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）。

所属学校和学院（请填写完整的全名）：国防科技大学系统工程学院

参赛队员（打印后签名）：1. 江俊辉 江俊辉

2. 尤李浙 尤李浙

3. 张毅 张毅

指导教师或指导教师组负责人（打印后签名）：李文桦 李文桦

日期：2024年 7 月 5 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。）

基于传感器信息的人员活动识别

摘要

随着智能手机的普及, 如何根据传感器信息对人员活动进行识别是智能手机判断使用者能量消耗的主要问题. 本文针对基于传感器信息的人员活动识别问题首先建立了基于 t-SNE 流形学习与高斯核 K-means 聚类的人员活动类别分类模型. 之后, 通过提取 12 类活动类别的典型特征, 建立基于 GPLearn 符号回归与三次核 SVM 判别模型. 最后, 通过计算不同人员在不同活动类别上的合角速度, 构建了未知人员判别模型.

针对问题一, 首先对原始数据进行中值滤波、数据归一化两步预处理操作. 在此基础上, 对每一维的速度提取 **8 个时域特征**和 **1 个频域特征**构建特征向量. 之后, 对提取所得的特征向量进行 t-SNE 流形学习降维, 并将降维后的数据使用高斯核 K-means 聚类方法聚类. 为提高聚类匹配性, 每簇有且仅有 5 个数据点作为约束条件加入至高斯核 K-means 聚类方法中. 聚类结果见表3.

针对问题二, 首先通过动力学分析, 将 12 类活动类别划分为 4 种运动基元——转动、水平移动、垂直移动和其他的简单组合. 之后, 对 4 种运动基元的典型特征进行分析提炼以得到 12 类活动类别的典型特征. 在此基础上, 以这些典型特征对问题一所得的 12 个类别进行活动类别匹配, 并计算得问题一的分类模型分类准确度为 27.6%. 在此之后, 采用 GPLearn 对提取的 54 维特征进行进一步地信息挖掘, 得到 80 个深层特征. 最后, 依次构造三次核、二次核、线性核 SVM 模型与以决策树为基分类器的集成学习方法的判别模型, 选取判别准确率最高 **(89.5%)** 的三次核 SVM 模型对附件 3 数据进行判别, 判别结果见表5.

针对问题三, 首先选择带有准确标签的实验人员 4-13 计算其合加速度与合角速度. 通过量纲与量级分析, 选择合角速度, 以人员类别为因素,5 次实验结果的均值为观测值进行单因素方差分析, 得出**仅向前走、躺下、乘坐电梯向上或向下移动不会因为人员的不同而产生差异**. 之后, 通过对合角速度的排列, 分析出**年龄影响行走与跑步, 身高影响跳跃以及体重影响坐下与站立**的结论. 最后, 选取**合角速度均值极差最小的活动类别——坐下**作为判别特征, 对 unknow1-5 进行人员判别. 判别结果显示, unknow1-5 分别为 **Person10、7、5、8、13**.

关键词： 人体活动识别　流形学习　核聚类　快速傅里叶变换

目录

1 问题综述

1.1 问题背景

随着智能手机的普及, 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. 而评估使用者日常活动消耗的热量是评估使用者当前可能的运动状态, 即根据一定的信息来测量人体当前的活动状态. 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——加速度计与陀螺仪来实现. 加速度计和陀螺仪主要测量的是 (X, Y, Z) 方向的加速度和角速度, 手机根据这些测量所得信息, 通过内置的算法进行计算, 得到使用者当前的活动状态.

目前有许多学者对人体活动行为识别算法做了研究. 传统的人体活动识别主要基于可穿戴式感知和视频感知技术, 例如: 穿戴智能手表等, 这种研究途径在一定程度上会被一些物理因素限制. 例如, 视频感知技术会因实际工作环境的可见度低而导致可靠度下降. 随着智能手机的普及, 利用物理传感器信息对人体活动行为识别成为了一种可行性强、可靠性高的方式.

1.2 问题提出

基于传感器信息的人体活动识别涉及多方面的问题, 这些问题可以精炼成如下 3 个问题:

(1) 问题 1: 如何确定不同人体活动的传感器信息特征;

(2) 问题 2: 如何根据传感器信息对人体活动进行判别;

(3) 问题 3: 人体不同信息, 例如: 不同身高、体重, 对人体活动识别的影响.

1.3 资料条件

附件 1-5 提供了若干个实验人员活动状态数据, 各文件的详细说明如下:

- 实验人员活动状态数据位 Excel 问题, 包含 6 列, 分别为加速度计和陀螺仪在 X, Y, Z 三个轴向的记录值. 传感器固定在实验人员右下腹位置, 采样率为 100Hz, 加速度计和陀螺仪的记录范围分别为 ±6g 和 ±500dps. 三个轴向分别被定义为:X 轴沿重力方向,Y 轴沿实验人员前进方向,Z 轴与 XY 平面垂直指向身体一侧, 如图1所示.
- 附件 1 提供了 3 名实验人员的运动数据, 包含每名实验人员 12 种活动状态的 5 组加速度计和陀螺仪数据, 但不包含数据所代表的活动状态.
- 附件 2 提供了 10 名实验人员的活动数据, 包含每名实验人员 12 种活动状态的 5 组加速度计和陀螺仪数据, 也包含数据所代表的活动状态.
- 附件 3 提供了某名实验人员 30 次活动的状态数据.
- 附件 4 提供了参与实验的 13 名实验人员的年龄、身高、体重等数据.

- 附件 5 给出了附件 2 中 10 名实验人员中 5 位的某次活动数据, 数据包含了每人的 12 类活动状态.

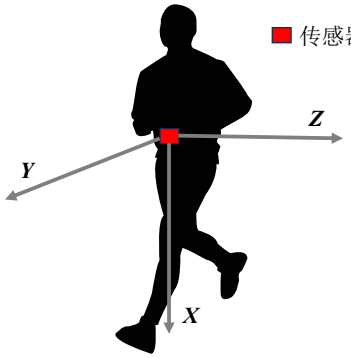


图 1　XYZ 轴向示意图

2 模型假设与符号说明

2.1 模型基本假设

(1) 假设 1: 假设附件 1 中实验数据已打乱, 即基本不存在连续 5 组实验数据来自同一个活动.

(2) 假设 2: 假设加速度计不能测出因为身高差导致的加速度差异.

(3) 假设 3: 假设身高、体重不会对传感器的摆放方向产生影响.

2.2 符号说明

本文定义了如下 7 个使用次数较多的符号, 其余符号已均在使用时注明.

表 1　符号说明

符号	含义	单位
a_x	X 轴向加速度	$\mathrm{m{\cdot}s^{-2}}$
a_y	Y 轴向加速度	$\mathrm{m{\cdot}s^{-2}}$
a_z	Z 轴向加速度	$\mathrm{m{\cdot}s^{-2}}$
ω_x	X 轴向角速度	dps
ω_y	Y 轴向角速度	dps
ω_z	Z 轴向角速度	dps

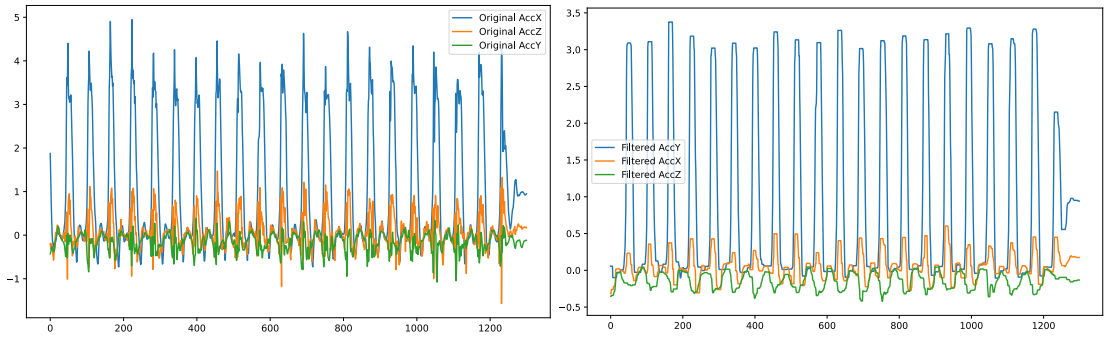
3 数据预处理

由于传感器在实际使用的过程中难免会受到环境噪声的影响, 因此需要对附件中的传感器采集的数据进行预处理. 在本文中, 原始数据将经过中值滤波、数据归一化两步操作进行处理.

3.1 中值滤波

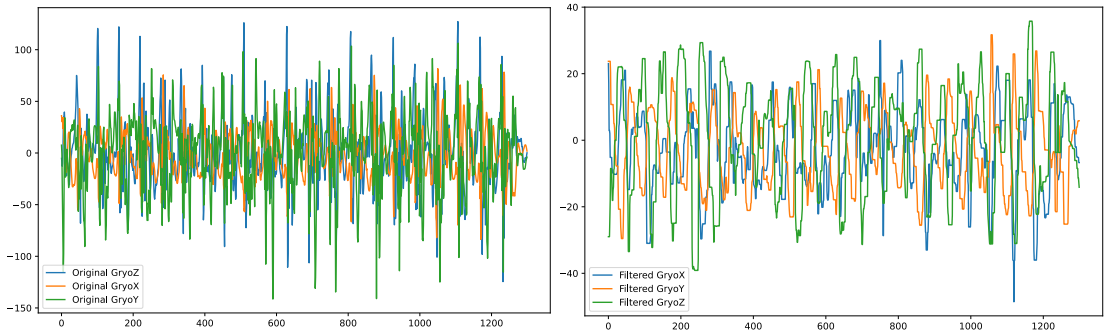
中值滤波是一种非线性信号处理技术, 能有效的将数据进行平滑、去噪. 中值滤波的基本原理是将某一点的值由该点邻域中各个点的中值代替, 从而可以消除孤立的噪声点.

图2和图3分别展示了加速度和角速度数据在中值滤波前后的波形图. 明显可见, 滤波前加速度和角速度的波形图具有许多尖锐的峰值以及密集性的抖动. 在滤波后, 数据变得更加稀疏与平稳. 滤波后的数据也较大程度的保留了人体活动信息特征, 可提高后续模型对人体活动识别的准确度.



(a) 滤波前加速度数据　　(b) 滤波后加速度数据

图 2　加速度中值滤波前后对比图



(a) 滤波前角速度数据　　(b) 滤波后角速度数据

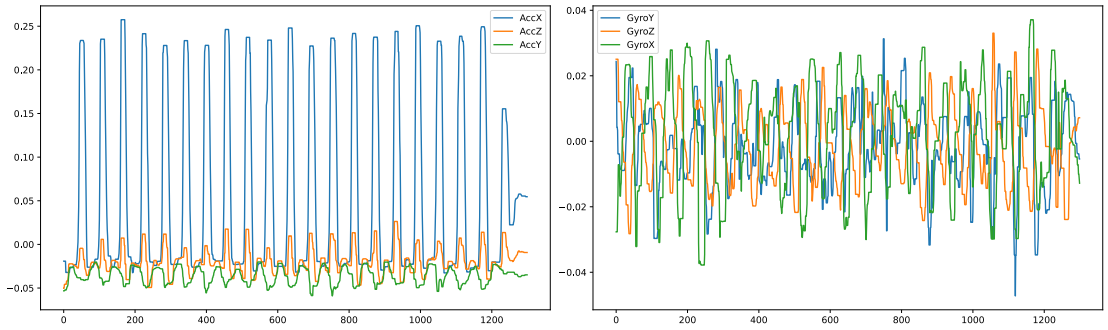
图 3　角速度中值滤波前后对比图

3.2 数据归一化

由于加速度计和陀螺仪的精度、单位不同, 不利于后续特征的提取, 因此需要通过数据归一化消除加速度计和陀螺仪测量数据的量纲和量级带来的影响. 本文采用式 (1) 对滤波后的数据进行归一化, 归一化后的数据在 $[-1, 1]$ 上. 图4为加速度和角速度归一化后的波形图.

$$x_{i,scale} = \frac{x_i - \overline{x}}{X_{\max} - X_{\min}} \tag{1}$$

式中 $x_{i,scale}$ 表示第 i 个归一化后的数据, x_i 表示第 i 个滤波后的数据, $\bar{x}$ 表示滤波后数据的平均值, $X_{\max}$ 表示加速度计或陀螺仪量程的最大值, $X_{\min}$ 表示加速度计或陀螺仪量程的最小值.



(a) 归一化后加速度数据　　(b) 归一化后角速度数据

图 4　加速度与角速度归一化后的波形图

4 问题一分析与求解

4.1 问题一的分析

问题一旨在对附件 1 中的 3 名实验人员的运动数据进行聚类分析. 聚类的类别数量为 12, 即题中活动的总数. 显然, 在问题一只需完成聚类, 而并不需要考虑每一类对应的活动状态. 用于聚类的数据为预处理后的加速度与角速度数据, 这些数据都是时域上的数据, 且具有较强的重复性, 因此能表征的特征信息是有限的, 需要进一步对这些数据进行加工, 以提前更多特征. 在提取完特征后, 根据提取的特征对实验数据进行聚类, 可提高聚类的准确度. 图5展示了问题一的解决方案流程图.

