

中国研究生创新实践系列大赛

“华为杯”第十七届中国研究生

数学建模竞赛

山东大学

学 校

20104220033

参赛队号

1.齐霖

队员姓名

2.李炜

3.颜冠鹏

中国研究生创新实践系列大赛

“华为杯”第十七届中国研究生

数学建模竞赛

题 目

摘要：

近年来，基于 P300 电位的脑机接口系统逐渐得到广泛的关注和研究。该系统通过分析识别在不同视觉刺激下人体脑部产生的脑电数据，检测 P300 电位是否产生，进而转换成相应的指令来实现对外部设备的操控。其中，P300 电位的分类识别在脑机接口系统中占据着至关重要的地位。

本文针对脑电信号的分析和判别问题，在对原始数据进行样本选取和降噪处理后，分别构建基于 BP 神经网络的字符分类识别模型、基于卷积神经网络的字符分类识别模型以及基于 SVM 的睡眠分期预测模型，使用 R 软件和 python 编程，对被试者实验中待识别的目标进行识别。

针对问题一，要求建立基于被试实验数据分析的分类识别模型，在已知 12 个字符的实验刺激电信号值的前提下，准确识别剩余 10 个待确定的字符。首先，考虑到 P300 电位的延迟性，本文选取行或列闪烁后 10 到 150 个采点时序的样本作为一个样本点，将目标字符所在行和列对应的样本点标记为 1，即说明在收到目标闪烁刺激后，脑电波产生 P300 电位，其余样本点标记为 0；其次，对于每个样本点，为防止在识别过程中受到噪声干扰，本文使用 Loess 平滑对数据进行降噪，为了提高模型分类识别的精确度，对样本点进行抽帧并标准化，形成 3600×142 的样本矩阵以进行进一步的模型构建；再次，通过构建 BP 神经网络，我们将样本矩阵输入模型，识别训练数据集中的 P300 电位用于与实验中目标采样点进行对应，并使用 k 折检验计算模型的正确率，将处理好的测试样本放入训练好的数学模型中，识别测试采样点的 P300 电位，并判断闪烁的行与列是否为目标行列；最后通过五位被试的分类结果，投票得到最终的字符。训练出的模型在识别方面的正确率达到了【】，相比较卷积神经网络的分类结果得到了很好的分类效果，为下面的进一步研究提供了条件。

针对问题二，由于多余的通道数据会导致信息冗余，干扰模型分类效果，在第一题建立 BP 神经网络分类识别模型的基础上，本文通过逐一筛选通道的方法，对具有相同数量通道的不同通道组合的模型识别效果进行排序比较，最终得到每个被试的最优通道组合，对五个最优通道组合进行正确率比较，选出最适合五名被试的通道组合。得到的最优通道组合正确率达到【】，可见随通道的筛选提高了模型分类识别的效果。

针对问题三，根据题目要求，本文考虑使用半监督模型，将从训练集样本点中适当选取部分样本做为有标签样本，剩余样本做为无标签样本。首先，基于问题二中建立的优化后的 BP 神经网络分类模型，先将有标签样本放入模型中进行训练，得到训练好的模型；其次，将未标签的样本数据放入训练好的模型中生成样本标签，得到新的有标签样本数据；再次，利用新旧有标签数据训练 BP 神经网络分类识别模型；最后利用训练好的分类模型对问题二中已知测试集样本数据进行分类以检验模型有效性，进而识别剩余未识别的目标字符。通过建立半监督学习模型，待识别字符正确率达【】。

针对问题四，由于仅仅已知不同个体在不同睡眠期不同频率的能量占比，信息量比较少，直接使用用原始数据训练 BP 神经网络可能会导致识别准确率不高，所以本文通过观察分布统计箱式图，计算定义了脑电能量占比的方差、四种脑电信号能量占比的比例等 20 个额外的特征变量。为了消除量纲的影响并且方便计算，本文还对增加了新的特征的数据进行了归一化处理。将处理好的数据输入 BP 神经网络分类识别模型进行识别，并建立支持向量机（SVM）模型进行比较，得到 BP 神经网络分类识别模型正确率为【】，SVM 模型

得到的正确率为【】。

一、问题重述

1.1 背景介绍

脑电信号是一种非平稳、非线性随机信号，其产生机理非常复杂，主要负责传递大脑对人体的控制信息[1]。通常大脑对肢体的控制过程可以表述为：大脑产生特定的思维活动，通过中枢神经系统传递给脑外周神经系统，脑外周神经系统去控制肌肉系统映射到四肢的运动。但是对于患有运动障碍的患者，由于神经组织病变等原因不能将脑电信号传递给肌肉，从而无法完成大脑相应的指令。

针对该类患者，Vidal 于 1973 年第一次提出了脑机接口（Brain computer interface，BCI）技术的研究成果。BCI 作为一种新型的人机交互方式，直接利用计算机等外部设备对 EEG 进行采集，进而采用模式识别方法对 EEG 提取特征和分类识别，从而将 EEG 与特定的运动（例如举起右手）联系起来，达到不经过大脑外周神经和肌肉系统等传统大脑输出回

路也能向外界传输信息的目的[2]。脑机接口系统由最初用于帮助运动神经疾病患者，发展到了老年人护理、医疗康复、军事、危险环境作业等诸多领域，有着广泛的应用前景和社会经济效益，成为脑科学研究领域的一个前沿热点研究方向[3]

1.2 问题重述

基于上述研究背景，本文研究需要完成以下问题：

问题一：针对脑机系统的分类识别问题，本文需要在保证目标分类精确度的前提下，利用附件 1 中的训练数据训练分类识别模型，依据尽可能少轮次的测试数据对待识别目标字符进行识别，进而提高脑际接口系统的信息传输速率。可使用其它数种方法进行识别比较以证明该模型方法的合理性。

问题二：在信息量巨大的原始脑电数据中，包含有较多复杂且无益于识别 P300 电位及待识别目标的冗余信息。由题中图表，无关或冗余的通道信号数据不仅作为多余的特征，且可能会影响分类识别模型的准确率和脑际系统的信息传输速率。因此，本文研究需要对通道数据进行筛选，针对每一个试验者选择更有利于其识别分类的通道组合，并研究是否存在适用于所有五个试验者识别目标的通道组合。

问题三：由于在 P300 系统中，需要花费较长时间获取有标签样本数据，本题根据附件 1 所给数据选取部分有标签样本，将剩余作为无标签样本，建立半监督模型，对模型进行训练与识别，并用问题二的测试数据检验方法的有效性，同时找出测试集中的其余待识别目标。

问题四：睡眠过程是一个动态的过程，其可以被划分为五个时期，分别为睡眠 I 期，睡眠 II 期，睡眠 III 期和睡眠 IV 期。题目要求建立一个分期预测模型，用尽可能少的样本去得到相对较高的预测准确率。同时，本文需要给出训练数据和测试数据的选取方式和分配比例，对预测模型的分类效果进行分析。

二、模型假设与符号说明

2.1 模型假设

考虑到现实与数据集的情况，本文做出以下假设：

- 1) 假设每个被试具有一定的个体差异，P300 信号的响应时间和峰值都不一定相同。
- 2) 假设同一个被试的不同的 P300 信号不会出现太大波形差异。
- 3) 附件所给数据大部分真实可靠。
- 4) 睡眠分期数据中，同一个睡眠期中的每一条记录都来自不同的个体。

2.2 符号说明

表 1 符号说明

序号	符号	符合说明
1	T_w	刺激发生后的时间
2	f	采样频率
3	T_s	字符矩阵闪烁时间
4	T_j	字符闪烁间隔时间
5	U_{P300}	P300 电位信号幅值
6	R	单个被试的识别正确率

