


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

学　　校	南京大学
参赛队号	20102840089
队员姓名	1.项祺
	2.袁康
	3.梅海强

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　面向康复工程的脑电信号分析和判别模型

摘　　要：

本文主要研究了诱发脑电信号（P300 脑-机接口）和自发脑电信号建模分类的问题。主要创新点在于，针对诱发脑电信号（P300 信号）识别分类模型构建问题，结合卷积神经网络和 PCA 主成分分析对模型进行改进，优化 P300 信号通道选择，最后构建出基于 CNN 的字符识别模型；针对自发脑电信号，结合数据特征分析和 K-Means 算法，提出基于 K-Means 的睡眠阶段分类模型。

问题一：我们首先对原始采集的数据进行预处理，由于该数据来源于实验测得，数据存在较多的噪声，直接进行分析干扰多且难度大，我们首先对数据用带通滤波器作预处理，并采用时域降采样法提取特征。对问题进行剖析，虽然是 36 字符分类问题，但是 P300 信号对于不同字符的刺激都是相同的，所以该问题本质上就是一个二分类问题，即在识别过程中得出目标字符所在的行和列。结合 P300 信号特性，本文截取时间窗 150-650ms 的实验数据作为特征数据。由于数据量比较庞大，为了提高训练的速度，在时域上采用降采样的方法，即每 4 个采样点选取 1 个，将采样点的数量由 225 个减少到 32 个，故数据样本简化为 32×20 的矩阵。同时考虑到一轮闪烁当中目标行或者列只出现 2 次，非目标行或者列出现 10 次，也就是正样本和负样本的数量是 1：5。如果用这个比例的样本进行训练，会导致模型直接把所有样本判断为负样本，这样的训练结果是不准确的，故考虑把正样本直接扩大 5 倍，使正负样本的比列达到 1：1。模型选取基于 SVM、基于朴素贝叶斯算法、以及基于 CNN 网络结构，经过对比，最终 CNN 网络取得了更好的效果。模型训练采用批归一化（BN）的方法，这样可以有效的解决梯度易消失的问题。最后对含有 10 个待预测字符的测试集进行预测，我们规定对于任一待预测字符，当某一轮次的预测值与之前某轮次的预测结果一致，即为预测成功，最终模型可以在 3 轮测试下成功找出 10 个待识别字符。对 S1 预测结果为：[“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”、“0”]；S2 预测结果为：[“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”]；S3 预测结果为：[“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”]；S4 预测结果为：[“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”、“0”]；S5 预测结果为：[“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”、“0”]。所以，我们预测的最终结果为：[“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”、“0”]。

问题二：针对通道选择优化问题，我们采用 PCA 主成分分析进行降维操作，对各个通道进行权重排序。然后选择问题一中基于 CNN 的字符识别模型进行测试，基于测试集给出的前五个标签字符，改变通道数目，通过比较预测准确率高低，确定一个数目适量的最优通道。对于 S1 通道数为 16，最优通道组合为：[1，3，4，5，6，7，8，9，10，12，13，14，16，18，19，20]。同理可得出 S2、S3、S4、S5 的最优通道组合。最后通过对各个被

试者的通道加权平均，确定一个适用于 5 个被试者平均加权的通道排名。对五个被试者的最优通道选择 16 个通道：[1，3，4，5，6，8，9，10，11，12，13，14，16，18，19，20]。

问题三：根据问题二结果，选取 20 个通道中的 16 个通道(Fz，F4，Cz，C3，C4，T8，CP3，CP4，CP5，CP6，Pz，P3，P7，Oz，O1，O2)作为最优通道组合。对于附件 1 中所给数据，选取部分为有标签数据，部分为无标签数据，设计基于 SVM 的半监督分类模型。基于半监督分类的特征提取，在传统的判别分析方法上进行改进，通过有标签数据与无标签数据之间的欧式距离来获得二者之间的关系，进而来优化目标函数，求解投影向量，提取特征，然后通过投票分类的方法来进行最终的识别。在采用 16 个通道作为最优组合通道的条件下，半监督 SVM 算法在含有 10 个待预测字符的测试数据下预测结果分别为：“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“M”和“0”。

问题四：针对睡眠分期预测，我们设计了一种基于改进 K-Means 聚类的睡眠分期预测模型。首先，我们在原来 K-Means 算法的基础上，采用小批量数据子集减小计算时间，并同时尝试优化目标函数，其寻找质心的方法依然与 K-Means 一致，但是采用小批量数据进行迭代，可以更快得到收敛结果，该改进算法可以对附件中的数据进行更高效地处理。模型识别过程主要为根据数据中的各个特征相似度进行样本分组，根据各个特征类的相似性将数据中的集合划分成若干聚类或分组，从实现对数据的 K 分类。其次，我们通过分析模型在二维特征和三维特征下的聚类结果，得出的结论是处于“快速眼动期”和“睡眠Ⅰ期”的数据很难与其余类别的数据区分开。随后，我们通过比较迭代次数与模型预测准确率的关系，得出模型最优迭代次数为 30 次。之后我们比较了模型在选取不同比例训练数据下的准确率，发现模型可以在 40%-50%训练数据下取得较好的效果（测试集准确率 72.4%）。最后，我们给出了模型预测的混淆矩阵，通过计算模型在各个类别下的准确率与召回率，得出的结论是：模型在“深睡眠期”与“清醒期”两个类别上分类效果最好，而在“睡眠Ⅰ期”类别下的分类效果较差。

最后，对针对各问题设计的模型，进一步对各性能指标进行分析，从模型预测准确率，训练次数等方面分析，我们设计的模型在一定程度上有着良好的预测性能，为后续脑机融合研究落地化提供方向和思路。

关键词：CNN　SVM　PCA　半监督　K-Means

目 录

1 问题重述

1.1 问题背景

大脑作为人体当中最重要的一部分，承载了人的很多活动。其连着数以亿计的各种神经单元，它是人类各种命令的发出者和控制者。脑机接口（BCI）是人脑与外部设备之间的直接通信路径。这样的系统允许人们通过直接测量大脑活动进行交流。对于由于严重的运动障碍（例如脊髓损伤或肌萎缩性侧索硬化症，也称为 Lou Gehrig 病）而无法通过常规手段进行交流的人们，BCI 可能是唯一的交流手段[1]。在监测大脑活动的非侵入性方法中，可以考虑使用脑电图（EEG）技术。它们具有多种实用性：可以使用相对便宜的设备轻松记录数据，它们是无创 BCI 的常见解决方案。随着科学技术的进步，机器学习和模式识别领域取得很大的发展，使得生物工程中脑波信号的分析成为了研究热点。

其中 P300 是研究当中最常见的一种脑波信号，它在小概率事件的刺激下人脑会产生相应的响应，其 P300 电位在刺激产生后的 300ms 会达到一个正向的峰值。P300 电位对一般性刺激（大概率刺激）不会产生过大反应。与此同时，其电位的产生一般不依赖受试者本身的物理特性，和受试者认知层面有很大关系。当然 P300 是因为刺激开始以后的 300ms 电位产生正向电位的峰值，并不意味对于每一个受试者而言，这个电位都是 300ms 做出反应。经过科学研究 P300 电位产生的时间范围大概是 200~500ms 之间[2]。如下图所示就是刺激发生后 450 毫秒左右的 P300 波形。

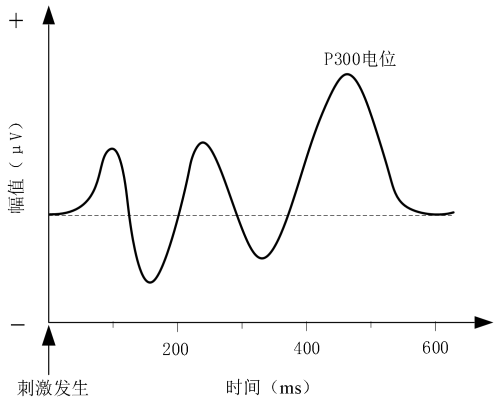

图 1.1 P300 波形示意图

对于被试者而言，采集的通道主要来源于 64 个不同的电极规范通道。本文只给出了 20 个通道的数据。具体的 20 个通道图 1.2，表 1.1 是采集的各个通道的名称。实验过程大致如下：

给予被试者一个 6 行 6 列的字符矩阵，由 26 个英文字数和 10 个阿拉伯数字组成，一共 30 个字符，字符矩阵如图 1.3。实验过程中，首先出现要识别的字符，其次，每次以随机顺序闪烁其中的一行或者一列，每次闪烁的时间是 80ms，间隔是 80ms。每一轮闪烁 12 次，对于一个被试着而言需要闪烁 5 轮，减少误差带来的影响。如下图 1.4 所示是单个字符闪烁实验过程。

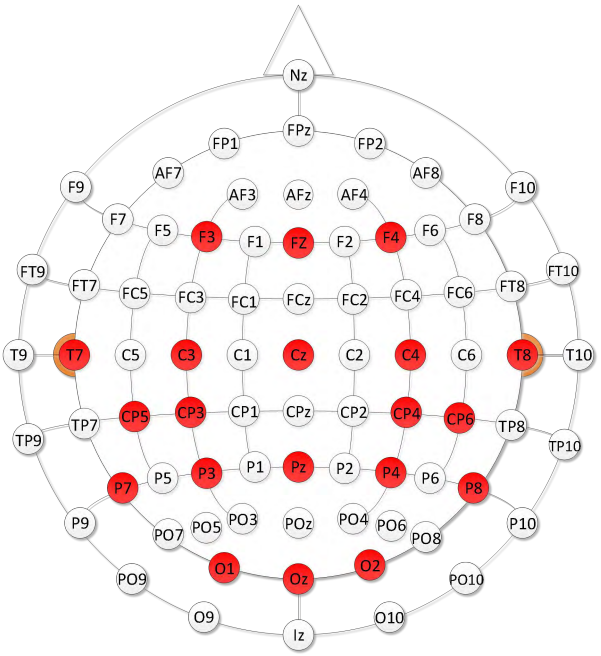

图 1.2　脑电信号采集通道图

表 1.1　采集通道的标识符

标识符	通道名称	标识符	通道名称
1	Fz	**11**	CP5
2	F3	**12**	CP6
3	F4	**13**	Pz
4	Cz	**14**	P3
5	C3	**15**	P4
6	C4	**16**	P7
7	T7	**17**	P8
8	T8	**18**	Oz
9	CP3	**19**	O1
10	CP4	**20**	O2

图 1.3　字符矩阵界面

睡眠是人类每天的必修课，睡眠质量的好坏对一个人的精神和健康有着重要的影响。在睡眠过程中监测脑电信号，可以用来分析一个人的健康状况进而诊断出相关疾病。除去清醒期以外，睡眠周期是由两种睡眠状态交替循环，分别是非快速眼动期和快速眼动期；人的睡眠是一个动态的过程，在非快速眼动期可以大致分为四个时期，分别是睡眠 I 期，睡眠 II 期，睡眠 III 期和睡眠 IV 期；睡眠 III 期和睡眠 IV 期又可合并为深睡眠期。由于脑电信号再不同的时期呈现的信号不同，因此可以根据信号不同的变化进行人体健康诊断。

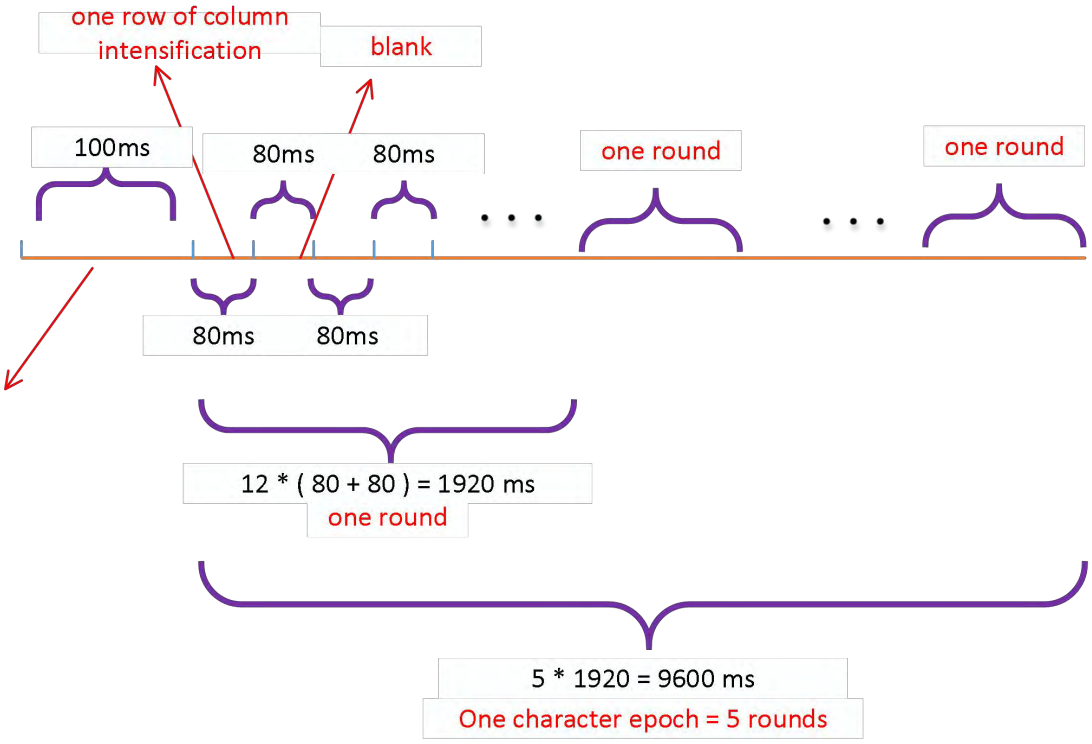

图 1.4　单个字符闪烁实验过程

1.2　问题提出

题目中给出了 2 个附件数据，附件 1 是 P300 脑机接口数据，采样频率是 250Hz，附件 2 是睡眠脑电数据。要求利用该数据解决如下问题：

问题一：脑机接口信息系统中，在保证分类准确率、信息传输速率和较少轮次的情况下给出 5 个被试者中的 10 个待识别目标的分类过程和结果，要求使用几种不同的方法做对比，以便给出所设计方法的合理性。

问题二：本文采集的是 20 个通道的数据，数据量庞大，其中可能包含一些对于目标分类和识别无关的通道，因此需要针对每个被试者选出有利于分类的最优通道。然后再综合每个被试者选择最适合所有人的通道。

问题三：在问题二中所得的一组最优通道的基础上，选取适量的样本，一部分作为有标签样本，一部分作为无标签样本利用问题所给的测试数据（char13-char17）作为检验，并找出剩余待识别目标（char18-char22）。

问题四：利用附件 2 给出的数据，建立睡眠分期预测模型，给出分类的过程（训练和测试数据选取方式和分配比例）和结果，并对分类。效果进行评价。

经过查阅文献和实践，目标识别和分类主要有 CNN、SVM、朴素贝叶斯等算法，本文主要使用多种算法对脑电信号分析和判别，确立本文的研究思路如图 1.5 所示。

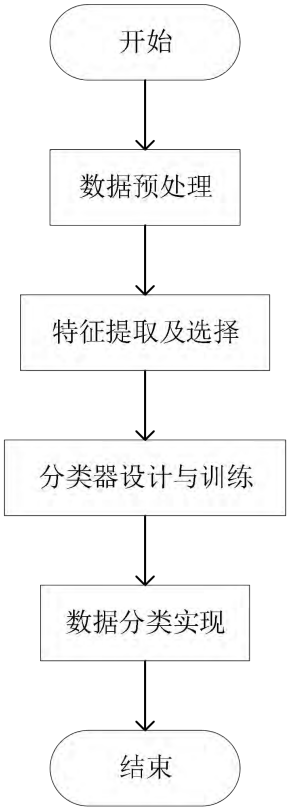

图 1.5　脑电信号分类判别流程图

2 问题分析

本题要求使用所给定的附件数据一和附件数据二进行 36 个字符识别和睡眠分期预测，并对几个模型进行对比和性能评价。对于题目中提出的问题具体分析如下：

针对问题一：首先，将脑电信号用带通滤波器作预处理,并用时域降采样法提取特征。其次，分别采用 5 层 CNN 网络、SVM 和贝叶斯逻辑回归模型进行对比训练，因为测试数据字符结果未知，训练数据字符结果已知，考虑将训练数据中的 80%数据作为训练数据，20%作为测试数据，利用训练数据中分得的测试数据来验证所设计的几种模型的合理性。最后，利用前面所设计的几种模型去识别 5 个被试着测试集中的 10 个待识别目标。

针对问题二：首先，将脑电信号用带通滤波器作预处理,并用时域降采样法提取特征。其次，选用 PCA 方法对 20 个通道进行降维操作，并按照贡献率进行排序。接着选用问题一的 5 层 CNN 网络进行训练，通过字符识别准确率的高低，确立分别适合每个被试者的最优通道数目。然后对五名被试者的 20 维贡献率进行加权平均，计算出五人贡献率平均排名，利用卷积神经网络进行训练，改变通道数目，通过识别准确率的高低确立一组适合五个被试者的最优通道。

最后，用给定测试数据（char13-char17）检验降维的有效性。

针对问题三：首先，在问题二得到的一组最优通道组合的基础上，选取训练集中的数据，按照一定的比例，分为有标签和无标签，对于比例的选取通过实验训练得到最优组合比例。其次，采用基于半监督 SVM 字符预测模型对数据进行训练，按照特征提取流程图，确定最优迭代次数，并选取测试集 char13-17 进行测试，确认模型的有效性。最后，对测试集中的数据（char18-22）进行预测，输出预测结果。

针对问题四：首先，对睡眠数据进行分析，设计基于 K-Means 的睡眠分期预测模型。其次，将数据归一化处理，比较数据归一化对预测准确率的影响。接着，对数据集进行 train/test 数据集划分，选取不同的分割比例，比较字符识别准确度。最后，在睡眠分期预测模型上，研究睡眠数据集四个特征以及特征组合对模型分类的影响，对结果进行预测，验证模型准确性，并结合模型预测性能指标分析模型。

3　模型假设

1.受试者应头发干净，过多的头油会导致头皮电阻过大，采集波形失真。
2.采集数据的过程应当远离电磁干扰并保证采集环境安静。
3.选择的特征参数集能够反映原数据的主要特征。
4.采集过程中受试者不应进行任何运动，保持肌肉放松并尽量减少眨眼，转动眼珠，吞咽口水等行为。
5.对于小概率事件，P300 信号总是会出现。
6.被试者没有眼睛方便的疾病。
7.被试者没有认知方面的困难。
8.脑电信号去噪效果比较理想。

4 符号说明

符号	含义
D	样本
μ_D	样本均值
σ_D^2	样本方差
x_j^l	第l层的j个特征图
M_j	输入的特征
k_{ij}^l	l层的卷积核
b_j^l	l层的偏置
f	激活函数
$h_{w,b}(x)$	输出的结果
R	实数矩阵
C	惩罚函数
α^*	最优解
cov	协方差
λ_i	特征值

