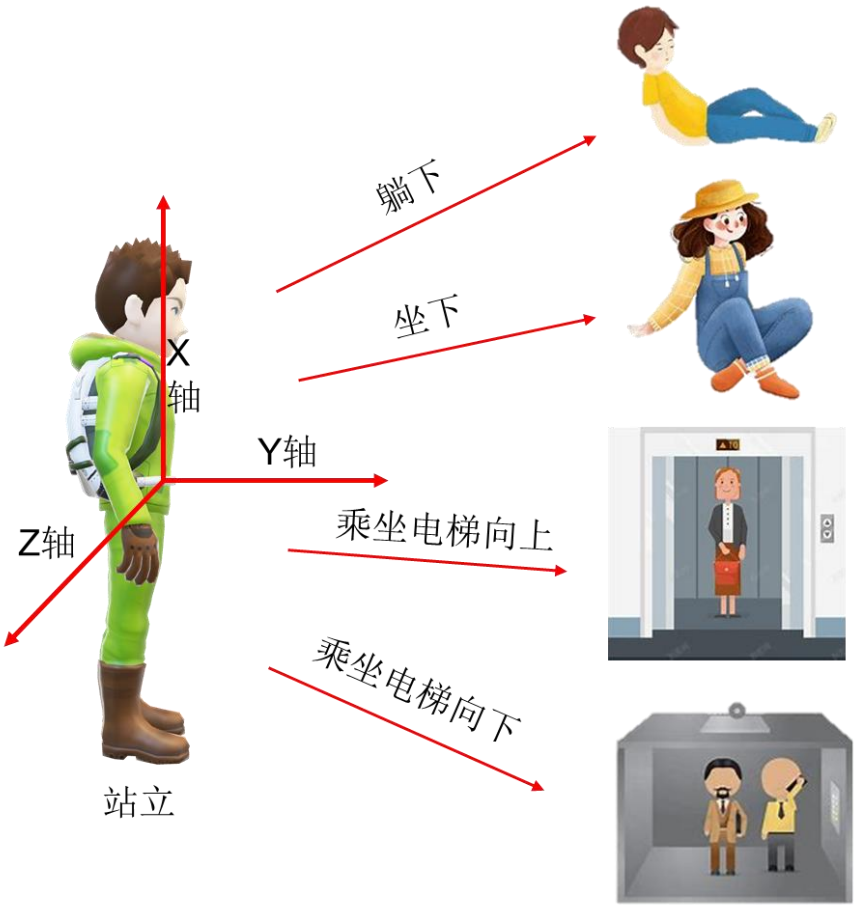
问题三首先要求我么根据附件 4 分析不同人员的同一活动状态是否存在差异？活动状态数据与实验人员的年龄、身高、体重有无关系？对于这一问题，可以计算出不同人员在同一运动状态下加速度数据的均值、方差、最大值等特征值，判断这些特征值是否存在规律性差异，从而分析身高、年龄以及体重对同一运动状态的影响规律，本文仅以向前走为样本分析年龄、身高、体重对运动数据的影响；其次，问题三还要求我们能否使用活动传感器数据进行人员画像，进一步，请使用你们的模型判断附件 5 中 10 位实验人员分别最可能来源于问题 2 中哪一名实验人员。在在本文题中，可以首先利用已

知运动数据对应的人员画像，训练神经网络模型，然后，利用附件 5 中的运动数据相应的人员特征预测；最后，再利用已知数据构建支持向量机分类模型，对附件 5 预测得到的人员画像进行判别分类。

4.2 数据分析

首先进行运动状态分析，对于站立状态，由于此时身躯并未运动，故在 X、Y 和 Z 三轴的加速度和角加速度应存在明显的无波动特征，如图中红色虚线框所示；而对于乘坐电梯向上和乘坐电梯向下的运动状态，在 X 轴上应呈现明显的加速、匀速以及减速状态，如图所示。依次对每种运动状态进行力学与数据分析，可以得到表所示的典型数据特征图。