三、问题一模型建立与求解

3.1 问题一分析

问题一要求依据附件 1 数据建立分类识别模型，附件 1 中给出训练数集 train_data 和训练数据的事件标签 train_event，包括 12 个已知字符的数据（char13~char22）。本文首先对数据用 Loess 平滑进行预处理，通过训练数据的事件标签可以知道在实验当中各行各列在何时闪烁，考虑到 P300 电位产生的延时性，本文在对数据预处理后，依据被试者实验中行和列闪烁的采样点序号截取各行与列自闪烁后 150 个采样点作为一个样本点。由于在训练数据集中，各个采样点的频率为 250Hz，因此每个样本点都包含 600ms 时段的采样点。本文应用 BP 人工神经网络算法将从 S1 至 S5 采集的样本进行分类，识别出产生 P300 电位的样本点，并将训练出的分类器用于测试数据集的电位识别，通过判断数据的事件标签中每个行与列闪烁后是否出现 P300 电位来确定待识别字符所在的行与列，进而识别出待识别目标。

3.2 数据预处理

在建立模型模型之前，先将模型进行预处理，根据问题分析，本次研究分别将五名被试的样本数据进行截取，得到 3600×142 的样本矩阵，考虑到脑电波信号容易受到扰动项的影响，本文通过 Loess 平滑对截取的每个样本点进行降噪处理。局部加权回归是一种非参数学习算法，通过以下步骤拟合信号值：

(1) 计算 θ 使得 $\sum_i w^{(i)} (y^{(i)} - \theta^T x^{(i)})^2$ 最小，其中 $w^{(i)} = \exp(-\frac{(x^{(i)} - x)^2}{2\tau^2})$ ；

(2) $y^{(i)}$ 的预测值为 $\theta^T x$ 。

目标所在行闪烁后样本点处理前后的对比图如图所示。在降噪以后，本文通过抽帧减少特征变量个数，

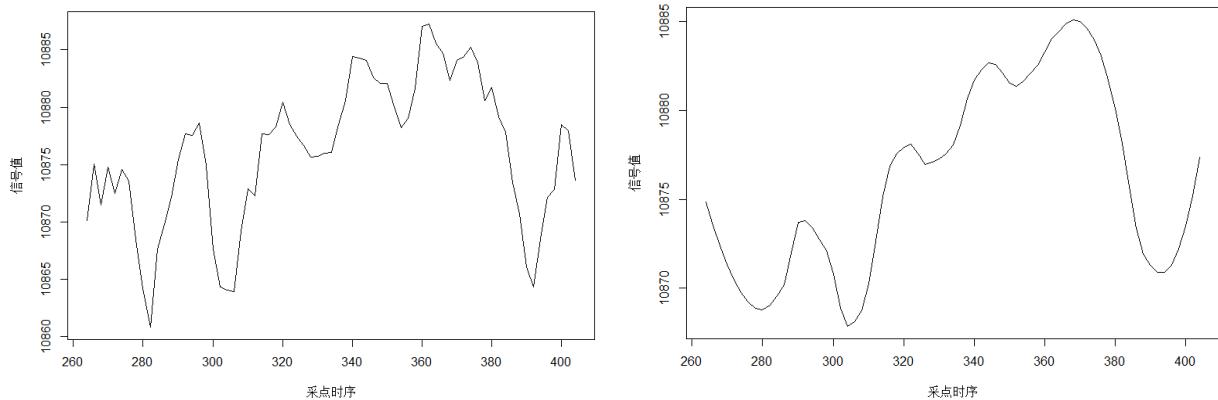


图 1 loess 平滑处理前（左）和处理后（右）信号值图

在经过降噪处理后，可以明显看出，目标行在闪烁后产生了一个明显的正向波峰，即认为出现了 P300 电位且具有可能被识别的特征，因此下面将构建基于 BP 神经网络的分类识别模型。

3.3 模型建立

3.3.1 基于 BP 人工神经网络的字符分类识别模型

由于样本点类型是 0 和 1，该问题是一个二分类问题，本文建立人工神经网络模型对样本点进行分类。BP 神经网络为一种多层次结构的前馈网络，由三部分组成，分别为输入层、隐含层、输出层，通过自身拓扑结构进行自动学习，神经网络拓扑结构如图：

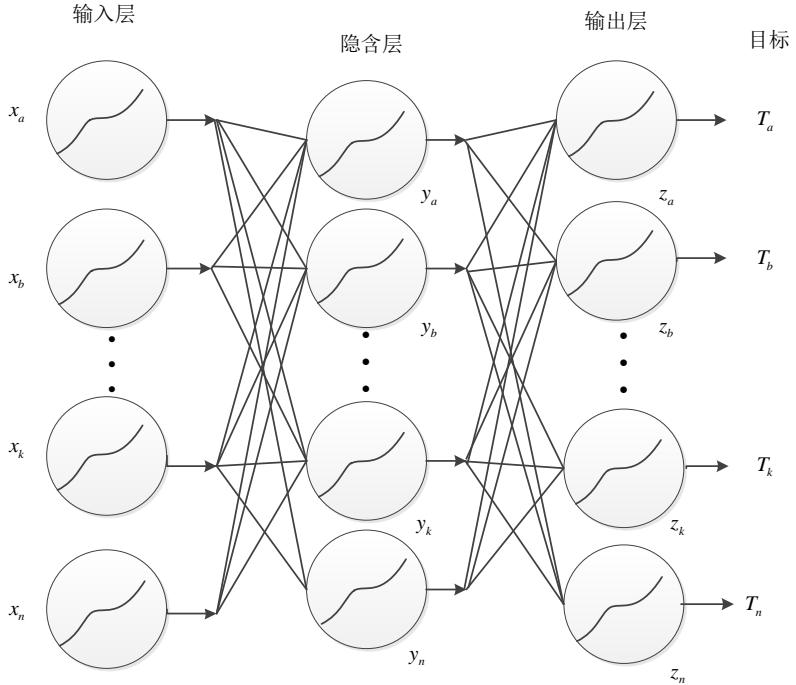


图 2 BP 神经网络拓扑结构

Step1:BP 神经网络构建

BP 神经网络算法如下：

(1) 网络初始化。该模型中设立一层含有 3 个隐蔽单元的隐含层，根据输入 $X = (x_1, x_2, \dots, x_n)$ 和期望输出 $T = (t_1, t_2, \dots, t_l)$ 来确定网络输入层、隐含层和输出层神经元（节点）个数，初始化各层神经元之间的连接权值 w_{ji} ， w_{kj} ，初始化隐含层阈值 a ，输出层阈值 b ，给定学习速率和神经元传递函数。

(2) 隐含层输出计算。根据输入向量 X ，输入层和隐含层间连接权值 w_{ji} 以及隐含层阈值 a ，计算隐含层输出。这里输入每个通道在每个时间点不同行列闪烁时的采样信号。

$$y_i = f\left(\sum_{j=1}^n w_{ji} x_i - a_j\right) = f\left(\sum_{j=0}^n w_{ji} x_i\right) \quad i=1, 2, \dots, m$$

式中， m 为隐含层节点数； $w_{ji} = -1, x_0 = a_j$ ； f 为隐含层传递函数。这里我们采用传递函数为

$$f(x) = \frac{1}{1+e^{-x}}$$

(3) 输出层输出计算。根据隐含层输出 Y ，连接权值 w_{kj} 和阈值 b ，计算 BP 神经网络的实际输出 T 。

$$T_k = f(\sum_{j=1}^m w_{kj}x_j - b_k) = f(\sum_{j=0}^m w_{kj}x_j) \quad k=1,2,\cdots,l$$