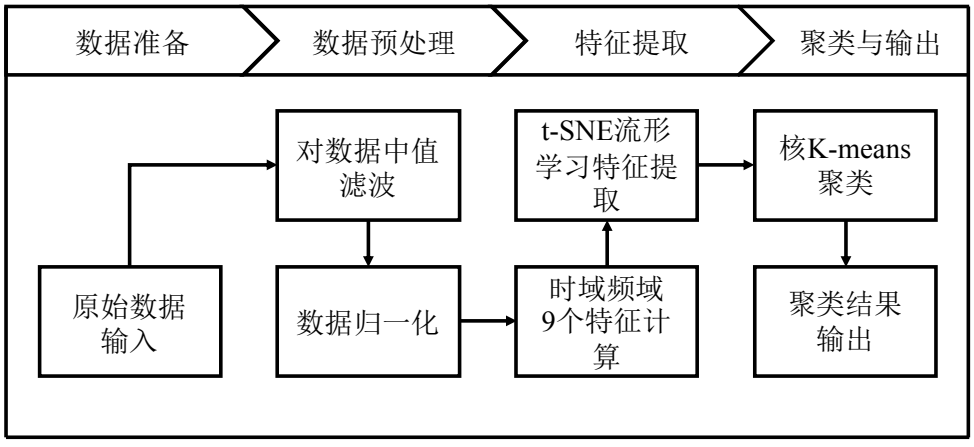


图 5　问题一解决方案

4.2 基于时域与频域的特征提取

对于信号的处理往往可以从时域与频域两个层面入手. 信号在时域上的特征是指信号随时间变化的特征, 它往往由原始数据的统计指标, 如均值、标准差等进行反映. 信号在频域上的特征是指信号在不同频率区间上的信号强度. 在本文中, 预处理后的每一维数据将被提取平均值、标准差、方差、最大值、最小值、峰度、四分位距和均方根 8 个时域特征, 频谱能量 1 个频域特征, 共 54 个特征. 这些特征的计算方式如下:

(1) 均值

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2}$$

(2) 标准差

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{3}$$

(3) 方差

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2 \tag{4}$$

(4) 最大值

$$x_{\max} = \max_i x_i \tag{5}$$

(5) 最小值

$$x_{\min} = \min_i x_i \tag{6}$$

(6) 峰度

$$\gamma = \frac{\mu_4}{\sigma^4} - 3 \tag{7}$$

式中 μ_4 表示 4 阶中心距.

(7) 四分位距

$$I_r = Q_3(x_i) - Q_1(x_i) \tag{8}$$

式中 $Q_3(x_i)$ 表示第三四分位数,$Q_1(x_i)$ 表示第一四分位数.

(8) 均方根

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} \tag{9}$$

(9) 频谱能量

$$E = \sum_{i=1}^{N} |F(x_i)|^2 \tag{10}$$

式中 $F(x_i)$ 表示信号 x_i 的快速傅里叶变换结果,$|F(x_i)|$ 表示其模长.

表2展示了部分实验人员 1 的 60 次实验数据经过特征提取后所得的特征, 具体数据以及其他实验人员的特征提取结果见支撑材料. 其中 a_x, a_y, a_z 表示 XYZ 三个轴向加速度,$\omega_x, \omega_y, \omega_z$ 表示 XYZ 三个轴向角速度.

表 2　实验人员 1 的 60 次实验数据特征提取结果

	a_x 的均值	a_y 的均值	a_z 的均值	$\cdots$	ω_x 的频谱能量	ω_y 的频谱能量	ω_z 的频谱能量
SY1	0.05525	-0.01925	-0.03600	$\cdots$	0.00106	0.00093	0.00193
SY2	0.05519	-0.01399	-0.04121	$\cdots$	0.00344	0.00026	0.00033
SY3	0.04782	-0.00989	-0.03793	$\cdots$	0.00057	0.00004	0.00013
SY4	0.05516	-0.01294	-0.04221	$\cdots$	0.00316	0.00038	0.00088
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
SY58	0.05715	-0.01692	-0.04023	$\cdots$	0.01331	0.00370	0.00705
SY59	0.05568	-0.01993	-0.03574	$\cdots$	0.02891	0.00493	0.00975
SY60	0.05633	-0.01214	-0.04419	$\cdots$	0.00371	0.00052	0.00067

4.3 基于 t-SNE 流形学习的特征降维

通过对传感器信息的特征提取后, 每个实验人员的每次实验可用一个 1×54 的特征向量进行表征. 然而处理后的特征向量维度较高, 不利于后续聚类的进行. 因此, 需将特征向量进行降维.

流形学习是机器学习的一种分支, 多用于处理高维数据并将数据处理成低维流形结果. 流形学习的核心思想是假设高维数据可用低维的流形结构进行描述. 相较于传统的降维方法——主成分分析, 流形学习更能处理一些具有非线性特征的数据. 因此, 在本文中采用流形学习的方法对提取的特征向量进行降维.

t-SNE 是一种用于高维数据可视化的流形学习方法. 它通过在低维空间中保持数据点之间的局部相似性进行降维, 并在可视化中展示数据的聚类结构. 其核心思想是用条件概率表示点与点之间的相似度, 在高维数据空间中, 条件概率由高斯联合分布表示, 在低维空间中由 t 分布表示. 之后, 通过最小化两个分布的 KL 散度, 以达成降维. t-SNE 的基本运行流程如下, 流程图如图6所示.

Step.1 计算高维空间的相似度.

利用高斯联合分布来刻画高维空间中点与点的相似性, 记高维空间数据点集合为 $D = \{z_1, z_2, \cdots, z_i, \cdots, z_n\}$, 数据点 z_i 与数据点 z_j 之间的相似度为条件概率 $p_{j|i}$, 如式 (11) 所示.

$$p_{j|i} = \frac{\exp\left(-\|z_i - z_j\|_2^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|z_i - z_k\|_2^2 / 2\sigma_i^2\right)} \tag{11}$$

式中 $\|z_i - z_j\|_2$ 表示 $z_i - z_j$ 的 L_2 范数,σ_i 表示数据点 z_i 对应的高斯分布的标准差.

Step.2 随机初始化低维空间的数据点.

Step.3 计算低维空间中数据点的相似度.

低维空间中的相似度由自由度为 1 的 t 分布刻画, 记低维空间数据点集合为 $D' = \{y_1, y_2, \cdots, y_i, \cdots, y_n\}$. 数据点 y_i 与数据点 y_j 之间的相似度为条件概率 $q_{j|i}$, 如式 (12) 所示.

$$q_{j|i} = \frac{\left(1 + \|y_i - y_j\|_2^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|_2^2\right)^{-1}} \tag{12}$$

Step.4 用 KL 散度计算低维空间与高维空间之间的损失函数. 损失函数的定义如式 (13) 所示.

$$C = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{13}$$

Step.5 用梯度下降法优化低维空间中的数据点.

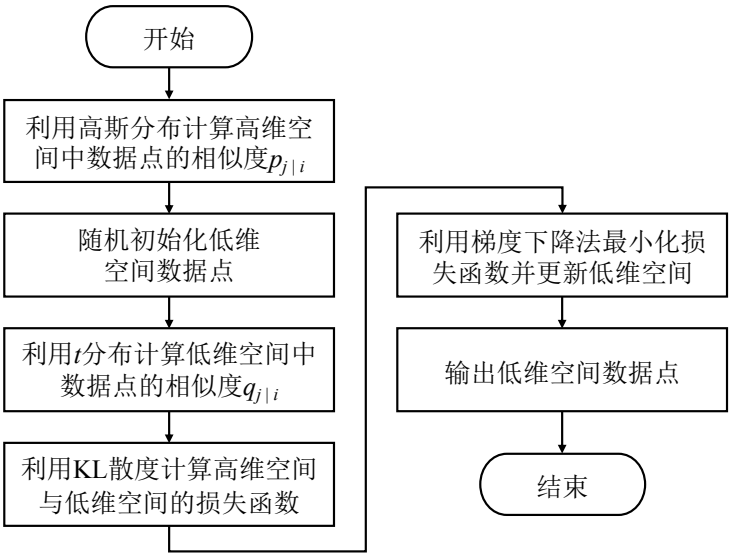


图 6　t-SNE 流程图

4.4 基于核 K-means 聚类的人体活动分类模型

核 K-means 聚类是在 K-means 聚类的基础上引入核函数, 将数据映射到高维空间, 并在高维空间中进行聚类. 引入核函数的优势在于, 数据在原始空间中线性不可分, 通过非线性映射到高维空间后线性可分. 核 K-means 聚类的基本流程如下:

Step.1 从数据点中随机选择 k 个点作为初始聚类中心.

Step.2 将数据点用核函数进行映射, 计算点与所有聚类中心的相似度, 并将点分配给相似度最高聚类中心所在的簇.

Step.3 在高维空间更新聚类中心.

Step.4 重复 Step.2 和 Step.3 至终止条件.

然而, 按照上述流程并不能实现本文需要的聚类效果. 本文需要固定每一簇中有也仅有 5 个数据点. 因此, 需要对 Step.2 进行调整. 在分配点时, 若相似度最高的聚类中心所在的簇的数据点个数超过 5 , 则需将数据点分配给相似度次高的聚类中心, 以此类推, 直到将点分配完成. 修改后的核 K-means 聚类流程图如图7所示.

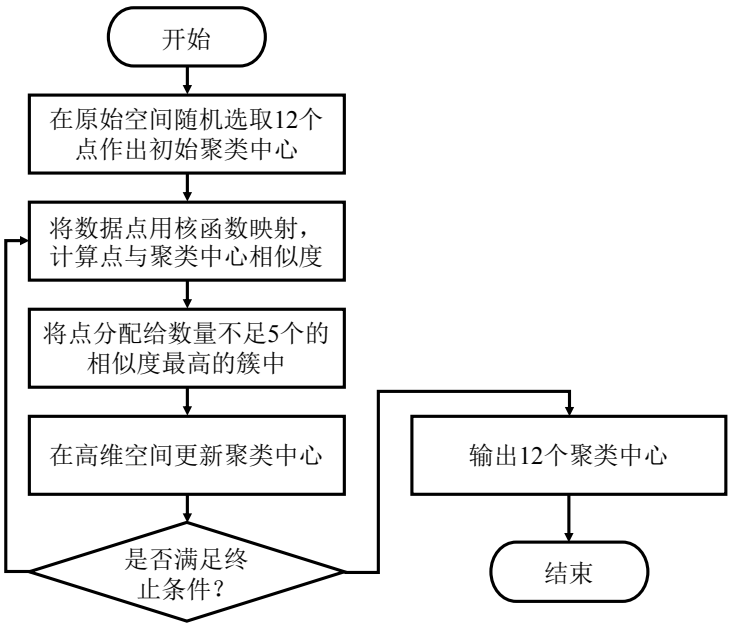


图 7　核 K-means 聚类流程图

在本文中, 核 K-means 聚类中的 k 被设定为 12,K-means 的终止条件被设置为迭代达到最大迭代次数 100. 核函数选用高斯核函数, 其表达式如式 (14) 所示.

$$\kappa(x,y)=e^{-\gamma\|x-y\|_2^2} \tag{14}$$

式中 γ 为高斯核函数参数, 在本文中被设定为 0.1.

表3展示了 3 个实验人员 60 次实验的分类结果. 图8为 Person1- Person3 的 60 次实验数据聚类可视化结果. 图中 X,Y 轴分别表示 t-SNE 提取出的两个特征. 图中每个点都有其编号 0-11, 该编号表示当前点被归于的类别标签.

由图可知, 在 t-SNE 两个特征张成的平面上,Person1-Person3 的 60 次实验数据的分散情况均匀, 但数据点并不是线性可分的. 因此, 需要使用核函数将低维线性不可分数据映射到高维数据. 此外, 从表中结果可知, 并不存在连续 5 组实验来自同一类别, 这进一步论证了本文提出的假设是合理的.

表 3　问题 1 结果

分类	Person1	Person2	Person3
第 1 类	10,34,50,54,55	51,55,57,58,59	28,32,51,55,56
第 2 类	27,30,32,37,49	35,38,42,47,53	18,22,25,39,54
第 3 类	6,21,41,42,48	20,34,43,44,46	24,31,38,49,53
第 4 类	7,11,13,19,20	2,12,16,19,22	2,5,6,10,16
第 5 类	9,29,36,38,43	1,13,14,17,27	3,20,37,45,47
第 6 类	8,17,23,33,53	30,40,45,54,56	40,41,42,43,46
第 7 类	31,39,44,45,46	4,8,23,32,33	11,36,44,48,52
第 8 类	2,15,26,40,52	6,9,18,29,50	1,8,9,14,21
第 9 类	4,14,18,22,24	10,21,28,31,39	7,19,30,34,35
第 10 类	3,5,35,47,51	7,15,25,36,49	4,12,13,26,27
第 11 类	1,12,16,25,28	5,11,24,37,41	15,17,23,29,33
第 12 类	56,57,58,59,60	3,26,48,52,60	50,57,58,59,60

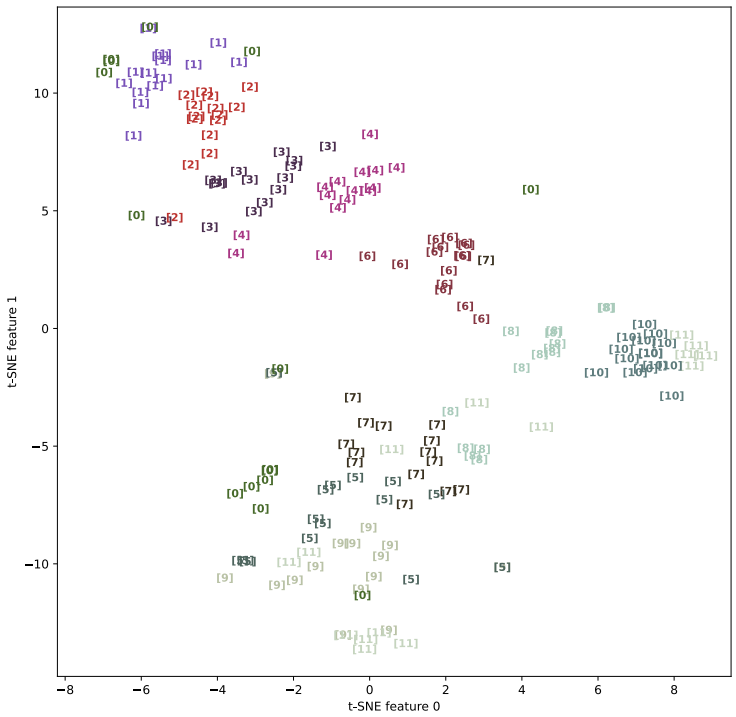


图 8　三名实验人员聚类结果图

5 问题二分析与求解

5.1 问题二分析

问题二旨在根据附件 2 中 10 名实验人员的活动数据, 提取 12 类人员活动状态的典型特征, 构建人员活动状态判别模型. 与问题一不同的是, 问题二中的数据都是已知活动状态的, 因此, 在提取完人员活动状态的典型特征后, 可对问题一中的 12 个类别进行判别, 得出每一个类别对应的人员活动状态. 进而, 可以分析出问题一的模型的分类准确率.