通过表 1 可以粗略判断给定数据属于哪一种运动状态



。 。 。

图 4.1 运动状态图

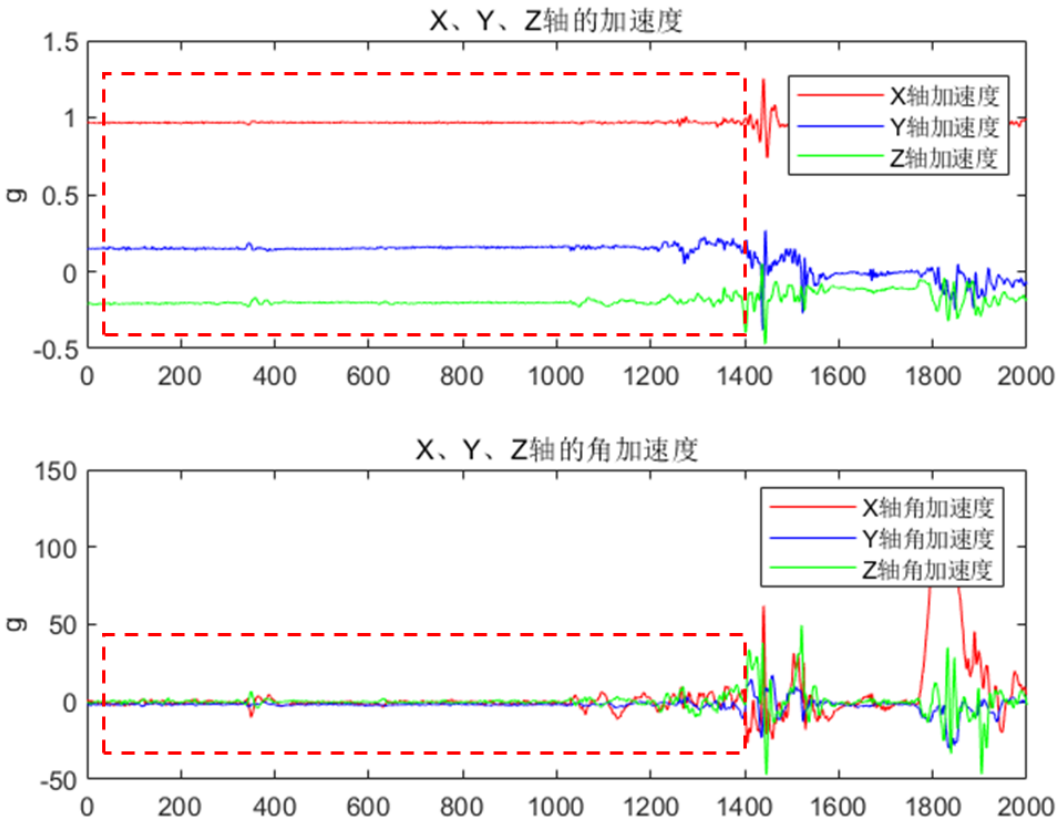


图 4.2 站立运动状态数据图

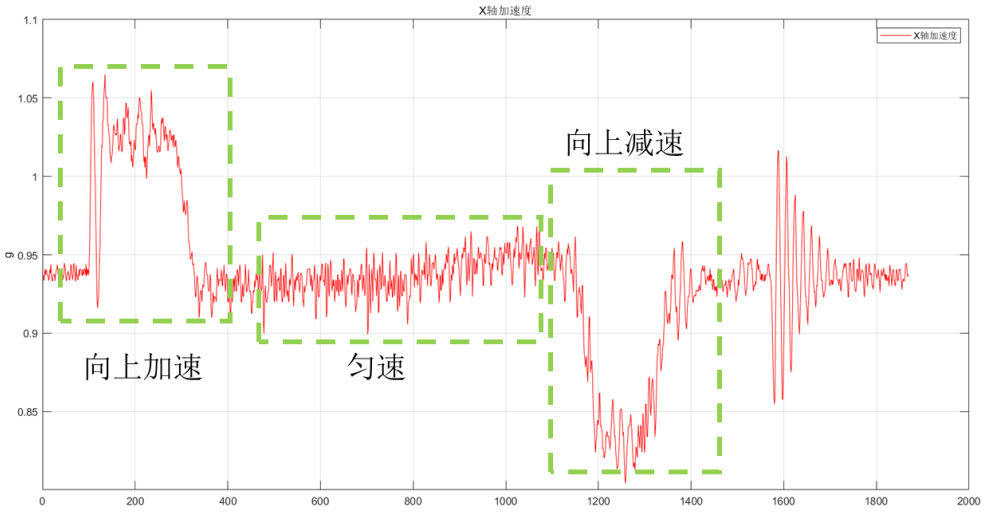


图 4.3 乘坐电梯向上运动状态数据图

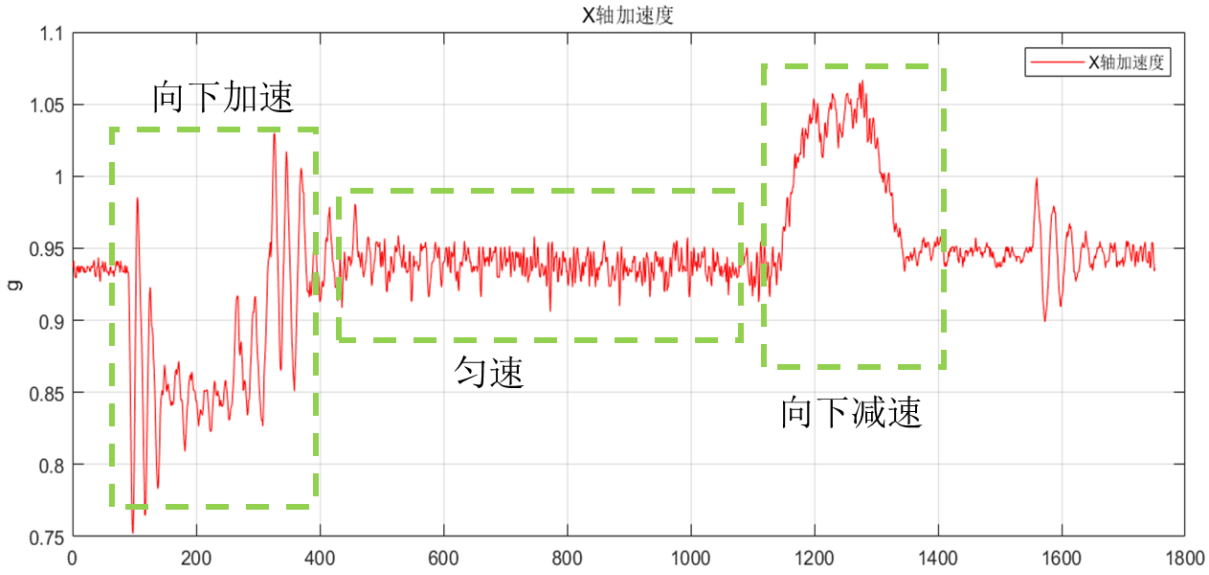
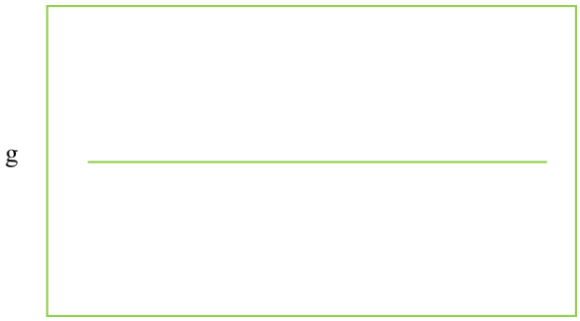
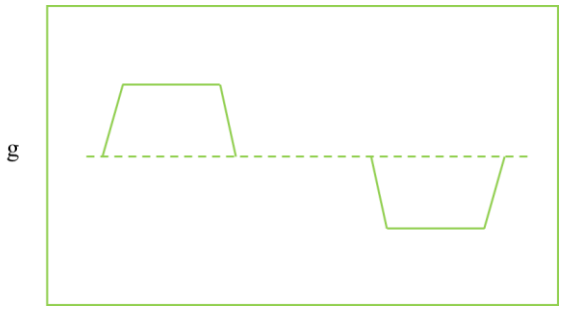
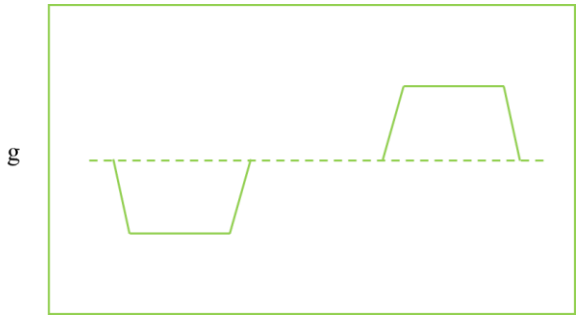
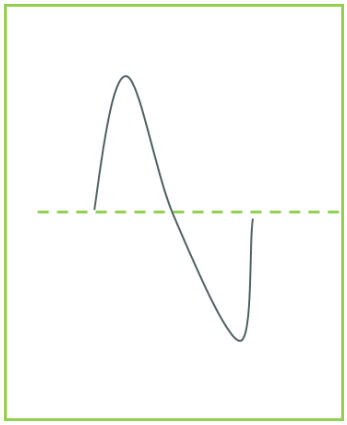
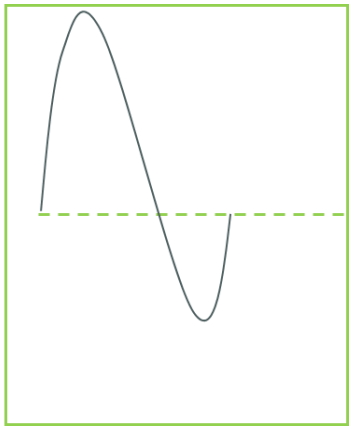
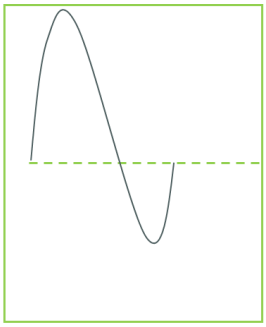
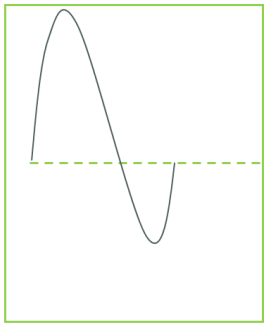
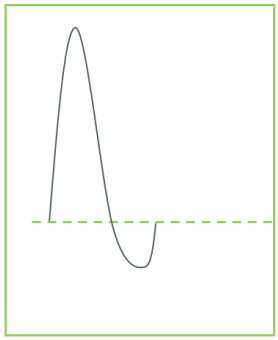
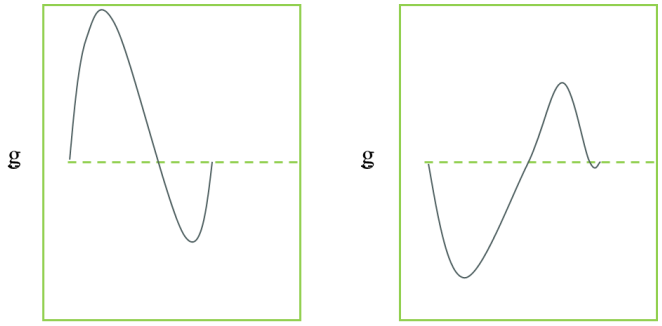
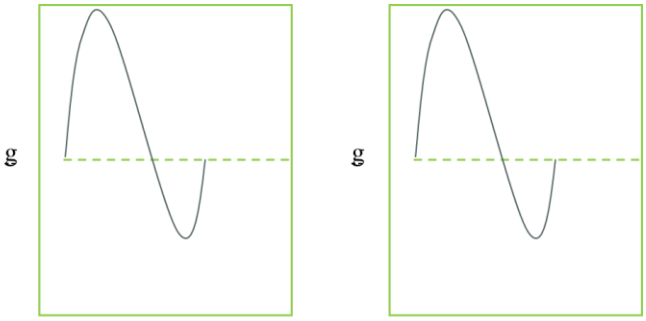
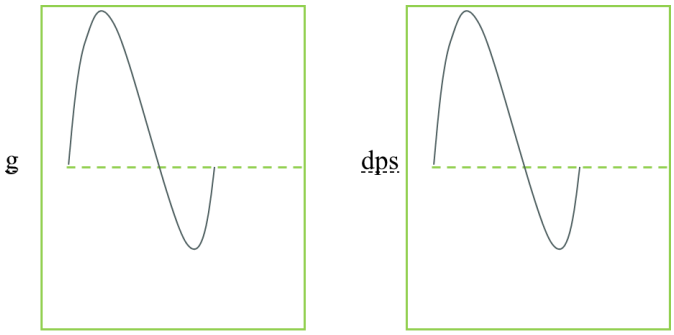
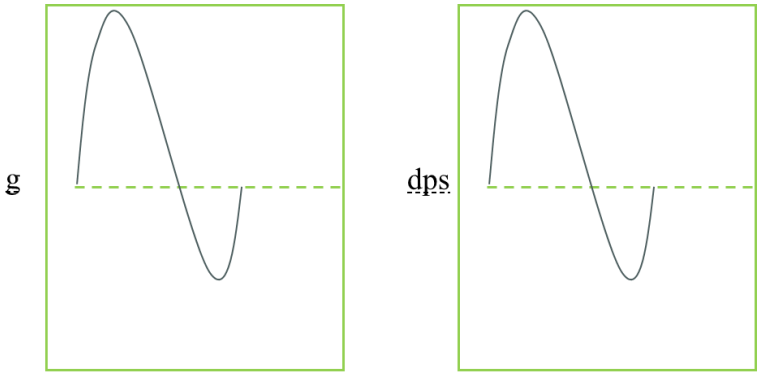


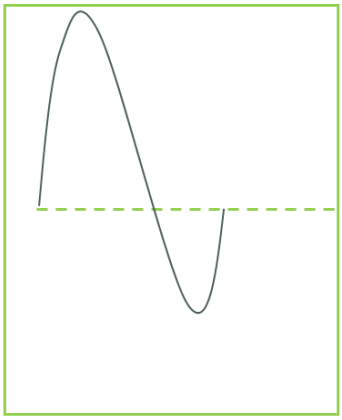
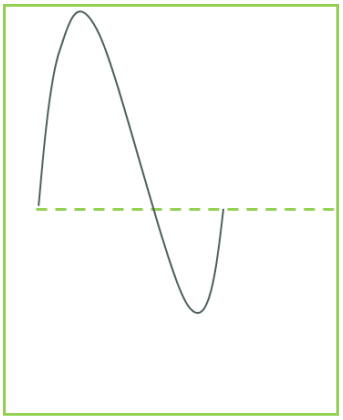
图 4.4 乘坐电梯向下运动状态数据图

表 4.1 类运动的典型特征

运动种类	典型数据特征
站立状态	
乘坐电梯向上	
乘坐电梯向下	

运动种类	典型数据特征
躺下	dps 
坐下	dps 
向前走	g  X轴 g  Y轴
向前跑	g  X轴

运动种类	典型数据特征
向左走	 X轴　　Z轴
向右走	 X轴　　Z轴
步行上楼	 X轴　　Z轴
步行上楼	 X轴　　Z轴

运动种类	典型数据特征
跳跃	g  X轴 dps  Z轴

5 模型建立与求解

5.1 问题一模型建立与求解

5.1.1 建模流程

首先对附件 1 中进行进行数据分析，预先筛选出乘坐电梯向上、站立、乘坐电梯向下等五类易于辨别的类别。然后对剩余数据进行四阶巴特沃斯滤波器滤波、数据分片以及特征提取，接着对特征值进行 PCA 降维后，对其 K-Means 聚类。具体流程如图 5.1 所示。

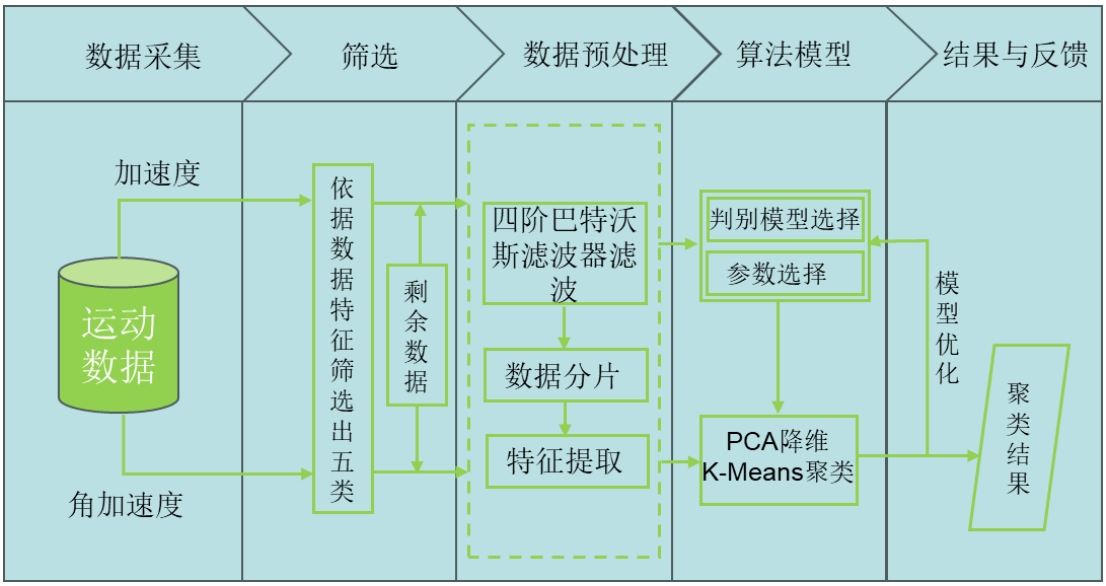


图 5.1 问题一建模流程图

5.1.2 算法介绍

（1）K-Means 算法：

K-Means 算法的工作原理：首先随机从数据集中选取 K 个点，每个点初始地代表每个聚类中心，然后计算剩余每个样本到聚类中心的距离，将它赋予最近的簇，接着重新计算每一簇的平均值，整个过程不断重复，如果相邻两次调整没有明显变化，说明数据聚类形成的簇已经收敛。该算法的一个特点是在每次迭代中都要考察每个样本的分类是否正确。若不正确，就要调整，在全部样本调整后，在修改聚类中心，进行下一次迭代。这个过程将不断重复直到满足某个终止条件，终止条件可以是以下任何一个。

（1）没有对象被重新分配给不同的簇聚类；

（2）聚类中心不再发生变化；

（3）误差平方和局部最小。

K-Mean 算法步骤如图 5.2 所示：

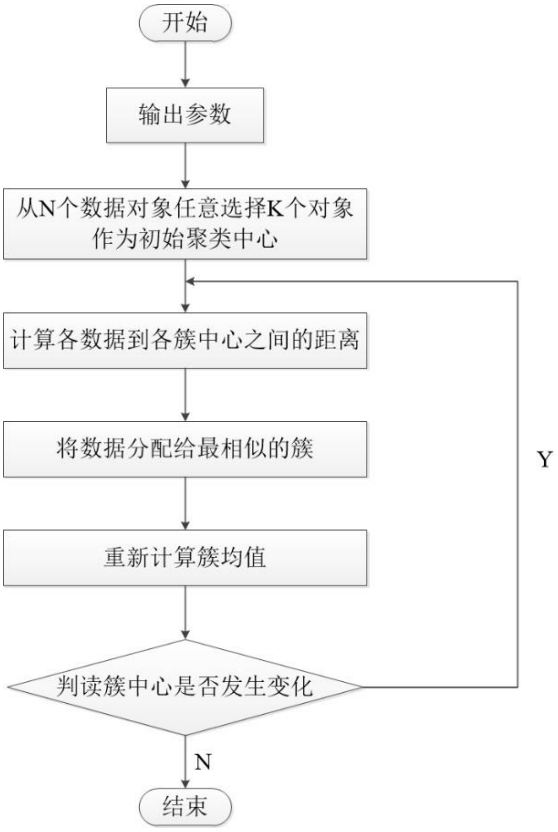


图 5.2 K-Mean 算法流程图

（2）PCA 算法：

主成分分析（PCA）是一种无监督学习的降维技术，其目的是将 N 维数据投影到 K 维空间（其中 0<K<N），以便在新的 K 维特征空间中，各特征向量间的协方差归零，同时特征向量的方差最大化。这种方式可以确保这些向量在尽可能保留原始数据信息的同时，彼此之间不具有线性相关性。具体地，PCA 通过选择 K 个正交的主成分来实现降维，这些主成分基于 N 维特征重构而成。首先，选取的第一个主成分是数据中方差最大的方向，接着第二个主成分则是方差次之且与第一个主成分正交的方向。类似地，第三个及后续的主成分将在与前面主成分都正交的方向中选择方差最大的。这样，大部分数据的方差将被这 K 个主成分所覆盖，剩余的方差几乎可以忽略。通过这种方式，PCA 不仅减少了数据的维度，还能通过计算各主成分的总方差贡献率来评估重要特征，常见的做法是选取累积方差贡献率达到 90%的主成分，以确保重要信息得以保留，同时去除更多的噪声和无关信息，从而使数据更加简洁和有效。PCA 降维的算法步骤如图 5.3 所示：