(4) 误差计算。根据网络实际输出 T 与期望输出 D ，计算网络总体误差 E 。

$$E = \frac{1}{2}(D-T)^2 = \frac{1}{2} \sum_{k=1}^l (d_k - o_k)^2 \quad k=1,2,\cdots,l$$

(5) 权值更新。根据网络总体误差 E ，按照以下公式更新网络连接权值 w_{ji} ， w_{kj} 。

$$w_{ji} = w_{ji} + \Delta w_{ji} \quad w_{kj} = w_{kj} + \Delta w_{kj}$$

$$\Delta w_{ji} = \eta \left(\sum_{k=1}^l \delta_k^o w_{kj} \right) y_j (1-y_j) x_i$$

$$\Delta w_{kj} = \eta \delta_k^o y_j$$

其中 $\delta_k^o = (d_k - t_k)t_k(1-t_k)$ 。式中 η 为学习速率。

(6) 判断算法迭代是否结束（可用网络总误差是否达到精度要求等方式来判断），若没有结束，返回步骤（2）。

通过构建与训练 BP 神经网络，可以识别出测试集中每个通道在采样时是否出现 P300 电位，进而为下一步目标行与列的识别提供条件。

Step2: 目标字符识别

将测试集数据放入训练好的神经网络模型中，识别出每个通道中的样本的分类，将每行和列闪烁时采样点对应样本的分类结果数值求出，若对应的样本识别为 P300 电位，则认为目标字符在该行或该列。通过五轮重复实验数据，通过投票确定最终的目标行和目标列，并识别出待识别字符。投票规则如下：

- (1) 依据每个被试者训练模型的正确度，计算每个分类正确度在所有正确度的比重；
- (2) 对于每个待识别字符，都会有五个识别结果，依据每一类预测结果对应的正确度的比重（若不同人识别结果相同，则将比重相加）确定最终投票结果。

3.3.2 基于卷积神经网络的 P300 电位检测

卷积神经网络是一种多层的神经网络，因其独特的卷积核机理，具备了强大的特征提取能力，目前被广泛地应用在计算机视觉和图像处理上面。卷积神经网络一般具有输入层，卷积层，池化层和输出层这几个结构。卷积层是对于输入的数据进行特征提取，内部一般具有多个卷积核，与前一层中位置接近的多个神经元相连，区域的大小是由卷积核的大小所

决定的，常被称为“感受野”，当卷积核工作时，会按照一定的规则扫过输入特征，对输入特征做乘法求和并叠加偏差。

池化层是把经过卷积层提取后的信息，再进行特征选择和信息过滤。池化层通过预先设定好的池化函数把特征图中单个点的结果替换为其相邻区域的特征图统计量。一般采用最大池化或者是平均池化，其步骤与卷积核工作大抵相同，由池化大小和步长来控制。

卷积层与池化层配合，逐层提取数据特征，卷积神经网络具有三个最为重要的特性：局部连接，权值共享以及下采样。这样既减少了权值数量使得网络易于优化，而且降低模型复杂度减小过拟合风险。如图 1 所示，这是经典卷积神经网络 LeNet5 的模型结构，共包含 8 层，C1 是一个卷积层，由 6 个特征图（Feature Map）构成。S2 是一个下采样层，有 6 个 14×14 的特征图。C3 层也是一个卷积层，通过 5×5 的卷积核去卷积 S2 层。S4 是一个下采样层，C5 层是一个卷积层。F6 层有 84 个单元，与 C5 层全连接，最后有一个输出层。

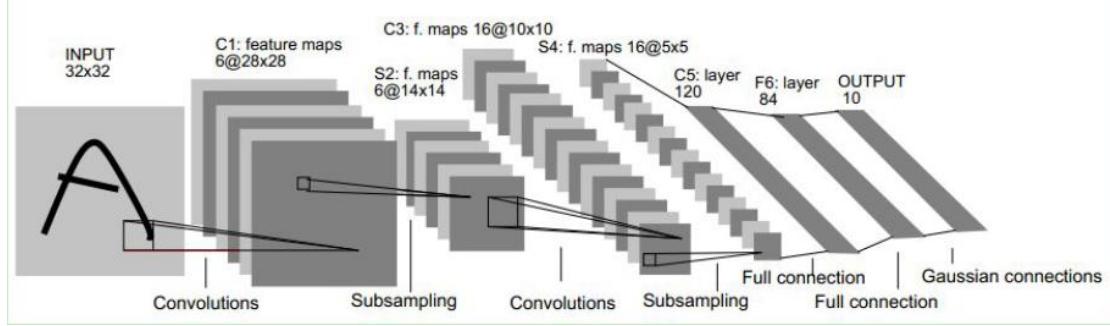


图 3 卷积神经网络结构

在这里我们把所有通道的电位信号都一起考虑，为了更好地区分 P300 信号，我们对数据分别进行了时域和频域上的处理。在频域处理中，P300 的频域特性主要集中在 1Hz 到 20Hz 之间，而题目所给的数据往往受到噪声和伪迹干扰而导致包含了其他频率段的无用信息，因此我们先将数据通过一个 4 阶的带通滤波器，只留下 1-20Hz 之间的数据。在时域处理中，由于 P300 信号往往在刺激发生后的 300ms 左右产生，考虑到个体差异和生理因素，我们决定取刺激后 0ms-600ms 的时间段作为特征数据。又因为脑电数据的采样频率是 250Hz，这样每次刺激后就相当于产生一个 150×20 的矩阵。为了方便计算，我们对数据做了归一化处理，采用 min-max 标准化转换由向量列组成的数据集，将每个特征调整到 $[0,1]$ 的范围，它通过每个特征内的最大绝对值来划分。它不会移动和聚集数据，因此不会破坏任何的稀疏性。min-max 标准化计算数据集上的统计数据，然后使用生成的模型分别的转换特征到范围 $[0,1]$ 。其原理如公式 (1) 所示

$$X_{scaled} = \frac{(X - X_{min})}{(X_{max} - X_{min})}$$

由于本模型采用有监督的训练方法，所以会对已有脑电数据做相对应的标签 y ，针对一个二分类问题，网络输出即代表是否为 P300 信号，按照以下规则对数据加上相应标签。

$$y = \begin{cases} 1 & (x \in P300) \\ 0 & (x \notin P300) \end{cases}$$

模型搭建

因为我们把数据划分成了 150×20 的矩阵，所以就相当于在处理一个 150 像素 * 20 像素且为单通道的图片，这时候就考虑到采用卷积神经网络来进行 P300 信号的识别。我们搭建了一个具有 n 层的网络，分别命名为 $L_1 \square L_n$ ，具体网络架构描述如下：