5　模型建立与求解分析

5.1　问题一

对于二分类问题，有很多优秀的算法。本题通过不同算法之间进行优劣对比，选取最适合模型的分类，下面是问题一的求解思路流程图。

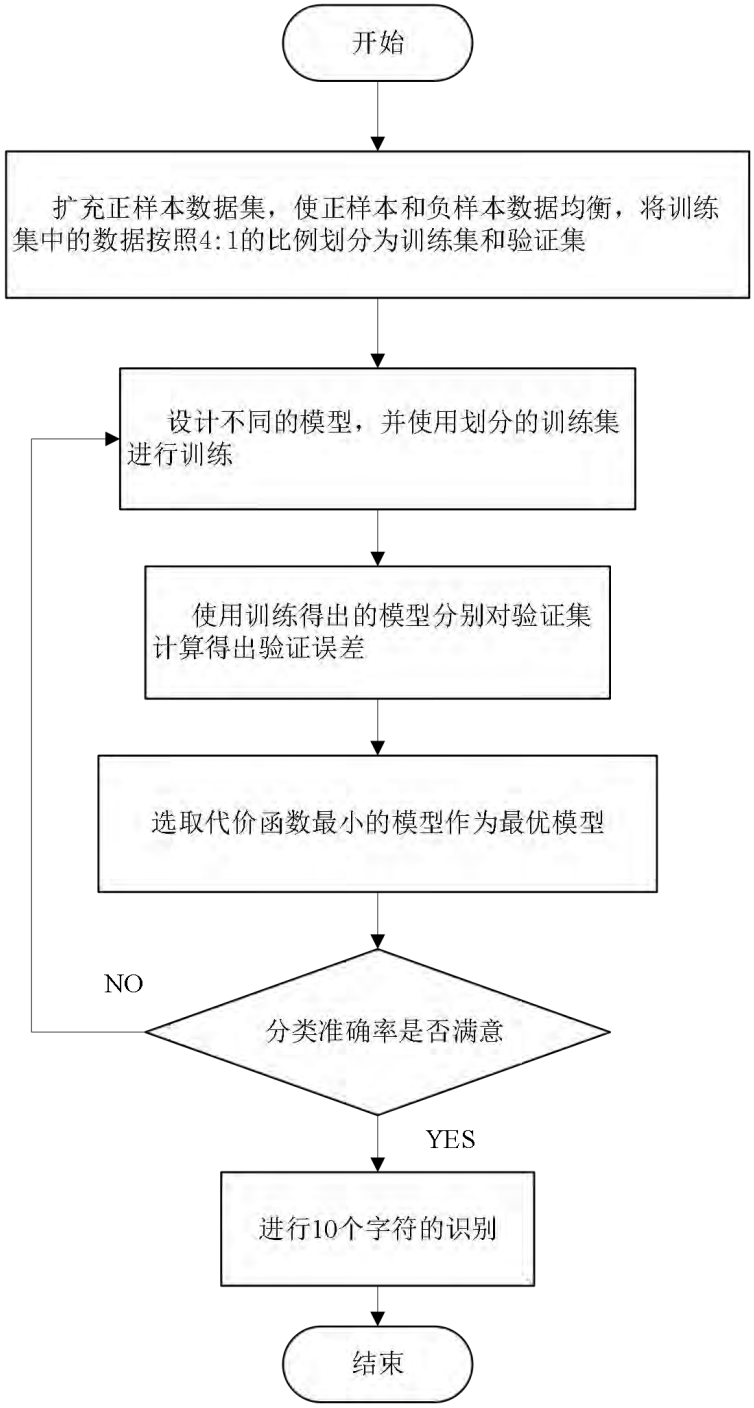

图 5.1　问题一的求解思路流程图

5.1.1 实验数据描述

本研究所使用的数据均为比赛官方所给定的数据，采样频率为 250Hz。一共有五名被试者，分别是 S1，S2，S3，S4 和 S5，每名被试者的脑电数据由 20 个通道构成（Fz、F3、F4、Cz、C3、C4、T7、T8、CP3、CP4、CP5、CP6、Pz、P3、P4、P7、P8、Oz、O1、O2）。若被试者要输入某个字符(称为目标字符，例如字符“B”)，则集中注意力注视这个字符即可。由于目标字符所在的行/列闪烁的概率为 1/6，因此会诱发出一个 P300 电位(如图 5.2 所示)。

本文截取每行/列开始闪烁前 100 ms 至闪烁后 800 ms 为一个时间窗，共 900ms，255 采样点。即一个数据样本为 255x20 的数据样本。字符需要所有行和列(12 行/列)重复闪 5 次，一个字符产生的样本为 60 个。训练数据集中每个被试者需要输入 12 个字符，因此对于一个被试者而言整个数据样本大小为 12x60=720 个。

P300 电位产生与刺激发生后的 200~500ms 之间，本文从截取时间窗中选取 150~650ms 作为特征数据。由于数据量比较庞大，为了提高训练的速度，可以采用在时域上降采样的方法，即每 4 个采样点选取 1 个。所以通过降采样的方式可以将采样点的数量由 225 个减少到 32 个，故数据样本的维数可以简化为 32x20 的矩阵。

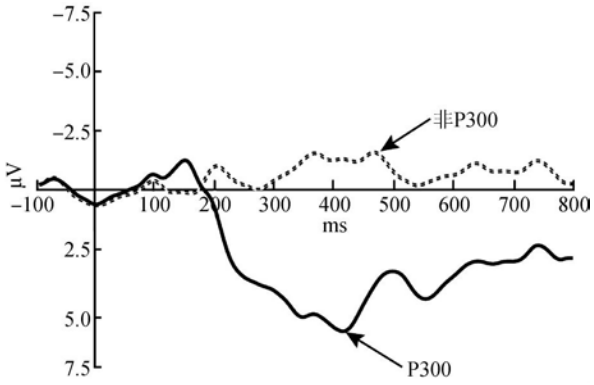

图 5.2　P300 波形

5.1.2 实验数据预处理

对于本文字符识别而言，本质上一个二分类问题，也就是当行或者列闪烁，检测是否有无 P300 电位。有就是 1，没有就是 0，打上相应的标签。根据所在的行或者列锁定相应的字符，这个时候就是 36 分类问题。但是由于一轮闪烁当中目标行或者列只出现 2 次，非目标行或者列出现 10 次，也就是正样本和负样本的数量是 1:5。如果用这个比例的样本进行训练，会导致模型直接把所有样本都会判断为负样本，这样的训练结果显示是不准确的，故考虑把正样本直接扩大 5 倍，使正负样本的比列达到 1:1。对于测试集而言，则不需要考虑这个问题。

对于激活函数 sigmoid 和 tanh 而言，二者均属于 S 型饱和非线性函数。对于神经网络而言，当数据穿过很多层时，很可能会发生数据偏移。考虑到很多模型用梯度下降法进行训练，然而激活函数 sigmoid 和 tanh 在饱和区的梯度很小，当采用反向传播的方法时，很容易发生梯度消失的现象。为了解决这个问题，本文拟采用批归一化（BN）的方法，这种算法可以有效地解决梯度容易消失的问题。

表 5.1　批归一化算法描述

批归一化算法
Input: $D=\{x_1,x_2,\cdots,x_m\}$表示一个 mini-batch 的 train 样本，m 表示样本的数量。x_i 为其中的第 i 个样本，$1\le i\le m$。
Output: $\hat{x}_i=BN(x_i)$ （5-1）
First step:求取 mini-batch 的均值：$$\mu_D\leftarrow\frac{1}{m}\sum_{i=1}^{m}x_i$$ （5-2）
Second step:求取 mini-batch 的方差：$$\sigma_D^2\leftarrow\frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_D)^2$$ （5-3）
Third step:对每个样本进行归一化处理：$$\hat{x}_i\leftarrow\frac{x_i-\mu_D}{\sqrt{{\sigma_D}^2+\in}}$$ （5-4）

5.1.3　方法一（CNN）

5.1.3.1　经典的卷积神经网络结构

经典的 CNN 属于深度前馈神经网络，它的训练主要使用反向传播算法。主要有全连接层、下采样层和卷积层组成。与此同时，由于其特有的网络结构，其具有三种良好的特性，分别是局部连接，下采样和权重共享[3]。这些特性造就了 CNN 的平移、缩放和旋转不变形[4]。

LeNet5 是一种应用比较广泛的 CNN 模型，其网络结构如图 5.3 所示，一共有 8 层[5]，其中输入层时一层，卷积层是 2 层，下采样层是 2 层，全连接层是 2 层，输出层是 1 层。

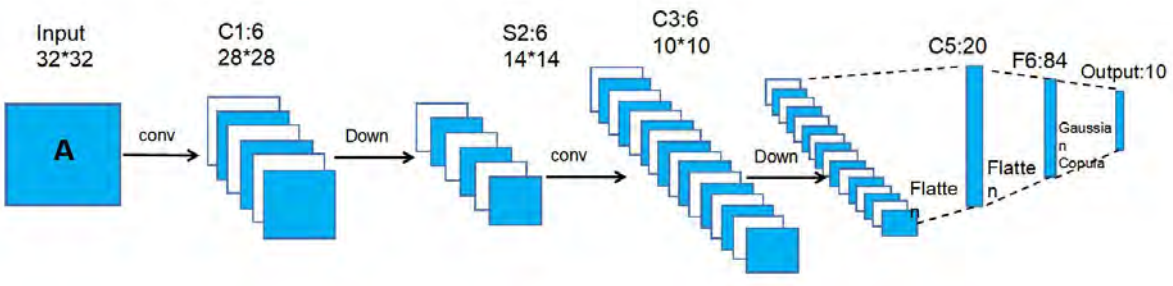

图 5.3　LeNet5　CNN 结构模型

5.1.3.2 基于 P300 脑电信号识别的 CNN 结构

本文采用基于 LeNet5 网络模型，在图 5.3 的特征数据 32x20 中，20 代表通道数目，表现了电信号的空间特征，32 是降采样中的采样点数目。根据所研究的数据特征对 LeNet5 中的卷积核的个数以及大小做出了一些改变，最终结构如图 5.4 所示。

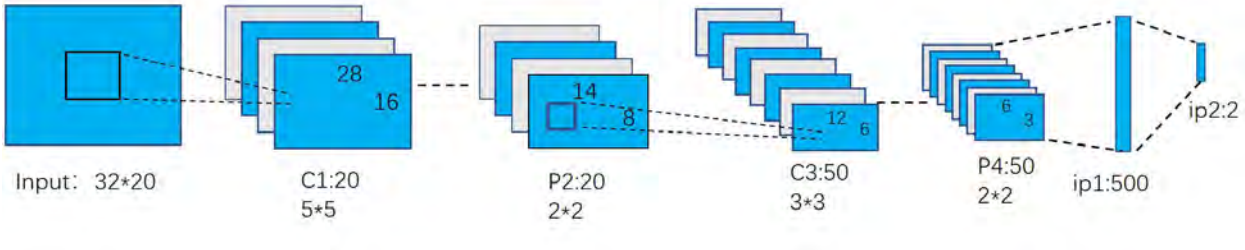

图 5.4 基于 P300 的 CNN 结构

（1）卷积层 C1 和下采样层 P2

对于输入层而言，其数据维数为 32x20。C1 有 20 个 5x5 的核。将输入层数据和 C1 进行运算可以得到一个特征映射图。初始化权重和初始化偏差分别采用 xavier 和 constant。P2 使用的方法是大池化。

（2）卷积层 C3 和下采样层 P4

C3 有 50 个 3x3 的核，C3 通过与 P2 的运算可以得到 50 个 12x6 的矩阵，P4 和 P2 一样均采用最大池化的方法。得到 50 个 6x3 的特征图。

（3）全连接层 ip1 和 ip2

ip1 设置了 500 个神经元，其输入数据的维数是 50x6x3。ip2 只设置 2 个神经元。本文的激活函数选择是 tanh，计算损失率采用的是 Softmax。

5.1.3.3 前项计算

首先需要对数据做相应的转换，将特征数据转换成 LMDB 格式[6]，对于卷积层的计算，其公式为：

$$x_j^l = f\left(\sum_{i \in M_j} x_i^{l-1} * k_{ij}^l + b_j^l\right) \tag{5-5}$$

x_j^l 代表第 l 层的 j 个特征图，M_j 代表输入的特征，k_{ij}^l 代表 l 层的卷积核，b_j^l 代表 l 层的偏置，f 代表采用的激活函数。

其次，卷积层的池化采用均值池化的方式，其计算公式如下：

$$x_j^l = f\left(\beta_j^l down\left(x_j^{l-1}\right) + b_j^l\right) \tag{5-6}$$

β_j^l 代表 l 层权重参数，$down$ 为下采样函数。

最后，全连接层承载着分类的功能，本文利用 Softmax 函数实现分类功能，其计算公式如下：

$$p\left(y^{(i)}=j \mid x^{(i)};\theta\right)=\frac{e^{\theta_j^T x^{(i)}}}{\sum_{i=1}^{k}\theta_j^T x^{(i)}} \tag{5-7}$$

5.1.3.4 反向误差计算

首先，经过各个层的连接输出后，反向误差的代价函数公式如下：

$$E^N=\frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{C}\left(t_k^n-y_k^n\right)^2 \tag{5-8}$$

公式（5-8）中 N 为样本个数，t_k^n 代表第 n 个样本的第 k 维，y_k^n 代表第 n 个样本的第 k 个输出。

其次是进行网络参数调整，利用 BP 算法优化各个网络层权重，残差的公式如下：

$$\delta_i^{n_l}=\frac{\partial}{\partial z^{nl}}\bullet\frac{1}{2}\left\|y-h_{w,b}(x)\right\|^2=-\left(y_i-a_i n_l\right)\bullet f'\left(z_i n_l\right) \tag{5-9}$$

n_l 代表的是输出层，z_i^{nl} 代表没有经历激活函数加权的输出层，$h_{w,b}(x)$ 代表输出的结果。y 代表标准的输出，$\delta_i^{n_l}$ 代表第 i 个输出。

计算残差按照公式（5-10）公式：

$$\delta_i^l=\left(\left(w^l\right)^T\delta^{(l+1)}\right)\bullet f'\left(z^l\right) \tag{5-10}$$

如果 l 为卷积层，需要下采样层进行相应的操作，具体公式如下：

$$\delta_i^l=\beta_j^{l+1}\left(f'\left(\mu_j^l\right)\bullet up\left(\delta_i^{l+1}\right)\right) \tag{5-11}$$

如果 l 为下采样层，则卷积核的权值为：

$$\delta_i^l=f'\left(\mu_j^l\right)\bullet conv2\left(\delta_j^{l+1},rot180\left(k_j^{l+1}\right),full'\right) \tag{5-12}$$

$conv2$ 代表卷积运算的相关函数，$rot180$ 代表对卷积核进行翻转。

最后一个重要的步骤就是更新权重参数。

$$w^l=w^l-a\left[\left(\frac{1}{m}\Delta w^l\right)\right] \tag{5-13}$$

$$b^l=b^l-a\left[\left(\frac{1}{m}\Delta b^l\right)\right] \tag{5-14}$$

则相应的卷积层的更新公式为：

$$\frac{\partial E}{\partial k_{ij}^l}=rot\left(conv2\left(x_i^l,rot180\left(\delta_i^l\right),valid'\right)\right) \tag{5-15}$$

接着下采样的权重公式也要相应的更新，其公式为：

$$\frac{\partial E}{\partial \beta_j} = \sum_{u,v} \left(\delta_j^i \bullet d_j^l\right)_{uv} \tag{5-16}$$

$$d_j^l = down\left(d_j^{l-1}\right) \tag{5-17}$$

综上 CNN 的算法描述如下所示：

表 5.2　CNN 算法描述

CNN 算法
Input：训练数据集 $\mathrm{T} = \{(x_1, y_1),(x_2, y_2),...,(x_N, y_N)\}$，其中，$\mathrm{x}_i \in \Re^n$， $\mathrm{y}_i \in \{0,1\}, i = 1,2,..., N$； **Output**：CNN 模型。 Input 输入层：用于数据的输入，大小为 32*20； Convolution 卷积层 1：使用卷积核进行特征提取和特征映射，卷积核个数为 100，大小为 5*5； Pool 池化层 1：进行下采样，对特征稀疏处理，大小为 2*2； Convolution 卷积层 2：作用同卷积层 1，大小为 3*3； Pool 池化层 2：作用同池化层 1，大小为 2*2； Full connect 全连接层 1：重新拟合，减少特征信息损失，大小为 500； Full connect 全连接层 2（Output 层）：作用同全连接层 1，大小为 2。

5.1.3.5　性能评价指标

为了检测模型建立的好坏，我们可以从 P300 的检测效果和 36 个字符识别的准确率方面加以评价。目前常用的指标有 *TP*、*TN*、*FP*、*FN* 。其中 *TP* 代表正样本中识别正确的对应的数量，*TN* 代表负样本被识别为正确的对应数量，*FP* 代表正样本被错误识别的相对应的数量，FN 代表正样本中被识别为错误的对应数量[7]。故识别准确的定义为：

$$Accury = \frac{TP + TN}{TP + TN + FP + FN} \tag{5-18}$$

与此同时，二分类还有以下性能指标，例如 Recall 、 Precision 、 *FI* - s*ocre* 。对应计算公式分别如下：

$$\mathrm{Re}\,call = \frac{TP}{TP + FN} \tag{5-19}$$

$$\mathrm{Pr}\,ecision = \frac{TP}{TP + FP} \tag{5-20}$$

$$FI - socre = 2\frac{\mathrm{Re}\,call * \mathrm{Pr}\,ecision}{\mathrm{Re}\,call + \mathrm{Pr}\,ecision} \tag{5-21}$$

从理论上说 $\mathrm{Re}\,call$ 和 $\mathrm{Pr}\,ecision$ 相互影响，在实际训练过程中，一般不太容易达到最优，一般可以选择 $FI - socre$ 作为综合性指标进行度量。

5.1.3.6 结果分析

从图 5.5 可知，对于五个被试者（S1、S2、S3、S4、S5）而言，随着迭代次数的增加，其识别的准确率均在不同程度地增加，当迭代次数将近 100 次时，准确率达到了 80%左右，说明所建立的模型有一定的有效性。其中，S1 准确率最高，S3 准确率最低。图 5.6 是将 5 名被试者准确率取平均，发现整体而言。其识别准确率随着迭代次数的增加在逐渐提高。图 5.7 中损失（loss）随着迭代次数的增加而逐渐降低，最终稳定在一个值附近。表 5.3 是 CNN 算法的 P300 检测的一些性能指标，其在一定程度上验证了前面的曲线结果。故本次实验所设计的 CNN 模型具有一定的合理性。