对于附件 2 的数据, 也需要进行中值滤波, 数据归一化两步预处理, 并依次提取 8 个时域特征, 1 个频域特征. 之后, 为进一步挖掘特征因子, 可采用 GPlearn 对 54 个特征进行符号回归. 在此基础上, 与问题一类似, 采用 t-SNE 流形学习对符号回归所得的特征进行降维, 并用若干机器学习方法与降维后的特征构建判别模型. 图9是问题二解决方案的流程图.

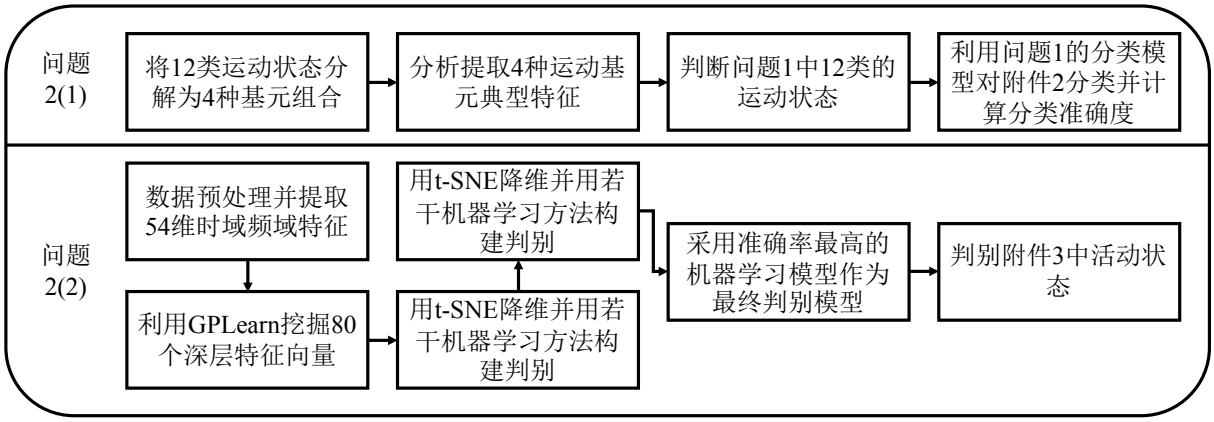


图 9　问题 2 流程图

5.2 人员活动状态的典型特征提取

在对人员活动状态的典型特征提取之前, 需要对人员活动状态进行简要分析. 人员活动状态可以由以下 4 个运动基元构成.

- 围绕人体垂直轴的旋转运动, 转体运动, 记为运动基元 1.
- 水平迈步, 以自身为中心, 向各个方向水平迈步、步行, 记为运动基元 2.
- 重心在垂直方向上的运动, 如垫脚、下蹲等, 记为运动基元 3.
- 姿态变化运动, 即人体水平位置保持不变, 其他部位发生姿态变化, 例如站着甩臂, 该运动基元能表示的姿态比较丰富, 记为运动基元 4.

以上 4 种运动基元进行组合可以表示本文考虑的 12 种人员活动状态. 因此, 只需要提取上述 4 种运动基元的典型特征, 便可表示 12 种人员活动状态的典型特征. 以下分析 12 种人员活动状态的运动基元分解.

(1) 向前走、向左走和向右走

如图10所示, 向前走、向左走和向右走可以拆解为水平方向迈步, 以及手臂姿态变化运动, 即运动基元 2 与运动基元 4 的组合.

(2) 步行上楼和步行下楼

如图11所示, 步行上下楼可以拆解为水平方向迈步, 重心在垂直方向上的运动, 以及手臂姿态变化运动, 即运动基元 2, 运动基元 3 和运动基元 4 的组合.

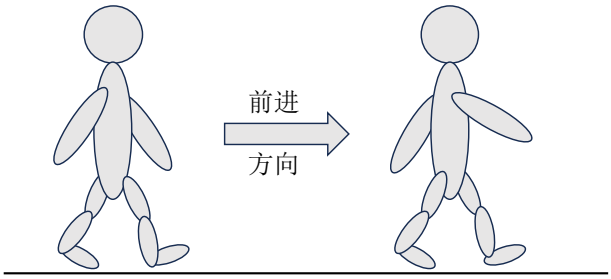


图 10　前进示意图

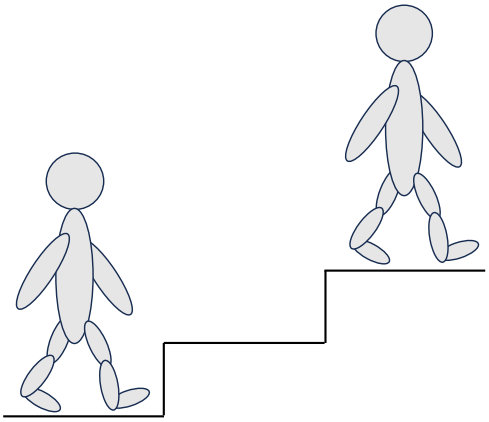


图 11　步行上下楼示意图

(3) 向前跑

如图12所示, 在向前跑的过程中, 人体的重心会略微前移, 会有略微跃起的动作, 因此向前跑可拆解为水平方向迈步, 重心在垂直方向上的运动, 以及手臂姿态变化运动, 即运动基元 2, 运动基元 3 和运动基元 4 的组合. 与步行上下楼不同的时, 向前跑在水平方向的迈步的加速度会更大.

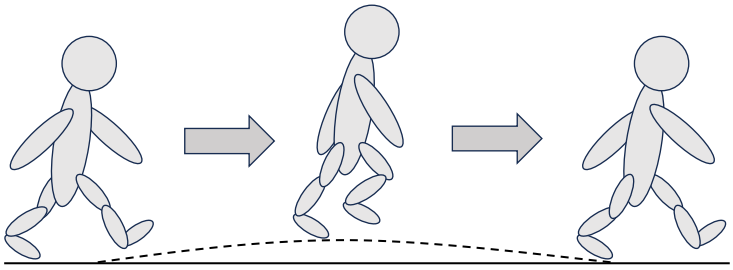


图 12　向前跑示意图

(4) 跳跃

如图13所示, 跳跃可以先拆解为蹲下再跃起的过程. 假设不考虑蹲下和跃起的重心移动, 则跳跃可以拆解为重心在垂直方向上的运动, 以及腿部姿态的变化运动, 即运动基元 3 与运动基元 4 的组合.

(5) 坐下和站立

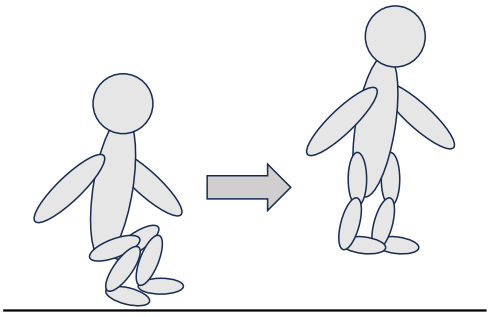


图 13　跳跃示意图

如图14所示, 坐下和站立可以拆解为重心在垂直方向上的移动以及腿部姿态的变化运动, 即运动基元 3 与运动基元 4 的组合. 与跳跃不同的是, 坐下和站立的加速度更小.

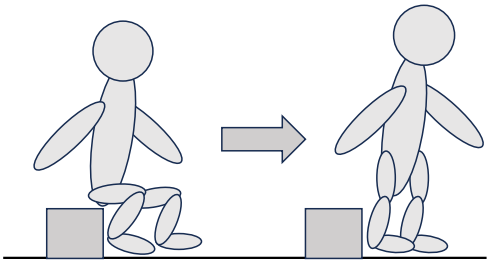


图 14　坐下和站立示意图

(6) 躺下

如图15所示, 躺下可以拆解为围绕垂直轴的旋转运动, 即运动基元 1.

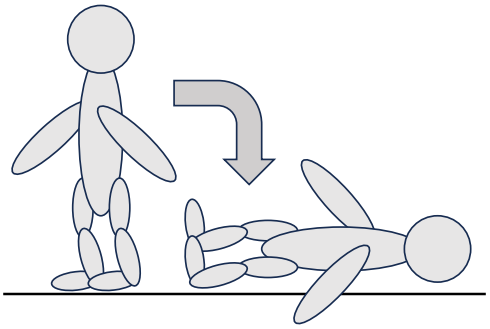


图 15　躺下示意图

(7) 乘坐电梯向上或向下移动

如图16所示, 乘坐电梯向上或向下移动并不能算为人体的活动, 但可拆解为重心在垂直方向上的运动, 即运动基元 3.

通过上述分析可知, 本文考察了 12 种人员活动状态均可表征为 4 种运动基元的简单组合. 而 4 种运动基元在空间上具有一定的独立性, 因此, 只需提取 4 种运动基元的典型特征. 通过简单的物理分析可知,4 种运动基元的典型特征如下:

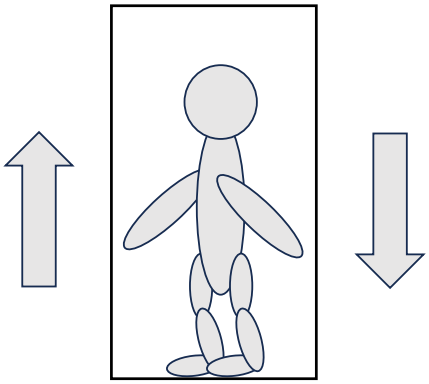


图 16　乘坐电梯向上或向下移动示意图

- 运动基元 1:Z 轴和 X 加速度发生变化,Z 轴存在角速度周期性波动.
- 运动基元 2:Y 轴加速度先减小后增大, 呈周期性波动,X, Y 轴存在角速度周期性波动.
- 运动基元 3:X 轴加速度绝对值先减小后增大, 角速度几乎无变化.
- 运动基元 4:X 轴加速度由正变负, 以正值开始逐渐减小到重力加速度 g, 角速度各方向均存在周期性变化.

由此可将问题一所得 12 个类别与题中 12 个人员活动状态进行对应, 如表4所示.

表 4　问题一所得 12 个类别与 12 个人员活动状态对应关系

问题一分类标签	12 个人员活动状态标签	含义
1	2	向左走
2	6	向前跑
3	4	步行上楼
4	5	步行下楼
5	9	站立
6	3	向右走
7	6	向前跑
8	12	乘坐电梯向下移动
9	7	跳跃
10	11	乘坐电梯向上移动
11	10	躺下
12	1	向前走

在此基础上, 可利用问题一的分类模型对附件 2 中的数据进行分类, 分类结果如图17所示, 具体结果见支撑材料. 由该图可见, 分类结果鲜明, 分类效果较好. 再根据表4中的类别对应关系, 可计算得问题一种的分类模型的准确率为 27.6%.