- 1) L_1 : 输入层, 输入尺寸为 $150*20$ 的待识别脑电信号。
 - 2) L_2 : 卷积层, 由 32 个尺寸为 $1*20$ 的卷积核组成, 这种操作相当于一次对数据的空域滤波, 可以去除脑电信号中冗余的信息。
 - 3) L_3 : 卷积层, 由 64 个尺寸为 $3*3$ 的卷积核组成对数据进行特征提取。
 - 4) L_4 : 池化层, 是用 $2*2$ 的池化滤波器对 L_3 层的输出结果进行最大池化, 池化的目的是为了减少网络的权值更好的训练网络。
 - 5) L_5 : 卷积层, 用 10 个 $3*3$ 的卷积核再次提取特征。
 - 6) L_6 : 全连接层, 将 L_5 层的输出结果进一步加权计算, 输出节点为 1, 输出对 P300 的识别结果。
- 在模型中, 我们采用二值交叉熵代价函数作为损失函数, 并且为了防止过拟合我们还采用了 dropout 的方法, 设置 $P=0.25$ 来进行随即丢弃。
- 在测试待识别目标时我们采用累积和最大的方法来确定目标所在位置, 即输入每次实验的脑电数据, 判断该段脑电信号是否为 P300 信号, 如果是则在相对应的位置计数(从 0 开始)上加 1, 直到多轮实验数据都判断完, 通过选择两个计数最大的位置来确定待识别目标的位置, 这样就可确定待识别目标为什么字符。

3.4 模型求解

将样本数据输入 BP 神经网络, 本文使用 k-fold 交叉验证对模型结果的测试误差进行度量。根据训练集的实验数据, 本次研究选取 k 值为 6, 将训练样本数据根据识别字符随机分成 6 等分, 即有 2 个字符的实验数据作为验证集, 最后将所有验证集的正确字母数/总字母数, 得到交叉验证正确率, 如表所示:

表 2 交叉验证结果

	S1	S2	S3	S4	S5	Total
N(True)	7	7	12	6	11	43
N	12	12	12	12	12	60
R	0.583	0.583	1.000	0.500	0.917	0.717

将训练好的分类模型用于识别测试集目标字符, 得到的识别结果如表所示:

表 3 模型识别结果

	S1	S2	S3	S4	S5	Final
Char13	M	A	M	5	M	M
Char14	L	V	F	A	F	F
Char15	5	5	M	B	5	5
Char16	2	V	2	Y	2	2
Char17	O	I	I	I	I	I
Char18	N	T	T	6	T	T
Char19	K	K	K	W	K	K
Char20	4	X	X	4	L	X
Char21	A	M	A	A	A	A
Char22	0			0	0	0
Weight	0.163	0.163	0.279	0.140	0.256	

可以看出, 在没有筛选通道的前提下, BP 人工神经网络分类识别模型仍然具有较好的识别

性效果。而 CNN 却未能很好的学习到数据存在的信息，不能完成分类任务。

四、问题二模型建立与求解

4.1 问题二分析

问题二在问题一的基础上对模型进一步优化，由问题二所给条件，部分通道的信息比较冗余，会影响模型分类识别效果，因此考虑对输入的特征变量进行缩减，筛选输入分类模型的特征变量，针对每一个被试者进一步确定最优的通道组合，并在每个最优组合中选取可以适用于所有人的通道组合。针对每一个被试者，通过逐步减少通道，并对每个通道组合下的预测精确度进行排序，选择出最适合被试者的通道数量和通道组合。

4.2 模型建立

依据问题分析，本文选用逐步筛选的方法，逐一剔除通道，并计算正确率。筛选步骤如下：

Step1:从已选取的通道组合中分别剔除单个通道，计算剔除通道后剩余通道组合下字符分类识别模型的识别正确率；

Step2:对计算的各个正确率进行排序，选取正确率最高的通道组合；

Step3:重复 step1 和 step2 直至剩余通道数量为 10。

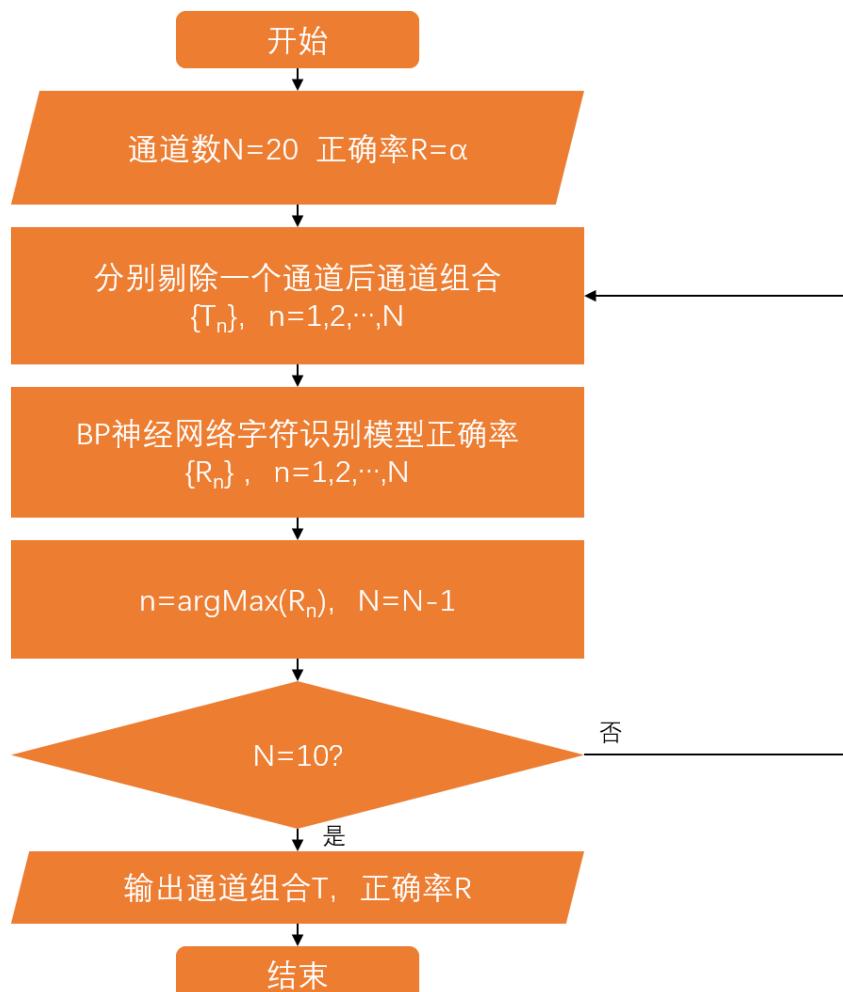


图 4 问题二解决步骤

通过上述步骤，可以得到每一个被试者的最优通道组合，最终通过比较这五个组合在其它被试者的正确率可以得到适用于各个被试者的最优通道组合。

4.3 模型求解

经过逐一筛选比较，通过上述步骤得到不同被试者最优通道。在此结果的基础上，取五个实验者超过半数(三人)共同选择的通道为适用所有被试者的最优通道，结果如表：

表 4 最优通道选择结果

	通道组合
S1	(Fz, F3, F4, Cz, C4, T7, T8, CP3, CP4, CP5, CP6, Pz, P3, P4, P7, P8, Oz, O1, O2)
S2	(F4, Cz, T7, T8, CP4, CP5, Pz, P3, P4, P8, O1, O2,
S3	(F3, CP6, Pz, P3, P4, P7, P8, Oz, O1, O2)
S4	(Fz, F3, F4, Cz, C3, C4, T7, T8, CP3, CP4, CP5, Pz, P3, P4, P8, Oz, O1, O2)
S5	(Fz, F3, F4, Cz, C4, T7, CP5, CP6, Pz, P3, P4, P7, P8, Oz, O1, O2)
ALL	Fz, F3, F4, Cz, C4, T7, T8, CP4, CP5, CP6, Pz, P3, P4, P7, P8, Oz, O1, O2

上述最优通道对应的预测结果如表：

	S1	S2	S3	S4	S5	Final
Char13	M	5	M	5	M	M
Char14	F	P	F	A	F	F
Char15	5	M	M	B	5	5
Char16	8	P	2	Y	2	2
Char17	O	I	I	I	I	I
Char18	B	Z	T	6	T	T
Char19	K	K	K	W	K	K
Char20	4	X	X	4	L	X
Char21	G	M	A	A	A	A
Char22	O			O	O	O