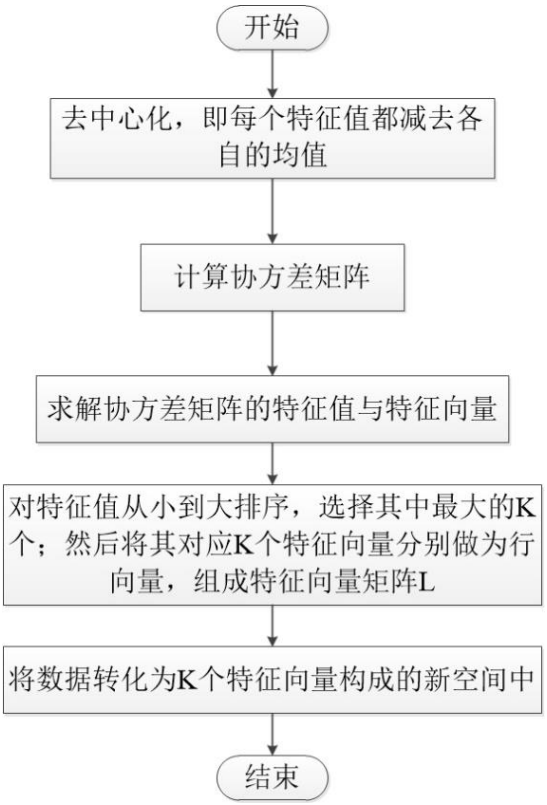
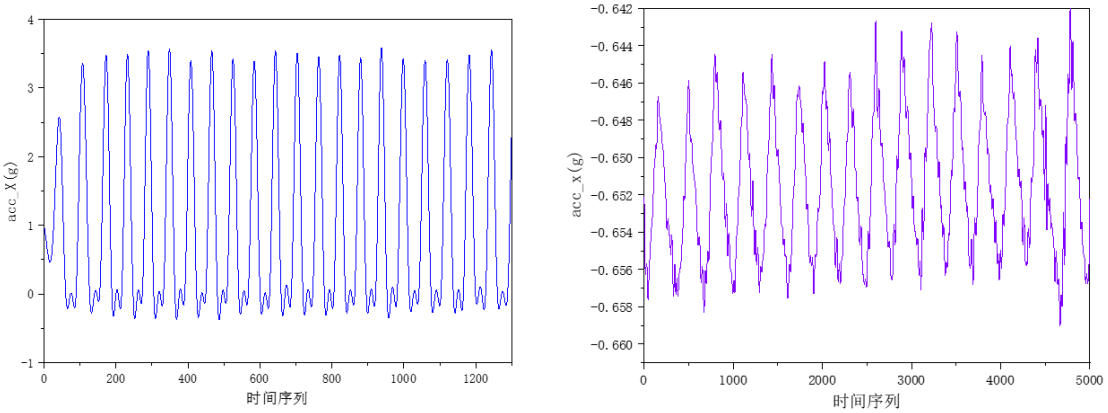


图 5.3 PCA 算法流程图

5.1.3 模型求解

（1）数据预分类

根据表 4 给出的测试者活动状态编码，可以大致判断出测试者的活动状态中 1-7 的运动状态对 X 轴方向加速度波动幅度影响较大，而 8-12 的活动状态在 X 轴方向加速度波动幅度较小，以 Person1 的 SY02 和 SY22 文件中的 X 轴方向加速度数据对比，如下图 5.4 所示：



样本数据 SY02　　样本数据 SY22

图 5.4 样本数据 SY01、SY02 对比图

在图中，样本数据 SY02 对应的 X 轴方向加速度波动幅度差在 4m/s^2，而样本数据 SY02 对应的 X 轴方向加速度波动幅度差小于 0.2m/s^2，综上，根据 X 轴方向加速度波动幅度差波动，可以将 1-7 的活动状态和 8-12 的活动状态进行分离。设定 X 轴方向加速度波动幅度差如果大于 0.3m/s^2，则可以认为测试者处于 1-8 活动状态。综上，把三位测试者的 1-7 活动状态和 8-12 活动状态进行划分，如下图 5.5 所示：

图 5.5 预分类结果图

其中 Person1-1、Person2-1 和 Person3-1 是三位测试者 1-7 活动状态数据文件，Person1-2、Person2-2 和 Person3-2 是三位测试者 8-12 活动状态数据文件。

根据上述预分类方法，使用附件 2 中带有活动状态标签的数据文件进行测试，测试预分类准确率如表 5.1 所示：

表 5.1 附件 2 测试结果准确率

测试者	预分类准确率	测试者	预分类准确率
Person4	100%	Person5	100%
Person6	100%	Person7	100%
Person8	100%	Person9	100%
Person10	96.7%	Person11	93.3%
Person12	100%	Person13	100%

通过带标签的附件 2 的数据文件测试发现，1-7 和 8-12 活动状态的分类准确率在93.3%以上，Person10 和 Person11 误分类了 2 个和 4 个数据，是因为使用数据滤波和清洗未能抑制较大的 X 轴向加速度幅值变化，使用幅值差分类方法并未进行有效的分类。但是总体而言，预分类效果较好，可以进一步对 1-7 和 8-12 活动状态进一步细分。

（2）活动状态数据分类

通过预分类的数据，对 1-7 活动状态数据文件中的六维数据（acc_x,acc_y,acc_z,gyro_x(dps), gyro_y(dps), gyro_z(dps)）数据先进行加速度和角速度合成，计算公式如下：

$$acc_sum = \sqrt{acc_x^2 + acc_y^2 + acc_z^2} \tag{5-1}$$

$$gvro_sum = \sqrt{gvro_x^2 + gvro_y^2 + gvro_z^2} \tag{5-2}$$

运动数据采样频率为 100Hz，在采集的这段时间内，信息量很大，需要进行窗口分割，即给定一个时间序列，以时间点为特征的有限样本集，将样本集划分为两个时间点 A 与 B 之间的连续样本的段(窗口)，这两个时间点内部对于程序来说是齐次的。在进行窗口分割时如何确定分割窗口是一个关键性问题。如果窗口太长，数据分片过大，有可能会重叠两个不相关的活动；如果过小则滑动窗口不能包含一个完整的活动状态。对于 1-7 活动状态而言，可以选择 200 的滑动窗口，同时每次窗口的分割会对前一个时间窗口有 50%的叠加率，这样保证每个动作有更好的完整性。

（3）特征提取与选择

运动状态识别可以提取的特征有很多，主要分为时域和频域特征，时域特征（Time Domain Features, TDF）主要是指在信号随时间变化的过程中，所具有的与时间相关的特征；频域特征（Frequency Domain Features, FDF）主要是被用来发现信号中的一些具有周期性的信号，频域主要用快速傅里叶变换（Fast Fourier Transform, FFT）来计算，以合成加速度为例，实验一在时域特征，使用了均值、标准差、最大值、最小值、高度、峰峰值、众数、过均值点个数；在频域特征中，使用了 FFT 振幅、FFT 频谱中零频率处的直流分类、FFT 频谱中的显著频率及对应的幅度、FFT 变换后的能量，以及功率谱的偏度和峰度，如表 5.2 所示。

表 5.2 相关特征值含义

编号	特征	计算公式
1	均值	一个窗口采样的平均值

编号	特征	计算公式
2	标准差	一个窗口采样的标准差
3	最大值	一个窗口采样的最小值
4	最小值	一个窗口采样的最大值
5	众数	出现频数最多的数
6	高度	一个窗口采样的最大值减去最小值
7	过均值点数	一个窗口内超过均值点的个数
8	直流分量 dc	FFT 变换后的第一个分量
9	谱峰位置 P1,p2,p3	FFT 变换后的前 5 个峰值
10	频率 fre	对应 5 个峰值的频率
11	能量 Energy	FFT 变换后的能量
12	四种幅度统计特征	平均值，标准差 std，偏度，峰度

对于给定的一组信号： $Y = \{y1, \ldots, yn\}$；经过 FFT 变换，其中 Fi 是 Y 的傅里叶变换的第 i 个分量，其中各个特征的计算方法如下：

均值：

$$mean = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{5-3}$$

标准差：

$$sid = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - mean)^2} \tag{5-4}$$

最大值：

$$max = max(y_i), i \in \{1,2,\ldots,n\} \tag{5-5}$$

最小值：

$$min = min(y_i), i \in \{1,2,\ldots,n\} \tag{5-6}$$

高度：

$$height = |max - min| \tag{5-7}$$

过均值点数：

$$cross\_mean = \begin{cases} i+1; (y_i > mean, 0 \le i \le n) \\ i, (y_i < mean) \end{cases} \tag{5-8}$$

能量：

$$Energy(Y) = \frac{\sum_{i=1}^{n} F_i^2}{n} \tag{5-9}$$

幅度统计特征，均值u_{amp}，标准差σ_{amp}，偏度γ_{amp}，峰度η_{amp}计算方法如下：

$$u_{amp} = \frac{1}{M}\sum_{i=1}^{M} D(i) \tag{5-10}$$

$$\sigma_{amp} = \sqrt{\frac{1}{M}\sum_{i=1}^{M} [D(i) - u_{amp}]^2} \tag{5-11}$$

$$\gamma_{amp} = \frac{1}{M}\sum_{i=1}^{M} [\frac{D(i)-u_{amp}}{\sigma_{amp}}]^3 \tag{5-12}$$

$$\eta_{amp} = \frac{1}{M}\sum_{i=1}^{M} [\frac{D(i)-u_{amp}}{\sigma_{amp}}]^4 - 3 \tag{5-13}$$

（4）PCA 数据特征降维

经过 PCA 降维后，前 8 个主成分的累积贡献率为 90.15%，前 11 个主成分的累积贡献率为 95.08%，前 17 个主成分的累积贡献率为 99.00%。如图 5.6 所示：

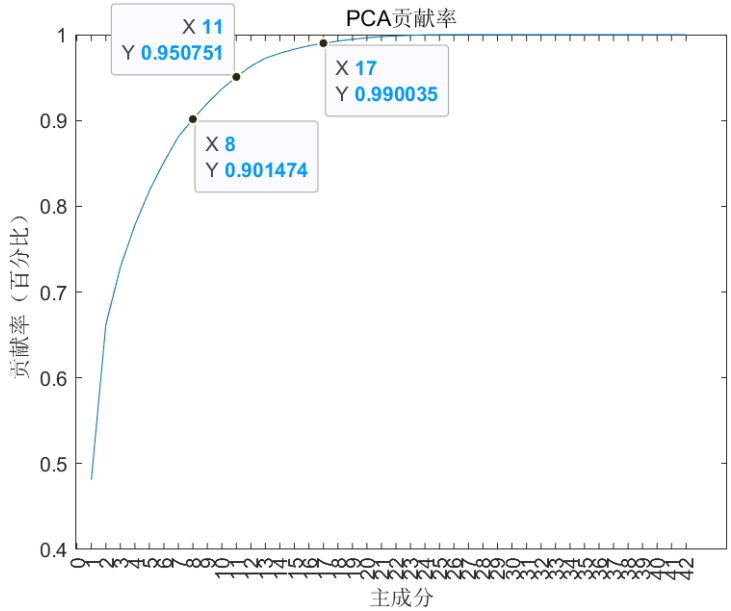


图 5.6 PCA 降维成分贡献率图

这里选择 11 个主成分进行数据特征降维，并使用 K-means 聚类算法进行聚类，采用附件 2 中的测试者 Person4-Person13 中的 1-7 的数据进行聚类测试，聚类数目为 7，将超维数据使用 t-SNE 进行降维可视化，观察数据降维至三维空间后的聚类情况，如下图 5.7 所示：

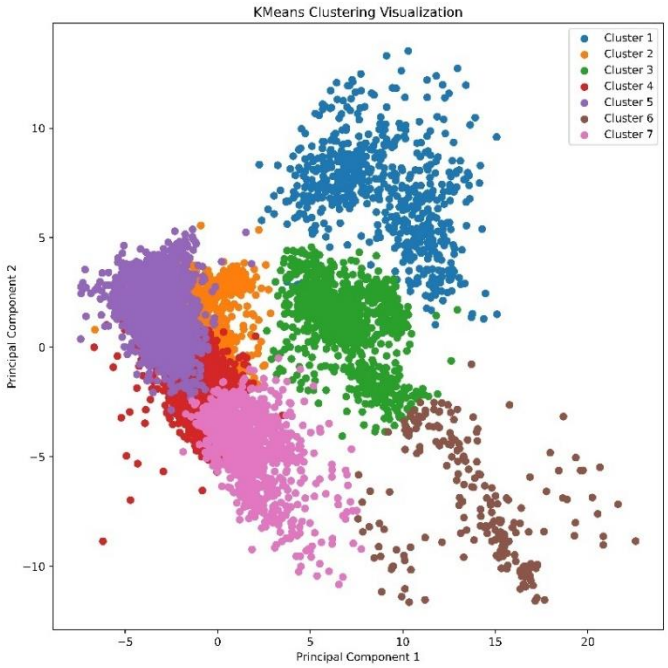


图 5.7 PCA 降维效果图

从图中可以看出，有部分数据产生了误分类问题，并统计其聚类后各类别中带有标签活动状态的分布情况，如表 5.3 所示：

表 5.3 聚类结果

标签标号	聚类 1	聚类 2	聚类 3	聚类 4	聚类 5	聚类 6	聚类 7	总共
标签 1	29	5	0	12	4	0	0	50
标签 2	20	48	0	6	6	0	0	50
标签 3	14	24	0	6	6	0	0	50
标签 4	10	1	1	29	9	0	0	50
标签 5	25	0	0	22	1	2	0	50
标签 6	5	0	11	0	0	34	0	50
标签 7	0	0	0	5	0	1	44	50