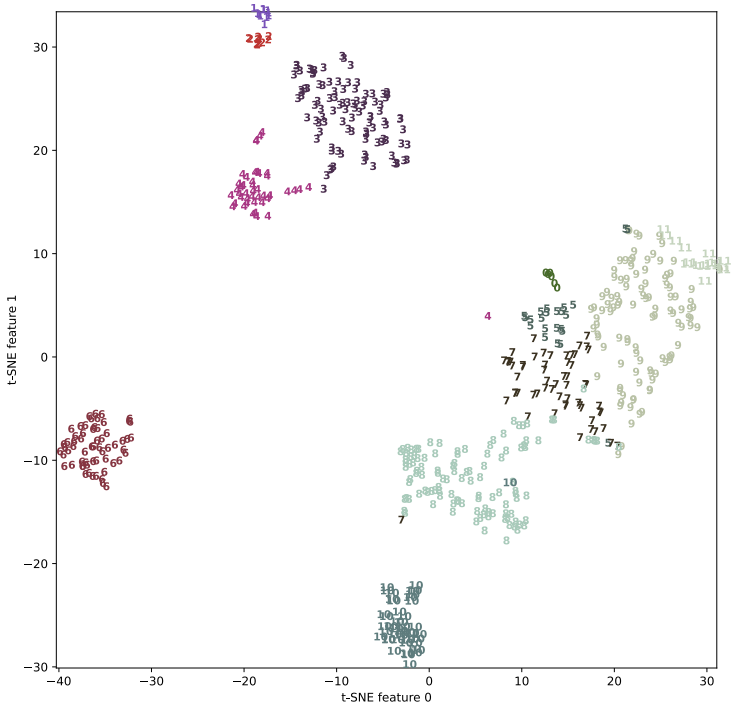


图 17　运用问题一中的分类模型对附件 2 的分类结果

5.3 基于 GPLearn 的符号回归特征提取模型

问题一中的模型可以归结为机器学习中的无监督学习, 但问题二中的判别模型与之不同. 由于附件 2 中的数据都已知标签, 因此, 问题二的判别模型可归结为机器学习中的监督学习. 然而, 仅考虑问题一中的 54 个特征可能并不足以实现高精度判别. 因此, 本文采用符号回归探索更深层次的特征变量.

符号回归是一种机器学习方法, 用于探索某种隐藏的数学表达式. 在本文中, 更深层次的特征变量是由 54 个特征中若干个, 通过基本的数学运算符或函数组合而成.GPLearn 是一种求解符号回归的方法, 其核心原理是遗传算法, 基本流程如下:

Step.1 随机初始化种群, 种群中的每个个体表示某种公式.

Step.2 计算每个个体的适应度.

Step.3 种群通过选择、交叉、变异等操作进行更新.

Step.4 判断是否满足终止条件, 若不满足, 则重复 Step.2-Step.3.

在本文中, 适应度函数为设置为均方误差回归器, 即利用遗传算法个体所得公式, 预测标签值, 并计算预测值与实际值的均方根误差. GPLearn 超参数设置如下:(1) 最大迭代次数:20;(2) 种群规模:500; (3) 精英保留个数:100;(4) 输出特征数:80;(5) 表达式复杂度系数:0.0005;(6) 训练集测试集划分比例:0.9. 表达式的运算符只考虑加法、减法、乘法、除法、对数、开方、绝对值、相反数、最大值和最小值 10 种运算.

图18为 GPLearn 所得的 80 个特征中适应度最优的结果, 它以树的形式进行呈现. 树

的叶节点为初始 54 个特征中某一个, 是可重复的, 其他节点为所需使用的运算符. 对于二元运算符, 它会有两个子节点, 对于一元运算符, 则仅有一个子节点.

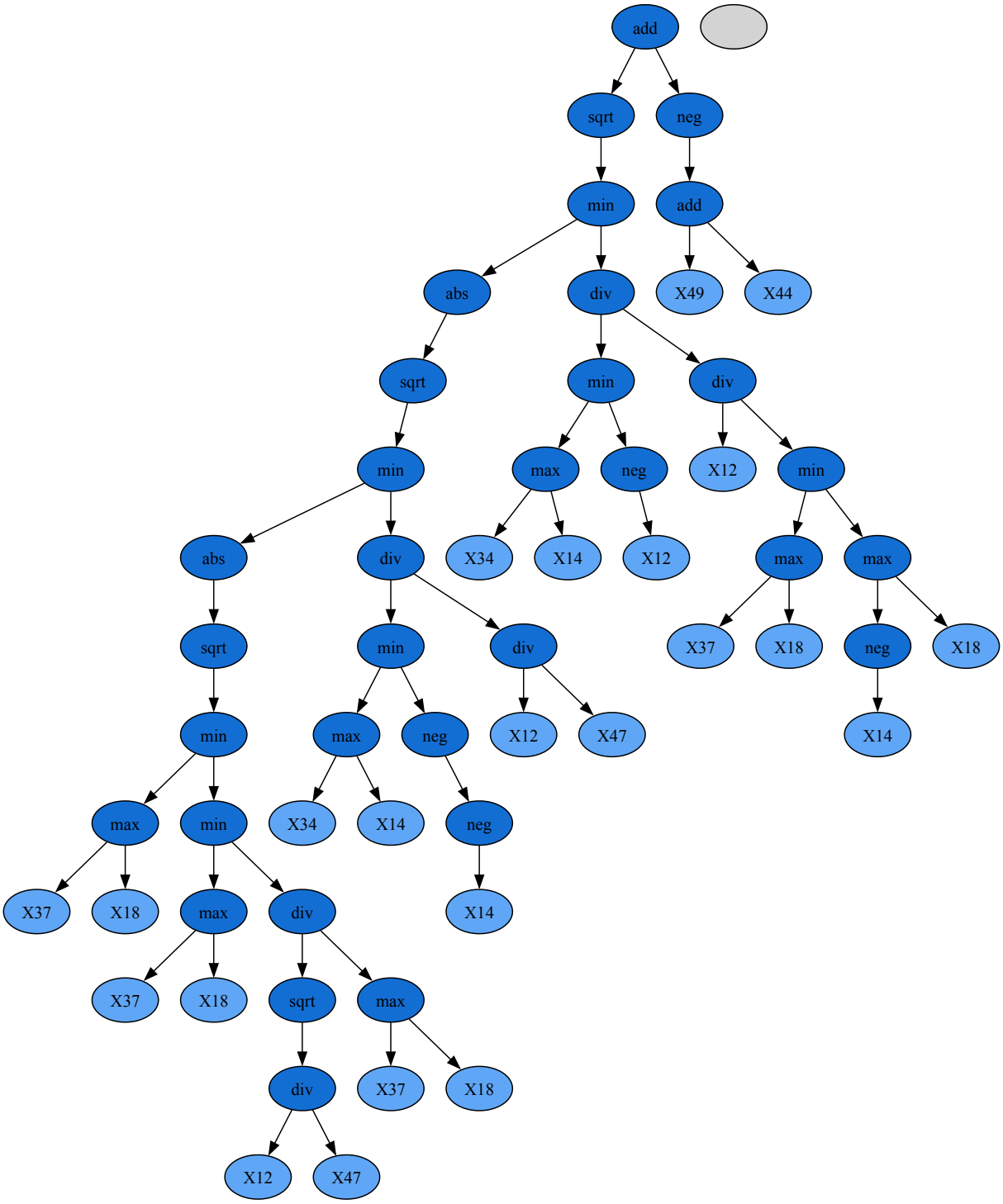


图 18　GPLearn 最佳特征树

5.4 基于多分类器的判别模型

在提取完特征后, 可运用分类器构建判别模型. 在本文中, 原始 54 维特征和 GPLearn 所得 80 维特征都将被使用, 并采用多种分类器进行构建判别模型, 准确率最高的模型将作为最终的判别模型. 原始 54 维特征采用三次核函数 SVM 模型、二次核函数 SVM 模型和线性核函数 SVM 模型进行构建判别模型.GPLearn 所得 80 维特征将用主成分分析进行降维, 依次保留 5、10、40 个主成分, 并均采用以决策树为基分类器的集成学习方法构建判别模型. 将这些模型分别记为模型 1、模型 2、$\cdots$、模型 6, 使用 Matlab 中的分类学习工具箱依次进行实验, 记录判别模型的准确率, 结果见表5.

表 5　六种分类模型准确率与总代价结果

模型	准确率 (%)	总代价
模型 1	89.50	63
模型 2	87.83	73
模型 3	85.67	86
模型 4	67.50	195
模型 5	71.00	174
模型 6	68.00	192

表中总代价表示该判别模型分类错误的数量, 由表中数据可知, 模型 1 与模型 5 分别在原始 54 维特征和 GPLearn 所得 80 维特征上取得最高准确率. 其中, 模型 1 为三次核函数 SVM 模型, 模型 5 为保留 10 个主成分的以决策树为基分类器的集成学习方法.

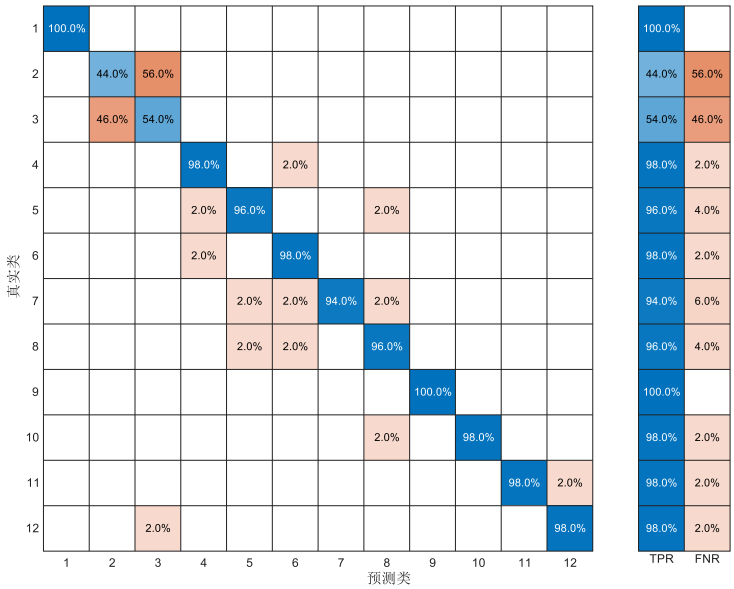


图 19　三次核函数 SVM 判别模型混淆矩阵

图19和图20分别为两个模型的混淆矩阵. 图中 TRP 表示真正率,FNR 表示假负率. 显然, 两种模型对第 2 类和第 3 类的判别准确率较低, 三次核函数 SVM 判别模型在其他类

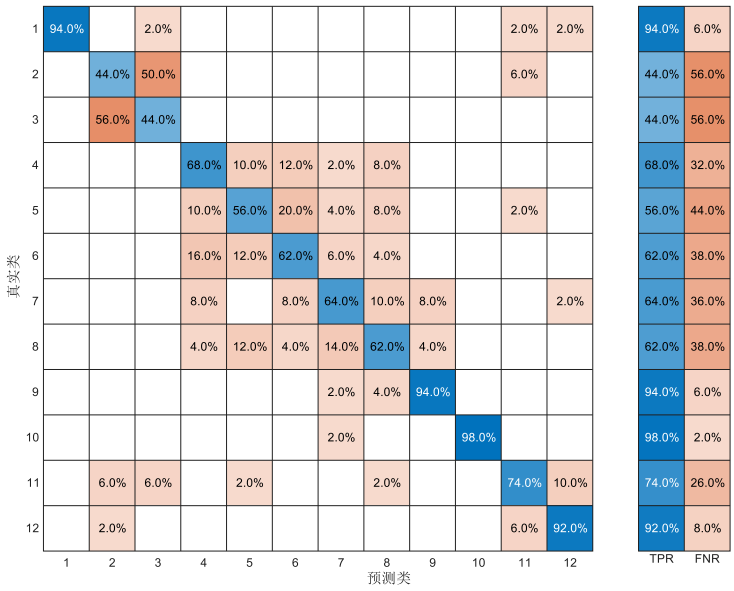


图 20　集成学习判别模型混淆矩阵

别上的判别准确率均大于 90%. 第 2 类和第 3 类分别为向左走和向右走, 二者的加速度和角速度特征均是相似度, 因此, 两种模型在这两类上的判别准确率较低.

因此, 本文采用三次核函数 SVM 模型作为最终的判别模型, 利用该模型可对附件 3 中数据进行判别, 判别结果见表 6.

表 6　附件 3 判别结果

实验编号	活动类别	实验编号	活动类别	实验编号	活动类别
SY1	5	SY11	11	SY21	5
SY2	1	SY12	4	SY22	7
SY3	12	SY13	11	SY23	10
SY4	10	SY14	7	SY24	5
SY5	6	SY15	5	SY25	2
SY6	6	SY16	11	SY26	8
SY7	7	SY17	5	SY27	1
SY8	4	SY18	5	SY28	5
SY9	6	SY19	12	SY29	7
SY10	8	SY20	11	SY30	10

6 问题三分析与求解

6.1 问题三分析

问题三旨在分析不同人员的同一活动状态是否存在差异, 以及活动状态数据是否与实验人员的年龄、身高和体重存在关系. 在分析前, 考虑到实验人员 1-3 的标签存在不准确性, 因此仅采用实验 4-13 的数据进行分析. 先求出 10 个实验人员的 6 个速度的平

均值. 之后, 分别求加速度和角速度的 “合速度”.“合速度” 可置于一个 10×12 的矩阵当中, 便于后续分析. 在分析不同人员的同一活动状态是否存在差异时, 可针对矩阵的每一列进行单因素方差分析, 筛选出存在差异的活动状态. 在此基础上, 对有差异的活动状态, 按 “合速度” 大小进行排列, 以分析年龄、身高和体重与活动状态的关系. 最后, 可选择极差最大的活动状态, 并计算 5 名 unknow 人员在该活动状态下的 “合速度”, 寻找与之差值的绝对值最小者作为人员画像.