五、问题三模型建立与求解

5.1 问题三分析

在题目所给的 P300 脑电数据中，有大量数据并且都是未做好标签的，这就给利用监督学习方法带来极大的不便，往往需要花大量的时间来给原始数据加上标签。因此考虑可以选择相对比较少的数据加上标签后作为已知样本，其余的训练样本是无标签的样本，以此进行半监督学习。

5.2 模型建立

一般，半监督学习算法可分为：self-training (自训练算法)、Graph-based Semi-supervised Learning (基于图的半监督算法)、Semi-supervised supported vector machine (半监督支持向量机，S3VM)。在半监督学习成为一个热门领域之后，出现了许多利用无类标签的样例提高学习算法预测精度和加快速度的学习方法，因此出现了大量改进的半监督学习方法。Nigam 等人将 EM 和朴素贝叶斯结合，通过引入加权系数动态调整无类标签的样例的

影响提高了分类准确度，建立每类中具有多个混合部分的模型，使贝叶斯偏差减小。Zhou 和 Goldman 提出了协同训练改进算法，不需要充分冗余的视图，而利用两个不同类型的分类器来完成学习。Shang 等人提出一种新的半监督学习方法，能同时解决有类标签样本稀疏和具有附加无类标签样例成对约束的问题[3]。

深度学习需要用到大量有标签数据，即使在大数据时代，干净能用的有标签数据也是不多的，由此引发深度学习与半监督学习的结合。在有标签数据与无标签数据混合成的训练数据中使用的深度学习算法可总结为三类：（1）无标签数据预训练网络后用有标签数据微调（fine-tune）；（2）有标签数据训练网络，利用从网络中得到的深度特征来做半监督算法；（3）让网络工作在真正的半监督方式上。

第一类是通过例如聚类算法等给无标签数据加上伪标签信息，先用这些伪标签来训练网络，再用已知正确标签的数据对网络进行微调。

第二类是先用已有标签的数据训练模型，此时由于已知标签数据较少通常网络呈过拟合状态，从该网络中提取所有数据的特征，以这些特征来用某种分类算法对无标签数据进行分类，挑选认为分类正确的无标签数据加入到训练集，再训练网络；如此循环。由于网络对无标签数据预测后就形成了新的有标签数据，可以再用新的数据进行训练提升网络但这种方式由于网络的过拟合和无标签数据存在的噪声，很容易导致模型精度较差。

因此我们决定采用第三类，其本质就是先用有标签数据训练得到一个网络，然后对无标签数据进行预测并将结果作为无标签数据的标签，再用加上新标签好数据的数据集来训练网络即网络的自训练可以提升网络的效果。这种模型一般采用的损失函数如下：

$$L = \sum_{m=1}^n \sum_{i=1}^C L(y_i^m, f_i^m) + \alpha(t) \sum_{m=1}^{n'} \sum_{i=1}^C L(y_i'^m, f_i'^m)$$

代价函数的第一项是有标签数据的代价，第二项是无标签数据的代价，其中 y' 表示的是无标签数据经过网络预测后取预测值为最大值的作为标签。在代价函数中最重要的就是 $\alpha(t)$ 的引入，它决定了无标签数据在网络参数更新的重要程度，在一开始训练的时候由于预测结果并不是很准， $\alpha(t)$ 应设置比较小：即从 0 开始。随着训练轮次的增加， $\alpha(t)$ 再随之增大，可采用高斯型的爬升函数。

为了保证数据的均衡，我们给小部分的数据人工标了标签，并且 P300 脑电信号和非 P300 脑电信号占比各位 50%。对于每一位被试，我们共标记了 50 组数据，其中 25 组为 P300 信号。

首先搭建结构与问题一求解模型一致的人工神经网络，先用标记好的数据进行训练得到一个初始的模型。然后加入无标签数据进行训练，随着训练轮次的增加，代价函数也会有所改变。

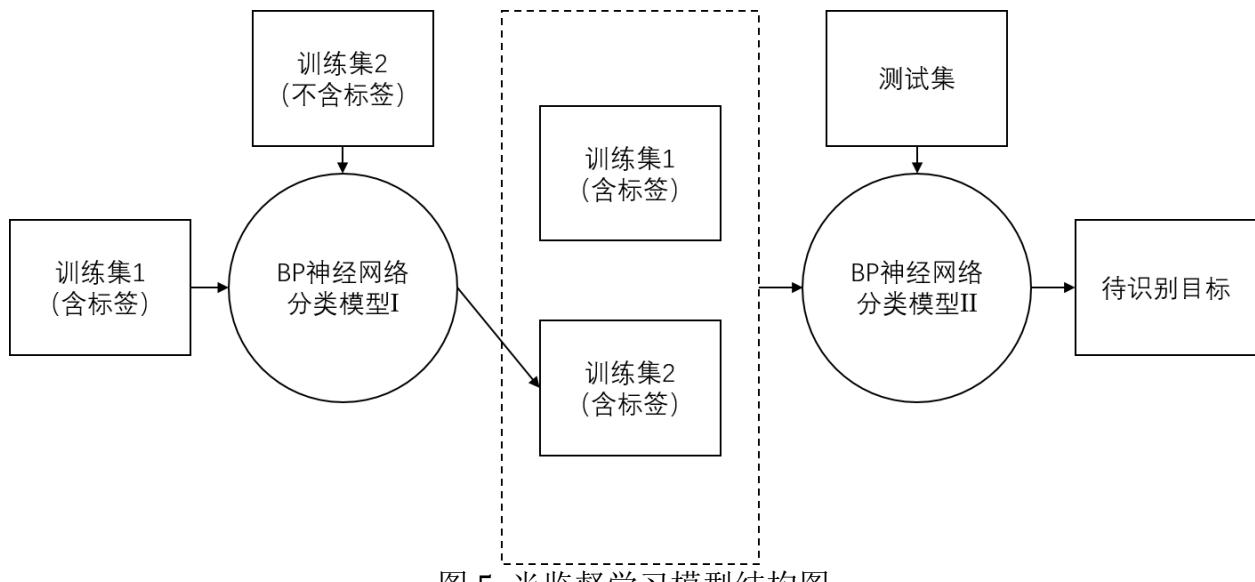


图 5 半监督学习模型结构图

5.3 模型求解

表 5 char13-char17 分类识别正确率

	Char13	Char14	Char15	Char16	Char17
R	0.6	0.4	0.8	0.8	0.4

表 6 char18-char22 识别结果

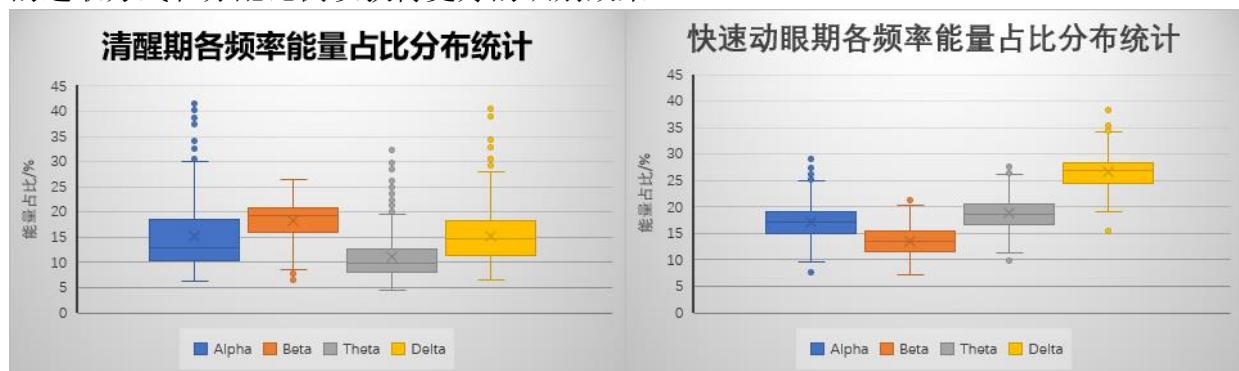
	Char18	Char19	Char20	Char21	Char22
字符识别	T	K	X	A	0