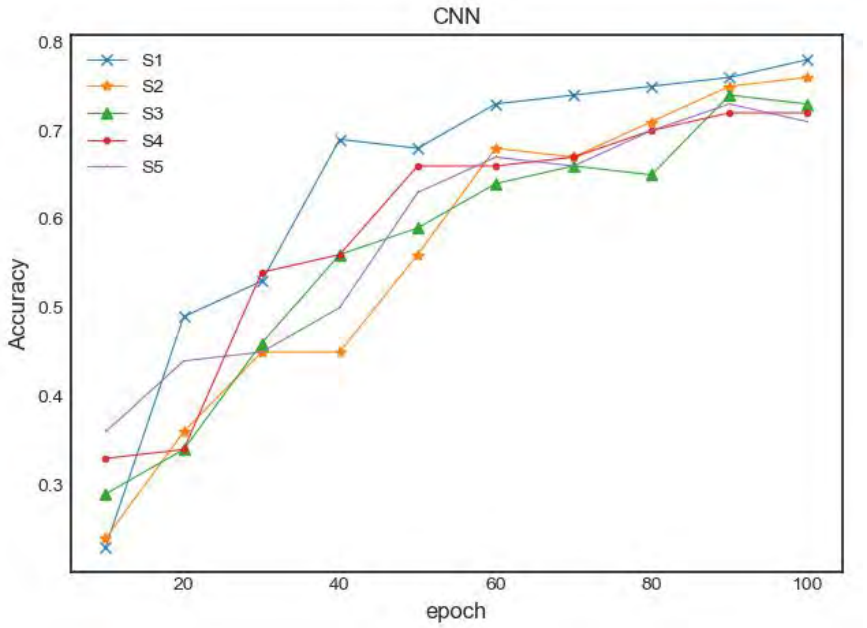

图 5.5　CNN 算法在不同被试数据上的准确率

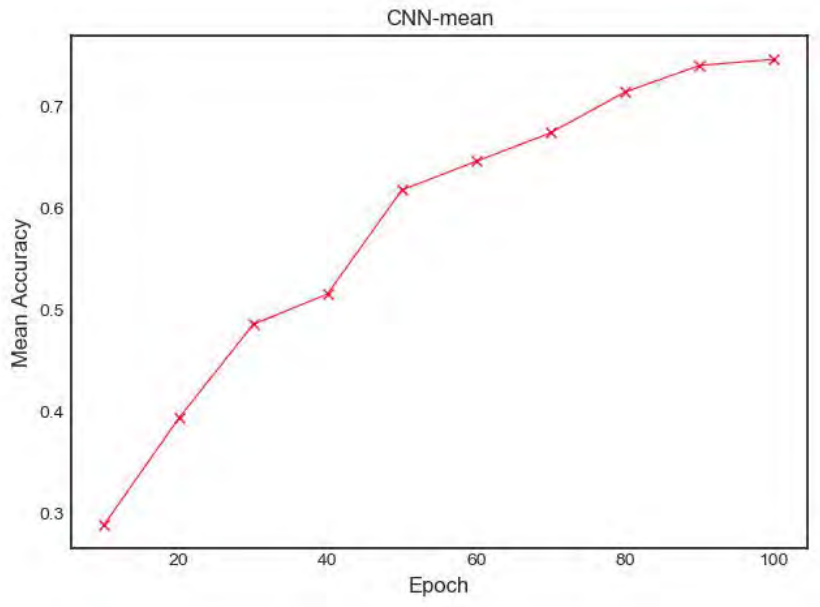

图 5.6　CNN 算法在 5 位被试者数据上的平均准确率

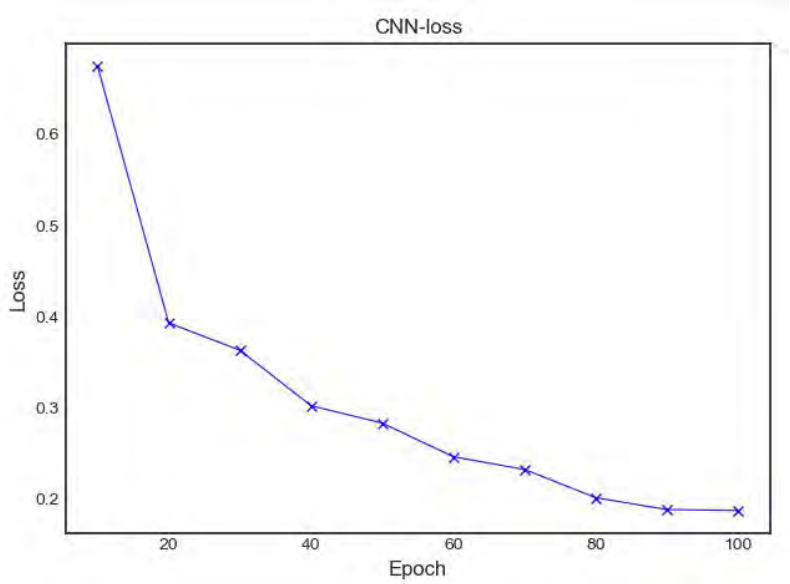

图 5.7　CNN 算法在验证集上的平均损失函数

表 5.3　CNN 的 p300 检测性能

Subject	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-score
S1	126	27	37	114	0.783	0.773	0.525	0.625
S2	144	38	29	126	0.765	0.832	0.533	0.650
S3	135	42	38	130	0.733	0.780	0.509	0.616
S4	126	37	40	127	0.724	0.759	0.498	0.601
S5	130	29	34	136	0.712	0.793	0.489	0.605

表 5.4　CNN 模型在测试数据上的预测结果

	char13	char14	char15	char16	char17	char18	char19	char20	char21	char22
r1	(1,7)	(1,12)	(6,7)	(5,10)	(1,9)	(4,8)	(1,11)	(4,12)	(1,7)	(4,12)
r2	(2,7)	(4,12)	(3,7)	(3,10)	(2,9)	(4,8)	(3,11)	(4,12)	(3,7)	—
r3	(3,7)	(1,12)	(6,7)	(5,10)	(2,9)	(4,8)	(2,11)	(4,12)	(1,7)	—
r4	(3,7)	(2,12)	(6,8)	(5,10)	(2,8)	(4,8)	(2,11)	(4,12)	(3,7)	(6,12)
r5	(5,7)	(1,12)	(6,7)	(5,10)	(2,9)	(1,10)	(2,11)	(2,12)	(4,7)	(6,12)

上表中，r1-r5 表示轮数，char13-char22 表示测试数据中要预测的 10 个字符，我们希望在较少轮次的测试数据的条件下，对 10 个待预测字符进行预测，我们规定对于任一待预测字符，当某一轮次的预测值与之前的轮次的预测结果一致，即为预测成功。根据上表可知，大多数字符可以在三轮内（包括三轮）成功预测（见表中标为浅蓝色的预测结果）。根据上表中的行列预测结果，查表可知待预测的 10 个字符分别是：“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”、“0”。但预测结果也存在一些歧义（如 char21 标为黄色的预测结果），两类预测结果各自出现两次，所以该字符无法准确预测。

5.1.4 方法二（贝叶斯）

5.1.4.1 基于贝叶斯的脑电信号模型

对于 P300 信号而言，靶刺激与非靶刺激对其影响是不同的，靶刺激闪烁可以检测到 P300 电位，而非靶刺激无法检测到 P300 电位，对于靶刺激而言，搜集 K 个刺激的脑电信号如下：

$$X=\left(X^{(1)},\cdots,X^{(l)},\cdots,X^{K}\right),X^{(l)}\in R^{C\times T} \tag{5-22}$$

$X^{(l)}$ 是第 l 个刺激作用下的序列，C和T 分别是通道数量和时间。主要是从 K 个刺激中选出靶刺激 $a\in(1,\cdots,K)$，公式如下：

$$f_{\theta}\left(X^{l}\right)=\left\langle W,X^{(l)}\right\rangle+b \tag{5-23}$$

$$p_{\theta}(a\mid X)=\frac{e^{f_{\theta}\left(X^{(a)}\right)}}{\sum_{1}^{K}e^{f_{\theta}\left(X^{(l)}\right)}} \tag{5-24}$$

$p_{\theta}(a\mid X)$代表在第 a 个刺激下产生 P300 点位的概率，$W\in R^{C\times T}$ 属于系数矩阵。对于 P300 目标检测任务而言，其最大化为[8]：

$$\hat{a}=\arg\max_{a}\lg p_{\theta}(a\mid X)=\arg\max_{a}f_{\theta}\left(X^{(a)}\right) \tag{5-25}$$

5.1.4.2 组稀疏贝叶斯逻辑回归通道选择

从本质上来说，检测靶刺激和非靶刺激下脑电信号是否包含 P300 信号，属于一个二分类问题。首先可以将 P 维特征向量 $x\in R^{P}$ 进行映射，其映射到类别标签 $t\in\{-1,1\}$ 上的线性模型为：

$$t=\omega^{T}x+\varepsilon \tag{5-26}$$

ω 属于权重向量，ε 为精度 β 的零均值高斯变量，*GLASSO* 可以表示为：

$$\hat{\omega}=\arg\min_{\omega}\left\|y-t\right\|_{2}^{2}+\lambda\sum_{i=1}^{G}\left\|\omega_{i}\right\|_{2} \tag{5-27}$$

ω_{i} 作为第 i 组的重要权重向量，G 是通道组的数量。对于一定量的数据，可以假设权重向量服从下面的高斯分布：

$$p(t\mid X,\omega,\beta)=N\left(t\mid X^{T}\omega,\beta^{-1}\right) \tag{5-28}$$

$$\alpha=\left(\alpha_{1},\alpha_{2,\cdots,}\alpha_{g},\cdots\alpha_{G}\right) \tag{5-29}$$

其后验概率也必然服从高斯分布，其形式如下：

$$p(\omega\mid t,X,\alpha,\beta)=N\left(\omega\mid m,\sum\ \right) \tag{5-30}$$

综上，贝叶斯算法描述如下：

表 5.5　贝叶斯算法描述

贝叶斯算法
First stage--preparation： **Input：**训练数据集 $\mathrm{T}=\{(x_1,y_1),(x_2,y_2),...,(x_n,y_n)\}$，其中，$\mathrm{x}_i=\{\mathrm{a}_1,\mathrm{a}_2,...,\mathrm{a}_m\}$ 为一个分类项，而每个 a 为 x 的一个特征属性，$\mathrm{y}_i\in\{0,1\},i=1,2,...,N$； **Output：**特征属性和训练样本。 **Second stage--train：** **Input：**特征属性和训练样本； **Output：**贝叶斯分类器。 **Third stage--apply：** **Input：**分类器和待分类项 $\mathrm{x}=\{a_1,a_2,...,a_m\}$，其中每个 a 为 x 的一个属性； **Output：**待分类项与类别的映射关系。 First step：有类别集合 $\mathrm{C}=\{y_1,y_2,...,y_n\}$，计算 $\mathrm{P}(y_i\mid x)$，根据训练样本集统计得到在各类别下各个特征属性的条件概率估计，即： $P(a_1\mid y_1),P(a_2\mid y_1),...,P(a_m\mid y_1);P(a_1\mid y_2),P(a_2\mid y_2),...,P(a_m\mid y_2);...;P(a_1\mid y_n),$ $P(a_2\mid y_n),...,P(a_m\mid y_n)$ Second step：设各个特征属性是条件独立的，根据贝叶斯定理有： $P(y_i\mid x)=\dfrac{P(x\mid y_i)P(y_i)}{P(x)}$ Third step：分母对于所有类别为常数，则将分子最大化，有： $P(x\mid y_i)P(y_i)=P(a_1\mid y_i)P(a_2\mid y_i)...P(a_m\mid y_i)P(y_i)=P(y_i)\prod_{j=1}^{m}P(a_j\mid y_i)$ Forth step：若 $P(y_k\mid x)=\max\{P(y_1\mid x),P(y_2\mid x),...,P(y_n\mid x)\}$，则 $\mathrm{x}\in y_k$。

5.1.4.3　结果分析

从图 5.8 可以看到，当采用贝叶斯算法时，从五名被试者随迭代次数增加识别准确率变化曲线可以看到，准确率和迭代次数呈正相关关系。经过 100 次迭代左右，五人的识别准确率均达到了 70%以上，可以判断模型基本有效。与此同时，贝叶斯算法在 5 位被试者数据上的平均准确率也随着迭代次数的增加，而逐渐增加。当然其损失率也随着迭代次数增加而逐步降低。从表 5.5 也可以侧面验证这一点。综上，可以用其去识别 10 个待识别字符。

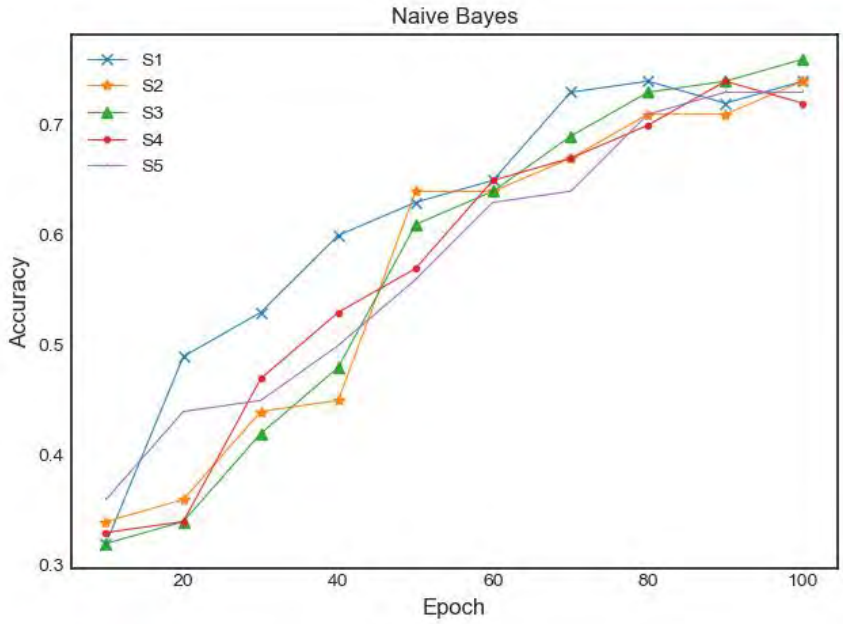

图 5.8　贝叶斯算法在不同被试数据上的准确率

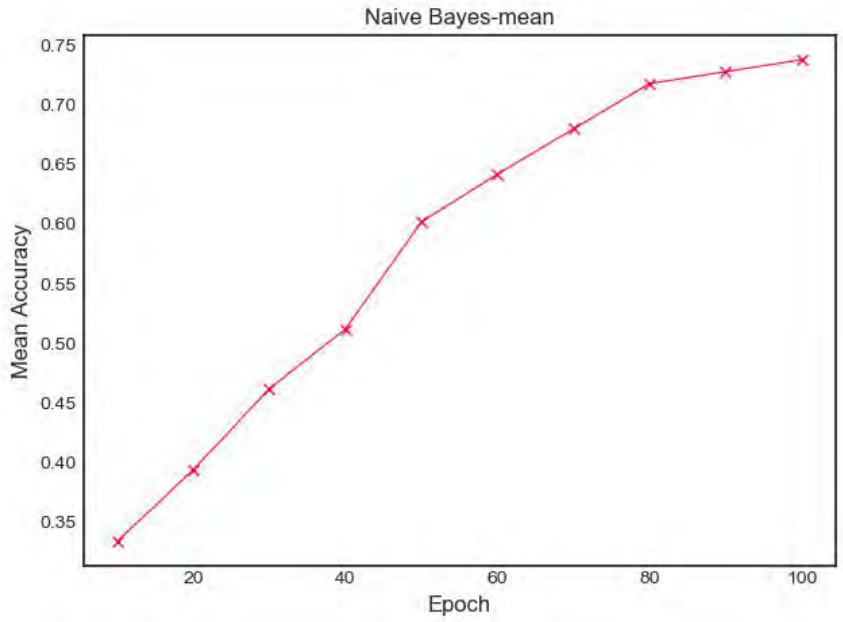

图 5.9　贝叶斯算法在 5 位被试者数据上的平均准确率

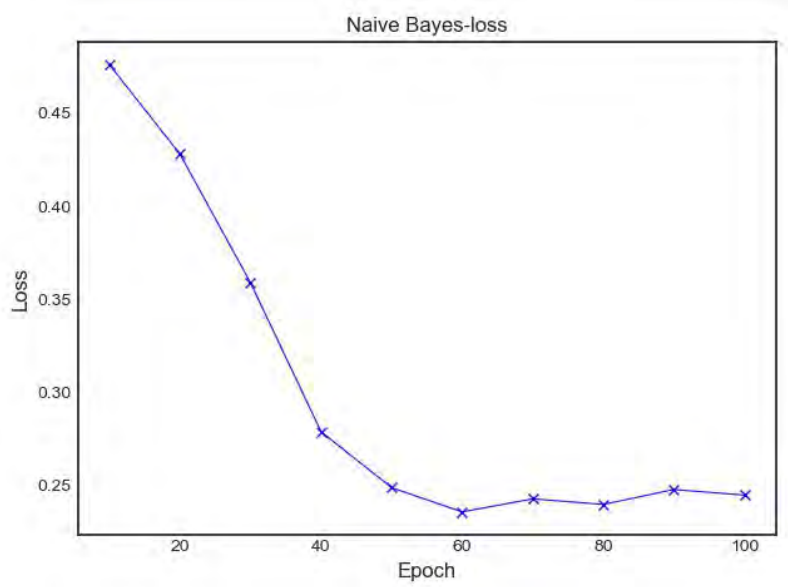

图 5.10　朴素贝叶斯算法在验证集上的平均损失函数

表 5.6　贝叶斯的 P300 检测性能

Subject	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-score
S1	129	27	37	144	0.713	0.777	0.473	0.588
S2	138	39	29	136	0.765	0.826	0.504	0.626
S3	145	42	38	125	0.690	0.792	0.537	0.640
S4	126	33	34	137	0.724	0.788	0.479	0.596
S5	120	22	43	136	0.684	0.736	0.469	0.573

表 5.7　贝叶斯模型在测试数据上的预测结果

	char13	char14	char15	char16	char17	char18	char19	char20	char21	char22
r1	(1,7)	(2,11)	(6,7)	(6,10)	(1,9)	(2,8)	(1,7)	(4,12)	(2,7)	(4,12)
r2	(2,7)	(4,10)	(3,7)	(5,10)	(2,10)	(4,8)	(2,11)	(4,12)	(2,8)	—
r3	(1,7)	(1,12)	(5,7)	(5,11)	(2,9)	(4,8)	(1,11)	(4,12)	(1,12)	—
r4	(3,7)	(4,12)	(6,7)	(4,10)	(2,7)	(5,9)	(2,10)	(5,12)	(1,7)	(6,12)
r5	(1,8)	(1,12)	(4,7)	(5,10)	(2,9)	(1,11)	(2,11)	(1,12)	(1,7)	(6,12)

根据上表可知，在和上述相同的判别标准下，贝叶斯算法的训练结果较差（大多数字符在第 5 轮测试后才能成功判别，且对于各个字符来讲，各轮预测结果差别也较大，如上表“char14”、“char16”、“char17”、“char19”和“char21”），所以，预测出来的 10 个字符是“A”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”、“0”。

5.1.5　方法三（SVM）

5.1.5.1 原理介绍

SVM 的特点是采用小样本进行训练学习，不同于传统的一些方法。对于传统方法而言，通常所训练的数据越大，其性能越好。SVM 并不需要这样，因此可以避免出现过拟合的情况。SVM 本质上属于二分类，但是也可以通过算法解决多分类问题。

SVM 的基本原理是通过一个非线性变换，将输入通过这个非线性变换成高维空间。这样在当前空间无法通过线性分类解决的问题，在高维空间就可以通过线性分类进行解决，如图 5.11 所示。

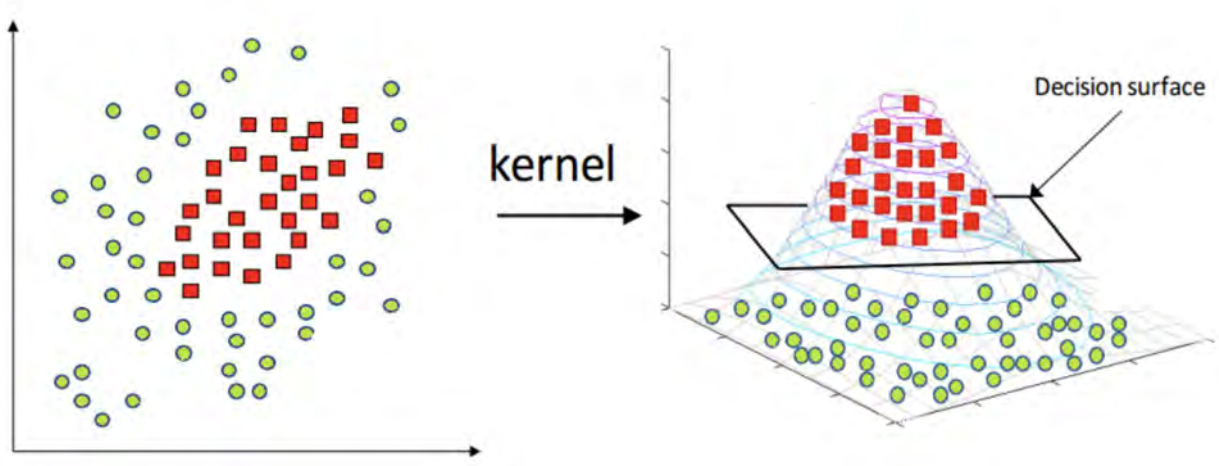

图 5.11 SVM 算法理论解释

对于 SVM 算法而言，可以这样描述，对于已知 n 个样本$(x_1,y_1),(x_2,y_2),(x_3,y_3),\cdots,(x_n,y_n)$

其中$x_i(i=1,2,\cdots,n)$为观测数据，$y_i(i=1,2,\cdots,n)$为相对应的类别标号。y_i取值为−1 或者 1。利用一个非线性变换，将输入数据变换到高维空间，再进行分类，其分类判别公式为：

$$f(x)=\mathrm{sgn}\left[\sum_{i=1}^{N} y_i a_i <\Phi(\mathrm{x}_i),\Phi(\mathrm{x})>+b\right] \qquad (5\text{-}31)$$