6.2 实验人员不同活动状态特征数计算

实验人员不同活动状态最基本的表征是 2 种传感器 6 个方向上的速度值, 因此可以用 (合加速度, 合角速度) 这一对数进行特征表示. 合速度的计算公式如式 (15) 所示. 不妨称这一对数为不同活动状态的特征数.10 名实验人员的每种活动状态的每次实验的合加速度见支撑材料.

$$V = \sqrt{v_x^2 + v_y^2 + v_z^2} \tag{15}$$

6.3 基于特征数的活动状态差异方差分析模型

通过计算每位实验人员在每种活动状态下每次实验的特征数, 可针对每种活动状态进行单因素方差分析. 在单因素方差分析中, 人员类型为因素, 具体的人员为水平, 共有 10 个水平. 每个人员在每种活动状态下的 5 次实验的特征数为观察值, 每个水平共有 5 个观察值. 取置信度水平 $\alpha = 0.05$, 分别对合加速度和合角速度进行单因素方差分析, 方差分析的 p 值记录在表7中.

表 7　单因素方差分析结果

活动类型编号	合加速度	合角速度
1	1.91×10^{-8}	0.2859
2	3.68×10^{-6}	0.0162
3	4.60×10^{-7}	0.0022
4	1.14×10^{-17}	1.22×10^{-10}
5	4.54×10^{-5}	9.45×10^{-10}
6	1.69×10^{-3}	6.99×10^{-8}
7	5.99×10^{-8}	8.39×10^{-6}
8	1.08×10^{-12}	4.59×10^{-14}
9	4.73×10^{-24}	6.98×10^{-5}
10	7.43×10^{-9}	0.3381
11	7.90×10^{-23}	0.1299
12	6.00×10^{-19}	0.1458

由表中数据可知, 若仅对合加速度进行方差分析, 则 12 个活动状态均因不同人员而产生差异. 但仅对合角速度进行方差分析, 则仅有 8 个活动状态因不同人员而产生差异.

因此, 本文采取合角速度的方差分析结果作为不同人员的同一活动状态差异性分析结果. 即向前走、躺下、乘坐电梯向上或向下移动不会因为人员的不同而产生差异.

6.4 基于特征数的活动状态与年龄、身高、体重关系分析

由前文的分析与计算可知, 所有人员在所有活动状态所有实验次数上的合加速度都十分接近, 但其量级都十分的小, 因此出现了 12 个活动状态均因不同人员而产生差异的现象. 这恰恰说明, 合加速度并不适合于分析活动状态与年龄、身高、体重之间的关系. 故本文仅采取合角速度对其进行分析.

首先, 对每个实验人员在 8 种有显著性差异的活动状态的实验数据取均值, 并按升序排列, 排列结果见支撑材料. 分析升序排列后的数据, 可得出如下几个结论:

(1) 对于向左走、向右走、步行上下楼和向前跑, 年龄越小, 则合角速度往往越大, 这是因为年龄小的关节活动度高, 完成动作的速度更快, 角速度更大.

(2) 对于跳跃, 身高矮的合角速度更大, 这是因为往往身高矮的爆发力更强且更灵活.

(3) 对于坐下, 体重大的人员合角速度更小, 这是因为体重基数大, 动作更加迟缓.

(4) 对于站立, 结果与坐下类似, 不过体重过轻的, 合角速度也会过小.

由此可见, 年龄主要影响行走、跑步等关节跨度大的活动状态, 而身高则影响重心在垂直平面内运动的活动状态, 如跳跃. 年龄和身高对合角速度的影响具有单调影响性. 体重则会影响一些变化幅度较小的活动状态, 与年龄和身高的单调影响性不同, 体重的两级都会导致合角速度的变小.

6.5 基于特征数极差的人员判别模型

在上述分析的基础上, 选择均值极差最小的活动类型来进行人员判断. 由于人员只有 10 个, 极差越小, 则人员之间的差异越小, 越紧致, 越利于判别. 其中极差最小的活动类型为坐下. 因此需计算 5 名 unknow 人员在坐下实验上的合角速度, 其计算结果见表8.

表 8　5 名 unknow 人员在坐下实验上的合角速度

	unknow1	**unknow2**	**unknow3**	**unknow4**	**unknow5**
合角速度	0.4879	1.4686	3.8293	0.0334	0.1168

由表中数据可见,5 名 unknow 人员在坐下实验中的合角速度差异显著, 与附件 2 中 5 次实验的均值进行对比, 按数值接近度进行判别, 可得结果, 如表9所示.

表 9　5 名 unknow 人员的判别结果

	unknow1	**unknow2**	**unknow3**	**unknow4**	**unknow5**
判别结果	Person10	Person7	Person5	Person8	Person13

7 模型评价与推广

7.1 模型优点

本文提出的模型共有以下 3 个优点.

- 优点 1: 采用多种模型进行组合, 具有较好的耦合性.
- 优点 2: 问题 1 中所得分类模型分类效果好, 类与类之间的差异性鲜明.
- 优点 3: 问题 2 中构建的判别模型准确率高.

7.2 模型缺点

本文提出的模型具有以下 2 个缺点.

- 缺点 1: 分类模型所得类别与实际类别对应性较低.
- 缺点 2: 不易区分向左走和向右走两种活动状态.

7.3 模型改进与推广

本文所构建的模型可做如下推广.

- 推广 1: 进一步增加时域和频域特征的提取, 以提高对左右走的判别准确率.
- 推广 2: 增加活动类别数量, 应用于其他复杂动作, 如体操, 吊环等.
- 推广 3: 增加数据量, 以提高模型的泛化性能.

参考文献

[1] 杨佳现. 基于智能手机多传感器融合技术的人体活动识别研究[D]. 北京工业大学, 2020.

[2] Van der Maaten, Hinton. Visualizing Data using t-SNE[J]. JOURNAL OF MACHINE LEARNING RESEARCH,2008,Vol.9(11):2579-2605.

[3] 陈翠玲. 基于块对角表示的多核聚类方法研究[D]. 广西师范大学,2023.

[4] 李路. 基于多传感器的人体运动模式识别研究[D]. 山东大学,2013.

[5] Pérez-Moroyoqui, René Rodrí guez-Romo, Suemi; Ibáñez-Orozco, Oscar. Genetic algorithm for the design of a multi-

band microstrip antenna with self-avoiding geometry obtained by backtracking.[J]. International Journal of Communication Systems,2022,Vol.35(18): 1-11

[6] Vimarsh Sathia;Venkataramana Ganesh;Shankara Rao Thejaswi Nanditale. Accelerating Genetic Programming using GPUs[J]. 2021.

[7] Y.Sasaki, K.Tanemura, Y.Tokuni, R.Miyadera and H.Manabe. Application of Symbolic Regression to Unsolved Mathematical Problems. 2023 International Conference on Artificial Intelligence and Applications (ICAIA) Alliance Technology Conference (ATCON-1), Bangalore,India,2023, pp.1-6

附录 A　文件列表

文件名	功能描述
fileImport.py	数据读取
filterData.py	数据预处理与特征计算
t-SNE.py	t-SNE 流形学习
k-Cluster.py	核 K-means 聚类函数
k-Cluster-Predict.py	用核 K-means 聚类对问题 1 进行聚类
GPLearn.py	GPLearn 程序
gplearntrainClassifier.m	GPLearn 所得 80 维特征对应的 3 个判别模型
trainClassifier.m	3 个 SVM 判别模型
HNQ3.m	问题三程序
问题一聚类结果.xlsx	问题 1 结果
附件 2 用问题一模型分类结果.xlsx	问题 2(1) 结果
问题二判别结果.xlsx	问题 2(2) 结果
时域频域特征	文件夹中含 Person1-Person13 以及 5 个 Unknown 的 54 维特征
合加速度.xlsx	合加速度计算结果与方差分析结果
合角速度.xlsx	合角速度计算结果与方差分析结果
活动状态与年龄、身高、体重的关系.xlsx	问题三分析过程数据

附录 B　代码

fileImport.py

```
import os

import pandas as pd

from filterData import *

# 获取文件路径
def import_excel_file(path):
    excel_files = [f for f in os.listdir(path)]
    for file in excel_files:
```

```
        file_path = os.path.join(path, file)
        file_name_file.append(file_path)

def import_excel_xlsx(path):
    excel_files = [f for f in os.listdir(path) if f.endswith('.xlsx') or f.endswith('.xls')]
    for file in excel_files:
        file_path = os.path.join(path, file)
        file_name_xlsx.append(file_path)

file_name_file = []
import_excel_file('D:\SoftFamily\Toolbox-App\PyCharm Professional\Project\HunanProject\QA2\Persons\Person9')

file_name_xlsx = []

file_name = []
for file in file_name_file:
    import_excel_xlsx(file)
    file_name.append(file_name_xlsx)
    file_name_xlsx=[]

features_data = []

# 提取特征
for i, files in enumerate(file_name, 1):
    for file in files:
        features = features_extracted(file)
        features_data.append(features)
    # 将列表转换为DataFram
    df = pd.DataFrame(features_data)
    # 保存 DataFrame 到 Excel文件中
    df.to_excel(r'D:\SoftFamily\Toolbox-App\PyCharm Professional\Project\HunanProject\QA2' + '\Person' + str(i) + '.xlsx' , index=False)  # index=False表示不保存索引
```

```
    features_data=[]
```

filterData.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.ndimage import median_filter
from scipy.stats import kurtosis
from scipy.signal import stft

def import_file(file_path):
    """
    导入文件并返回DataFrame

    参数:
    file_path - 文件路径

    返回:
    DataFrame - 导入的文件数据
    """

    # 使用pandas读取文件并返回 DataFrame
    df = pd.read_excel(file_path)
    return df

def calculate_fft(data):
    """
    计算给定数据的快速傅里叶变换（FFT）。

    该函数使用短时傅里叶变换（STFT）将时域信号转换为频域信号，
    以便分析信号在不同频率下的表现。

    参数:
    data: 一维数组，包含待分析的时域信号。

```

```
    返回值:
    f: 一维数组，频率轴。
    t: 一维数组，时间轴。
    Zxx: 二维数组，时频谱图。
    """
    """
    执行快速傅里叶变换（STFT，Short-Time Fourier Transform），将时域信号转换为频域信号
    STFT参数
    """
    # 定义每个段的长度，用于确定STFT中每个窗口的大小。
    nperseg = 200  # 每个段的长度
    # 定义段与段之间的重叠点数，用于控制时频谱图的分辨率。
    noverlap = 100 # 段之间重叠的点数
    # 定义使用的窗口函数类型，这里选择汉明窗。
    window = 'rectangular'  # 窗口类型

    # 调用scipy库的stft函数计算快速傅里叶变换。
    # 返回值包括频率轴f、时间轴t和时频谱图Zxx。
    f, t, Zxx = stft(data, fs=100, nperseg=nperseg, noverlap=noverlap, window=window)
    return f, t, Zxx

def calculate_spectrum_energy(f, Zxx):
    """
    计算频谱能量。

    参数:
    f: 一维数组，频率轴。
    Zxx: 二维数组，时频谱图。

    返回值:
    energy: 一维数组，每个频率上的能量。
    """

```

```
    # 计算频谱幅度
    Pxx = np.abs(Zxx) ** 2

    # 提取总的频谱能量（对所有时间窗口和频率进行累加）
    energy = np.sum(Pxx)

    return energy


def features_extracted(file):
    fig_file = file.replace('Persons', 'Pictures')

    # 1.中值滤波
    np.random.seed(0)  # 为了可重复性设置随机种子

    df = import_file(file)

    df = df.to_numpy()

    # 注意：对于一维数据，窗口大小必须是奇数
    window_size = 21  # 示例窗口大小

    filtered_data = np.zeros_like(df)  # 创建一个与原始数据形状相同的数组来存储结果

    for i in range(df.shape[1]):  # 遍历每个维度（列）
        filtered_data[:, i] = median_filter(df[:, i], size=window_size)  # 对每个维度的值应用中值滤波

    # Acc 对比
    plt.figure(figsize=(10, 6))
    plt.plot(df[:, :3], label={'Original AccX', 'Original AccY', 'Original AccZ'})
    plt.legend()
    fig_file_filter = fig_file.replace('.xlsx', 'AccFilter1.
```

```
svg')
 plt.savefig(fig_file_filter)
 plt.show()

 # Acc 对比
 plt.figure(figsize=(10, 6))
 plt.plot(filtered_data[:, :3], label={'Filtered AccX', '
Filtered AccY', 'Filtered AccZ'})
 plt.legend()
 fig_file_filter = fig_file.replace('.xlsx', 'AccFilter2.
svg')
 plt.savefig(fig_file_filter)
 plt.show()

 # Gryo对比
 plt.figure(figsize=(10, 6))
 plt.plot(df[:, 3:], label={'Original GryoX', 'Original
GryoY', 'Original GryoZ'})
 plt.legend()
 fig_file_filter = fig_file.replace('.xlsx', 'GryoFilter1.
svg')
 plt.savefig(fig_file_filter)
 plt.show()

 # Gryo对比
 plt.figure(figsize=(10, 6))
 plt.plot(filtered_data[:, 3:], label={'Filtered GryoX', '
Filtered GryoY', 'Filtered GryoZ'})
 plt.legend()
 fig_file_filter = fig_file.replace('.xlsx', 'GryoFilter2.
svg')
 plt.savefig(fig_file_filter)
 plt.show()