根据附件 1，采用来自测试者 Person1-Person3 的 1-7 活动状态数据，先根据时域和频域特征得到上述 42 维度的数据特征。然后再使用 PCA 降维，降维后前 8 个主成分的累积贡献率为 90.82%，前 11 个主成分的累积贡献率为 95.17%，前 17 个主成分的累积贡献率为 99.13%，具体如图 5.8 所示。

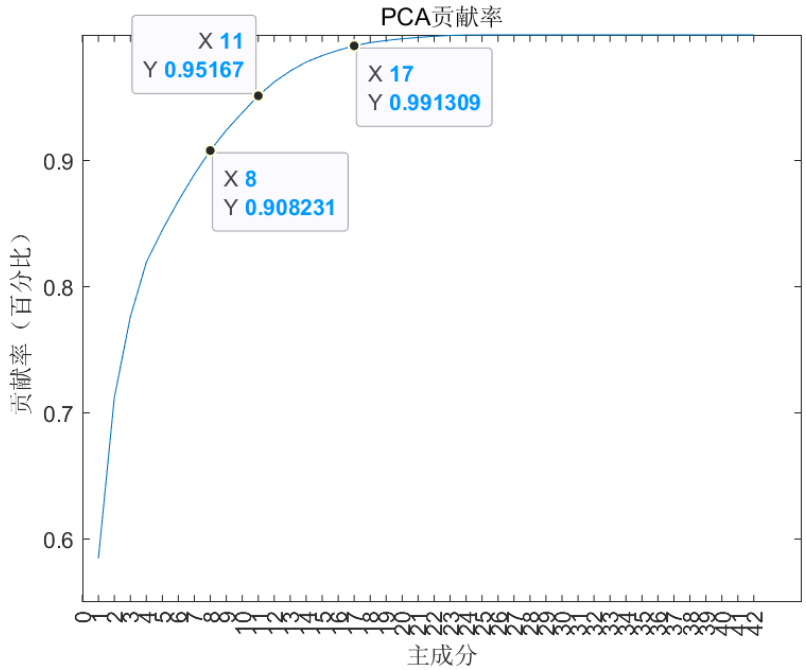


图 5.8 PCA 降维成分贡献率图

这里选择 11 个主成分进行数据特征降维，并使用 K-means 聚类算法进行聚类，聚类结果如图 5.9 所示：

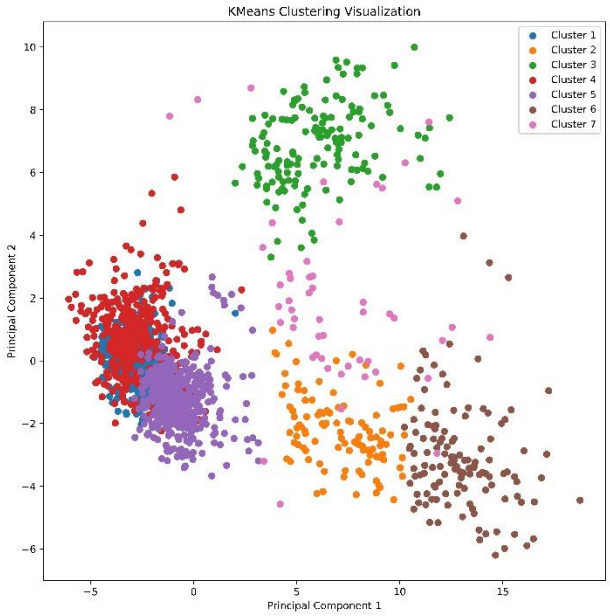


图 5.9 PCA 降维效果图

附件 1 的 1-7 活动状态聚类结果如 5.4 表所示：

表 5.4 分类结果

类	Person1	Person2	Person3
1 类	SY5/SY19/SY28/SY41/SY42/SY50/	SY2/SY10/SY20/SY26/SY38/SY47/SY51	SY9/SY13/SY14/SY23/SY27/SY29/SY37/SY39/SY42/SY43/SY51
2 类	SY7/SY8/SY14/SY25/SY33/SY38/SY52/SY53/SY56	SY12/SY21/SY40/SY60	SY30/SY52/
3 类	SY17/SY43/SY46/SY47/SY58/SY59	SY16/SY29/SY43/	SY56/
4 类	SY21/SY37/SY48/SY49	SY3/SY7/SY19/SY23/SY24/SY30/SY33/SY37/SY44/SY46/	SY11/SY19/SY24/SY25/SY31/SY35/SY36/SY44/SY49/SY60
5 类	SY27/SY29/SY31		SY8/
6 类	SY26/SY44	SY1/SY4/SY27/SY48/SY49/SY50/SY57	SY4/SY10/SY18/SY22/SY57/SY58/SY59
7 类	SY1/SY2/SY15/SY23/SY36/	SY13/SY28/SY34/SY42/	SY5/SY41/SY53

5.2 问题二模型建立与求解

5.2.1 建模流程

首先对附件 2 中 Person 4-13 总共 10 个人员加速度和角加速度的特点进行分析，主要分析采集数据中能够表征运动类型的典型特征，根据分析结果对数据进行预处理，包括数据滤波、数据切片、数据时-频特征提取三个主要步骤。之后，构建运动数据的判别模型。由于人类活动的复杂性，通过数据判别人类运动类型需要深度挖掘三轴监测数据在连续时间上的规律和不同轴之间的关联关系。为此，建模过程中采用能够深度挖掘数据规律的人工神经网络作为判别模型，预期通过模型构建提取的时频特征与运动类型的映射关系。完成模型构建后，对模型进行训练和测试评估，选择性能最优的模型作为最终的运动判别模型。具体流程如图 5.10 所示。

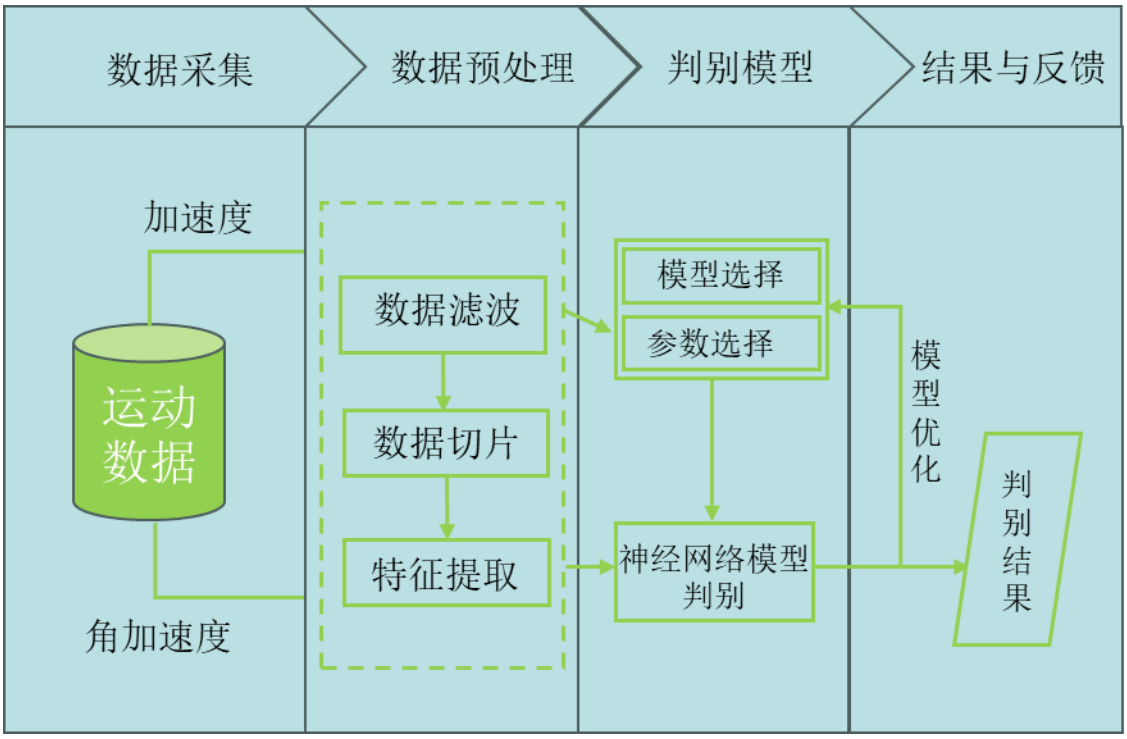


图 5.10 问题二建模流程图

5.2.2 特征值计算

按照表 5.1 以及相关的计算公式[7]可以得到不同人员在不同运动状态下的均值、标准差、最大值、最小值、众数、高度、过均值点数等特征值。

5.2.3 算法介绍

多层感知机神经网络（Multilayer Perceptron， MLP）的网络结构如图所示，是一种具有三层或者三层以上神经元的神经网络,包括输入层、中间层(隐含层)和输出层,上下层之间实现全连接，而同一层的神经元之间无连接，输人层神经元与隐含层神经元之间的是网络的权值，即两个神经元之间的连接强度，隐含层或输出层任一神经元将前层所有神经元传来的信息进行整合，通常还会在整合过的信息中添加一个阈值,这主要是模仿生物学中神经元必须达到一定的阈值才会触发的原理，然后将整合过的信息作为该层神经元输入。当一对学习样本提供给输入神经元后,神经元的激活值(该层神经元输出值)从输入层经过各隐含层向输出层传播，在输出层的各神经元获得网络的输入响应,然后按照减少网络输出与实际输出样本之间误差的方向，从输出层反向经过各隐含层回到输入层,从而逐步修正输出各连接权值，这种算法称为“误差反向传播算法”，即 Back Propagation（BP）算法。随着这种误差逆向传播修正的反复进行，网络对输入模式响应的正确率也不断上升。BP 算法的核心是数学中的“负梯度下降”理论，即 BP 网络的误差调整方向总是沿着误差下降最快的方向进行。

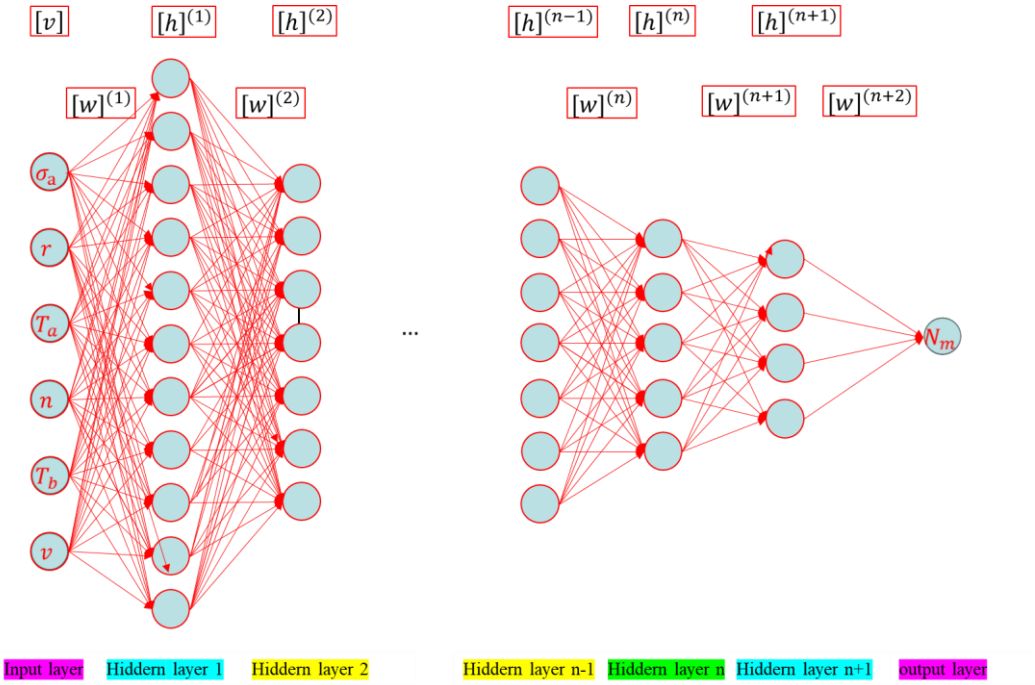


图 5.11 神经网络结构图

“负梯度下降”理论的核心思想是，通过迭代地调整参数，沿着函数的负梯度方向寻找函数的局部最小值。这里的“梯度”是损失函数对模型参数的偏导数的向量，指向损失函数增长最快的方向；而“负梯度”则是梯度的反方向，即损失函数减少最快的方向，其数学表达为：

$$\theta := \theta - \eta \cdot \nabla_\theta L(\theta)$$

$L(\theta)$为损失函数；θ是模型的参数；η是学习率，其一个正的常数，决定了参数更新的步长。$\nabla_\theta L(\theta)$是损失函数L对参数θ的梯度。