六、问题四模型建立与求解

6.1 问题四分析

题目中所给的睡眠分期的数据，是不同个体在不同睡眠期不同频率的能量占比。从题目例中可以观察到，脑电信号在不同睡眠分期所呈现的特点有所不同，再结合所给的数据考虑到根据各频率段能量占比不同来识别当前所属睡眠分期。

我们首先对不同睡眠分期的不同频率的能量占比分布做一个统计，如下箱式图所示。可以看出，随着睡眠程度的深入，Delta 频率范围内的能量占比总体呈上升趋势，而 Alpha 和 Beta 频率范围内的能量占比虽有个体差异但总体还是呈下降趋势，Theta 的总体变化在五个睡眠期中并不明显。需要通过观察数据图形和箱型图，构造 20 个特征变量，将特征变量输入神经网络分类模型，依据特征变量识别睡眠模式，并不断调整训练数据和测试数据的选取方式和分配比例以获得更好的识别效果。



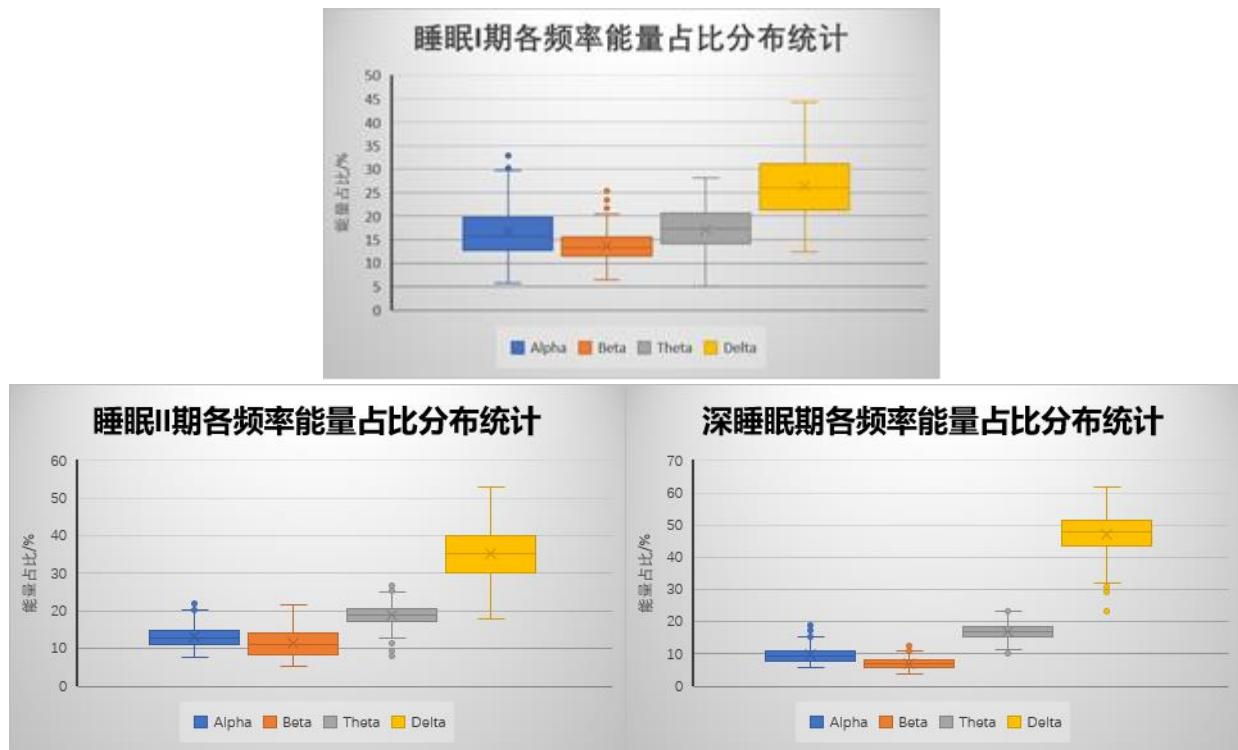


图 6 数据箱型图

6.2 模型建立

6.2.1 数据预处理

由于已知信息仅仅只有不同个体在不同睡眠期不同频率的能量占比，信息比较少，直接用原始数据训练神经网络的话可能会导致识别准确率不高，所以我们想到人为制造一些额外特征。

通过上图的分布统计箱式图，我们发现对于大多数人来说不同睡眠期的能量占比差异很大，像快速动眼期和清醒期，不同频率脑电的能量占比差距不是很大，所以我们考虑到可以把每个个体的不同频率脑电能量占比的方差作为一个新的特征。另外我们观察到，每个个体在不同睡眠期都不止仅包括这四种频率的脑电信号，因为这四个的能量占比之和小于100%，于是我们考虑可以增加这四种不同频率脑电信号占四者之和的比例作为新的特征。另外还增加了各频率能量占比的差值以及能量占比的平均值作为新的特征。

为了消除量纲的影响并且方便计算，我们还对增加了新的特征的数据进行了归一化处理。然后再固定随机种子，对数据进行乱序处理，并保证标签和数据的一致性。

6.2.2 神经网络模型

这一题我们仍然采用神经网络进行求解，并把 SVM 方法作为对比。对于神经网络方法，我们设计了一个具有四层全连接层的神经网络，第一层包含 128 个神经元，第二层和第三层分别包含 256 和 128 个神经元。最后一层是输出层，有 5 个神经元，代表着 5 个分类的概率输出，激活函数采用 softmax 将输出转换为概率形式。由于标签是 2-6 的数字表示，而网络的输出是各分类的概率，所以损失函数和 metrics 分别为 SparseCategoricalCrossentropy 和 sparse_categorical_accuracy。

6.2.3 支持向量机（SVM）

支持向量机（Support Vector Machine, SVM）是一类按监督学习（supervised learning）方式对数据进行二元分类的广义线性分类器（generalized linear classifier），其决策边界是对学习样本求解的最大边距超平面（maximum-margin hyperplane）。

SVM 使用铰链损失函数（hinge loss）计算经验风险（empirical risk）并在求解系统中加入了正则化项以优化结构风险（structural risk），是一个具有稀疏性和稳健性的分类器。SVM 可以通过核方法（kernel method）进行非线性分类，是常见的核学习（kernel learning）方法之一。

在分类问题中给定输入数据和学习目标： $X = \{X_1, \dots, X_N\}$, $y = \{y_1, \dots, y_N\}$ ，其中输入数据的每个样本都包含多个特征并由此构成特征空间（feature space）： $X_i = [x_1, \dots, x_n] \in X$ ，而学习目标为二元变量 $y \in \{-1, 1\}$ 表示负类（negative class）和正类（positive class）。