公式（5-31）N 为支持向量的数目，a_i为拉格朗日乘子，b 为分类的阈值。

对于 SVM 算法而言，其核心是和核函数，通过不同的核函数可以实现不用的非线性决策面的学习机。这样可以造就不同 SVM 算法。对于上述的高维分类判别公式，可以通过核函数的介入，变换成低维的写法，如下所示：

$$f(x)=\mathrm{sgn}\left[\sum_{i=1}^{N} y_i a_i K(\mathrm{x}_i,\mathrm{x})-b\right] \qquad (5\text{-}32)$$

其中$K(\mathrm{x}_i,\mathrm{x})$为核函数，核函数有很多种，各有各的特点。本研究采用的是线性核函数，其公式如下。

$$K(\mathrm{x},\mathrm{x}_i)=\langle \mathrm{x},\mathrm{x}_i\rangle \qquad (5\text{-}32)$$

综上 SVM 算法描述如下：

表 5.8　SVM 算法描述

SVM 算法描述

Input：训练数据集 $\mathrm{T}=\{(x_1,y_1),(x_2,y_2),...,(x_N,y_N)\}$，其中，$\mathrm{x}_i\in\Re^n$，

$\mathrm{y}_i\in\{0,1\},i=1,2,...,N$；

Output：分离超平面和分类决策函数。

First step：选择惩罚函数 $C>0$，构造并求解凸二次规划问题：

$$\min_{\alpha}\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(x_i\cdot x_j)-\sum_{i=1}^{N}\alpha_i$$

$$\text{s.t.}\quad\sum_{i=1}^{N}\alpha_i y_i=0$$

$$0\le\alpha_i\le C,i=1,2,...,N,$$

得到最优解 $\alpha^*=(\alpha_1^*,\alpha_2^*,...,\alpha_N^*)^T$；

Second step：计算 $\omega^*=\sum_{i=1}^{N}\alpha_i^* y_i x_i$，选择 α^* 的一个分量 α_j^* 满足条件 $0<\alpha_j^*<C$，计

算 $\mathrm{b}^*=y_j-\sum_{i=1}^{N}\alpha_i^* y_i(x_i\cdot x_j)$；　　（5-33）

Third step：求分离超平面 $\omega^*\cdot x+b^*=0$，以及分类决策函数 $\mathrm{f}(x)=sign(\omega^*\cdot x+b^*)$。

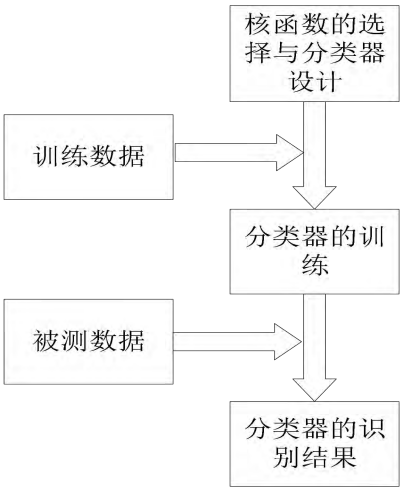

图 5.12　SVM 算法流程图

5.1.5.2　结果分析

从图 5.13 可以看到，当采用 SVM 算法时，从五名被试者随迭代次数增加识别准确率变化曲线可以看到，准确率和迭代次数呈正相关关系。经过 100 次迭代左右，五人的识别准确率均达到了 70%以上，可以判断模型基本有效。与此同时，SVM 算法在 5 位被试者数据上的平均准确率也随着迭代次数的增加，而逐渐增加。当然其损失率也随着迭代次数增加而逐步降低。从表 5.9 也可以侧面验证这一点。综上，可以用其去识别 10 个待识别字符。

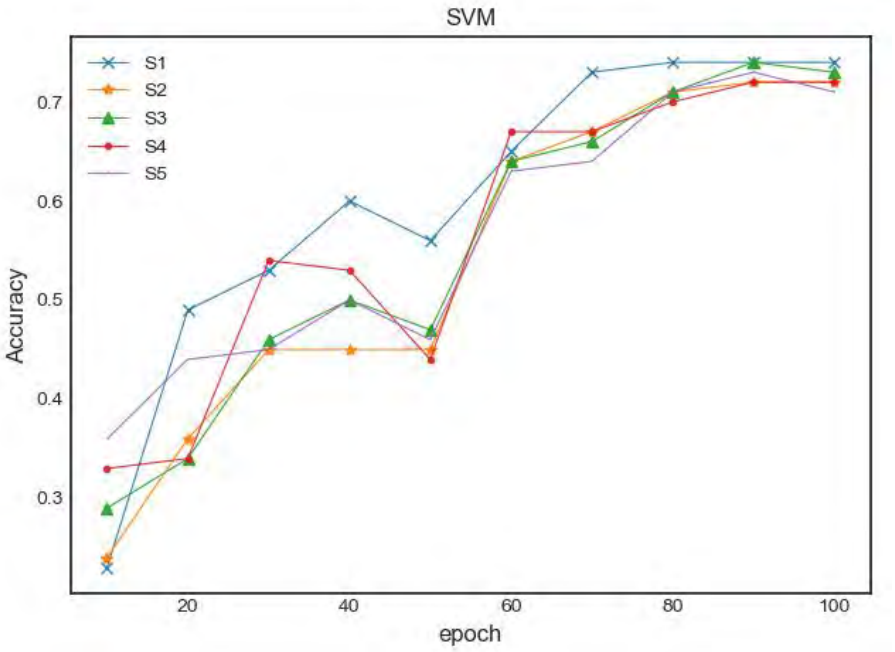

图 5.13　SVM 算法在不同被试者数据上的准确率

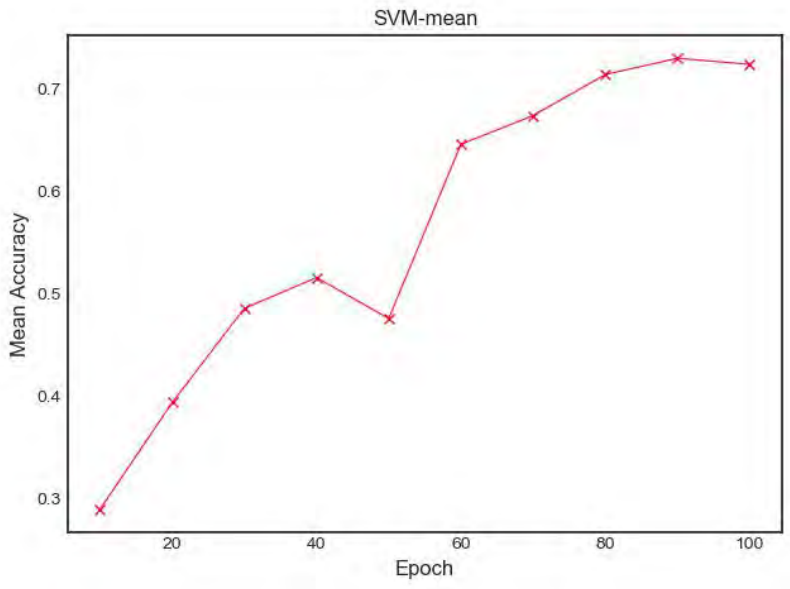

图 5.14　SVM 算法在 5 位被试者数据上的平均准确率

表 5.9　SVM 的 p300 检测性能

Subject	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-score
S1	113	36	37	120	0.683	0.753	0.485	0.590
S2	128	38	36	126	0.727	0.780	0.504	0.612
S3	125	42	29	130	0.712	0.811	0.490	0.611
S4	147	37	33	134	0.729	0.816	0.523	0.637
S5	110	39	34	147	0.692	0.763	0.428	0.548

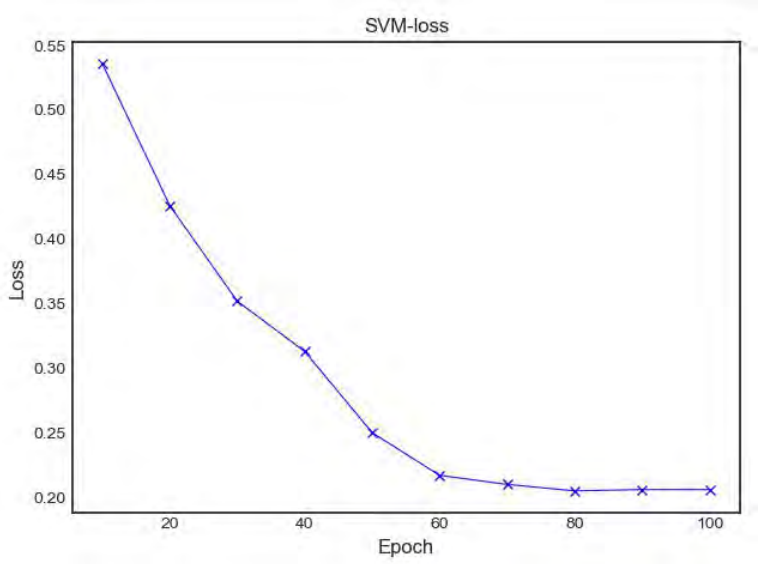

图 5.15　SVM 算法在验证集上的平均损失函数

表 5-10　SVM 模型在测试数据上的预测结果

	char13	char14	char15	char16	char17	char18	char19	char20	char21	char22
r1	(1,7)	(1,12)	(6,7)	(5,10)	(1,9)	(4,8)	(1,11)	(4,12)	(1,7)	(4,12)
r2	(6,7)	(4,12)	(3,7)	(3,10)	(2,10)	(4,8)	(3,11)	(4,12)	(3,7)	—
r3	(3,7)	(1,12)	(6,7)	(5,10)	(2,9)	(4,8)	(2,11)	(4,12)	(1,7)	—
r4	(3,7)	(2,12)	(6,8)	(5,10)	(2,8)	(4,8)	(4,11)	(4,12)	(3,7)	(6,12)
r5	(6,7)	(1,12)	(6,7)	(5,10)	(2,9)	(1,10)	(2,11)	(2,12)	(4,7)	(6,12)

根据上表可知，SVM 算法的训练效果介于 CNN 与贝叶斯算法之间，即多数字符在第 2-3 轮被成功预测，而预测结果与 CNN 的预测结果较为相似。但不足的方面是有两个字符可能存在歧义（见“char13”和“char21”中标为黄色的数据），所以，预测的 10 个字符是“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”、“0”。

5.1.6　三种方法对比分析

由上文分析可知，本研究选取的三种方法（CNN、贝叶斯、SVM）均具有一定的有效性。都可以作为预测 10 个字符的标准。由于要选出一种较优方法，评价一个模型的好坏

重要标准就是其准确率，图 5.16 展示了三类算法在 5 位被试者数据上的平均准确率比较，可以看到 CNN 算法相比另外两种算法表现稍优秀一些，因此本研究最终选择 CNN 算法来进行 10 个未知字符的识别，又上文可知，最终 10 个待识别字符分别为“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”、“0”。

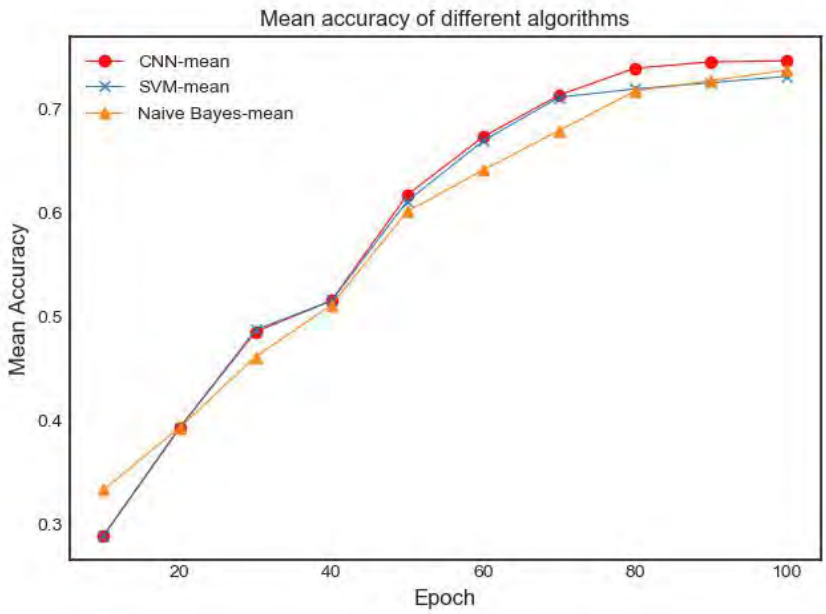

图 5.16　三类算法在 5 位被试者数据上的平均准确率比较

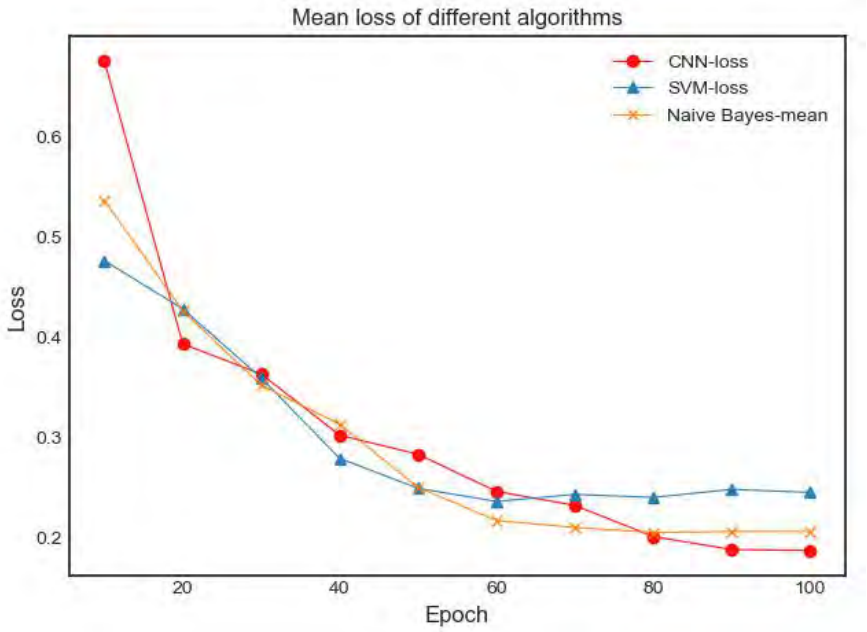

图 5.17　三类算法在 5 位被试者数据上的平均损失函数比较

5.2 问题二

对于问题二而言，首先应该对各个通道进行降维操作，选择主成分分析（PCA）进行降维操作，对各个通道进行权重排序。确定一个适用于 5 个被试者平均加权的通道排名。然后选择上文 CNN 算法进行测试，改变通道数目，通过准确率高低，确定一个数目适量

的最优通道，其流程图如下所示：

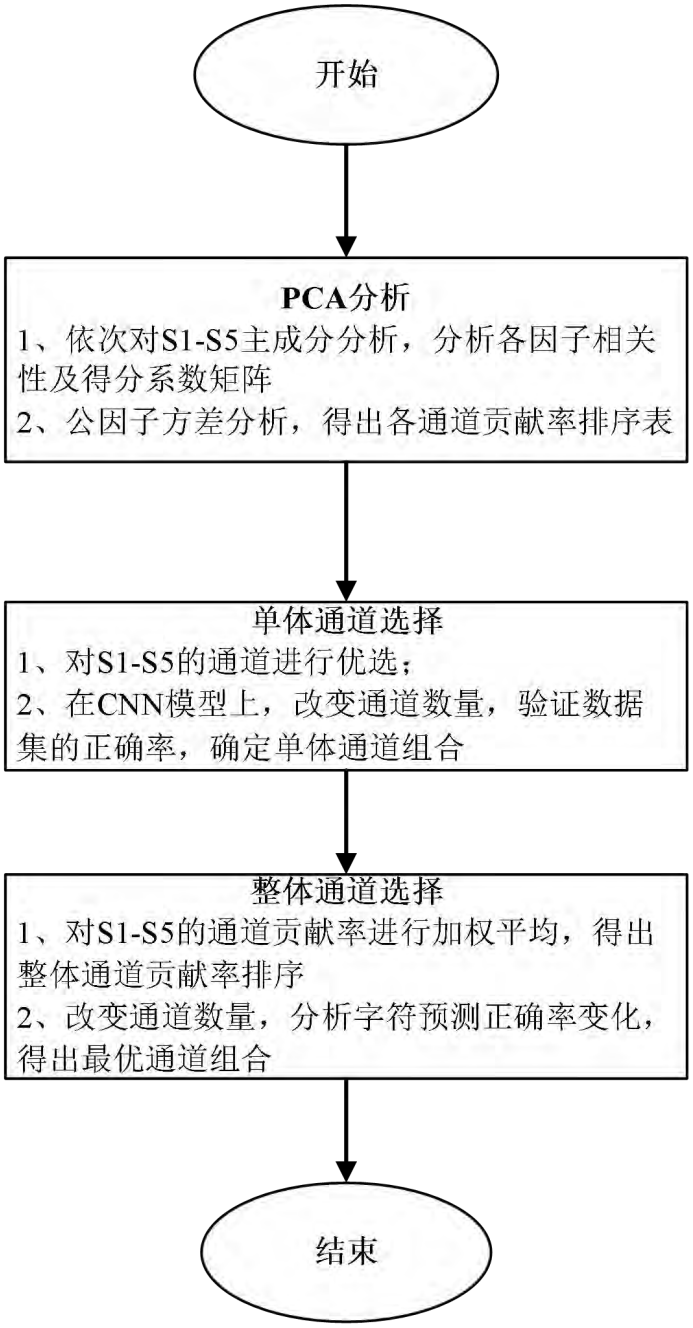

图 5.18　问题二研究思路流程图

5.2.1　主成分分析

主成分分析法其目标是将数据中众多具有一定相关性的变量进行重新组合， 组合为新的相互没有关联的综合变量。在数学上其处理方法是通过原来的变量做线性组合，从而形成新的综合变量，其一般流程图如图 5.19 所示：