5.2.4 模型求解

在本题提供的附件 2 中所有人员的数据经预处理后共得到 27424 个样本，114 个特征，所有特征均进行归一化处理。一行表示一个样本，每一列表示一个特征，这就构成了用于运动状态识别的数据集。

依据上述模型，判别模型采用 python 编程语言和 pytorch 神经网络框架编写了具有三个全连接层的多层感知机，其网络结构表 5.5 所示：

表 5.5 神经网络结构

层级	类型	输入维度	输出维度	激活函数
输入层	输入	114	256	ReLU

隐藏层	全连接	256	128	ReLU
输出层	全连接	128	12	
输出层	全连接	128	12	

每个样本通过神经网络计算后，得到 12 个元素的一维向量，与 12 个运动类型一致。神经网络采用交叉熵损失函数计算损失，使用 Adam 优化器进行参数更新，学习率设定为 1e-3，200 轮训练后稳定收敛。在模型评估采用十折交叉验证，其中训练集和测试集的样本量比例为 9：1，每次训练和评估均随机抽样训练集和测试集，重复 10 次。模型性能评估指标包括准确率（accuracy），精确率（precision），召回率（recall）和 F1 score。十折交叉验证后得到 10 个不同样本训练和测试的模型，性能评估结果如表 5.6 所示：

表 5.6 性能评估结果

指标 \ 模型	1	2	3	4	5	6	7	8	9	**10**
准确率	0.9864	0.9809	0.9863	0.9751	0.9861	0.9884	0.9836	0.9783	0.9869	0.9895
精确率	0.9824	0.9754	0.9817	0.9713	0.9807	0.9846	0.9784	0.9732	0.9824	0.9865
召回率	0.9827	0.9768	0.9817	0.9702	0.9811	0.9840	0.9773	0.9709	0.9826	0.9861
F1 score	0.9826	0.9757	0.9816	0.9706	0.9809	0.9843	0.9773	0.9717	0.9825	0.9863

根据表中结果所示，十折交叉验证过程中各个模型均取得了较高的识别精度，其平均准确率均大于 0.98，其中模型 10 取得了最高精度，如图 5.12 为该模型对各类运动的识别准确率。

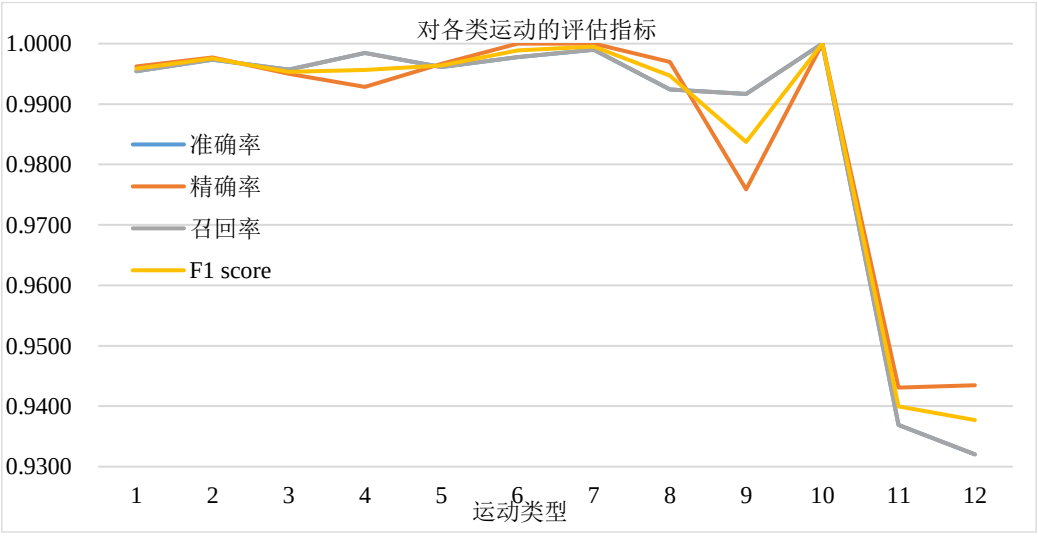


图 5.12 识别准确率

根据表中结果所示，模型 10 对 12 个中的 9 个运动类型的判别精度高于 99%，对第 9、11、12 类运动的判别精度相对较低，但也达到了 93%以上，如图 5.13 为该模型通过测试数据评估得到的混淆矩阵：

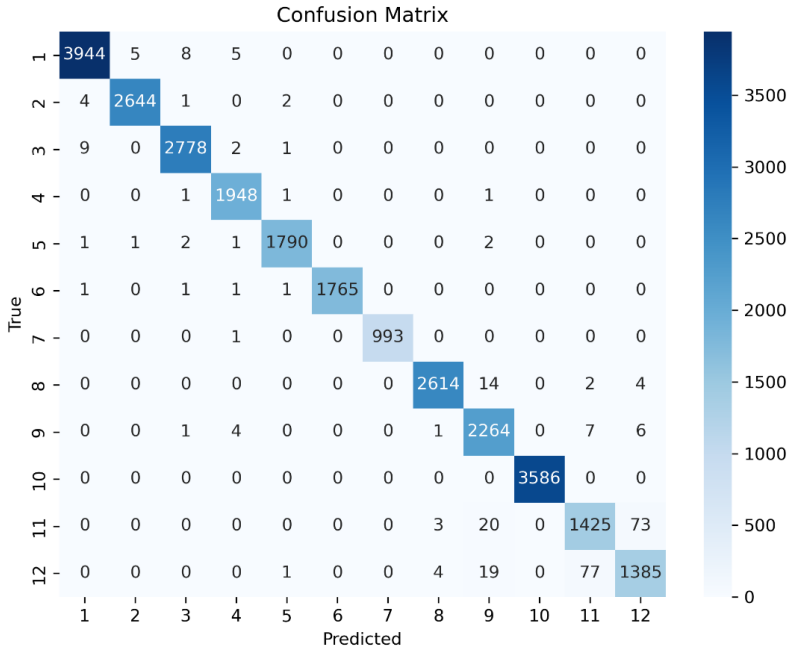


图 5.13 混淆矩阵

从混淆矩阵中可以看出，该判别模型对运动类型 8（坐下）、11（乘坐电梯向上运动）、12（乘坐电梯向下运动）存在相对较多的误判，其中，坐下的 2634 个样本中 14 个被识别为站立，乘坐电梯向上运动的 1521 个样本中 20 个被错误识别为站立，73 个被识别为乘坐电梯向下运动。乘坐电梯向下运动的 1486 个样本中 19 个被错误识别为站立，77 个被识别为乘坐电梯向上运动。事实上，上述样本被错误分类的主要原因在于站立、坐下、乘坐电梯上下运动数据中切分的数据切片存在较高的相似性，如下图举例 Person4 数据中上述三种运动的 X 轴加速度曲线：

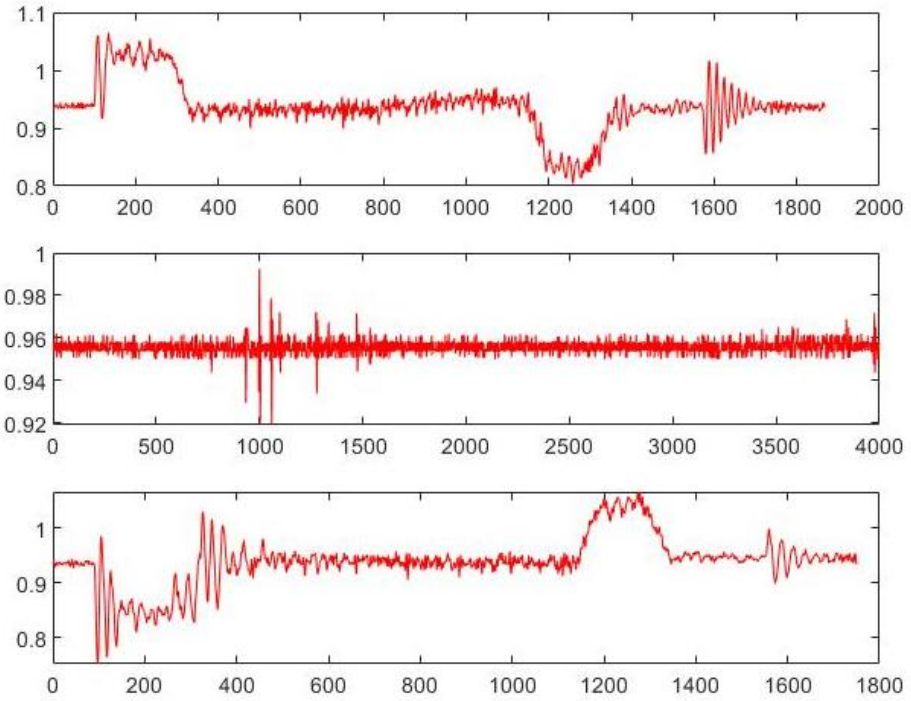


图 5.14 加速度曲线

从图 5.14 中可以看出，坐下和站立的很多数据片段高度相似，而乘坐电梯上下行的数据中在电梯平稳运行时的数据切片也非常相似，并且都和站立的数据片段相似。因此，错误区分的主要问题在于不同类型运动的数据切片存在相似性，数据切片的选取方式还有待进一步改进。

对于问题 2（2），将附件 3 中所有的数据按照本节所述流程依次进行预处理，然后将特征数据输入效果较好的模型 10 中，由于题中要求判别整个文件的状态，而每个文件又在预处理后又提取为多个特征样本，这些样本特征可能被判别模型识别为不同的运动类型。因此，在对文件判别的过程中，使用投票机制确定文件最终的运动类型。各文件中提取得到的特征样本判别结果如表 5.6（(与该文件相关的完整统计信息见附件 q2_2_result.xlsx 中)所示：

表 5.6 附件 3 数据判别结果

活动类型	判别状态
SY1	5
SY2	1
SY3	7

活动类型	判别状态
SY4	9
SY5	7
SY6	10
SY7	2
SY8	6
SY9	7
SY10	10
SY11	9
SY12	7
SY13	4
SY14	3
SY15	4
SY16	1
SY17	4
SY18	5
SY19	8
SY20	8
SY21	6
SY22	2
SY23	8
SY24	5
SY25	2
SY26	9
SY27	8
SY28	5

活动类型	判别状态
SY29	6
SY30	5

5.3 问题三模型建立

5.3.1 建模流程

首先计算了 person 4-13 总共 10 个人员加速度数据进行重力加速度消除以及光滑处理后，计算其均值、标准差、最大值、最小值、众数、高度、过均值点数七个特征值；随后，进行神经网络模型训练，即通过输入不同人员不同测试数据的七个特征值，训练得到人员身高年龄、身高以及体重；然后，计算 person1-3 的七个特征值，并输入到已经训练好的神经网络模型，预测得到 person1-3 共 3 名人员的年龄、身高和体重，并进行准确度评估。最后，再计算出附件 5 数据的七个特征值，输入到已经训练好的神经网络模型，得到预测的年龄、身高以及体重。构建支持向量机模型（SVM），根据附件 5 预测得到的年龄、身高和体重判别其最可能来源于问题 2 中哪一名实验人员，具体流程如图 5.15 所示。