 # 2.数据归一化
```

```
126     df_normalized_Acc = (filtered_data[:, :3] - filtered_data
    [:, :3].mean()) / 12
127     df_normalized_Gyro = (filtered_data[:, 3:] - filtered_data
    [:, 3:].mean()) / 1000
128 
129     df_normalized = np.concatenate([df_normalized_Acc,
    df_normalized_Gyro], axis=1)
130 
131     plt.figure(figsize=(10, 6))
132     plt.plot(df_normalized_Acc, label={'AccX', 'AccY', 'AccZ'
    })
133     plt.legend()
134     fig_file_normalized_acc = fig_file.replace('.xlsx', '
    NormalizedAcc.svg')
135     plt.savefig(fig_file_normalized_acc)
136     plt.show()
137 
138     plt.figure(figsize=(10, 6))
139     plt.plot(df_normalized_Gyro, label={'GyroX', 'GyroY', '
    GyroZ'})
140     plt.legend()
141     fig_file_normalized_gyro = fig_file.replace('.xlsx', '
    NormalizedGyro.svg')
142     plt.savefig(fig_file_normalized_gyro)
143     plt.show()
144 
145     # 4.时域特征提取
146     # 4.1 均值
147     mean_features = np.mean(df_normalized, axis=0)
148     # 4.2 标准差
149     std_features = np.std(df_normalized, axis=0)
150     # 4.3 方差
151     var_features = np.var(df_normalized, axis=0)
152     # 4.4 最大值
153     max_features = np.max(df_normalized, axis=0)
```

```
    # 4.5 最小值
    min_features = np.min(df_normalized, axis=0)
    # 4.6 峰度
    kurtosis_features = kurtosis(df_normalized, axis=0)
    # 4.7 四分位距
    quartile_features = np.transpose(np.quantile(df_normalized, [0.25, 0.5, 0.75], axis=0))

    quartile_features = quartile_features[:, 2] - quartile_features[:, 0]

    # 4.8 均方根
    rms_features = np.sqrt(np.mean(df_normalized ** 2, axis=0))

    # 5.频域特征提取
    # 5.1 滑动窗口快速傅里叶变换
    energy = []

    # 5.2 计算频谱能量
    for i in range(df_normalized.shape[1]):
        f, t, Zxx = calculate_fft(df_normalized[:, i])
        energy_temp = calculate_spectrum_energy(f, Zxx)
        energy.append(energy_temp)

    energy = np.array(energy)

    # 6. 特征合并
    features = np.concatenate(
        [mean_features, std_features, var_features, max_features, min_features, kurtosis_features, quartile_features,
         rms_features, energy], axis=0)

    return features
```

t-SNE.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import NMF,PCA
from sklearn.manifold import TSNE

# 编写使用GPU运行的代码

file_path_person1 = r'D:\SoftFamily\Toolbox-App\PyCharm Professional\Project\HunanProject\QA1\Person1.xlsx'
file_path_person2 = r'D:\SoftFamily\Toolbox-App\PyCharm Professional\Project\HunanProject\QA1\Person2.xlsx'
file_path_person3 = r'D:\SoftFamily\Toolbox-App\PyCharm Professional\Project\HunanProject\QA1\Person3.xlsx'

df_person1 = pd.read_excel(file_path_person1)
df_person2 = pd.read_excel(file_path_person2)
df_person3 = pd.read_excel(file_path_person3)

df_persons = pd.concat([df_person1, df_person2, df_person3], axis=0)

file_path_labels = r'D:\SoftFamily\Toolbox-App\PyCharm Professional\Project\HunanProject\QA1\labels.xlsx'
df_labels = pd.read_excel(file_path_labels)

df_persons = df_persons.reset_index(drop=True)
df_labels = df_labels.reset_index(drop=True)

df = pd.concat([df_persons, df_labels], axis=1)

tsne = TSNE(random_state=42)
data_tsne = tsne.fit_transform(df)

```

```
pca=PCA(n_components=2)
pca.fit(df)

data_pca=pca.transform(df)

# tsne = TSNE(random_state=42)
# df_P1 = pd.concat([df_person1, df_labels.iloc[:60, :]], axis=1)
# df_P2 = pd.concat([df_person2, df_labels.iloc[60:120, :].reset_index(drop=True)] , axis=1)
# df_P3 = pd.concat([df_person3, df_labels.iloc[120:, :].reset_index(drop=True)], axis=1)
# data_tsne_person1 = tsne.fit_transform(df_P1)
# data_tsne_person2 = tsne.fit_transform(df_P2)
# data_tsne_person3 = tsne.fit_transform(df_P3)
#
# data_tsne = np.concatenate([data_tsne_person1, data_tsne_person2, data_tsne_person3])

colors=['#476A2A','#7851B8','#BD3430','#4A2D4E',
        '#A83683','#4E655E','#853541','#3A3120',
        '#A8CABA','#B9C2A6','#5E7C7D','#C5D4BE']

# plt.figure(figsize=(10, 10))
# plt.xlim(data_tsne[:, 0].min(), data_tsne[:, 0].max())
# plt.ylim(data_tsne[:, 1].min(), data_tsne[:, 1].max())
#
# df_labels = np.array(df_labels)
#
# for i in range(len(data_tsne)):
#     plt.text(data_tsne[i, 0], data_tsne[i, 1], df_labels[i],
#              color=colors[df_labels[i, 0]],
#              fontdict={'weight': 'bold', 'size': 9})
# plt.xlabel('t-SNE feature 0')
# plt.ylabel('t-SNE feature 1')
```

```
# plt.show()

plt.figure(figsize=(10, 10))
plt.xlim(data_tsne[:, 0].min()-1, data_tsne[:, 0].max()+1)
plt.ylim(data_tsne[:, 1].min()-1, data_tsne[:, 1].max()+1)

# df_labels_person1 = np.array(df_labels.iloc[:60, :])
#
# for i in range(len(data_tsne_person1)):
#     plt.text(data_tsne_person1[i, 0], data_tsne_person1[i, 1], df_labels_person1[i],
#              color=colors[df_labels_person1[i, 0]],
#              fontdict={'weight': 'bold', 'size': 9})
# plt.xlabel('t-SNE feature 0')
# plt.ylabel('t-SNE feature 1')
# plt.savefig('person1_kCluster.svg')
# plt.show()
#
#
# plt.figure(figsize=(10, 10))
# plt.xlim(data_tsne_person2[:, 0].min()-1, data_tsne_person2[:, 0].max()+1)
# plt.ylim(data_tsne_person2[:, 1].min()-1, data_tsne_person2[:, 1].max()+1)
#
# df_labels_person2 = np.array(df_labels.iloc[60:120, :])
#
# for i in range(len(data_tsne_person2)):
#     plt.text(data_tsne_person2[i, 0], data_tsne_person2[i, 1], df_labels_person2[i],
#              color=colors[df_labels_person2[i, 0]],
#              fontdict={'weight': 'bold', 'size': 9})
# plt.xlabel('Person2 t-SNE feature 0')
# plt.ylabel('Person2 t-SNE feature 1')
# plt.savefig('person2_kCluster.svg')
```

```
# plt.show()
#
# plt.figure(figsize=(10, 10))
# plt.xlim(data_tsne_person3[:, 0].min()-1, data_tsne_person3[:, 0].max()+1)
# plt.ylim(data_tsne_person3[:, 1].min()-1, data_tsne_person3[:, 1].max()+1)
#
# df_labels_person3 = np.array(df_labels.iloc[120:, :])
#
# for i in range(len(data_tsne_person3)):
#     plt.text(data_tsne_person3[i, 0], data_tsne_person3[i, 1], df_labels_person3[i],
#              color=colors[df_labels_person3[i, 0]],
#              fontdict={'weight': 'bold', 'size': 9})
# plt.xlabel('Person3 t-SNE feature 0')
# plt.ylabel('Person3 t-SNE feature 1')
# plt.savefig('person3_kCluster.svg')
# plt.show()

df_labels = np.array(df_labels)

for i in range(len(data_tsne)):
    plt.text(data_tsne[i, 0], data_tsne[i, 1], df_labels[i],
             color=colors[df_labels[i, 0]],
             fontdict={'weight': 'bold', 'size': 9})
plt.xlabel('t-SNE feature 0')
plt.ylabel('t-SNE feature 1')
plt.savefig('person1_kCluster.svg')
plt.show()
```

k-Cluster.py

```
import numpy as np
import pandas as pd
from sklearn.metrics.pairwise import rbf_kernel
```

```
from joblib import dump
from sklearn.metrics.pairwise import polynomial_kernel

file_path_person1 = 'D:\SoftFamily\Toolbox-App\PyCharm Professional\Project\HunanProject\QA1\Person1.xlsx'
file_path_person2 = 'D:\SoftFamily\Toolbox-App\PyCharm Professional\Project\HunanProject\QA1\Person2.xlsx'
file_path_person3 = 'D:\SoftFamily\Toolbox-App\PyCharm Professional\Project\HunanProject\QA1\Person3.xlsx'

df_person1 = pd.read_excel(file_path_person1)
df_person2 = pd.read_excel(file_path_person2)
df_person3 = pd.read_excel(file_path_person3)


def strictly_balanced_constrained_kernel_kmeans(X, n_clusters, max_iter=100, gamma=None, max_cluster_size=15, dataset_sizes=[60, 60, 60]):
    """
    严格平衡约束核 K均值聚类算法，确保每个聚类严格包含来自多个数据集的指定数量样本。

    参数:
    X : array-like, shape (n_samples, n_features)
        训练实例向量。

    n_clusters : int
        聚类数量。

    max_iter : int, optional, default: 100
        最大迭代次数。

    gamma : float, optional
        RBF核函数参数。

```

```
    samples_per_dataset_per_cluster : int
        每个聚类中每个数据集的样本数。

    dataset_sizes : list of int
        各数据集样本数。

    返回:
    labels : array, shape (n_samples,)
        样本聚类标签。
    """
    # 初始化聚类中心
    centers = X[np.random.choice(X.shape[0], n_clusters, replace=False)]

    # 计算每个数据集样本在每个聚类中的目标数量
    target_dataset_counts = [max_cluster_size // len(dataset_sizes)] * len(dataset_sizes)
    remainder = max_cluster_size % len(dataset_sizes)
    for i in range(remainder):
        target_dataset_counts[i] += 1

    for _ in range(max_iter):
        # 计算核矩阵
        K = rbf_kernel(X, centers, gamma=gamma)

        # 创建一个空的标签数组和一个跟踪每个簇大小的计数器
        labels = np.full(X.shape[0], -1, dtype=int)
        cluster_sizes = np.zeros(n_clusters, dtype=int)
        dataset_counts = np.zeros((n_clusters, len(dataset_sizes)), dtype=int)

        # 遍历每个数据集，分配样本到簇
        start_idx = 0
        for dataset_idx, dataset_size in enumerate(dataset_sizes):
```

```
64              end_idx = start_idx + dataset_size
65
66              # 尝试分配样本到满足条件的簇
67              for sample_idx in range(start_idx, end_idx):
68                  # 找到当前样本与所有簇中心的最大相似度
69                  max_sim = -np.inf
70                  best_cluster = -1
71
72                  # 遍历每个簇
73                  for cluster_idx in range(n_clusters):
74                      # 如果当前簇未满，且数据集分布符合要求，检
    查相似度
75                      if cluster_sizes[cluster_idx] <
    max_cluster_size and \
76                         dataset_counts[cluster_idx][dataset_idx
    ] < target_dataset_counts[dataset_idx] and \
77                         K[sample_idx, cluster_idx] > max_sim:
78                          max_sim = K[sample_idx, cluster_idx]
79                          best_cluster = cluster_idx
80
81                  # 分配样本到找到的最佳簇
82                  if best_cluster != -1:
83                      labels[sample_idx] = best_cluster
84                      cluster_sizes[best_cluster] += 1
85                      dataset_counts[best_cluster][dataset_idx]
    += 1
86
87              start_idx = end_idx
88
89          # 检查是否所有簇都满足数据集分布的严格要求
90          valid_clusters = [i for i in range(n_clusters) if all(
    dataset_counts[i][k] == target_dataset_counts[k] for k in
    range(len(dataset_sizes)))]
91
92          # 如果所有簇都满足要求，或者达到最大迭代次数，停止迭代
```

```
93          if len(valid_clusters) == n_clusters or _ == max_iter
    - 1:
94              break
95
96          # 更新聚类中心
97          new_centers = np.zeros((len(valid_clusters), X.shape
    [1]))
98          for i, valid_cluster in enumerate(valid_clusters):
99              if cluster_sizes[valid_cluster] > 0:
100                 new_centers[i] = X[labels == valid_cluster].
    mean(axis=0)
101
102         # 如果聚类中心不再变化，则停止迭代
103         if np.all(centers[valid_clusters] == new_centers):
104             break
105
106         centers = new_centers
107
108     # 如果迭代结束但仍有簇未满足条件，进行额外处理
109     if len(valid_clusters) < n_clusters:
110         # 对于未满足条件的簇，重新分配剩余样本
111         remaining_samples = np.where(labels == -1)[0]
112         remaining_clusters = set(range(n_clusters)) - set(
    valid_clusters)
113         for sample_idx in remaining_samples:
114             # 找到剩余簇中与当前样本相似度最高的簇
115             max_sim = -np.inf
116             best_cluster = -1
117             for j in remaining_clusters:
118                 if K[sample_idx, j] > max_sim:
119                     max_sim = K[sample_idx, j]
120                     best_cluster = j
121             labels[sample_idx] = best_cluster
122             cluster_sizes[best_cluster] += 1
123             dataset_index = next(k for k, size in enumerate(
```

```
    dataset_sizes) if sum(dataset_sizes[:k]) <= sample_idx < sum
    (dataset_sizes[:k+1]))
124             dataset_counts[best_cluster][dataset_index] += 1
125 
126         # 更新所有聚类中心
127         new_centers = np.zeros((n_clusters, X.shape[1]))
128         for i in range(n_clusters):
129             if cluster_sizes[i] > 0:
130                 new_centers[i] = X[labels == i].mean(axis=0)
131         centers = new_centers
132 
133     return labels, centers
134 
135 
136 df_person1 = np.array(df_person1)
137 df_person2 = np.array(df_person2)
138 df_person3 = np.array(df_person3)
139 df = np.concatenate([df_person1, df_person2, df_person3])
140 # 应用核 K均值聚类
141 labels_persons, centers_persons =
    strictly_balanced_constrained_kernel_kmeans(df, n_clusters
    =12, gamma=0.1)
142 
143 
144 # 保存模型参数
145 model_params_persons = {
146     'centers': centers_persons,  # 从constrained_kernel_kmeans
    函数中获取的centers
147 }
148 
149 # labels_person2, centers_person2 =
    strictly_balanced_constrained_kernel_kmeans(df_person2,
    n_clusters=12, gamma=0.1)
150 # labels_person3, centers_person3 =
    strictly_balanced_constrained_kernel_kmeans(df_person3,
```

```
    n_clusters=12, gamma=0.1)
#
# model_params_person2 = {
#     'centers': centers_person2,  # 从
    constrained_kernel_kmeans函数中获取的centers
# }
#
# model_params_person3 = {
#     'centers': centers_person3,  # 从
    constrained_kernel_kmeans函数中获取的centers
# }
# dump(model_params_person2, 'kernel_kmeans_model_person2.
    joblib')
# dump(model_params_person3, 'kernel_kmeans_model_person3.
    joblib')

# 保存模型
dump(model_params_persons, '
    strictly_balanced_kernel_kmeans_model_persons.joblib')

print("Cluster labels:", labels_persons)
labels = pd.DataFrame(np.concatenate([labels_persons]))
labels.to_excel(r'D:\SoftFamily\Toolbox-App\PyCharm
    Professional\Project\HunanProject\QA1\labels.xlsx', index =
    False)
```