若输入数据所在的特征空间存在作为决策边界（decision boundary）的超平面将学习目标按正类和负类分开，并使任意样本的点到平面距离大于等于 1：

$$\text{decision boundary: } \omega^T X + b = 0$$

$$\text{point to plane distance: } y_i(\omega^T X_i + b) \geq 1$$

则称该分类问题具有线性可分性，参数 ω, b 分别为超平面的法向量和截距。

满足该条件的决策边界实际上构造了 2 个平行的超平面作为间隔边界以判别样本的分类：

$$\omega^T X_i + b \geq +1 \Rightarrow y_i = +1$$

$$\omega^T X_i + b \leq -1 \Rightarrow y_i = -1$$

所有在上间隔边界上方的样本属于正类，在下间隔边界下方的样本属于负类。两个间隔边界的距离 $d = \frac{2}{\|\omega\|}$ 被定义为边距（margin），位于间隔边界上的正类和负类样本为支持向量（support vector）。

一些线性不可分的问题可能是非线性可分的，即特征空间存在超曲面（hypersurface）将正类和负类分开。使用非线性函数可以将非线性可分问题从原始的特征空间映射至更高维的希尔伯特空间（Hilbert space），从而转化为线性可分问题，此时作为决策边界的超平面表示如下

$$\omega^T \phi(X) + b = 0$$

由于映射函数具有复杂的形式，难以计算其内积，因此可用核方法来回避内积的显示计算。常见的核函数一般有多项式核，径向基函数核，拉普拉斯核等。使用非线性函数将输入数据映射到高维空间后应用线性 SVM 即可求解非线性分类问题，完成非线性 SVM。有如下优化问题：

$$\min_{\omega, b} \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^N \xi_i$$

$$\text{s.t. } y_i [\omega^T \phi(X_i) + b] \geq 1 - \xi_i, \xi_i \geq 0$$

在这里我们采用核函数为径向基函数（rbf）的 SVM。

6.3 模型求解

采用 50% 的数据作为训练集，在经过 300 轮的迭代后，神经网络模型在训练集上可以达到 80% 左右的正确率，在测试集上可以达到 74% 左右的正确率。其训练过程的 loss 曲线和 acc 曲线如下

而 SVM 模型，在训练集上可以达到 100% 的正确率，在测试集上的正确率却只有 65% 左右，可以发现 SVM 模型在这种情况下出现了过拟合现象。

可以看出，在对于这种样本分类的问题中，神经网络模型的效果会优于 SVM。

七、模型评价与改进

7.1 模型的优点

本文针对附件所给的 P300 脑电数据和睡眠分期数据，利用 python, Matlab, Excel 等统计软件进行了数据预处理，带通滤波器的使用在保存大部分有用信息数据的前提下，得到了较为合理的清洗后的数据并进行分段加注标签形成可训练的数据。针对 P300 脑电信号的识别，我们采用了多种方法对比试验，最终得到了以 BP 神经网络为基础的模式识别模型，可以较为准确判别 P300 信号，并且据此判断了字符矩阵闪烁的位置从而识别出待确定字符。采用的半监督学习模型可以使得在尽可能少标注标签的情况下，仍然可以比较准确的识别出待确定字符。通过人为的增加特征，使得神经网络可以在原始数据特征较少的情况下依然可以较为准确地对睡眠期进行判别，提高了模型的可靠度。

7.2 模型的缺点

由于脑电 P300 的数据文件中的通道数目较多，每个文件的记录条数也比较多，给数据量化处理工作带来一些困难，因此在筛选过程中可能会因为方法选择的不恰当会遗漏部分可能存在重要作用的变量信息，并由此可能会对结果造成一定程度的影响。另外在睡眠分期数据的人为特征构造中，可能存在的特征的重复与不合理性，导致在网络深度不足时容易出现过拟合。

八、参考文献

附录

数据处理

```
library(reshape2)
library(e1071)
library(sampling)
library(randomForest)
library(dplyr)
library(neuralnet)
library(nnet)
library(mFilter)
library(pspline)
trainlevel <- c(2,4,7,12,15,17,19,22,26,30,33,35)
testlevel <- c(13,6,25,28,9,-1,-2,-3,-4)
C <- c("A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T",
      "U","V","W","X","Y","Z","1","2","3","4","5","6","7","8","9","0")
S <- paste("S",1:5, sep = "")
result <- NULL
result0 <- NULL
result2 <- NULL
```

```

colStandardization <- function(data){
  result <- apply(data, 2, function(x){
    max <- max(x)
    min <- min(x)
    (x-min)/(max-min)
  })
  return(result)
}

getRC <- function(tab,i){
  if(i<=0) return(c(0,0))
  else{
    row <- (tab[i] - 1)%/6+1
    col <- (tab[i] - 1)%6 +7
  }
  return(c(row,col))
}

start <- 10
end <- 149
by <- 1
result <- NULL
result2 <- NULL
#Get Train Data
for(k in 1:5){
  for(i in 1:length(trainlevel)){
    print(i)
    data <- read.xlsx(paste0(S[k],"/",S[k],"_train_data.xlsx"),sheet = i,colNames = F)
    data <- apply(data,2,function(x) x)
    data <- data.frame(apply(data,2,function(x) x))
    id <- as.numeric(rownames(data))
    data <- cbind(id,data)
    head(data)
    event <- data.frame(read.xlsx(paste0(S[k],"/",S[k],"_train_event.xlsx"),sheet = i,colNames = F))
    head(event)
    colnames(event) <- c("Flash", "x")
    head(event)
    event <- dplyr::filter(.data = event, Flash<100)
    temp <- NULL
    for(j in 1:dim(event)[1]){
      temp2 <- merge(event[j,2],event[j,2]+seq(start,end,by))
      temp2$Time <- 1:length(seq(start,end,by))
      colnames(temp2)[2] <- "id"
      temp3 <- merge(temp2,data)
    }
  }
}

```

```

head(temp3)
for(l in paste("X",1:20,sep = "")){
  loess.fit <- loess(as.formula(paste(l,"~id",sep= "")),data=temp3,span=0.3)
  temp3[,l] <- predict(loess.fit)
}
temp <- rbind(temp, temp3)
}
colnames(temp)
temp[,paste("X",1:20,sep = "")] <-
  colStandardization(temp[,paste("X",1:20,sep = "")] )
event$Group <- sort(rep(1:5,12))
temp <- merge(event,temp)
data2 <- temp
head(data2)
RC <- getRC(trainlevel,i)
data2$Y <- apply(data2["Flash"],1,function(x){
  if(x==RC[1]||x==RC[2]) 1 else 0
})
head(data2)
data2$Char <- trainlevel[i]
data2$S <- S[k]
temp3 <- melt(data2,measure = paste0("X",1:20,""))
temp3 <- temp3[,c(7,8,3,2,6,5,9,10)]
head(temp3)
formula <- paste0("Char+S+Group+Flash+Y","~","variable+Time",collapse = "+")
temp4 <- dcast(temp3,as.formula(formula))
head(temp4)
result <- rbind(result,temp4)
}
}
#Get Test Data
for(k in 1:5){
  if(k %in% c(1,4,5)) testlevel <- c(testlevel,0)
  for(i in 1:length(testlevel)){
    print(i)
    data <- read.xlsx(paste0(S[k],"/",S[k],"_test_data.xlsx"),sheet = i,colNames = F)
    data <- apply(data,2,function(x) x)
    data <- data.frame(apply(data,2,function(x) x))
    id <- as.numeric(rownames(data))
    data <- cbind(id,data)
    head(data)
    event <- data.frame(read.xlsx(paste0(S[k],"/",S[k],"_test_event.xlsx"),sheet =
i,colNames = F))
    head(event)
  }
}