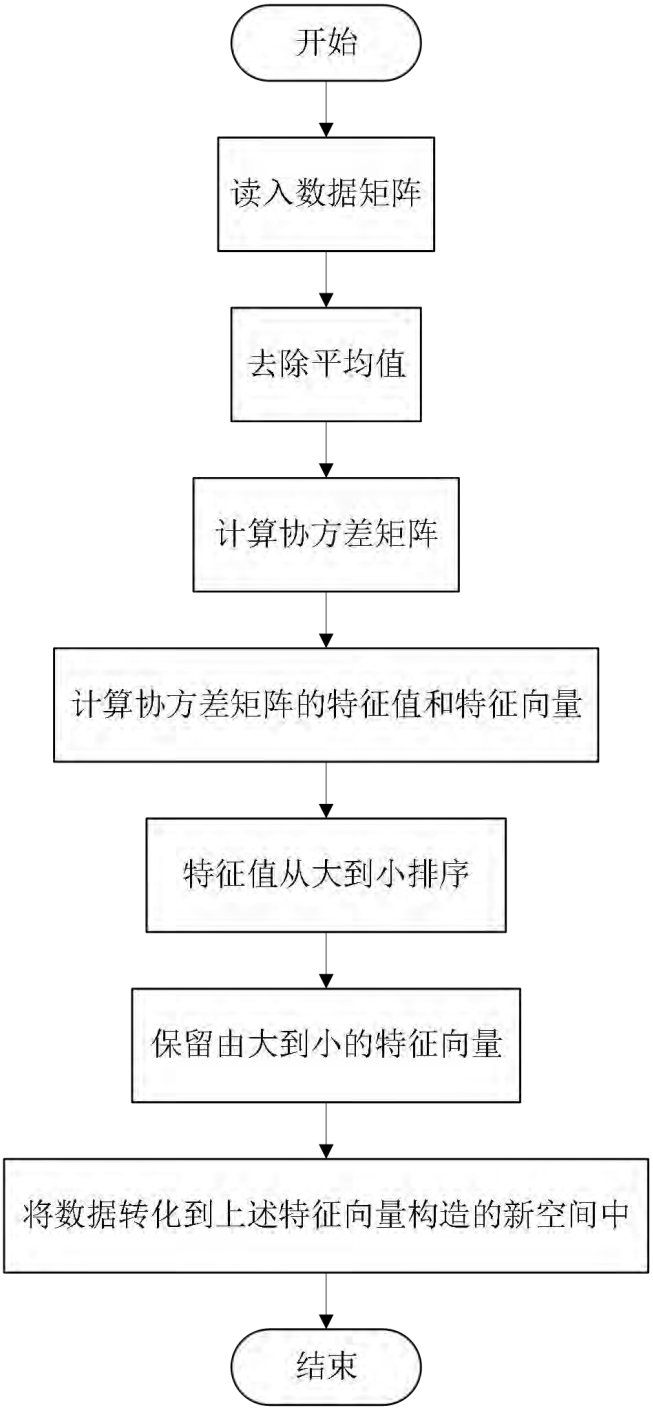

图 5.19　主成分分析原理图

$$Z_{mxn}=\begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{21} & z_{22} & \cdots & z_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ z_{m1} & z_{m2} & \cdots & z_{mm} \end{bmatrix} \tag{5-33}$$

该矩阵中 $z_{ij}\left(i=1,2\ldots,m,j=1,2,\ldots n\right)$ 作为第 i 个采样点的第 j 个参数。在本研究中 n=20，在这个基础上进行标准化计算求出矩阵 X，如下所示：

$$X=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mm} \end{bmatrix} \tag{5-34}$$

其中：

$$x_{ij}=\frac{z_{ij}-\bar{z}_j}{s_j} \tag{5-35}$$

$$\bar{z}_j=\frac{1}{m}\sum_{i=1}^{m}z_{ij} \tag{5-36}$$

$$s_j^2=\frac{1}{m-1}j\sum_{1}^{m}\left(z_{ij}-\bar{z}_j\right)^2 \tag{5-37}$$

其中我们针对矩阵 X 求其协方差矩阵，明确定义如下：

$$\sum=\begin{bmatrix} s_1^2 & \operatorname{cov}(1,2) & \cdots & \operatorname{cov}(1,n) \\ \operatorname{cov}(2,1) & s_2^2 & \cdots & \operatorname{cov}(2,n) \\ \cdots & \cdots & \cdots & \cdots \\ \operatorname{cov}(n,1) & \operatorname{cov}(n,2) & \cdots & s_n^2 \end{bmatrix} \tag{5-38}$$

其中：

$$s_x^2=\operatorname{cov}(x,x) \tag{5-39}$$

$$\operatorname{cov}(x,y)=\operatorname{cov}(y,x)=\frac{1}{m-1}\sum_{i=1}^{n}\left(x_i-\bar{x}\right)\left(y_i-\bar{y}\right) \tag{5-40}$$

根据矩阵 Z 可以求得矩阵 R，其定义如下：

$$R=\frac{1}{m-1}\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ r_{m1} & r_{m2} & \cdots & r_{mm} \end{bmatrix} \tag{5-41}$$

其中：

$$r_{xy}=\frac{\operatorname{cov}(x,y)}{s_x s_y} \tag{5-42}$$

其中用 λ_i 代表矩阵 R 的特征值，对这些特征值的大小排序为：$\lambda_1\geq\lambda_2\geq\ldots\lambda_n\geq 0$，算出其相应的正交化特征向量如下所示：

$$[e_1, e_2, e_3 \cdots e_n] = \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1n} \\ e_{21} & e_{22} & \cdots & e_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ e_{n1} & e_{n2} & \cdots & e_{nn} \end{bmatrix} \tag{5-43}$$

利用主成分分析法对不同的通道进行主成分分析，针对某特征值贡献率进行排序，利用 $\lambda_i \Big/ \sum_{j=1}^{n} \lambda_i$ 计算出属性的贡献率，求出主成分比较高的特征值代表大部分原始特征参数的信息。

5.2.2 主成分结果分析

我们对各属性之间的相互关系进行处理，进行协方差计算，求得属性之间的进行协方差计算，求得属性之间的相互关系，对于相互关系越大的属性颜色表示越浅， 对于相互关系越小的属性，相互关系越深，如下图 5.20 所示。

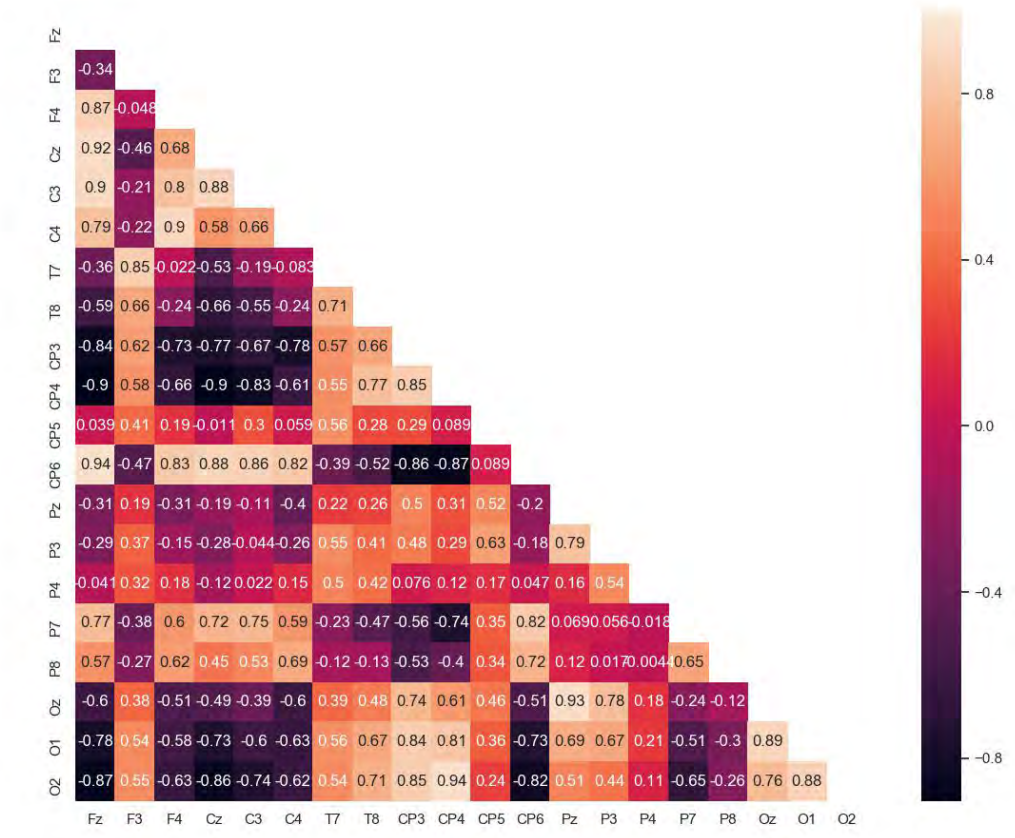

图 5.20 S1 个通道之间相互关系效果图

通过主成分分析，选择被试者的通道名称组合。对特征参数标准化进行 KMO 检验和 Barlett 球形检验，然后进行主成分分析。以被试者 S1 为例：总方差解释表给出了 S1 的初始特征值、主成分的方差百分比和累积贡献率。如图所示。主成分得分的方差，在数值上等于相关系数矩阵的各个特征根λ，全部特征根的总和则等于变量数目。提取载荷平方和显示我们提取λ>1 的主成分，其对应主成分方差的累计贡献度>85%,我们也可以通过碎石图观察特征根数值的衰减点变化，主成分数目为 3，再结合前述原则，选取主成分数目为 3。

表5.11　总方差解释

成分	初始特征值			提取载荷平方和		
	总计	方差百分比	累积 %	总计	方差百分比	累积 %
1	10.821	54.103	54.103	10.821	54.103	54.103
2	3.842	19.212	73.315	3.842	19.212	73.315
3	2.034	10.170	83.485	2.034	10.170	83.485
4	.974	4.871	88.355			
5	.914	4.572	92.928			
6	.415	2.076	95.003			
7	.265	1.327	96.330			
8	.186	.931	97.261			
9	.163	.816	98.078			
10	.094	.472	98.549			
11	.070	.352	98.902			
12	.059	.297	99.198			
13	.039	.197	99.395			
14	.035	.173	99.568			
15	.028	.138	99.706			
16	.022	.109	99.815			
17	.020	.100	99.916			
18	.009	.044	99.959			
19	.006	.029	99.989			
20	.002	.011	100.000			

提取方法：主成分分析法。

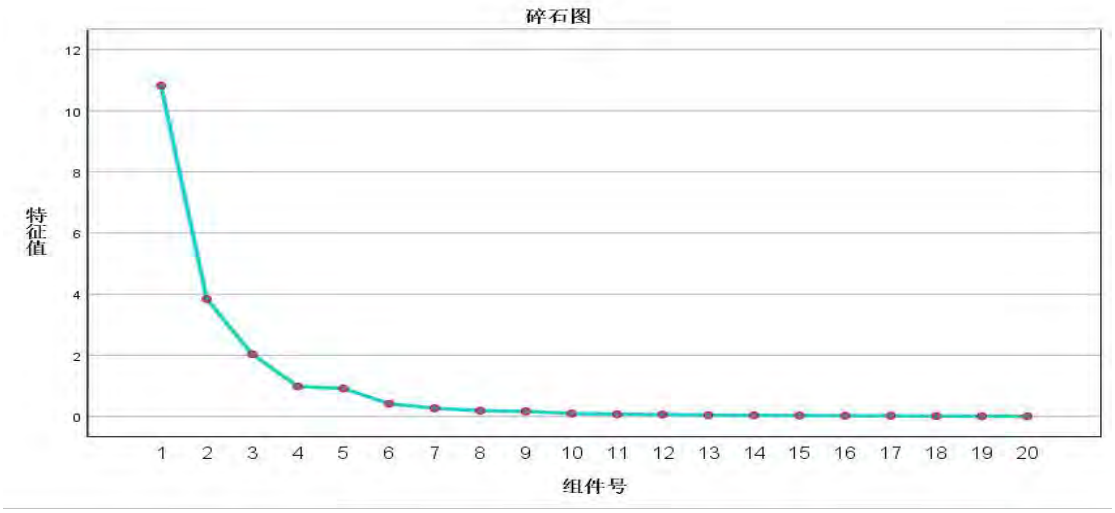

图 5.21　S1 特征根数值衰减图

因子载荷图反映某个指标在该因子上的载荷多大。在因子载荷图中，可以看到 S1 的 20 个通道在 3D 载荷图的分布情况。

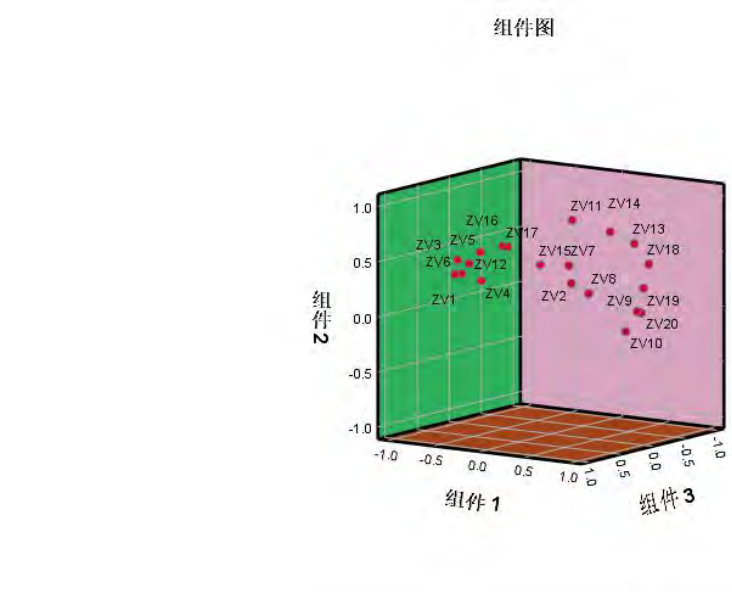

图 5.22　S1 主成分分析因子载荷图

在成分矩阵图中，给出主成分载荷矩阵。矩阵中的每一列载荷值都显示了各个变量与有关主成分的相关系数。如下 0.942 表示主成分 1 与变量 1 的相关系数。同时，表中也给出了提取了 3 个成分的提示。

表5.12　S1成分矩阵[a]

	成分		
	1	2	3
Zscore(V1)	-.942	.255	.046
Zscore(V9)	.940	.072	-.066
Zscore(V10)	.937	-.061	.174
Zscore(V20)	.934	.094	-.005
Zscore(V12)	-.922	.336	-.040
Zscore(V19)	.902	.280	-.154
Zscore(V4)	-.894	.160	-.190
Zscore(V5)	-.808	.450	-.050
Zscore(V6)	-.768	.317	.399
Zscore(V3)	-.758	.447	.377
Zscore(V18)	.749	.438	-.457
Zscore(V16)	-.716	.481	-.270
Zscore(V8)	.716	.297	.434
Zscore(V2)	.580	.387	.512
Zscore(V7)	.563	.549	.524
Zscore(V17)	-.528	.521	-.081

Zscore(V11)	.163	.831	-.099
Zscore(V14)	.475	.734	-.247
Zscore(V15)	.162	.497	.398
Zscore(V13)	.472	.574	-.624

提取方法：主成分分析法。

a. 提取了 3 个成分。

随后计算主成分载荷矩阵转置矩阵的逆或逆矩阵的转置，也即主成分载荷除以相应的特征根得到的结果，从而得到对应成分得分系数矩阵的数值。主成分得分通过原始数据标准化结果与各自对应的成分得分系数相乘并累加即可求得。通过主成分分析，我们总共选取了5 个主成分分量，主成分方差的累计贡献率83.485。为了更好地展现主成分分析的结果，绘制第一成分和第二主成分的散点图。

表5.13　成分得分系数矩阵

	成分		
	1	2	3
Zscore(V1)	-.087	.066	.023
Zscore(V2)	.054	.101	.252
Zscore(V3)	-.070	.116	.185
Zscore(V4)	-.083	.042	-.093
Zscore(V5)	-.075	.117	-.024
Zscore(V6)	-.071	.082	.196
Zscore(V7)	.052	.143	.258
Zscore(V8)	.066	.077	.213
Zscore(V9)	.087	.019	-.032
Zscore(V10)	.087	-.016	.085
Zscore(V11)	.015	.216	-.049
Zscore(V12)	-.085	.088	-.020
Zscore(V13)	.044	.149	-.307
Zscore(V14)	.044	.191	-.122
Zscore(V15)	.015	.129	.196
Zscore(V16)	-.066	.125	-.133
Zscore(V17)	-.049	.136	-.040
Zscore(V18)	.069	.114	-.225
Zscore(V19)	.083	.073	-.076
Zscore(V20)	.086	.024	-.002

提取方法：主成分分析法。

组件得分。

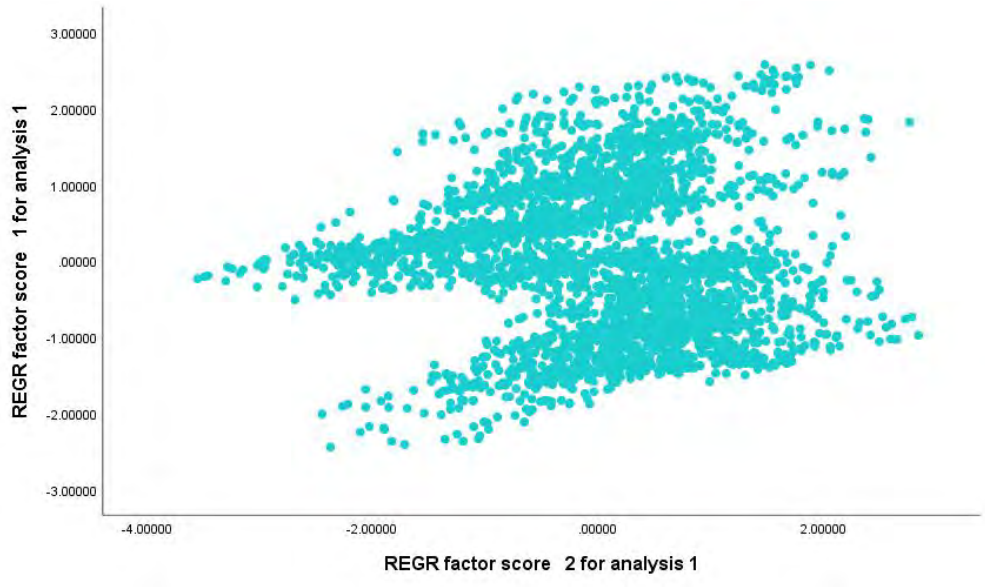

图 5.23　S1 的 PCA 降维结果可视化

同理，对 S2-S5 应用 PCA 分析，进一步确定通道的优化选择。

对 S1 的全部变量的公因子方差进行分析，公因子方差是几个公因子方差的累计贡献率，累计贡献率越高，说明提取的这几个公因子对于原始变量的代表性或者说解释率越高，整体的效果就越好。累计贡献率越低，说明提取的公因子的代表性或者说解释率越差，效果就越差。对 S1，选取贡献率大于 80%的通道，并对通道数在 10 到 20 范围内测试预测准确率，最终选取 16 个通道：[1, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 16, 18, 19, 20]。

表5.14　公因子方差

	初始	提取
V1	1.000	.954
V2	1.000	.748
V3	1.000	.917
V4	1.000	.862
V5	1.000	.859
V6	1.000	.849
V7	1.000	.893
V8	1.000	.789
V9	1.000	.893
V10	1.000	.913
V11	1.000	.726
V12	1.000	.965
V13	1.000	.942
V14	1.000	.825
V15	1.000	.432
V16	1.000	.817
V17	1.000	.556