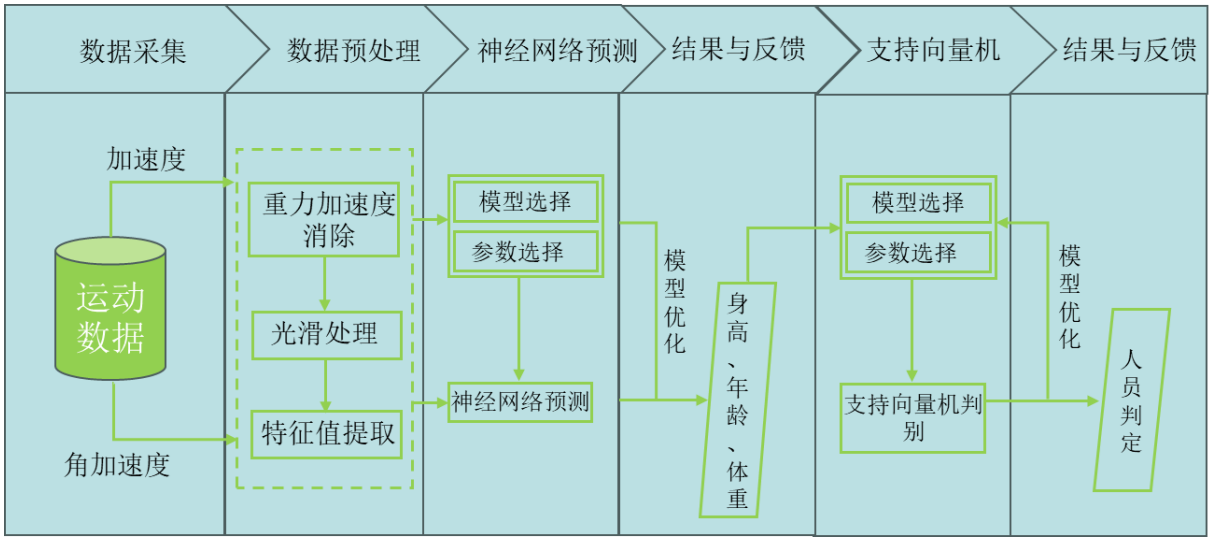


图 5.15 问题三建模流程图

5.3.2 特征值计算

按照表 5.7 中的计算公式依次计算不同人员在不同运动状态下的均值、标准差、最大值、最小值、众数、高度、过均值点数七个特征值。

表 5.7

编号	特征	计算公式
1	均值	$mean = \frac{1}{n}\sum_{i=1}^{n} y_i$
2	标准差	$sid = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - mean)^2}$
3	最大值	$max = max(y_i), i \in \{1,2,\ldots,n\}$
4	最小值	$min = min(y_i), i \in \{1,2,\ldots,n\}$
5	众数	出现频数最多的数
6	高度	$height = \lvert max - min \rvert$
7	过均值点数	$cross\_mean = \begin{cases} i+1; (y_i > mean, 0 \le i \le n) \\ i, (y_i < mean) \end{cases}$

根据表计算得到的部分特征值，如图 5.16 所示。

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	-0.6935	-0.7066	-0.7218	-0.7211	-0.6743	0.9370	0.9331	0.9290	0.9176	0.9266	0.9389	0.9283	0.9243
2	0.0334	0.0415	0.0070	0.0050	0.0343	0.0425	0.0458	0.0411	0.0458	0.0385	0.0456	0.0444	0.0407
3	-0.5836	-0.2578	-0.6527	-0.7069	-0.4527	1.0650	1.0882	1.0964	1.0548	1.0470	1.0666	1.0410	1.0402
4	-0.8176	-0.8687	-0.7877	-0.7361	-0.8072	0.8042	0.7444	0.7995	0.7961	0.8100	0.7521	0.7857	0.7885
5	-0.8176	-0.8687	-0.7877	-0.7361	-0.8072	0.8042	0.7444	0.7995	0.7961	0.8100	0.7521	0.7857	0.7885
6	0.2340	0.6110	0.1350	0.0292	0.3545	0.2608	0.3438	0.2969	0.2587	0.2370	0.3145	0.2553	0.2517
7	1890	1579	2633	2269	3583	896	1165	984	1085	945	967	980	894

图 5.16 部分特征值

5.3.3 算法介绍

支持向量机（SVM）：

支持向量机（SVM）的数学理论基础是围绕着优化问题的构建和解决。以下是 SVM 数学理论的几个关键点：

线性可分情况：

对于线性可分的数据，SVM 的目标是找到一个线性超平面，该超平面能够最大化不同类别数据点之间的间隔。

间隔最大化：

间隔是数据点到超平面的最短距离。SVM 试图找到一个超平面，使得这个间隔最大化。对于线性超平面，间隔可以表示为：

$$Margin = \frac{2}{\| w \|}$$

其中，w是超平面的法向量。

线性分类：

SVM 的决策函数可以表示为：

$$f(x) = w \cdot x + b$$

其中，x是特征向量，w是权重向量，b是偏置项。

优化问题：

SVM 的优化问题可以表述为一个凸二次规划问题，其目标是最小化以下目标函数：

$$\min_{w,b} \frac{1}{2} \| w \|^2 \text{ 受约束: } y_i(w \cdot x_i + b) \geq 1, \forall i$$

其中，y_i是第i个样本的类别标签。

软间隔与正则化：

为了处理非线性可分的数据，SVM 引入了软间隔的概念，允许一些数据点违反间隔规则。这通过引入松弛变量ξ_i来实现，优化问题变为：

$\min_{w,b,\xi} \left(\frac{1}{2} \| w \|^2 + C \sum_{i=1}^{n} \xi_i \right)$受约束：$y_i(w \cdot x_i + b) > 1 - \xi_i, \xi_i > 0, \forall i$

其中C是正则化参数，控制着分类误差和间隔宽度之间的权衡。

核技巧：

对于非线性问题，SVM 使用核函数将数据映射到高维空间，以在高维空间中寻找线性可分的超平面。常用的核函数包括：

线性核：$K(x_i, x_j) = x_i \cdot x_j$

多项式核：$K(x_i, x_j) = (\gamma x_i \cdot x_j + r)^d$

径向基函数（RBF）核：$K(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|^2)$

Sigmoid 核：$K(x_i, x_j) = \tanh(\gamma x_i \cdot x_j + r)$

对偶问题：

SVM 的优化问题通常通过拉格朗日乘子法转换为对偶问题来求解，这有助于简化计算并引入核技巧。

拉格朗日乘子：

在对偶问题中，拉格朗日乘子α_i用于表示每个样本对最终决策函数的贡献，只有满足$0 < \alpha_i < C$的样本对应的数据点才是支持向量。

决策函数：

最终的决策函数可以表示为：

$$f(x) = \sum_{i-1}^{n} \alpha_i y_i K(x_i, x) + b$$

其中，$K(x_i, x)$是核函数，α_i和y_i分别是拉格朗日乘子和样本类别标签。

5.3.4 模型求解

根据前述思路，计算得到向前走状态下 X 轴加速度均值与年龄、身高以及体重之间的关系，从图 5.17 中可以看到，X 轴加速度与年龄、身高以及体重之间的变化并无明显的规律性，这可能跟两方面原因有关，其一，与我们所选的特征值—均值有关，也就是说均值并不能反应年龄、身高以及体重对运动数据的影响；其二，运动数据的变化还与人员的身体素质有关，因此，在身体素质因素影响下，运动数据的变化与年龄、身高以及体重之间无明显规律性，但从图依旧可以发现实验人员的年龄、身高、体重对活动状态数据存在影响。在此，我们提供了分析思路，可以根据此进一步分析年龄、身高、体重对其它特征值得影响规律。

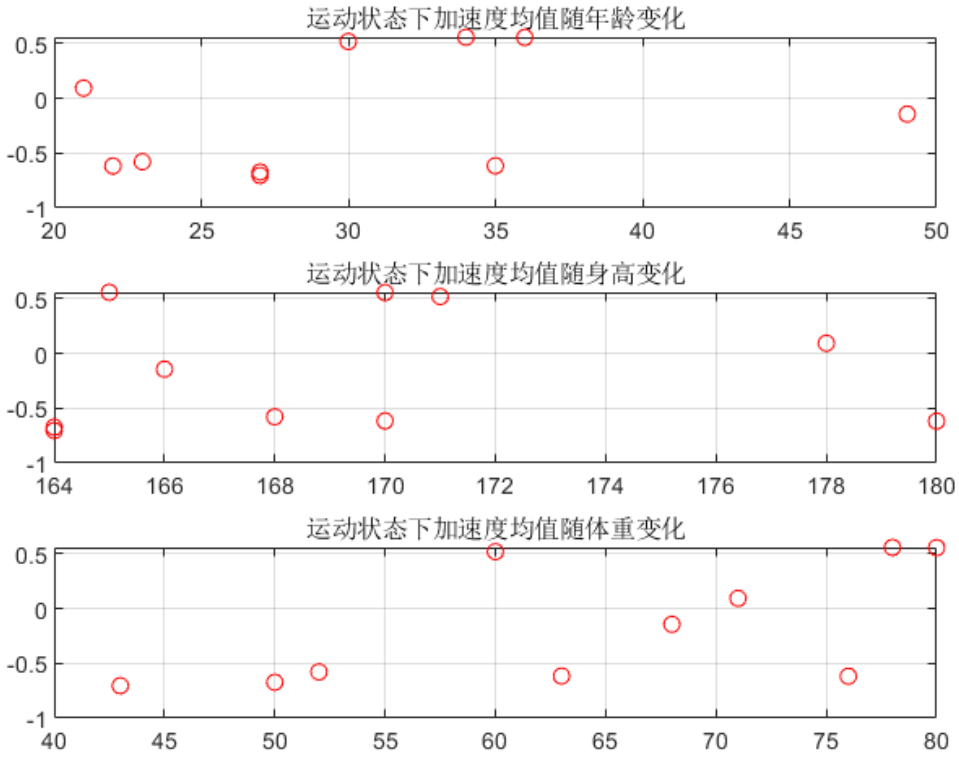


图 5.17 加速度随年龄、身高和体重变化图

依据相关模型，通过 matlab 编程首先计算得到同一人员的运动数据经由神经网络预测得到的预测值与实际值对比图，如图 5.18 所示。求均值后得到验证人员 person1 的年龄、身高、体重分别为 31，171 和 69，与实际的误差值分别为 19%、7.6%和 8.1%；验证人员 person2 的年龄、身高、体重分别为 30，171 和 68，与实际的误差值分别为 3.6%、1.1%和 0.8%；验证人员 person3 的年龄、身高、体重分别为 30，171，66，与实际的误差值分别为 6.4%、6.6%和 11.9%。由此，可见，该模型人员特征预测效果较好。输入附件 5 的运动数据，得到预测的年龄、身高以及体重，如表 5.8 所示。

构建支持向量机模型（SVM），根据附件 5 预测得到的年龄、身高和体重判别其最可能来源于问题 2 中哪一名实验人员；同时，对向量机模型进行较差验证，计算泛化误差，其值为 0.53。

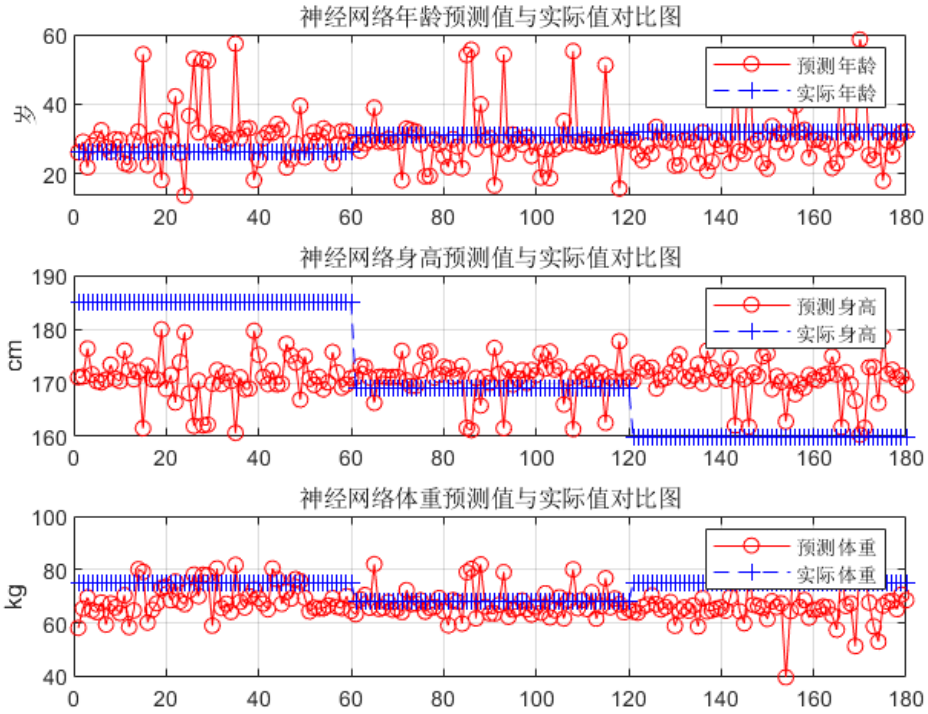


图 5.18 神经网络预测得到的预测值与实际值对比图

表 5.8 计算结果

活动类型	预测结果			判别结果
	年龄	身高	体重	
Unknow1	29	170	62	Person7
Unknow2	26	171	55	Person5

活动类型	预测结果			判别结果
	年龄	身高	体重	
Unknow3	26	171	57	Person7
Unknow4	25	172	59	Person5
Unknow5	28	171	67	Person8