k-Cluster-Predict.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from joblib import load
from sklearn.manifold import TSNE

from sklearn.metrics.pairwise import rbf_kernel

```

```
# 加载模型参数
model_params_person1 = load('kernel_kmeans_model_person1.joblib')
model_params_person2 = load('kernel_kmeans_model_person2.joblib')
model_params_person3 = load('kernel_kmeans_model_person3.joblib')


# 提取模型参数
centers_person1 = model_params_person1['centers']
centers_person2 = model_params_person2['centers']
centers_person3 = model_params_person3['centers']


# 定义预测函数
def predict_kernel_kmeans(X_new, centers, gamma=None):
    # 计算新样本与聚类中心的核矩阵
    K = rbf_kernel(X_new, centers, gamma=gamma)

    # 聚类分配
    labels = K.argmax(axis=1)

    return labels


# 使用模型预测新样本，使用投票机制实现投票
X_new = pd.read_excel(r'D:\SoftFamily\Toolbox-App\PyCharm Professional\Project\HunanProject\QA2\PersonsQ2.xlsx')  # 新样本数据
predicted_labels_1 = pd.DataFrame(predict_kernel_kmeans(X_new.iloc[:, :-1], centers_person1))
predicted_labels_2 = pd.DataFrame(predict_kernel_kmeans(X_new.iloc[:, :-1], centers_person2))
predicted_labels_3 = pd.DataFrame(predict_kernel_kmeans(X_new.
```

```
    iloc[:, :-1], centers_person3))
assert (predicted_labels_1.columns == predicted_labels_2.
    columns).all() and (predicted_labels_1.columns ==
    predicted_labels_3.columns).all()

# 聚合预测结果
predicted_labels = pd.DataFrame(index=predicted_labels_1.index
    , columns=predicted_labels_1.columns)

for i in predicted_labels_1.columns:
    # 将三个 DataFrame 的列堆叠在一起
    stacked_series = pd.concat([predicted_labels_1[i],
    predicted_labels_2[i], predicted_labels_3[i]], axis=1)
    # 应用多数投票
    predicted_labels.iloc[i, :] = stacked_series.mode(axis=1)
    [0]

df = pd.concat([X_new.iloc[:, :-1], predicted_labels], axis=1)

tsne = TSNE(random_state=42)
data_tsne = tsne.fit_transform(df)

colors=['#476A2A', '#7851B8', '#BD3430', '#4A2D4E',
        '#A83683', '#4E655E', '#853541', '#3A3120',
        '#A8CABA', '#B9C2A6', '#5E7C7D', '#C5D4BE']

plt.figure(figsize=(10, 10))
plt.xlim(data_tsne[:, 0].min(), data_tsne[:,0].max())
plt.ylim(data_tsne[:, 1].min(), data_tsne[:,1].max())

for i in range(len(data_tsne)):
    plt.text(data_tsne[i,0],data_tsne[i,1], df.iloc[i, -1],
             color=colors[df.iloc[i, -1]],
             fontdict={'weight':'bold','size':9})
plt.xlabel('t-SNE feature 0')
```

```
plt.ylabel('t-SNE feature 1')
plt.show()
```

GPLearn.py

```
from gplearn.genetic import SymbolicTransformer
from IPython.display import Image
import pydotplus
import numpy as np
import pandas as pd

df = pd.read_excel('D:\\desktop\\hunan\\train.xlsx')
column_name = 'Lable'  # 要去掉的列名
train = df.drop(column_name, axis=1)
y_label = df['Lable']

function_set = ['add', 'sub', 'mul', 'div', 'log', 'sqrt', '
   abs', 'neg', 'max', 'min']

st = SymbolicTransformer(
    generations=20,
    population_size=500,
    hall_of_fame=100,
    n_components=80,
    function_set=function_set,
    parsimony_coefficient=0.0005,
    max_samples=0.9,
    verbose=1,
    random_state=0,
    n_jobs=3
)

st.fit(np.array(train.astype(np.float32)), np.array(y_label))


graph = st._best_programs[0].export_graphviz()
```

```
graph = pydotplus.graphviz.graph_from_dot_data(graph)

df = pd.read_excel('D:\\desktop\\hunan\\PersonUnknown.xlsx')

pd.DataFrame(
    st.transform(
        np.array(
            df
        )
    )
).to_excel('PersonUnknowGp.xlsx',index=False)

# 将图形保存为SVG文件
graph.write_svg('output.svg')
```

gplearntrainClassifier.m

```
function [trainedClassifier, validationAccuracy] =
   gplearntrainClassifier(trainingData)
% [trainedClassifier, validationAccuracy] = trainClassifier(
   trainingData)。
%
%  输入:
%      trainingData: 一个包含导入 App 中的预测变量和响应列的
   表。
%
%
%  输出:
%      trainedClassifier: 一个包含训练的分类器的结构体。该结构
   体中具有各种关于所训练分
%       类器的信息的字段。
%
%      trainedClassifier.predictFcn: 一个对新数据进行预测的函
   数。
%
%      validationAccuracy: 以百分比形式表示验证准确度的双精度
```

```
    值。在 App 中，"模型"
15  %       窗格显示每个模型的验证准确度。
16  %
17  % 使用该代码基于新数据来训练模型。要重新训练分类器，请使用原始
        数据或新数据作为输入参数
18  % trainingData 从命令行调用该函数。
19  %
20  % 例如，要重新训练基于原始数据集 T 训练的分类器，请输入：
21  %   [trainedClassifier, validationAccuracy] = trainClassifier(
        T)
22  %
23  % 要使用返回的 "trainedClassifier" 对新数据 T2 进行预测，请使
        用
24  %   [yfit,scores] = trainedClassifier.predictFcn(T2)
25  %
26  % T2 必须是一个表，其中至少包含与训练期间使用的预测变量列相同
        的预测变量列。有关详细信息，请
27  % 输入：
28  %   trainedClassifier.HowToPredict
29
30
31  % 提取预测变量和响应
32  % 以下代码将数据处理为合适的形状以训练模型。
33  %
34  inputTable = trainingData;
35  predictorNames = {'x0', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', '
        x7', 'x8', 'x9', 'x10', 'x11', 'x12', 'x13', 'x14', 'x15', '
        x16', 'x17', 'x18', 'x19', 'x20', 'x21', 'x22', 'x23', 'x24'
        , 'x25', 'x26', 'x27', 'x28', 'x29', 'x30', 'x31', 'x32', '
        x33', 'x34', 'x35', 'x36', 'x37', 'x38', 'x39', 'x40', 'x41'
        , 'x42', 'x43', 'x44', 'x45', 'x46', 'x47', 'x48', 'x49', '
        x50', 'x51', 'x52', 'x53', 'x54', 'x55', 'x56', 'x57', 'x58'
        , 'x59', 'x60', 'x61', 'x62', 'x63', 'x64', 'x65', 'x66', '
        x67', 'x68', 'x69', 'x70', 'x71', 'x72', 'x73', 'x74', 'x75'
        , 'x76', 'x77', 'x78', 'x79'};
```

```
predictors = inputTable(:, predictorNames);
response = inputTable.Lable;
isCategoricalPredictor = [false, false, false, false, false,
   false, false, false, false, false, false, false, false,
   false, false, false, false, false, false, false, false,
   false, false, false, false, false, false, false, false,
   false, false, false, false, false, false, false, false,
   false, false, false, false, false, false, false, false,
   false, false, false, false, false, false, false, false,
   false, false, false, false, false, false, false, false,
   false, false, false, false, false, false, false, false,
   false, false, false, false, false, false, false, false,
   false, false, false];
classNames = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12];

% 将 PCA 应用于预测变量矩阵。
% 仅对数值预测变量运行 PCA。PCA 不会对分类预测变量进行任何处
   理。
isCategoricalPredictorBeforePCA = isCategoricalPredictor;
numericPredictors = predictors(:, ~isCategoricalPredictor);
numericPredictors = table2array(varfun(@double,
   numericPredictors));
% 在 PCA 中必须将 'inf' 值视为缺失数据。
numericPredictors(isinf(numericPredictors)) = NaN;
numComponentsToKeep = min(size(numericPredictors,2), 10);
[pcaCoefficients, pcaScores, ~, ~, explained, pcaCenters] =
   pca(...
    numericPredictors, ...
    'NumComponents', numComponentsToKeep);
predictors = [array2table(pcaScores(:,:)), predictors(:,
   isCategoricalPredictor)];
isCategoricalPredictor = [false(1,numComponentsToKeep), true
   (1,sum(isCategoricalPredictor))];

% 训练分类器
```

```
% 以下代码指定所有分类器选项并训练分类器。
template = templateTree(...
    'MaxNumSplits', 599, ...
    'NumVariablesToSample', 'all');
classificationEnsemble = fitcensemble(...
    predictors, ...
    response, ...
    'Method', 'Bag', ...
    'NumLearningCycles', 500, ...
    'Learners', template, ...
    'ClassNames', classNames);

% 使用 predict 函数创建结果结构体
predictorExtractionFcn = @(t) t(:, predictorNames);
pcaTransformationFcn = @(x) [ array2table((table2array(varfun(
   @double, x(:, ~isCategoricalPredictorBeforePCA))) -
   pcaCenters) * pcaCoefficients), x(:,
   isCategoricalPredictorBeforePCA) ];
ensemblePredictFcn = @(x) predict(classificationEnsemble, x);
trainedClassifier.predictFcn = @(x) ensemblePredictFcn(
   pcaTransformationFcn(predictorExtractionFcn(x)));

% 向结果结构体中添加字段
trainedClassifier.RequiredVariables = {'x0', 'x1', 'x2', 'x3',
    'x4', 'x5', 'x6', 'x7', 'x8', 'x9', 'x10', 'x11', 'x12', '
   x13', 'x14', 'x15', 'x16', 'x17', 'x18', 'x19', 'x20', 'x21'
   , 'x22', 'x23', 'x24', 'x25', 'x26', 'x27', 'x28', 'x29', '
   x30', 'x31', 'x32', 'x33', 'x34', 'x35', 'x36', 'x37', 'x38'
   , 'x39', 'x40', 'x41', 'x42', 'x43', 'x44', 'x45', 'x46', '
   x47', 'x48', 'x49', 'x50', 'x51', 'x52', 'x53', 'x54', 'x55'
   , 'x56', 'x57', 'x58', 'x59', 'x60', 'x61', 'x62', 'x63', '
   x64', 'x65', 'x66', 'x67', 'x68', 'x69', 'x70', 'x71', 'x72'
   , 'x73', 'x74', 'x75', 'x76', 'x77', 'x78', 'x79'};
trainedClassifier.PCACenters = pcaCenters;
trainedClassifier.PCACoefficients = pcaCoefficients;
```

```
trainedClassifier.ClassificationEnsemble =
    classificationEnsemble;
trainedClassifier.About = '此结构体是从分类学习器 R2024b 导出
    的训练模型。';
trainedClassifier.HowToPredict = sprintf('要对新表 T 进行预
    测，请使用: \n [yfit,scores] = c.predictFcn(T) \n将 ''c'' 替
    换为作为此结构体的变量的名称，例如 ''trainedModel''。\n \n表
     T 必须包含由以下内容返回的变量: \n c.RequiredVariables \n变
    量格式(例如矩阵/向量、数据类型)必须与原始训练数据匹配。\n忽
    略其他变量。\n \n有关详细信息，请参阅 <a href="matlab:
    helpview(fullfile(docroot, ''stats'', ''stats.map''), ''
    appclassification_exportmodeltoworkspace'')">How to predict
    using an exported model</a>。');