V18	1.000	.960
V19	1.000	.916
V20	1.000	.881

提取方法：主成分分析法。

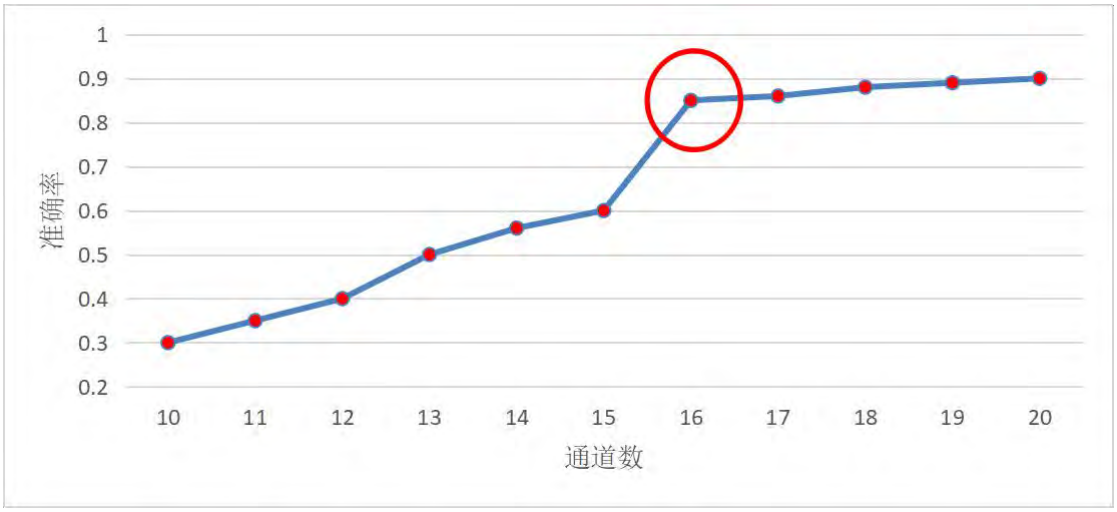

图 5.24　S1 通道选择字符预测准确率图

同理对 S2-S5 进行主成分分析，得出每个被试的通道贡献率表，对贡献率排序处理后如下表：

表 5.15　S2、S3、S4 和 S5 通道贡献率表

S2		S3		S4		S5	
通道	贡献率	通道	贡献率	通道	贡献率	通道	贡献率
V20	0.984	V4	0.969	V10	0.999	V17	0.997
V19	0.982	V15	0.965	V12	0.999	V15	0.997
V11	0.982	V9	0.961	V17	0.999	V18	0.996
V18	0.981	V10	0.953	V15	0.999	V20	0.996
V15	0.980	V20	0.940	V18	0.999	V12	0.994
V13	0.979	V11	0.926	V20	0.999	V10	0.987
V12	0.978	V1	0.912	V8	0.998	V2	0.975
V17	0.976	V18	0.909	V11	0.998	V19	0.970
V10	0.974	V2	0.893	V16	0.998	V1	0.969
V5	0.954	V3	0.884	V9	0.998	V11	0.964
V8	0.941	V14	0.883	V14	0.998	V8	0.955
V4	0.931	V8	0.881	V19	0.998	V7	0.947
V6	0.926	V16	0.877	V5	0.998	V16	0.939
V9	0.923	V6	0.838	V3	0.993	V14	0.929
V7	0.922	V5	0.830	V7	0.975	V6	0.910
V16	0.914	V17	0.800	V6	0.975	V3	0.883

V1	0.904	V13	0.784	V13	0.948	V4	0.862
V14	0.891	V19	0.734	V4	0.888	V13	0.858
V2	0.814	V7	0.670	V1	0.887	V5	0.853
V3	0.787	V12	0.507	V2	0.847	V9	0.842

同理，绘制 S2、S3、S4、S5 的通道选择字符预测准确率图如下所示：

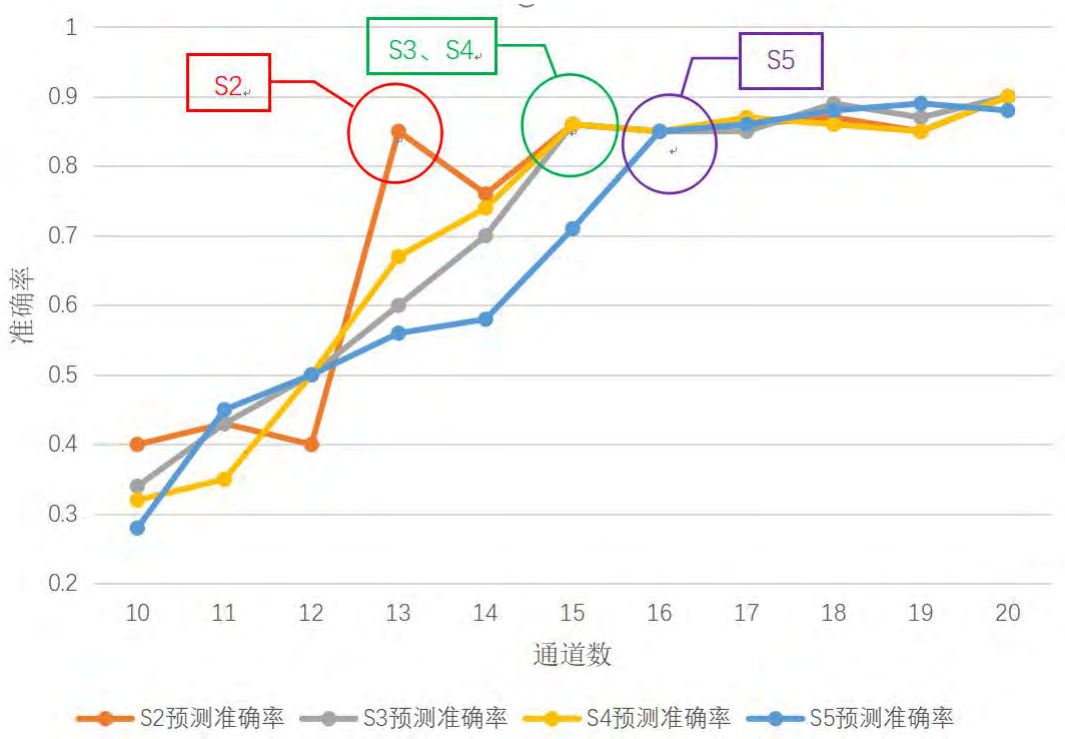

图 5.25　S2、S3、S4、S5 通道数目对预测准确率影响

所以可以得出：得出 S2 最优通道数为 13，通道为：[4, 5, 6, 8, 10, 11, 12, 13, 15, 17, 18, 19, 20]；S3 最优通道数为 15，通道为：[1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 14, 15, 16, 18, 20]；S4 最优通道数为 15：[3, 5, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20]；S5 最优通道数为 16，通道为 [1, 2, 3, 6, 7, 8,10, 11, 12, 14, 15, 16, 17, 18, 19, 20]。

由 PCA 分析并根据对字符测试的结果可以观察出 S1-S5 五个被试者的最优通道并不一致，且最优通道数也不相同，为了选择对于所有被试都较适用的一组最优通道名称组合，对每个被试者的通道贡献率进行加权后取平均，得出所有通道对五个被测试者的加权贡献率，例如对于每个被测试者，贡献率越靠前的，赋予权重越大，越靠后的权重越小，并对通道数进行选择测试，得出字符预测准备率变化图，最后确定通道数以及通道的选择。

表 5.16　对于所有被试者通道加权平均贡献率

通道	贡献率
V18	0.969
V10	0.965
V20	0.960
V1	0.925

V9	0.924
V19	0.920
V11	0.919
V8	0.913
V16	0.909
V14	0.905
V4	0.902
V13	0.902
V6	0.900
V5	0.899
V3	0.893
V12	0.889
V7	0.881
V15	0.874
V17	0.865
V2	0.855

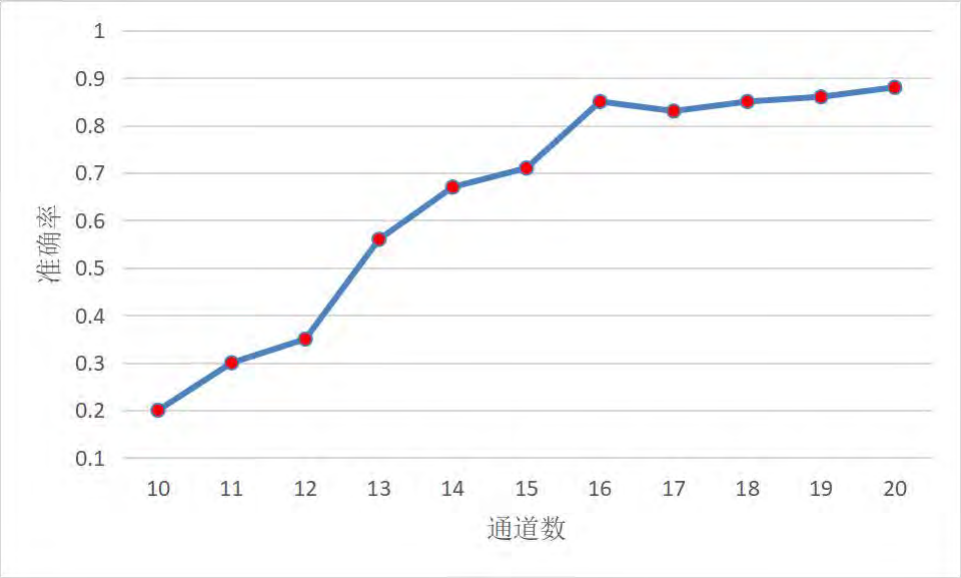

图 5.26　最优通道组合

综合 S1-S5，在提高模型泛化能力和准确能力的前提下，对五个被试者的最优通道选择 16 个通道：[1,3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 18, 19, 20]。

5.3　问题三

5.3.1 半监督 SVM

在问题 2 中，最终选取 20 个通道中的 16 个通道(Fz, F4, Cz, C3, C4, T8, CP3, CP4, CP5, CP6, Pz, P3, P7, Oz, O1, O2)作为最优通道组合.对于附件 1 中所给数据，选取部分为有标签数据，部分为无标签数据。

数据的搜集中，获得标记数据的成本是高昂的，而获得未标记的数据则是低廉的，为此人们提出了半监督的学习方法，旨在利用少量的标记数据和大量的未标记数据进行学

习，从而节约成本。这是可能做到的，比如有一些未被标记的数据和一个被标记的正样本数据非常相似，那么我们就可以将这些未被标记的样本视为正样本来学习。

本文提出使用基于半监督分类的特征提取，在传统的判别分析方法上进行改进，通过有标签数据与无标签数据之间的欧式距离来获得二者之间的关系，进而来优化目标函数，求解投影向量，提取特征，然后通过投票分类的方法来进行最终的识别。在小样本情况下，需要对样本进行特征提取，常用的方法是增加一个惩罚项对目标函数进行约束，获得新的目标函数：

$$a_{opt}=\arg\max_{a}\frac{a^{T}S_{b}a}{a^{T}S_{t}a+\alpha J(a)} \tag{5-44}$$

α 是用来约束惩罚项的系数，用来降低模型复杂度。$J(a)$ 是用来约束目标函数的惩罚项，惩罚项为 $J(a)=\|a\|^{2}$。在处理该问题时，依靠有标签数据与无标签数据之间的关系所建立的惩罚项。假设相同类别之间的数据，距离更近。假定一组数据 $\{\mathrm{X_i}\}_{i=1}^{m}$，利用 K-近邻图的方式来判断不同点之间的关系。如果 $\mathrm{X_i}$ 与 X_j 互为近邻，就将其视为一类，关系矩阵为 S_{ij}。

综合考虑本文选取半监督 SVM 算法，由于 SVM 算法的原理已经在问题一的方法三陈述过，于是次数不再重复介绍 SVM 算法的原理。半监督 SVM 算法描述如下：

表 5.17　半监督 SVM 算法描述

半监督 SVM 算法
Input：训练集矩阵 $X\in R^{l\times N}$，标签矩阵 $Y\in R^{N}$，测试集矩阵 $U\in R^{l\times M}$，惩罚系数 α； Output：特征分类结果。 First Step：构造关系矩阵 $s_{ij}=\begin{cases}e^{-\|x_i-x_j\|^{2}},if:x_i\in N_k(x_j),\ or:x_j\in N_k(x_i)\\0,others\end{cases}$， 式中 $N_k(x_j)$ 为 x_j 的 k 最近邻，并计算 $L=D-S$； Second Step：计算类间散度矩阵 S_b 与整体散度矩阵 S_l； Third Step：计算投影向量 $a_{opt}=\arg\max_{a}\frac{a^{T}S_{b}a}{a^{T}(S_{l}+aXLX^{T})a}$；（5-45） Forth Step：将原始数据利用投影向量进行特征提取； Fifth Step：利用投票分类对提取的特征进行分类。

整个过程处理流程图如下：

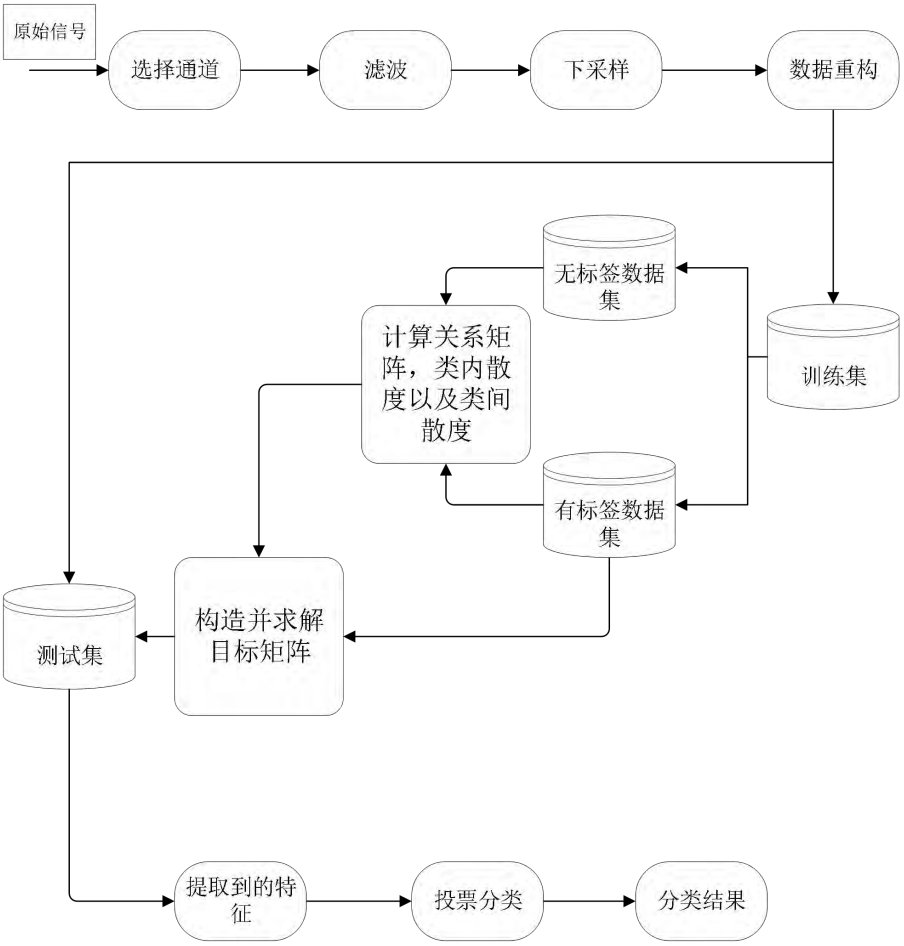

图 5.27　基于半监督特征分析提取方法的分类流程图

5.3.2　结果分析

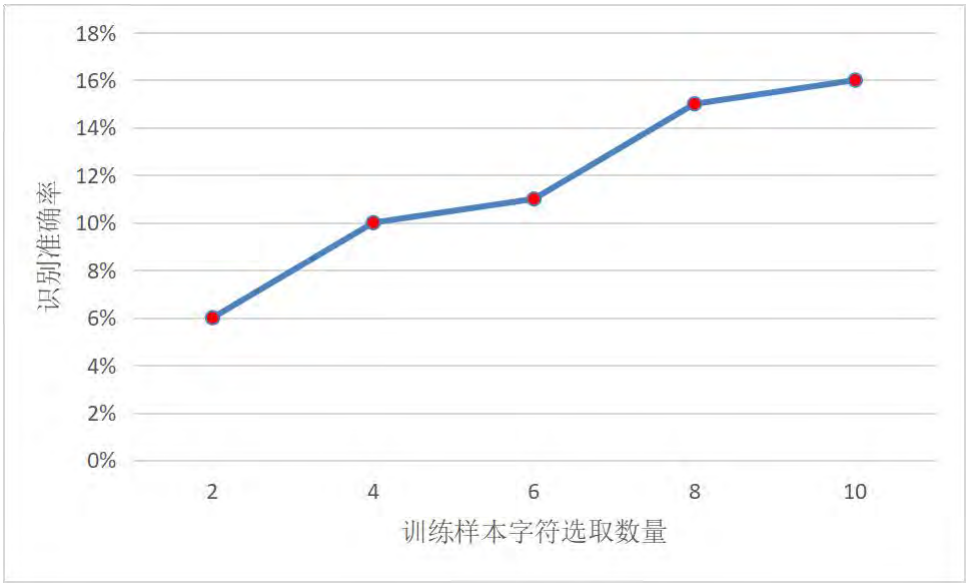

图 5.28　模型识别结果

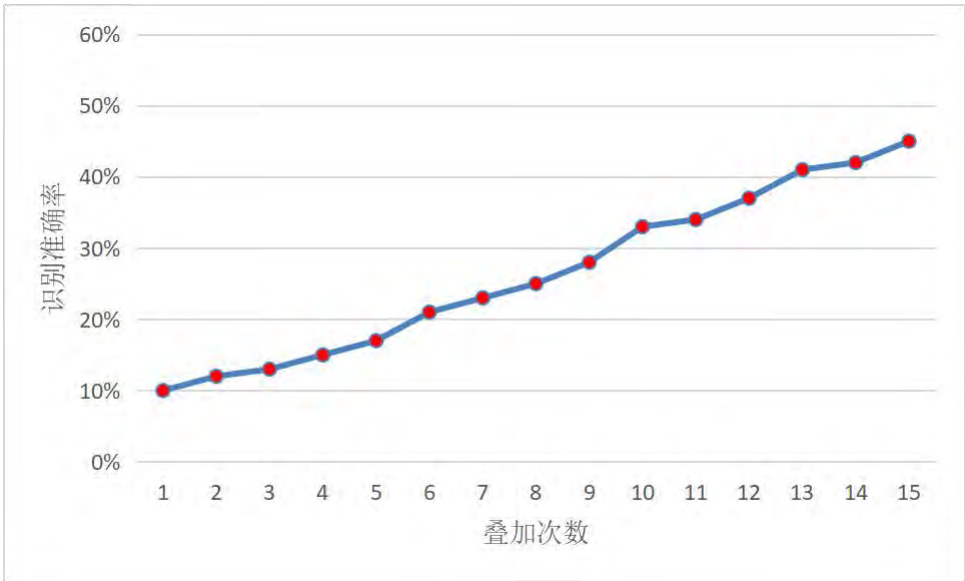

图 5.29　训练个数为 4，识别准确率随叠加次数的变化

从图中可以看出，4 个字符（4*25 个训练样本）做训练时，叠加次数为 1 时，识别准确率为 9.8%，随着叠加次数的增加，识别准确率在逐渐提升。同时在随着训练字符的增加，模型的预测准确率也逐渐增加。

表 5.18　半监督 SVM 算法在测试数据上的预测结果

	char13	char14	char15	char16	char17	char18	char19	char20	char21	char22
r1	(1,7)	(1,12)	(6,7)	(5,10)	(1,9)	(4,8)	(1,11)	(4,12)	(1,7)	(4,12)
r2	(2,7)	(4,12)	(3,7)	(3,10)	(2,10)	(4,8)	(3,11)	(4,12)	(3,7)	—
r3	(3,7)	(1,12)	(6,7)	(5,10)	(2,9)	(4,8)	(2,11)	(4,12)	(3,7)	—
r4	(3,7)	(2,12)	(6,8)	(5,10)	(2,8)	(4,8)	(4,11)	(2,12)	(3,7)	(6,12)
r5	(5,7)	(1,12)	(6,7)	(5,10)	(2,9)	(1,10)	(2,11)	(2,12)	(4,7)	(6,12)

由上表可知，在采用 16 个通道(Fz, F4, Cz, C3, C4, T8, CP3, CP4, CP5, CP6, Pz, P3, P7, Oz, O1, O2)作为最优组合通道的条件下，半监督 SVM 算法在含有 10 个待预测字符的测试数据下预测结果分别为：“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“M”和“0”，与问题一中 CNN 模型的预测结果不完全一致（“char21”在问题一中 CNN 模型输出为字符“A”，而本问题中半监督 SVM 算法输出为字符“M”）。

5.4　问题四

5.4.1　实验数据描述

本研究所使用的数据均为比赛官方所给定的数据，包括 3000 组带有标签的睡眠脑电特征样本，其中特征样本有 4 个维度，其物理属性是脑电信号在“8-13Hz”，“14-25Hz”，“4-7Hz”和“0.5-4Hz”频率范围内的能量占比，分别用“Alpha”、“Beta”、“Theta”和“Delta”表示。数据标签为不同的睡眠分期，包括清醒期、快速眼动期、睡眠 I 期、睡眠 II 期和深睡眠期，分别用数字 6、5、4、3、2 来表示。