6 模型评价与改进

6.1 模型的优点

（1）模型充分结合实际, 简化不同位置重力加速度分量变化对模型建立的影响，同时，在模型建立的过程中，充分考虑了运动数据的诸多特征，包括时域和频域特征，如: 均值，标准差等，这样得到的模型贴合实际, 具有较高的应用价值, 可以推广到声音识别等领域；

（2）模型运用神经网络和支持向量机分类思想，将复杂的利用运动数据刻画试验人员画像问题转化为由运动数据的特征数据预测试验人员年龄、身高以及体重问题并合理优化模型的参数合理设置参数，模型的输出结果符合题目要求，能解决实际问题；

本文使用的神经网络算法具有强大的非线性映射、泛化、自动特征提取等能力，特别适用于预测等问题；同时，本文建模思路能够广泛推广到其它领域，如声音信息识别、车辆运行状态识别等领域。

6.2 模型的不足

实际应用中，在不同位置的重力加速度分量不同，其对运动加速度存在影响，本文未考虑此因素的影响，一定程度上影响了模型的准确性；其次，在试验人员年龄、身高、体重预测中，未考虑到频域特征，这会对模型的精确度产生影响；此外，对于不等长数据可以进一步研究变长度数据特征提取方法以解决不同类型运动数据片段切分相似的问题。

6.3 模型的推广

在机器运维维修方面，可以将运动数据替换成机器特征部位的振动、噪声等数据，从而解决机械故障识别、判定问题；然后进行有针对性的维修、保养，减少人员故障排除成本。

总之，本文中涉及到的建模思路在各个领域的状态识别，如故障模式、车辆运行模型（加速、减速、左拐弯等），均可推广应用。

参考文献

[1]彭际群.基于加速度传感器的人体运动状态识别研究[D].哈尔滨工业大学,2014.

[2]周博翔.基于加速度传感器的人体运动姿态识别[D].长沙理工大学,2014.

[3]胡志亮.基于加速度传感器的人体运动状态识别研究[D].哈尔滨工业大学,2016.

[4]李锋,潘敬奎.基于三轴加速度传感器的人体运动识别[J].计算机研究与发展,2016,53(03):621-631.

[5]雷林平.基于 Savitzky-Golay 算法的曲线平滑去噪[J].电脑与信息技术,2014,22(05):30-31.DOI:10.19414/j.cnki.1005-1228.2014.05.011.

[6]卢锦芳,楼京俊,刘树勇,等.基于巴特沃斯滤波器的近场声全息滤波方法[J].舰船科学技术,2023,45(24):160-165.

[7]郭贵昌.穿戴式运动状态识别及心率监测的研究[D].贵州大学,2019.

附　录

附录 A: 数据预处理相关 matlab 程序

```
%%根据站立不动状态求取重力加速度 X、Y、Z 轴三个方向的分量
clc;
clear all;
close all;
%  数据读取
data=xlsread('a9t1.xlsx');
data1=data(:,1);%X 轴
data2=data(:,2);%Y 轴
data3=data(:,3);%Z 轴
data1_mean=mean(data1);
data2_mean=mean(data2);
data3_mean=mean(data3);
PointNumber=length(data1); %  信号点数
%%  识别和处理异常值
Q1=quantile(data1, 0.25);
Q3 = quantile(data1, 0.75);
IQR = Q3 - Q1;
lower_bound = Q1 - 1.5 * IQR;
upper_bound = Q3 + 1.5 * IQR;
data1(data1 < lower_bound | data1 > upper_bound) = NaN;
data1(isnan(data1))=data1_mean;

Q1=quantile(data2, 0.25);
Q3 = quantile(data2, 0.75);
IQR = Q3 - Q1;
lower_bound = Q1 - 1.5 * IQR;
upper_bound = Q3 + 1.5 * IQR;
data2(data2 < lower_bound | data2 > upper_bound) = NaN;
data2(isnan(data2))=data2_mean;

Q1=quantile(data3, 0.25);
Q3 = quantile(data3, 0.75);
IQR = Q3 - Q1;
lower_bound = Q1 - 1.5 * IQR;
upper_bound = Q3 + 1.5 * IQR;
data3(data3 < lower_bound | data3 > upper_bound) = NaN;
```

```
data3(isnan(data3))=data3_mean;

%% 初始化滤波器状态,对 X 轴加速度进行处理
y_last1 =data1(1);
output1 = zeros(PointNumber,1);
% 滤波系数
a=0.8;
% 滤波器实现
for i = 1:PointNumber
    x_now1= data1(i);
    y_now1= (1-a)*y_last1+a*x_now1;
    output1(i,1) = y_now1;
    y_last1= y_now1;
end
zhonglijiasudu_X=mean(output1);

% 初始化滤波器状态,对 Y 轴加速度进行处理
y_last2 =data2(1);
output2 = zeros(PointNumber,1);
% 滤波系数
a=0.8;
% 滤波器实现
for i = 1:PointNumber
    x_now2 = data2(i);
    y_now2= (1-a)*y_last2+a*x_now2;
    output2(i,1) = y_now2;
    y_last2 = y_now2;
end
zhonglijiasudu_Y=mean(output2);

% 初始化滤波器状态,对 Z 轴加速度进行处理
y_last3 =data3(1);
output3 = zeros(PointNumber,1);
% 滤波系数
a=0.8;
% 滤波器实现
for i = 1:PointNumber
    x_now3 = data3(i);
    y_now3= (1-a)*y_last3+a*x_now3;
    output3(i,1) = y_now3;
```

```
        y_last3 = y_now3;
end
zhonglijiasudu_Z=mean(output3);
%% 对数据进行重力加速度消除
% 数据读取
data=xlsread('a11t1.xlsx');
data1=data(:,1);%X 轴
data2=data(:,2);%Y 轴
data3=data(:,3);%Z 轴
data1_mean=mean(data1);
data2_mean=mean(data2);
data3_mean=mean(data3);
PointNumber=length(data1); % 信号点数

%%重力加速度消除
chuli_data1=data1-zhonglijiasudu_X;
chuli_data2=data2-zhonglijiasudu_Y;
chuli_data3=data3-zhonglijiasudu_Z;
% 绘图
%分解前加速度
figure(1)
subplot(2,1,1)
plot(1:PointNumber,data1, '-r');
hold on
plot(1:PointNumber,data2, '-b');
hold on
plot(1:PointNumber,data3, '-g');
ylabel('g');
legend('X 轴加速度','Y 轴加速度','Z 轴加速度');
title('重力加速度消除前 X、Y、Z 轴的加速度');
grid on
%分解后加速度
subplot(2,1,2)
plot(1:PointNumber,chuli_data1, '-r');
hold on
plot(1:PointNumber,chuli_data2, '-b');
hold on
plot(1:PointNumber,chuli_data3, '-g');
ylabel('g');
legend('X 轴加速度','Y 轴加速度','Z 轴加速度');
```

```
title('重力加速度消除后 X、Y、Z 轴的加速度');
grid on
%%两次 3 点均值平滑处理
signal = chuli_data1; %  对重力加速度消除后的 X 轴加速度进行平滑处理
%  调用函数进行两次 3 点均值平滑
smoothedSignal1=twice3PointMeanSmoothing(signal);
signal=chuli_data2; %  对重力加速度消除后的 Y 轴加速度进行平滑处理
%  调用函数进行两次 3 点均值平滑
smoothedSignal2=twice3PointMeanSmoothing(signal);
signal=chuli_data3; %  对重力加速度消除后的 Y 轴加速度进行平滑处理
%  调用函数进行两次 3 点均值平滑
smoothedSignal3=twice3PointMeanSmoothing(signal);
%绘图
figure(2)
subplot(3,1,1)
plot(1:PointNumber,data1, '-r');
hold on
plot(1:PointNumber,data2, '-b');
hold on
plot(1:PointNumber,data3, '-g');
ylabel('g');
legend('X 轴加速度','Y 轴加速度','Z 轴加速度');
title('重力加速度消除前 X、Y、Z 轴的加速度');
grid on
subplot(3,1,2)
plot(1:PointNumber,chuli_data1, '-r');
hold on
plot(1:PointNumber,chuli_data2, '-b');
hold on
plot(1:PointNumber,chuli_data3, '-g');
ylabel('g');
legend('X 轴加速度','Y 轴加速度','Z 轴加速度');
title('重力加速度消除后 X、Y、Z 轴的加速度');
grid on
subplot(3,1,3)
plot(1:length(smoothedSignal1),smoothedSignal1, '-r');
hold on
plot(1:length(smoothedSignal2),smoothedSignal2, '-b');
hold on
plot(1:length(smoothedSignal3),smoothedSignal3, '-g');
```

```
ylabel('g');
legend('X 轴加速度','Y 轴加速度','Z 轴加速度');
title('重力加速度消除后并进行两次三点均值平滑处理的 X、Y、Z 轴加速度');
grid on

%%Savitzky-Golay 平滑处理
windowSize=21;
polyOrder=(windowSize-1)/2;
originalData1=chuli_data1;
originalData2=chuli_data2;
originalData3=chuli_data3;
smoothedData1= sgolayfilt(originalData1, polyOrder, windowSize);
smoothedData2= sgolayfilt(originalData2, polyOrder, windowSize);
smoothedData3= sgolayfilt(originalData3, polyOrder, windowSize);
% 绘制原始数据和平滑后的数据
figure(3);
subplot(3,1,1)
plot(1:PointNumber,data1, '-r');
hold on
plot(1:PointNumber,data2, '-b');
hold on
plot(1:PointNumber,data3, '-g');
ylabel('g');
legend('X 轴加速度','Y 轴加速度','Z 轴加速度');
title('重力加速度消除前 X、Y、Z 轴的加速度');grid on
subplot(3,1,2)
plot(1:PointNumber,chuli_data1, '-r');
hold on
plot(1:PointNumber,chuli_data2, '-b');
hold on
plot(1:PointNumber,chuli_data3, '-g');
ylabel('g');
legend('X 轴加速度','Y 轴加速度','Z 轴加速度');
title('重力加速度消除后 X、Y、Z 轴的加速度');grid on
subplot(3,1,3)
plot(1:length(smoothedData1),smoothedData1, '-r');
hold on
plot(1:length(smoothedData2),smoothedData2, '-b');
hold on
plot(1:length(smoothedData3),smoothedData3, '-g');
ylabel('g');
```

```
legend('X 轴加速度','Y 轴加速度','Z 轴加速度');
title('重力加速度消除后并进行 Savitzky-Golay 平滑处理的 X、Y、Z 轴加速度');grid
on
%%两点三均值光滑处理算法评估
MSE1=mean((chuli_data1(1:end-4)-smoothedSignal1).^2);%均方误差
MAE1= mean(abs(chuli_data1(1:end-4)-smoothedSignal1));%平均绝对误差
MSE2=mean((chuli_data2(1:end-4)-smoothedSignal2).^2);%均方误差
MAE2= mean(abs(chuli_data2(1:end-4)-smoothedSignal2));%平均绝对误差
MSE3=mean((chuli_data3(1:end-4)-smoothedSignal3).^2);%均方误差
MAE3= mean(abs(chuli_data3(1:end-4)-smoothedSignal3));%平均绝对误差
% 输出评估结果
fprintf('MSE: %f\n', MSE1);
fprintf('MAE: %f\n', MAE1);
fprintf('MSE: %f\n', MSE2);
fprintf('MAE: %f\n', MAE2);
fprintf('MSE: %f\n', MSE3);
fprintf('MAE: %f\n', MAE3);
%%Savitzky-Golay 平滑处理算法评估
MSE1=mean((chuli_data1-smoothedData1).^2);%均方误差
MAE1= mean(abs(chuli_data1-smoothedData1));%平均绝对误差
MSE2=mean((chuli_data2-smoothedData2).^2);%均方误差
MAE2= mean(abs(chuli_data2-smoothedData2));%平均绝对误差
MSE3=mean((chuli_data3-smoothedData3).^2);%均方误差
MAE3= mean(abs(chuli_data3-smoothedData3));%平均绝对误差
% 输出评估结果
fprintf('MSE: %f\n', MSE1);
fprintf('MAE: %f\n', MAE1);
fprintf('MSE: %f\n', MSE2);
fprintf('MAE: %f\n', MAE2);
fprintf('MSE: %f\n', MSE3);
fprintf('MAE: %f\n', MAE3);
```