% 提取预测变量和响应
% 以下代码将数据处理为合适的形状以训练模型。
%
inputTable = trainingData;
predictorNames = {'x0', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', '
    x7', 'x8', 'x9', 'x10', 'x11', 'x12', 'x13', 'x14', 'x15', '
    x16', 'x17', 'x18', 'x19', 'x20', 'x21', 'x22', 'x23', 'x24'
    , 'x25', 'x26', 'x27', 'x28', 'x29', 'x30', 'x31', 'x32', '
    x33', 'x34', 'x35', 'x36', 'x37', 'x38', 'x39', 'x40', 'x41'
    , 'x42', 'x43', 'x44', 'x45', 'x46', 'x47', 'x48', 'x49', '
    x50', 'x51', 'x52', 'x53', 'x54', 'x55', 'x56', 'x57', 'x58'
    , 'x59', 'x60', 'x61', 'x62', 'x63', 'x64', 'x65', 'x66', '
    x67', 'x68', 'x69', 'x70', 'x71', 'x72', 'x73', 'x74', 'x75'
    , 'x76', 'x77', 'x78', 'x79'};
predictors = inputTable(:, predictorNames);
response = inputTable.Lable;
isCategoricalPredictor = [false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
```

```
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false];
classNames = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12];

% 执行交叉验证
KFolds = 5;
cvp = cvpartition(response, 'KFold', KFolds);
% 将预测初始化为适当的大小
validationPredictions = response;
numObservations = size(predictors, 1);
numClasses = 12;
validationScores = NaN(numObservations, numClasses);
for fold = 1:KFolds
    trainingPredictors = predictors(cvp.training(fold), :);
    trainingResponse = response(cvp.training(fold), :);
    foldIsCategoricalPredictor = isCategoricalPredictor;

    % 将 PCA 应用于预测变量矩阵。
    % 仅对数值预测变量运行 PCA。PCA 不会对分类预测变量进行任何处理。
    isCategoricalPredictorBeforePCA = foldIsCategoricalPredictor;
    numericPredictors = trainingPredictors(:, ~foldIsCategoricalPredictor);
    numericPredictors = table2array(varfun(@double, numericPredictors));
    % 在 PCA 中必须将 'inf' 值视为缺失数据。
    numericPredictors(isinf(numericPredictors)) = NaN;
    numComponentsToKeep = min(size(numericPredictors,2), 10);
    [pcaCoefficients, pcaScores, ~, ~, explained, pcaCenters] = pca(...
```

```
        numericPredictors, ...
        'NumComponents', numComponentsToKeep);
    trainingPredictors = [array2table(pcaScores(:,:)),
trainingPredictors(:, foldIsCategoricalPredictor)];
    foldIsCategoricalPredictor = [false(1,numComponentsToKeep)
, true(1,sum(foldIsCategoricalPredictor))];

    % 训练分类器
    % 以下代码指定所有分类器选项并训练分类器。
    template = templateTree(...
        'MaxNumSplits', 599, ...
        'NumVariablesToSample', 'all');
    classificationEnsemble = fitcensemble(...
        trainingPredictors, ...
        trainingResponse, ...
        'Method', 'Bag', ...
        'NumLearningCycles', 500, ...
        'Learners', template, ...
        'ClassNames', classNames);

    % 使用 predict 函数创建结果结构体
    pcaTransformationFcn = @(x) [ array2table((table2array(
varfun(@double, x(:, ~isCategoricalPredictorBeforePCA))) -
pcaCenters) * pcaCoefficients), x(:,
isCategoricalPredictorBeforePCA) ];
    ensemblePredictFcn = @(x) predict(classificationEnsemble,
x);
    validationPredictFcn = @(x) ensemblePredictFcn(
pcaTransformationFcn(x));

    % 向结果结构体中添加字段

    % 计算验证预测
    validationPredictors = predictors(cvp.test(fold), :);
    [foldPredictions, foldScores] = validationPredictFcn(
```

```
    validationPredictors);
142
143     % 按原始顺序存储预测
144     validationPredictions(cvp.test(fold), :) = foldPredictions
    ;
145     validationScores(cvp.test(fold), :) = foldScores;
146 end
147
148 % 计算验证准确度
149 correctPredictions = (validationPredictions == response);
150 isMissing = isnan(response);
151 correctPredictions = correctPredictions(~isMissing);
152 validationAccuracy = sum(correctPredictions)/length(
    correctPredictions);
```

trainClassifier.m

```
1  function [trainedClassifier, validationAccuracy] =
    trainClassifier(trainingData)
2  % [trainedClassifier, validationAccuracy] = trainClassifier(
    trainingData)
3  %
4  %  输入:
5  %      trainingData: 一个包含导入 App 中的预测变量和响应列的
    表。
6  %
7  %
8  %  输出:
9  %      trainedClassifier: 一个包含训练的分类器的结构体。该结构
    体中具有各种关于所训练分
10 %       类器的信息的字段。
11 %
12 %      trainedClassifier.predictFcn: 一个对新数据进行预测的函
    数。
13 %
14 %      validationAccuracy: 以百分比形式表示验证准确度的双精度
```

```
    值。在 App 中，"模型"
15  %       窗格显示每个模型的验证准确度。
16  %
17  % 使用该代码基于新数据来训练模型。要重新训练分类器，请使用原始
        数据或新数据作为输入参数
18  % trainingData 从命令行调用该函数。
19  %
20  % 例如，要重新训练基于原始数据集 T 训练的分类器，请输入：
21  %   [trainedClassifier, validationAccuracy] = trainClassifier(
        T)
22  %
23  % 要使用返回的 "trainedClassifier" 对新数据 T2 进行预测，请使
        用
24  %   [yfit,scores] = trainedClassifier.predictFcn(T2)
25  %
26  % T2 必须是一个表，其中至少包含与训练期间使用的预测变量列相同
        的预测变量列。有关详细信息，请
27  % 输入：
28  %   trainedClassifier.HowToPredict
29
30
31  % 提取预测变量和响应
32  % 以下代码将数据处理为合适的形状以训练模型。
33  %
34  inputTable = trainingData;
35  predictorNames = {'x0', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', '
        x7', 'x8', 'x9', 'x10', 'x11', 'x12', 'x13', 'x14', 'x15', '
        x16', 'x17', 'x18', 'x19', 'x20', 'x21', 'x22', 'x23', 'x24'
        , 'x25', 'x26', 'x27', 'x28', 'x29', 'x30', 'x31', 'x32', '
        x33', 'x34', 'x35', 'x36', 'x37', 'x38', 'x39', 'x40', 'x41'
        , 'x42', 'x43', 'x44', 'x45', 'x46', 'x47', 'x48', 'x49', '
        x50', 'x51', 'x52', 'x53'};
36  predictors = inputTable(:, predictorNames);
37  response = inputTable.Lable;
38  isCategoricalPredictor = [false, false, false, false, false,
```

```
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false];
classNames = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12];

% 训练分类器
% 以下代码指定所有分类器选项并训练分类器。
template = templateSVM(...
    'KernelFunction', 'polynomial', ...
    'PolynomialOrder', 3, ...
    'KernelScale', 'auto', ...
    'BoxConstraint', 1, ...
    'Standardize', true);
classificationSVM = fitcecoc(...
    predictors, ...
    response, ...
    'Learners', template, ...
    'Coding', 'onevsone', ...
    'ClassNames', classNames);

% 使用 predict 函数创建结果结构体
predictorExtractionFcn = @(t) t(:, predictorNames);
svmPredictFcn = @(x) predict(classificationSVM, x);
trainedClassifier.predictFcn = @(x) svmPredictFcn(
   predictorExtractionFcn(x));

% 向结果结构体中添加字段
trainedClassifier.RequiredVariables = {'x0', 'x1', 'x2', 'x3',
    'x4', 'x5', 'x6', 'x7', 'x8', 'x9', 'x10', 'x11', 'x12', '
   x13', 'x14', 'x15', 'x16', 'x17', 'x18', 'x19', 'x20', 'x21'
   , 'x22', 'x23', 'x24', 'x25', 'x26', 'x27', 'x28', 'x29', '
```

```
    x30', 'x31', 'x32', 'x33', 'x34', 'x35', 'x36', 'x37', 'x38'
    , 'x39', 'x40', 'x41', 'x42', 'x43', 'x44', 'x45', 'x46', '
    x47', 'x48', 'x49', 'x50', 'x51', 'x52', 'x53'};
63  trainedClassifier.ClassificationSVM = classificationSVM;
64  trainedClassifier.About = '此结构体是从分类学习器 R2024b 导出
    的训练模型。';
65  trainedClassifier.HowToPredict = sprintf('要对新表 T 进行预
    测，请使用: \n [yfit,scores] = c.predictFcn(T) \n将 ''c'' 替
    换为作为此结构体的变量的名称，例如 ''trainedModel''。\n \n表
     T 必须包含由以下内容返回的变量: \n c.RequiredVariables \n变
    量格式(例如矩阵/向量、数据类型)必须与原始训练数据匹配。\n忽
    略其他变量。\n \n有关详细信息，请参阅 <a href="matlab:
    helpview(fullfile(docroot, ''stats'', ''stats.map''), ''
    appclassification_exportmodeltoworkspace'')">How to predict
    using an exported model</a>。');
66
67  % 提取预测变量和响应
68  % 以下代码将数据处理为合适的形状以训练模型。
69  %
70  inputTable = trainingData;
71  predictorNames = {'x0', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', '
    x7', 'x8', 'x9', 'x10', 'x11', 'x12', 'x13', 'x14', 'x15', '
    x16', 'x17', 'x18', 'x19', 'x20', 'x21', 'x22', 'x23', 'x24'
    , 'x25', 'x26', 'x27', 'x28', 'x29', 'x30', 'x31', 'x32', '
    x33', 'x34', 'x35', 'x36', 'x37', 'x38', 'x39', 'x40', 'x41'
    , 'x42', 'x43', 'x44', 'x45', 'x46', 'x47', 'x48', 'x49', '
    x50', 'x51', 'x52', 'x53'};
72  predictors = inputTable(:, predictorNames);
73  response = inputTable.Lable;
74  isCategoricalPredictor = [false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
    false, false, false, false, false, false, false, false,
```

```
    false, false, false, false, false, false, false, false,
    false];
classNames = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12];

% 执行交叉验证
KFolds = 5;
cvp = cvpartition(response, 'KFold', KFolds);
% 将预测初始化为适当的大小
validationPredictions = response;
numObservations = size(predictors, 1);
numClasses = 12;
validationScores = NaN(numObservations, numClasses);
for fold = 1:KFolds
    trainingPredictors = predictors(cvp.training(fold), :);
    trainingResponse = response(cvp.training(fold), :);
    foldIsCategoricalPredictor = isCategoricalPredictor;

    % 训练分类器
    % 以下代码指定所有分类器选项并训练分类器。
    template = templateSVM(...
        'KernelFunction', 'polynomial', ...
        'PolynomialOrder', 3, ...
        'KernelScale', 'auto', ...
        'BoxConstraint', 1, ...
        'Standardize', true);
    classificationSVM = fitcecoc(...
        trainingPredictors, ...
        trainingResponse, ...
        'Learners', template, ...
        'Coding', 'onevsone', ...
        'ClassNames', classNames);

    % 使用 predict 函数创建结果结构体
    svmPredictFcn = @(x) predict(classificationSVM, x);
    validationPredictFcn = @(x) svmPredictFcn(x);
```

```
108
109     % 向结果结构体中添加字段
110
111     % 计算验证预测
112     validationPredictors = predictors(cvp.test(fold), :);
113     [foldPredictions, foldScores] = validationPredictFcn(
    validationPredictors);
114
115     % 按原始顺序存储预测
116     validationPredictions(cvp.test(fold), :) = foldPredictions
    ;
117     validationScores(cvp.test(fold), :) = foldScores;
118 end
119
120 % 计算验证准确度
121 correctPredictions = (validationPredictions == response);
122 isMissing = isnan(response);
123 correctPredictions = correctPredictions(~isMissing);
124 validationAccuracy = sum(correctPredictions)/length(
    correctPredictions);
```

HNQ3.m

```
1  clc, clear, close all
2  path1 = '¸½¼□2/Person';
3  path2 = '/a';
4  path3 = 't';
5  path4 = '.xlsx';
6  acc = zeros(13, 12, 5);
7  omega = zeros(13, 12, 5);
8  for i = 4:13
9      for j = 1:12
10         for k = 1:5
11             my_path = strcat(path1,num2str(i),path2,num2str(j)
    ,path3,num2str(k),path4);
12             data = xlsread(my_path);
```

```
            A = data(:,1:3);
            B = data(:,4:6);
            C = mean(A);
            D = mean(B);
            E = sum(C.^2);
            F = sum(D.^2);
            acc(i, j, k) = E;
            omega(i, j, k) = F;
        end
    end
    i
end
filename1 = 'Acc1.xlsx';
filename2 = 'Omega1.xlsx';
Sheet_name = 'Action';
for i = 1:12
    Sheet = strcat(Sheet_name, num2str(i));
    my_acc = zeros(13, 5);
    my_omega = zeros(13, 5);
    for j = 4:13
        my_acc(j, :) = squeeze(acc(j, i, :));
        my_omega(j, :) = squeeze(omega(j, i, :));
    end
    xlswrite(filename1, my_acc, Sheet);
    xlswrite(filename2, my_omega, Sheet);
    i
end
```