```

```

colnames(event) <- c("Flash", "x")
head(event)
event <- dplyr::filter(.data = event, Flash<100)
temp <- NULL
for(j in 1:dim(event)[1]){
  temp2 <- merge(event[j,2],event[j,2]+seq(start,end,by))
  temp2$Time <- 1:length(seq(start,end,by))
  colnames(temp2)[2] <- "id"
  temp3 <- merge(temp2,data)
  head(temp3)
  for(l in paste("X",1:20,sep = "")){
    loess.fit <- loess(as.formula(paste(l,"~id",sep= "")),data=temp3,span=0.4)
    temp3[,l] <- predict(loess.fit)
  }
  temp <- rbind(temp, temp3)
}
colnames(temp)
temp[,paste("X",1:20,sep = "")] <-
  colStandardization(temp[,paste("X",1:20,sep = "")])
event$Group <- sort(rep(1:5,12))
temp <- merge(event,temp)
data2 <- temp
head(data2)
RC <- getRC(testlevel,i)
data2$Y <- apply(data2["Flash"],1,function(x){
  if(x==RC[1]||x==RC[2]) 1 else 0
})
head(data2)
data2$Char <- testlevel[i]
data2$S <- S[k]
temp3 <- melt(data2,measure = paste0("X",1:20,""))
temp3 <- temp3[,c(7,8,3,2,6,5,9,10)]
head(temp3)
formula <- paste0("Char+S+Group+Flash+Y","~","variable+Time",collapse = "+")
temp4 <- dcast(temp3,as.formula(formula))
head(temp4)
result2 <- rbind(result2,temp4)
}
if(k %in% c(1,4,5)) testlevel <- testlevel[1:9]
}
#Combine and Output
data4 <- result
train <- data4[,c(1:5,seq(6,dim(data4)[2],5))]
data5 <- result2

```

```

test <- data5[,c(1:5,seq(6,dim(data4)[2],5))]
train$type <- "Train"
test$type <- "Test"
write.xlsx(rbind(train,test)[,c("Type",colnames(train)[1:length(colnames(train))-1])],"Data
Use.xlsx", row=F, col =T)
BP 神经网络分类识别模型
Seed = 1
Kfold <- 6
getCharR <- function(ans){
  for(q in seq(1,dim(ans)[1],60)){
    print(q)
    temp <- ans[q:(q+59),]
    t <- c(NA,NA)
    rowM <- temp[temp$Flash<=6,c("S","Flash","prey")]
    colM <- temp[temp$Flash>=7,c("S","Flash","prey")]
    rowS <- dcast(rowM, S ~ Flash, function(x) sum(x))[, -1]
    colS <- dcast(colM, S ~ Flash, function(x) sum(x))[, -1]
    t[1] <- names(which.max(rowS))[1]
    t[2] <- names(which.max(colS))[1]
    t <- as.numeric(t)
    ans[q:(q+59),"PreChar"] <- putRC(t[1],t[2])
  }
  return(ans)
}
putRC <- function(R,C){
  return((R-1)*6+C-6)
}
trainlevel <- c(2,4,7,12,15,17,19,22,26,30,33,35)
S <- paste("S",1:5, sep = "")
C <- c("A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T",
      "U","V","W","X","Y","Z","1","2","3","4","5","6","7","8","9","0")
      
data <- openxlsx::read.xlsx("Data Use.xlsx",sheet = 1,colNames = T)
train <- list()
test <- list()
set.seed(Seed)
L <- order(rnorm(length(trainlevel)))
for(i in seq(1,length(L),round(length(L)/Kfold))) {
  iT <- L[i:min((i+1),length(L))]
  temp <- filter(.data = data, Type == "Train", Char %in% trainlevel[-iT])
  train <- c(train,list(temp))
  temp <- filter(.data = data, Type == "Train", Char %in% trainlevel[iT])
  test <- c(test,list(temp))
}

```

```

formula <-
  as.formula(paste0("Y~",paste0(colnames(data)[-1:-6],collapse = "+"),collapse = ""))
Para <- matrix(0,nrow = 5,ncol = 5)
rownames(Para) <- S
colnames(Para) <- c("rang","decay","R","N","accuracy")
# Search Parameters
for(i in S){
  trainT <- train
  testT <- test
  for(j in 1:6){
    trainT[[j]] <- dplyr::filter(trainT[[j]],S==i)
    testT[[j]] <- dplyr::filter(testT[[j]],S==i)
  }
  for(rang in seq(0,1,0.2)){
    for(decay in seq(0,1,0.1)){
      R <- 0
      N <- 0
      for(j in 1:6){
        set.seed(Seed)
        fit <- nnet(as.formula(formula) ,data=trainT[[j]],size=3,rang=rang,decay =
decay,linout=F,maxit=10000,MaxNWts=10000)
        prey <- predict(fit,testT[[j]])
        temp4 <- cbind(testT[[j]],prey)
        temp4 <- getCharR(temp4)
        R <- R + sum(temp4$Char==temp4$PreChar)
        N <- N + dim(temp4)[1]
      }
      if((R/N)>Para[i,"accuracy"]){
        Para[i,"rang"] <- rang
        Para[i,"decay"] <- decay
        Para[i,"R"] <- R
        Para[i,"N"] <- N
        Para[i,"accuracy"] <- R/N
      }
      if((R/N)==1) break
    }
    if((R/N)==1) break
  }
}
write.xlsx(Para,"Ans1(table1).xlsx",row.names = T, col.names = T)
# Predict
data <- openxlsx::read.xlsx("Data Use.xlsx",sheet = 1,colNames = T)
Para <- openxlsx::read.xlsx("Ans1(table1).xlsx",rowNames = T )
Ans <- matrix(nrow = 6, ncol = 12)

```

```

rownames(Ans) <- c(S,"鎬昏")
colnames(Ans) <- c(1:10,"姝 g '鍗?',"鏉江噸")
for(i in S){
  train <- filter(.data = data, Type == "Train", S == i)
  test <- filter(.data = data, Type == "Test", S == i)
  para <- Para[i,]
  set.seed(Seed)
  fit <- nnet(
    as.formula(formula),
    data=train,
    size=3,
    rang=as.numeric(para["rang"]),
    decay=as.numeric(para["decay"]),
    linout=F,
    maxit=10000,
    MaxNWts=10000)
  prey <- predict(fit,test)
  ans <- cbind(test,prey)
  for(q in seq(1,dim(ans)[1],60)){
    print(q)
    temp <- ans[q:(q+59),]
    ##### Case 2
    t <- c(NA,NA)
    rowM <- temp[temp$Flash<=6,c("S","Flash","prey")]
    colM <- temp[temp$Flash>=7,c("S","Flash","prey")]
    rowS <- dcast(rowM, S ~ Flash, function(x) sum(x))[, -1]
    colS <- dcast(colM, S ~ Flash, function(x) sum(x))[, -1]
    t[1] <- names(which.max(rowS))[1]
    t[2] <- names(which.max(colS))[1]
    t <- as.numeric(t)
    ans[q:(q+59),"PreChar"] <- C[putRC(t[1],t[2])]
  }
  for(q in seq(1,dim(ans)[1],60)){
    Ans[i,(q%/%60)+1] <- ans[q,"PreChar"]
  }
}
Ans[1:5,11] <- Para[,"accuracy"]
Ans[1:5,12] <- Para[,"accuracy"]/sum(Para[,"accuracy"])
Ans[6,12] <- sum(as.numeric(Ans[1:5,12]))
write.xlsx(data.frame(Ans),"Ans1(table2).xlsx",row.names = T, col.names = T)

```