附录 B：问题二程序

Pathon 程序见附件。

附录 C：问题三相关 matlab 程序

```
clc;
```

```
clear all;
close all;
%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person4');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);

% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result4(:,i)= calculateStatistics(matrix);
end
person4=[27; 164; 43];
for i=1:1:60
    result4(8:10,i)=person4;
end
```

```
%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person5');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);

% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result5(:,i)= calculateStatistics(matrix);
end
person5=[23; 168; 52];
for i=1:1:60
    result5(8:10,i)=person5;
end

%%数据读取
desktopPath = getenv('USERPROFILE');
```

```
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person6');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);

% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result6(:,i)= calculateStatistics(matrix);
end
person6=[27; 164; 50];
for i=1:1:60
    result6(8:10,i)=person5;
end


%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person7');
```

```
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);

% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result7(:,i)= calculateStatistics(matrix);
end
person7=[35; 170; 63];
for i=1:1:60
    result7(8:10,i)=person7;
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person8');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
```

```
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);

% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result8(:,i)= calculateStatistics(matrix);
end
person8=[30; 171; 60];
for i=1:1:60
    result8(8:10,i)=person8;
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person9');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
```

```
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result9(:,i)= calculateStatistics(matrix);
end
person9=[22; 180; 76];
for i=1:1:60
    result9(8:10,i)=person9;
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person10');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);

% 检查是否有 Excel 文件
```

```
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result10(:,i)= calculateStatistics(matrix);
end
person10=[49; 166; 68];
for i=1:1:60
    result10(8:10,i)=person10;
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person11');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
```

```
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result11(:,i)= calculateStatistics(matrix);
end
person11=[21; 178; 71];
for i=1:1:60
    result11(8:10,i)=person11;
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person12');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
```

```
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result12(:,i)= calculateStatistics(matrix);
end
person12=[34; 165; 78];
for i=1:1:60
    result12(8:10,i)=person12;
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person13');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
```

```
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result13(:,i)= calculateStatistics(matrix);
end
person13=[36; 170; 80];
for i=1:1:60
    result13(8:10,i)=person13;
end
%%神经网络算法
result=[result4,result5,result6,result7,result8,result9,result10,result11,result12,result13];
result=result';
p=[result(:,1:7)];%输入数据矩阵
p=p';
o=[result(:,8:10)];%输出数据矩阵
o=o';
[pn,input_str] = mapminmax(p) ;
[tn,output_str] = mapminmax(o) ;
net=newff(pn,tn,[5 8 4],{'purelin','logsig','purelin'});
% 10 轮回显示一次结果
net.trainParam.show=10;
% 学习速度为 0.05
net.trainParam.lr=0.05;
% 最大训练次数为 5000 次
net.trainParam.epochs=10000;
% 均方误差
net.trainParam.goal=0.65*10^(-3);
% 网络误差如果连续 6 次迭代都没有变化，训练将会自动终止（系统默认的）
% 为了让程序继续运行，用以下命令取消这条设置
net.divideFcn = '';
```

```
% 开始训练，其中 pn,tn 分别为输入输出样本
net=train(net,pn,tn);

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 1', 'person1');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result1(:,i)= calculateStatistics(matrix);
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 1', 'person2');
```

```
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result2(:,i)= calculateStatistics(matrix);
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 1', 'person3');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
% 检查是否有 Excel 文件
if isempty(files)
```

```
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result3(:,i)= calculateStatistics(matrix);
end

i=[result1,result2,result3];
[in,input_str]=mapminmax(i);
op=sim(net,in);
% 利用函数 mapminmax 把仿真得到的数据还原为原始的数量级
a = mapminmax('reverse',op,output_str);

for i=1:1:60
    person (1:3,i)=[26;185;75];
end
for i=61:1:120
    person (1:3,i)=[31;169;68];
end
for i=121:1:180
    person (1:3,i)=[32;160;75];
end
%%绘图
```

```
figure(1)
subplot(3,1,1)
x=1:1:180;
plot(x,a(1,:),'r-o',x,person(1,:),'b--+')
grid on
legend('预测年龄','实际年龄');
ylabel('岁');
title('神经网络年龄预测值与实际值对比图');
subplot(3,1,2)
x=1:1:180;
plot(x,a(2,:),'r-o',x,person(2,:),'b--+')
grid on
legend('预测身高','实际身高');
ylabel('cm');
title('神经网络身高预测值与实际值对比图');
subplot(3,1,3)
x=1:1:180;
plot(x,a(3,:),'r-o',x,person(3,:),'b--+')
grid on
legend('预测体重','实际体重');
ylabel('kg');
title('神经网络体重预测值与实际值对比图');
data_mean1=mean(a(:,1:60),2);
data_mean2=mean(a(:,61:120),2);
data_mean3=mean(a(:,121:180),2);
person1=[26;185;75];
person2=[31;169;68];
person3=[32;160;75];
data_wuch1=abs(data_mean1-person1)./person1;
data_wuch2=abs(data_mean2-person2)./person2;
data_wuch3=abs(data_mean3-person3)./person3;

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 5', 'unknow1');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
% 检查是否有 Excel 文件
```

```
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:12
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    yuche1(:,i)= calculateStatistics(matrix);
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 5', 'unknow2');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% cell 来存储读取的数据
allData = cell(length(files), 1);
```

```
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:12
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    yuche2(:,i)= calculateStatistics(matrix);
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 5', 'unknow3');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
```

```
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:12
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    yuche3(:,i)= calculateStatistics(matrix);
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 5', 'unknow4');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
```

```
for i=1:1:12
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    yuche4(:,i)= calculateStatistics(matrix);
end
%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 5', 'unknow5');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:12
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    yuche5(:,i)= calculateStatistics(matrix);
end
input=[yuche1,yuche2,yuche3,yuche4,yuche5];
[in,input_str]=mapminmax(input);
```

```
op=sim(net,in);
%  利用函数 mapminmax 把仿真得到的数据还原为原始的数量级
a = mapminmax('reverse',op,output_str);
yuchedata_mean1=mean(a(:,1:12),2);
yuchedata_mean2=mean(a(:,13:24),2);
yuchedata_mean3=mean(a(:,25:36),2);
yuchedata_mean4=mean(a(:,37:48),2);
yuchedata_mean5=mean(a(:,49:60),2);
```

%%支持向量机判别

```
clc;
clear all;
close all;
%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person4');
%  构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); %  搜索所有.xlsx 文件
%  获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
%  检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
%  初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
%  循环读取每个 Excel 文件
for i = 1:length(files)
    %  构造完整的文件路径
```

```
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result4(:,i)= calculateStatistics(matrix);
end
person4=[27; 164; 43];
for i=1:1:60
    result4(8:10,i)=person4;
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person5');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
```

```
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result5(:,i)= calculateStatistics(matrix);
end
person5=[23; 168; 52];
for i=1:1:60
    result5(8:10,i)=person5;
```

```
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person6');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);

% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
```

```
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result6(:,i)= calculateStatistics(matrix);
end
person6=[27; 164; 50];
for i=1:1:60
    result6(8:10,i)=person5;
end
%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person7');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);

% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
```

```
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result7(:,i)= calculateStatistics(matrix);
end
person7=[35; 170; 63];
for i=1:1:60
    result7(8:10,i)=person7;
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person8');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
```

```
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result8(:,i)= calculateStatistics(matrix);
end
person8=[30; 171; 60];
for i=1:1:60
```

```
    result8(8:10,i)=person8;
end
%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person9');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);

% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
```

```
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result9(:,i)= calculateStatistics(matrix);
end
person9=[22; 180; 76];
for i=1:1:60
    result9(8:10,i)=person9;
end
%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person10');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);

% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
```

```
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result10(:,i)= calculateStatistics(matrix);
end
person10=[49; 166; 68];
for i=1:1:60
    result10(8:10,i)=person10;
end

%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person11');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
```

```
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result11(:,i)= calculateStatistics(matrix);
end
person11=[21; 178; 71];
for i=1:1:60
    result11(8:10,i)=person11;
```

```
end


%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person12');
% 构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); % 搜索所有.xlsx 文件
% 获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
% 检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% 初始化一个数组或 cell 来存储读取的数据
allData = cell(length(files), 1);
% 循环读取每个 Excel 文件
for i = 1:length(files)
    % 构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
```

```
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result12(:,i)= calculateStatistics(matrix);
end
person12=[34; 165; 78];
for i=1:1:60
    result12(8:10,i)=person12;
end
%%数据读取
desktopPath = getenv('USERPROFILE');
folderPath = fullfile(desktopPath, 'Desktop\附件 2', 'person13');
%  构造文件夹中 Excel 文件的搜索路径
searchPath = fullfile(folderPath, '*.xlsx'); %  搜索所有.xlsx 文件
%  获取文件夹下所有的 Excel 文件列表
files = dir(searchPath);
%  检查是否有 Excel 文件
if isempty(files)
    disp('文件夹中没有找到 Excel 文件。');
    return;
end
% cell 来存储读取的数据
allData = cell(length(files), 1);
%  循环读取每个 Excel 文件
for i = 1:length(files)
    %  构造完整的文件路径
    filePath = fullfile(folderPath, files(i).name);
```

```
    % 读取 Excel 文件
    data = readtable(filePath, 'ReadVariableNames', true); % 假设第一行包含列名
    % 存储数据到 cell 数组
    allData{i} = data;
end
% allData 中现在存储了所有 Excel 文件的数据
% 合并到一个单独的表格中
combinedData = cell2table(allData);
for i=1:1:60
    data=allData{i,1};
    calcute_data=data(:,1);
    matrix=table2array(calcute_data);
    result13(:,i)= calculateStatistics(matrix);
end
person13=[36; 170; 80];
for i=1:1:60
    result13(8:10,i)=person13;
end

%%神经网络算法
result=[result4,result5,result6,result7,result8,result9,result10,result11,result12,result13];
result=result';
p=[result(:,1:7)];%输入数据矩阵
p=p';
o=[result(:,8:10)];%输出数据矩阵
o=o';
[pn,input_str] = mapminmax(p) ;
[tn,output_str] = mapminmax(o) ;
net=newff(pn,tn,[5 8 4],{'purelin','logsig','purelin'});
```

```
% 10 轮回显示一次结果
net.trainParam.show=10;
% 学习速度为 0.05
net.trainParam.lr=0.05;
% 最大训练次数为 5000 次
net.trainParam.epochs=5000;
% 均方误差
net.trainParam.goal=0.65*10^(-3);
% 网络误差如果连续 6 次迭代都没有变化，训练将会自动终止（系统默认的）
% 为了让程序继续运行，用以下命令取消这条设置
net.divideFcn = '';
% 开始训练，其中 pn,tn 分别为输入输出样本
net=train(net,pn,tn);
op=sim(net,pn);
% 利用函数 mapminmax 把仿真得到的数据还原为原始的数量级
a = mapminmax('reverse',op,output_str);
a(4,1:60)=4;
a(4,61:120)=5;
a(4,121:180)=6;
a(4,181:240)=7;
a(4,241:300)=8;
a(4,301:360)=9;
a(4,361:420)=10;
a(4,421:480)=11;
a(4,481:540)=12;
a(4,541:600)=13;
a=a';%数据转置
%数据标准化
```

```
X_train = a(:,1:3); % 600 个训练样本，3 个特征
y_train = a(:,4); % 10 个类别
[X_train_norm, mu, sigma] = zscore(X_train); %  标准化训练集
t = templateSVM('Standardize',1);
% 3. 选择 SVM 模型和参数
Mdl = fitcecoc(X_train_norm,y_train,'Learners',t,'ClassNames',{'4','5','6','7','8','9','10','11','12','13'});
%  新点进行标准化
X_new = [29,170,62;
         26,171,55;
         26,171,51;
         25,172,59;
         28,171,67]; % 5 个新点
X_new_norm = (X_new - mu)./ sigma; %  使用训练集的均值和标准差标准化新点
%   分类新点
predictedClass = predict(Mdl, X_new_norm);
%使用 10 倍交叉验证交叉验证 Mdl
CVmodel_1 = crossval(Mdl);
%估算泛化误差
oosLoss_1 = kfoldLoss(CVmodel_1);
```

附录 D: 程序说明

代码环境	操作系统: Window10
	编程语言: matlab 2019b

代码清单

calculateStatistics.m	问题三中特征值计算函数程序
Disanti.m	问题三人员特征（年龄、身高、体重）神经网络模型预测

huitu.m	运动数据绘图程序
Shujuyuchuli.m	数据预处理程序
twice3PointMeanSmoothing.m	两次 3 点均值平滑处理程序
Xiangliangj.m	问题三支持向量机程序
yingxfenxi.m	问题三影响分析程序
CHECKPOINT	模型存档文件
Q2_filter_mlp.ipynb	问题二程序