5.4.2　模型提出

拟采用一种基于改进 K-Means 聚类的睡眠分期预测模型，K-Means 是聚类算法中的最常用的一种，该算法最大的特点是简单，好理解，运算速度快，但是只能应用于连续型的数据，并且一定要在聚类前手工指定要分成几类。

5.4.2.1 K-Means 聚类分析

K-Means 聚类算法主要就是用基于距离的聚类算法，根据数据中的各个特征相似度进行样本分组，把数据中的集合根据各个特征类的相似性，划分成若干聚类或分组，其本质是对数据进行分类。

基本思想：给定 n 个数据点$\{x_1, x_2, \cdots, x_n\}$，设置 k 个聚类中心$\{a_1, a_2, \cdots, a_k\}$是每个数据点$x_i(1 \le i \le n)$与中心点之间的距离最小。距离有多种计算方式，如曼哈顿距离、欧几里得距离、闵科夫斯基距离等.。比较普遍使用的是欧几里得距离，其距离公式如下所示：

$$S = \sum_{i=1}^{n} \min_{1 \le j \le k} \left| x_i - a_j \right|^2 \qquad (5\text{-}46)$$

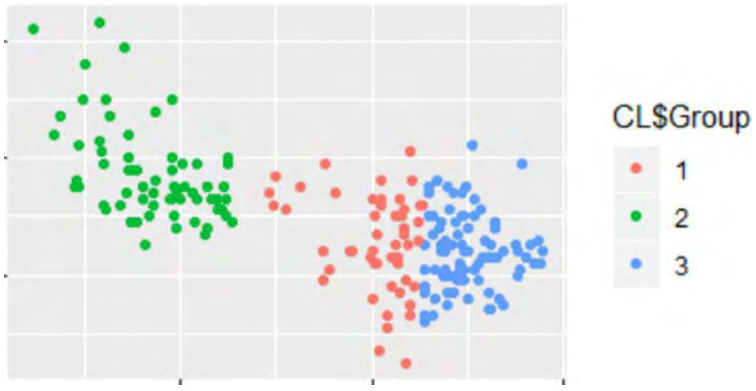

图 5.30　二维聚类样例

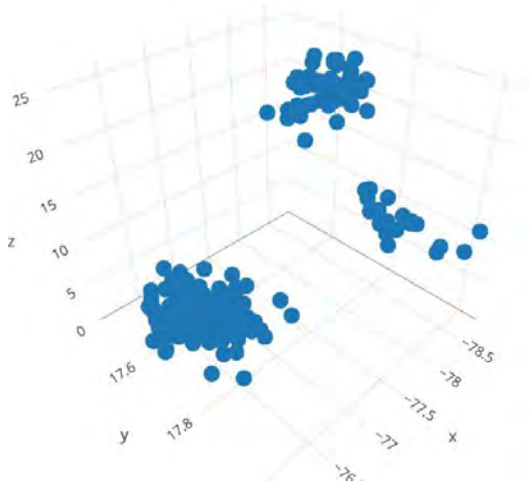

图 5.31　三维聚类样例

5.4.2.2 改进 K-Means 算法

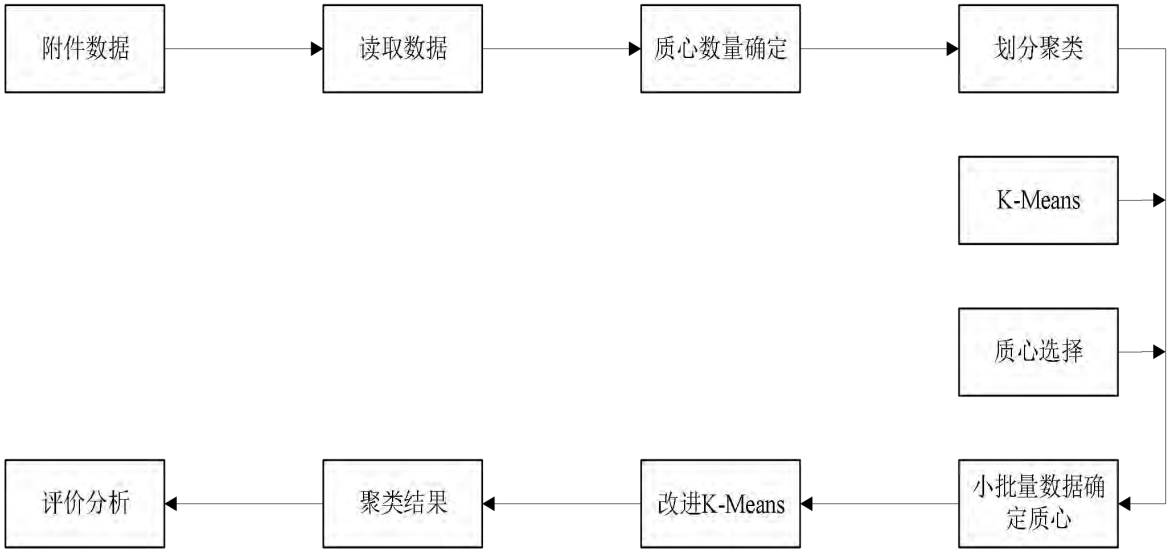

图 5.32　聚类分析流程图

在对原来 K-Means 算法的基础上，采用小批量数据子集减小计算时间，并同时尝试优化目标函数，其寻找质心的方法依然与 K-Means 一致，但是采用小批量数据进行迭代，从而可以更快的得到收敛结果，该改进算法可以对附件中的数据进行更高效地处理。

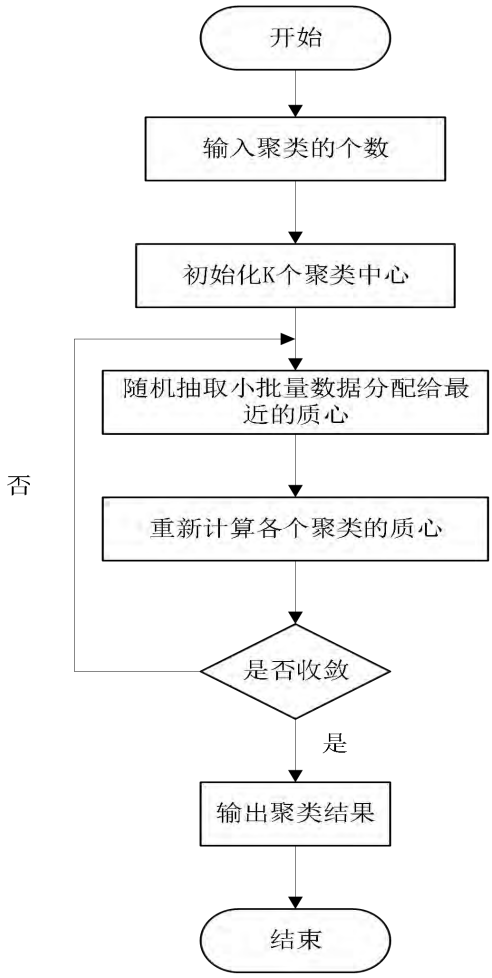

图 5.33　改进 K-Means 算法流程图

表 5.19　K-Means 算法描述

K-means 算法
Input：训练数据集 X； **Output**：按某些特征属性划分的 k 个子集。 First Step：确定需要分类的类别数 k，即数据集经过聚类得到 k 个集合； Second Step：从数据集 X 中随机选择 k 个数据点作为质心； Third Step：对数据集中每一个点，计算其与每一个质心的距离（如欧式距离），并将该点划分到距某个最近质心所属的集合； Forth Step：重复上步至所有点归入集合，随后重新计算每个集合的质心； Fifth Step：若新计算出的质心和原来的质心之间的距离小于某个设定的阈值，则可以认为聚类已经达到期望的结果，算法终止；若新质心与原质心的距离变化很大，则需重复迭代第 3~5 步骤（Third Step ~ Fifth Step）。

5.4.3 数据预处理

在预测问题中，任何数据送入模型训练之前需要考虑是否应该对数据进行预处理，本研究中采用归一化的方式对数据进行预处理。对数据进行归一化的目的就是使得预处理的数据被限定在一定的范围内，从而消除奇异样本对聚类产生的不良影响。我们分别对训练数据进行了最大最小归一化与均值归一化两种归一化方式，并在睡眠预测模型上进行了测试，测试结果如下：

表 5.20　数据归一化后对模型预测的影响

	原始数据	对数据进行最大最小归一化	对数据进行均值归一化
模型准确率	0.713	0.635	0.523
运行时间	0.097	0.087	0.082

以上训练过程中除了对数据进行了操作之外，训练参数保持不变。根据上表可知，对训练数据分别进行两类归一化操作后，模型的预测时间有所降低，但模型的预测准确率都有下降，出现这种情况的可能原因是真实的脑电数据存在噪声，导致模型不适合使用归一化，故接下来的模型训练都不采用归一化处理。

5.4.4　模型聚类分析

5.4.4.1 模型在二维特征下的聚类结果

为了直观感受 K-Means 在较少为特征下的聚类结果，我们对 4 个特征进行两两组合，并将剩余的两个特征人为的去除，随后将只有两维特征的训练数据送入睡眠预测模型训练，产生的结果如下图所示：

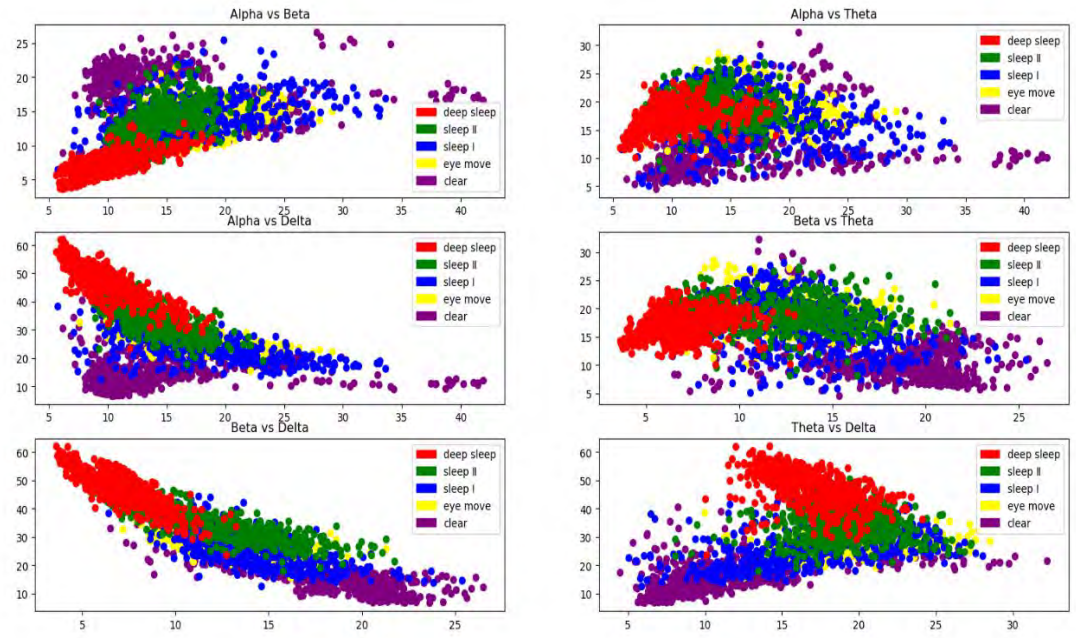

图 5.34　去除两维特征的模型聚类结果

由上图可知，在任意两维特征下，处于“清醒期”、“睡眠 I 期”、“睡眠 II 期”和“深睡眠期”的数据聚类效果很好，即各类数据点有明显的间隔（如图中红色、绿色、蓝色、紫色点所示），但处于“快速眼动期”的数据点明显嵌于四类数据点之间，没有明显的间隔（如图中黄色点所示），综上所述，处于“快速眼动期”的数据点很难与其余四类分开，分析原因可能由以下几点：一是人的脑电信号受不同程度的干扰会产生异常数据，这类数据对于模型预测来说都是噪声数据，而这类噪声数据即使通过某种过滤算法也很难处理干净；二是由于每个被试者个体具有差异性，采集到的数据也很大可能具有一定的个体差异，这对于模型训练是较复杂的情况，模型可能在训练数据上取得了较高的准确率而在不同的测试数据上准确率较低，模型泛化效果较差。

5.4.4.2　模型在三维特征下的聚类结果

为了进一步探究三维特征组合的方式下各类标签的聚类效果，我们选出任意四维特征中的任意三个，共四种组合方式作为训练样本的特征，模型的聚类结果如下图所示：

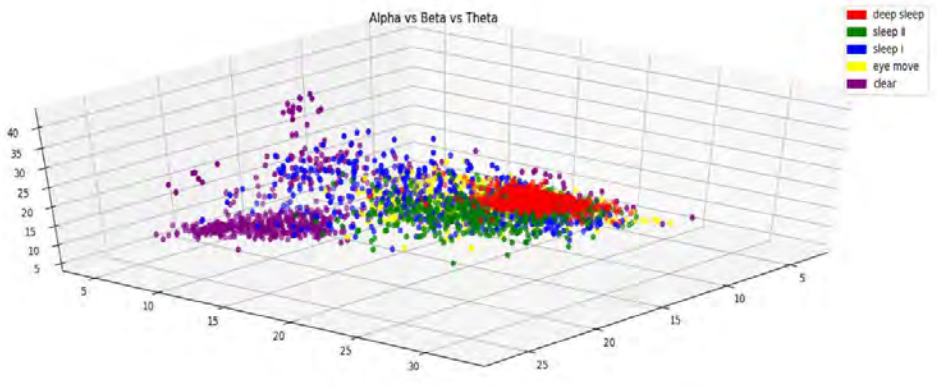

图 5.35　“Alpha vs Beta vs Theta”特征组合

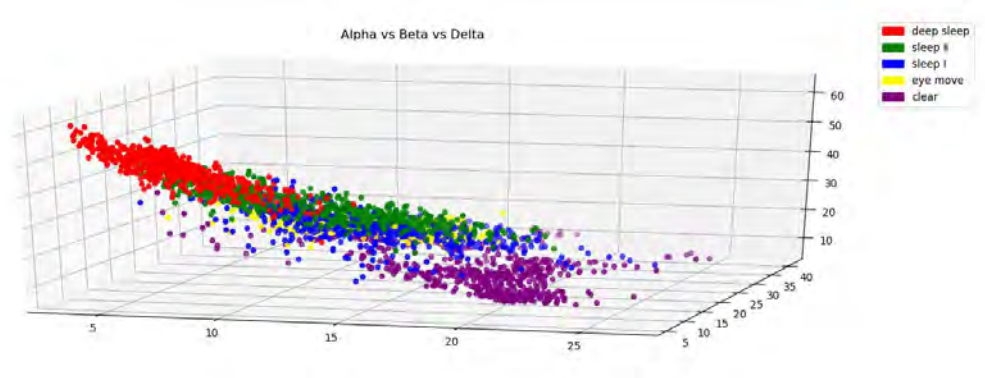

图 5.36 “Alpha vs Beta vs Delta”特征组合

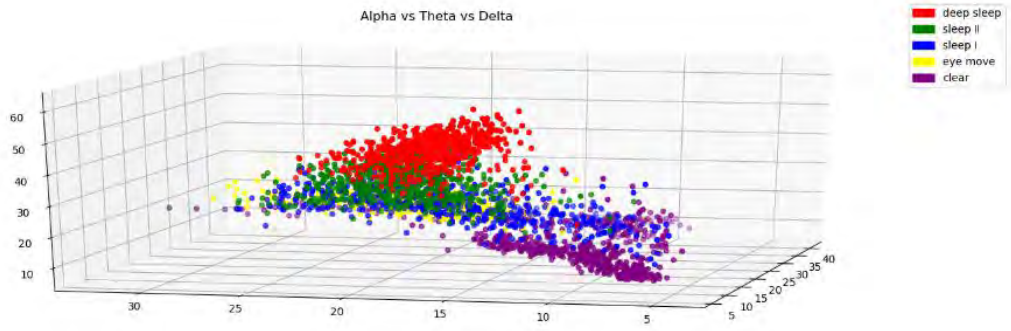

图 5.37 “Alpha vs Theta vs Theta”特征组合

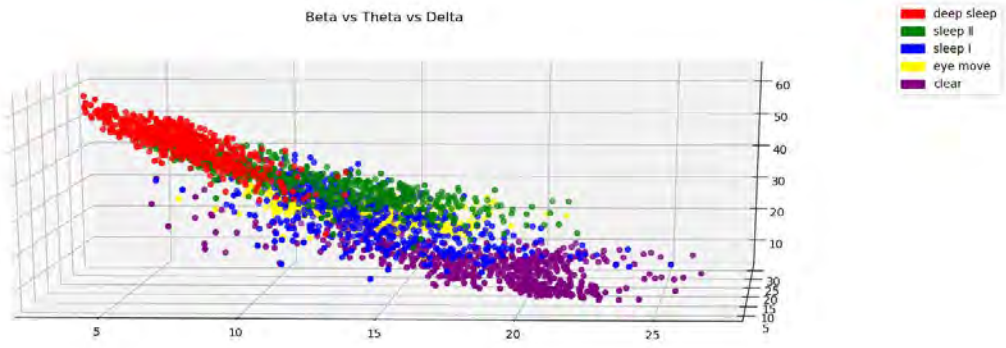

图 5.38 “Beta vs Theta vs Delta”特征组合

由以上四图同样可以得到上一小节得出的结论，即除“快速眼动期”之外的数据点聚类较为简单，其中分隔最明显的是“深睡眠期”与“清醒期”（见上图中红色与紫色数据点），这种聚类结果与我们在日常生活中的认知相似，即人处于清醒状态与深睡状态的脑电信号会有很大的差异，而“睡眠Ⅰ期”和“睡眠Ⅱ期”的数据点也具有某种交叉属性，模型聚类效果一般，同时处于“快速眼动期”的数据点仍与其余类别数据点融合，不容易区分。

5.4.4.3 模型在全部特征下的聚类结果

基于以上分析，采用全部特征的训练数据进行训练，由于四维特征聚类结果无法显示，故给出模型的聚类中心坐标矩阵：

表 5.21　全部特征下模型输出的聚类中心

类别	Alpha	Beta	Theta	Delta
深睡眠期	12.024	19.720	9.692	13.327
睡眠Ⅱ期	12.005	9.553	18.654	38.343
睡眠Ⅰ期	8.828	6.440	16.458	49.728
快速眼动期	23.891	16.091	13.860	19.122
清醒期	16.118	13.539	18.974	27.355

5.4.5 模型迭代次数与准确率的关系

通过设定不同的迭代次数对模型进行训练，以寻找当前设定参数下的最优迭代次数值。不同的迭代次数与测试准确率关系如下表：

表 5.22　模型迭代次数与准确率关系

迭代次数	15	20	25	30	35	40
模型 准确率	0.442	0.529	0.693	0.729	0.719	0.706

5.4.6　模型训练所用测试集大小与准确率的关系

上一小节可以得到模型迭代 30 次达到最优，则本节中我们选择 30 次的迭代次数，探究划分不同大小的训练集和测试集比例与模型测试准确率的关系，如下表所示：

表 5.23　不同测试集大小与模型准确率关系

测试集划分 (test 占比)	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
模型 准确率	0.708	0.762	0.732	0.724	0.720	0.697	0.685	0.669

根据上表可知，test 数据占比为 30%（训练集占比 70%）时模型的预测准确率最高，为 76.2%；同时，当测试集占比上升到 50%至 60%时（训练集占比为 40%至 50%），模型的预测准确率也可以达到相对于最高准确率的较好水平（72.4%）。故可以得出结论，当模型只采用 40%-50%的训练数据时，也可以输出较好的分类结果。

5.4.7　模型评估

基于上一小节中的结论，采用 50%的训练数据的情况下，我们输出了模型预测的混淆矩阵，包括 5 个待分类项，如下表所示：

表 5.24　50%训练数据下的模型预测混淆矩阵

类别	深睡眠期	睡眠Ⅱ期	睡眠Ⅰ期	快速眼动期	清醒期
深睡眠期	253	28	4	3	0
睡眠Ⅱ期	55	181	30	42	2
睡眠Ⅰ期	1	40	116	79	31
快速眼动期	2	19	37	244	2

清醒期	1	0	32	4	294

基于上表，可以计算当前模型对于各个类别的准确率与召回率，如下表：

表 5.25　模型准确率与召回率

	深睡眠期	睡眠Ⅱ期	睡眠Ⅰ期	快速眼动期	清醒期	平均
准确率	0.81	0.67	0.53	0.66	0.89	0.712
召回率	0.88	0.58	0.43	0.80	0.88	0.714

根据上表可知，模型在类别为“深睡眠期”和“清醒期”上的准确率和召回率最高，意味着模型在这两个类别上的分类比较准确；但在类别为“睡眠Ⅰ期”的准确率与召回率都很低，意味着模型在这个类别上的分类效果较差。

6 结论

对于问题一：通过不同轮次对比试验，在题目要求尽可能少的轮次的前提下，我们最终选用的轮次是 3。为了说明设计方法的合理性，关于问题一，我们设计了三种模型去解决字符识别问题，采用的方法分别是 CNN、贝叶斯和 SVM。当使用 CNN 方法时，预测的 10 个字符分别是：“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”、“0”；当使用贝叶斯方法时，预测的 10 个字符是“A”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”、“0”；当使用 SVM 方法时，预测的 10 个字符是“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”、“0”。通过不同模型的性能进行对比，我们选用的最终模型是 CNN 模型，故问题一的 10 个字符最终结果是“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“A”、“0”。

对于问题二：降维后，对于 S1，其最优通道数为 16，最优通道组合为：[1，3，4，5，6，7，8，9，10，12，13，14，16，18，19，20]。对于 S2，其最优通道数为 13，通道为：[4，5，6，8，10，11，12，13，15，17，18,19，20]；对于 S3，其最优通道数为 15，通道为：[1，2，3，4，5，6，8，9，10，11，14，15，16，18，20]；对于 S4，其最优通道数为 15：通道为：[3，5，7，8，9，10，11，12，14，15，16，17，18，19，20]；对于 S5，其最优通道数为 16，通道为[1，2，3，6，7，8,10，11，12，14，15，16，17，18，19，20]。最后通过对各个被试者的通道加权平均，确定一个适用于 5 个被试者平均加权的通道排名。对五个被试者的最优通道选择 16 个通道：[1，3，4，5，6，8，9，10，11，12，13，14，16，18，19，20]。

对于问题三：采用半监督的 SVM 学习方式，采用 16 个通道作为最优组合通道的条件下，半监督 SVM 算法在含有 10 个待预测字符的测试数据下，预测结果分别为：“M”、“F”、“5”、“2”、“I”、“T”、“K”、“X”、“M”和“0”。

对于问题四：采用 K-means 方法，我们得出：“快速眼动期”和“睡眠Ⅰ期”的数据很难与其余类别的数据区分开。随后，我们通过比较迭代次数与模型预测准确率的关系，得出模型最优迭代次数为 30 次。之后我们比较了模型在选取不同比例训练数据下的准确率，发现模型可以在 40%-50%训练数据下取得较好的效果（测试集准确率 72.4%）。最后，我们给出了模型预测的混淆矩阵，通过计算模型在各个类别下的准确率与召回率，得出的结论是：模型在“深睡眠期”与“清醒期”两个类别上分类效果最好，而在“睡眠Ⅰ期”类别下的分类效果较差。

7　模型的评价与改进

7.1　模型的优点

（1）针对问题 1，本文对 P300 脑机接口实验数据首先进行预处理，对脑电信号用带通滤波器作预处理，并采用时域降采样法提取特征，保证了数据的质量。

（2）在一轮闪烁过程中，目标字符所在行和列只出现了两次，非目标行和列出现 10 词，正样本和负样本的比例 1:5，正负样本失衡，会影响预测准确性，所以本文将正样本扩大 4 倍，使正负样本比例一致，保证模型训练的有效性。

（3）在设计问题 1 模型时，综合对比 SVM 和贝叶斯逻辑回归模型，最终确定基于 CNN 的字符预测模型。利用卷积对输入向量矩阵进行处理，提取数据特征，利用 20 个 20*1 的卷积核，5 层网络结构，提高预测的准确性。

（4）考虑到很多模型采用反向传播的方法时，用梯度下降法进行训练，激活函数 sigmoid 和 tanh 在饱和区的梯度很小，很容易发生梯度消失的现象。为了解决这个问题，本文拟采用批归一化（BN）的方法，有效的解决梯度容易消失的问题。

（5）针对问题 2，本文选择 PCA 主成分分析，对数据进行降维，综合考虑不同的被试者，得出贡献率排序表，再进行加权平均，得出各个通道贡献率，在问题 1 基于 CNN 字符预测模型上，改变通道数目，确定数目适合的最优通道，使得选取的通道和通道数量泛化能力更好。

（6）针对问题 3，本文设计一个基于半监督 SVM 的字符预测模型，基于半监督分类的特征提取，在传统的判别分析方法上进行改进，通过有标签数据与无标签数据之间的欧式距离来获得二者之间的关系，进而来优化目标函数，求解投影向量，提取特征，然后通过投票分类的方法来进行最终的识别。

（7）针对问题 4，本文设计一个基于 K-Means 的睡眠阶段预测模型，对 K-Means 算法进行了改进，进行了小批量数据的多次迭代，多次寻找最优质心点位置，并且利用轮廓系数计算出了最理想的质心数，保证了聚类模型的准确。

7.2　模型的缺点

（1）针对问题 2，由于时间问题，仅仅考虑了 PCA 主成分分析降维，未从低方差滤波 (Low Variance Filter)等别的降维方法对模型进行考量。

（2）针对问题 4，本文对聚类算法只是采用了改进的 K-Means 一种算法，因为时间原因未能再进行 DBSCAN 算法的聚类构建，如果能通过两种聚类算法的聚类相互比较，相信可以构建出可以构建更为准确的睡眠阶段预测模型。

（3）本文提出的字符分类算法是基于离线的训练数据集。若要投入到实际应用中，需要进一步完成在线实验分析，并尽量减少训练所需的时间。

7.3　模型的改进与推广

本文提出的方法和模型可推广应用于脑电波 P300 应用领域，例如教育娱乐领域，可以利用脑电信号控制游戏，训练对注意力的控制能力；人工智能领域，可以利用脑电信号控制智能轮椅等。

（1）本文中对于数据预处理的方法及操作因为其处理科学，在其他相关研究中如果

遇到数据质量问题，可以对应采纳和使用。

（2）构建的模型对以后的其他相关研究具有较强的现实参考意义，可根据具体情况对参数以及网络层进行增减和优化。

（3）模型 3 半监督分类模型对于实际中大量的无标签数据有重大的参考意义。数据本身蕴藏着大量有价值的信息，标签只是帮助我们提取信息的手段，如果没有标签，我们依然应该努力从数据中获取有价值的信息。

参考文献

[1]N. Birbaumer and L.G. Cohen, “Brain-Computer Interfaces:Communication and Restoration of Movement in Paralysis,”J. Physiology—London, vol. 579, no. 3, pp. 621-636, 2007.

[2]H. Cecotti and A. Graser, "Convolutional Neural Networks for P300 Detection with Application to Brain-Computer Interfaces," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 3, pp. 433-445, March 2011, doi: 10.1109/TPAMI.2010.125.

[3]SWIETOJANSKI P,GHOSHAL A,RENALS S． Convolutional neural networks for distant speech recognition［J］.IEEE Signal Proc Let，2014，21(9) : 1120-1124．

[4]李奇,卢朝华.基于卷积神经网络的 P300 电位检测及在脑机接口系统中的应用[J].吉林师范大学学报(自然科学版),2018,39(03):116-122.

[5]张政,徐杨.基于双分类器的自适应单双手手势识别[J/OL].激光与光电子学进展:1-18[2020-09-20].http://kns.cnki.net/kcms/detail/31.1690.tn.20200916.1301.012.html.

[6]申奉璨,张萍,罗金,刘松阳,冯世杰.目标检测中的尺度变换应用综述[J].中国图象图形学报,2020,25(09):1754-1772.

[7]刘宏马,王生进.基于匹配滤波器和度量学习的脑电信号分类[J/OL].清华大学学报(自然科学版):1-6[2020-09-20].https://doi.org/10.16511/j.cnki.qhdxxb.2020.22.024.

[8]冯宝,张绍荣.组稀疏贝叶斯逻辑回归的 P300 信号通道自动选择算法[J].东北大学学报(自然科学版),2019,40(09):1245-1251.

[9]Vo K， Pham T， Nguyen D N， et al． Subject-independent ERP based brain-computer interfaces ［J］．IEEE Transactions on Neural Systems and Rehabilitation Engineering，2018，26(4) : 719-728．

[10]Simbolon A I， Turnip A， Hutahaean J， et al． An experiment of lie detection based EEG-P300 classified by SVM algorithm［C］/ / International Conference on Automation， Cognitive Science ， Optics ， Micro Electro-Mechanical System ， and Information Technology．Bandung-Indonesia，2015: 68-71．

[11]Momennezhad A，Ebrahimnezhad H，Shamsi M，et al．Brain activity EEG-P300 signal categorization from LPC based estimation of signal using fisher linear discriminant analysis ［J］．International Journal of Intelligent Computing in Medical Sciences ＆Image Processing，2014，6(1) : 17-26．

[12]Cecotti H，Graser A．Convolutional neural networks for P300 detection with application to brain-computer interfaces［J］. IEEE Transactions on Pattern Analysis and Machine Intelligence，2011，33(3) : 433-445．

[13]Kutsenko D O，Ivonin A A，Shuvaev V T，et al．Spatial structure of EEG in depression patients with co-occurring anxiety disorders［J］．Human Physiology，2015，41 (1) :34-38.

[14]肖郴杰. 基于深度学习的 P300 脑机接口分类算法研究[D].华南理工大学,2019.

[15]Castillo I，Schmidt-Hieber J，Aad V D V．Bayesian linear regression with sparse priors ［J］．e-Print arXiv，2015，43(5) :1986-2018．

附录 A　问题一预处理

preprocess_data.py

```python
# -*- encoding : utf-8 -*-
# @Time : 2020-09-17
# author : xiangqi

"""
tarining for person S1
"""

import pandas as pd
import numpy as np
import math



io_S1 = r'D:/2020年中国研究生数学建模竞赛赛题/projectC/excel_data/S1/S1_train_data.xlsx'
io_S1_index = r'D:/2020年中国研究生数学建模竞赛赛题/projectC/excel_data/S1/S1_train_event.xlsx'

"""
oldest version : only pick up one data at 75*4=300ms
"""


def preprocess(io_S1, io_S1_index, sheet_name, a, b,  num):
    data_S1_char01 = pd.read_excel(io_S1, sheet_name=sheet_name, header=None)
    data_S1_char_01_index = pd.read_excel(io_S1_index, sheet_name=sheet_name, header=None)


    # print(data_S1_char01)
    index = []
    for i in range(data_S1_char_01_index.shape[0]):
        if data_S1_char_01_index.iloc[i,0] < 100:
            index_i = data_S1_char_01_index.iloc[i,:].tolist()
            index.append(index_i)
    index_df = pd.DataFrame(index, index=None)
    print(index_df)
    # input("2")

    index_df['2'] = 0
    for i in range(index_df.shape[0]):
        if index_df.iloc[i,0] == a or index_df.iloc[i,0] == b:
            index_df.iloc[i,2] = 1
        else:
            index_df.iloc[i,2] = 0
    print(index_df)
    # input("2.5")

    label_i = index_df['2']
    label_final = []
    for i in range(label_i.shape[0]):
        label = label_i[i]
```

```
    for i in range(label_i.shape[0]):
        label = label_i[i]
        for j in range(32):
            label_final.append(label)
    label_final = pd.Series(label_final)
    print(label_final)
    # label_final.to_csv("train_label_temp.txt",index=None,header=None,mode='a')


    train_data = []
    index_list = index_df.iloc[:,1].tolist()
    for index in index_list:
        for i in range(index+25,index+153):
            data = data_S1_char01.loc[i]
            train_data.append(data)
    train_data = pd.DataFrame(train_data)
    print(train_data)
    # input("3")

    train_data_2 = []
    for i in range(train_data.shape[0]):
        if i % 4 == 0:
            data = train_data.iloc[i,:]
            train_data_2.append(data)
    train_data_2 = pd.DataFrame(train_data_2)
    print(train_data_2)


    # train_data_2.to_csv("train_data_temp.txt",index=None,header=None,sep='\t',mode='a')



char_S1 = {'P1': ["char01(B)", 1, 8, 1], 'P2': ["char02(D)", 1, 10, 2],
           'P3': ["char03(G)", 2, 7, 3], 'P4': ["char04(L)", 2, 12, 4],
           'P5': ["char05(O)", 3, 9, 5], 'P6': ["char06(Q)", 3, 11, 6],
          'P7': ["char07(S)", 4, 7, 7], 'P8': ["char08(V)", 4, 10, 8],
          'P9': ["char09(Z)", 5, 8, 9],'P10': ["char10(4)", 5, 12, 10],
          'P11': ["char11(7)", 6, 9, 11], 'P12': ["char12(9)", 6, 11, 12]}


for pi in list(char_S1.keys()):
    preprocess(io_S1, io_S1_index,
    sheet_name=char_S1[pi][0],
    a=char_S1[pi][1],
    b=char_S1[pi][2],
    num=char_S1[pi][3])
```

附录 B　CNN

ql_CNN.py

```
184
185 def train(x, y, model):
186     '''
187         ori_shape = x.shape
188         x = x.reshape((x.shape[0], -1))
189         x = preprocessing.normalize(x, norm='12')
190         x = x.reshape(ori_shape)
191         print("X_minMax", x[:10])
192     '''
193
194     # do PCA
195     if args.do_PCA:
196         x = dimensionality_reduction2(x)
197         args.time_len = x.shape[1]
198         args.embedding_dim = x.shape[2]
199         print("After PCA", x.shape)
200
201     train_data, test_data, train_label, test_label = train_test_split(
202         x, y, test_size=0.2, random_state=1, shuffle=True)
203
204     print("shape:", train_data.shape, train_label.shape)
205
206     '''
207     print('\n Training LogisticRegression')
208     train_content = train_data.reshape(train_data.shape[0], -1)
209     test_content = test_data.reshape(test_data.shape[0], -1)
210     lrs = train_lr(train_content, train_label)
211     for clf in lrs:
212         score = clf_pred(clf, test_content, test_label)
213
214     '''
215     optimizer = torch.optim.Adam(model.parameters(), lr=args.lr)
216
217     steps = 0
218     best_acc = 0
219     last_step = 0
220     model.train()
221     for epoch in range(1, args.epoch + 1):
222         if train_data.shape[0] % args.batch_size == 0:
223             num_iters_per_epoch = train_data.shape[0] // args.batch_size
224         else:
225             num_iters_per_epoch = (train_data.shape[0] // args.batch_size) + 1
226
227         for iters in range(num_iters_per_epoch):
228             batch_data, batch_label = get_iteration_batch(iters, train_data, train_label)
229
230             optimizer.zero_grad()
231             logits, loss = model(batch_data, batch_label)
232             # print("logits, loss", logits, loss)
233             loss.backward()
234             optimizer.step()
235
236             steps += 1
237             if steps % args.log_interval == 0:
238                 pred = torch.max(logits, 1)[1]
239
240                 print('\nEpoch-{} - Batch[{}] - loss: {:.6f}'.format(epoch, steps, loss.item()))
241                 score = elevate_result(pred, batch_label.view(-1))
242
243             if steps % args.test_interval == 0:
244                 score = evaluating(test_data, test_label, model)
245                 dev_acc = score[0]
246                 if dev_acc > best_acc:
247                     best_acc = dev_acc
248                     last_step = steps
249                     if args.save_best:
250                         print('Saving best model, acc: {:.4f}%\n'.format(best_acc))
251                         save(model, args.save_dir, 'best', steps)
252                 else:
253                     if steps - last_step >= args.early_stopping:
254                         print('\nearly stop by {} steps, acc: {:.4f}%'.format(args.early_stopping, best_acc))
255                         return model
256
257
258     return model
```

q1_CNN.py

```
def main():
    root_dir = "./data2"
    sep_dir = ['S1', 'S2', 'S3', 'S4', 'S5']
    # sep_dir = ['S1', 'S2', 'S3']
    features, labels = [], []

    if args.model_name == "TextCNN":
        model = TextCNN(args)
    else:
        model = MLP(args)
    print("training")
    for user in sep_dir:
        print("\n\n current user:", user)
        root_path = os.path.join(root_dir, user)

        user_data, feature, label = get_data_fast(root_path, user, do_train="train", time_begin=0, time_end=args.time_len, band=True)

        # print(len(feature), len(feature[0]), len(feature[0][0]))
        features += feature
        labels += label

    # normed_data, feature, label = norm_by_epoch(user_data)
    x = np.array(features)
    y = np.array(labels)
    print("ori:", x.shape, y.shape)

    x, y = upsample(x, y, same=True)
    print("after upsample", x.shape, y.shape)

    # x = standardization(x)
    # print(x[:10])

    model = train(x, y, model)

    print("\n\ntesting")
    for user in sep_dir:
        print("\n\n current user:", user)
        root_path = os.path.join(root_dir, user)

        user_res = {}
        merge_k = 5
        row_features, col_features = get_test_data(root_path, user, do_train="test", time_len=args.time_len,
                                                   epoch_n=merge_k)
        for sheet in col_features.keys():
            sheet_col_features, sheet_row_features = col_features[sheet], row_features[sheet]
            print("shape", sheet, sheet_col_features.shape, sheet_row_features.shape)

            batch_row_data = torch.FloatTensor(sheet_row_features)
            batch_col_data = torch.FloatTensor(sheet_col_features)

            logits_row = model(batch_row_data)
            logits_col = model(batch_col_data)

            softmax = torch.nn.Softmax(dim=1)
            logits_row, logits_col = softmax(logits_row)[:, 1], softmax(logits_col)[:, 1]
            preds_row = logits_row.argmax(dim=0).numpy()
            preds_col = logits_col.argmax(dim=0).numpy()

            print("predict result:", user, sheet, preds_row + 1, preds_col + 6 + 1)
            user_res[sheet] = (preds_row + 1, preds_col + 6 + 1)


if __name__ == "__main__":
    main()
```

Python file

附录 C　SVM

q1_SVM.py

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import  accuracy_score
import pandas as pd
import numpy as np

data = pd.read_csv('train_data_copy.txt', header=None, sep='\s+')

for i in range(1,data.shape[0], 1):
    for j in range(1, data.shape[1], 1):
        data.ix[i][j] = int(data.ix[i][j])
X1 = data.iloc[:,1:]
a = data.iloc[:, 0]

X1 = np.array(X1)
y1 = np.array(a)
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=1/5., random_state=8)

print(X_train,'\n', X_test,'\n', y_train,'\n', y_test)

svc = svm.SVC(C=1.0, kernel='poly', degree=3)
svc.fit(X_train, y_train)

y_pred = svc.predict(X_test)

print("SVM准确率： ", accuracy_score(y_test, y_pred))


```

附录 D　贝叶斯

classer.py

```
73
74
75  def train_svm(train_content, train_label):
76      kernals = ["linear"]
77      clfs = []
78      for kernel in kernals:
79          clf = svm.SVC(kernel=kernel)
80          clf = clf_train(clf, train_content, train_label)
81          clfs.append(clf)
82      return clfs
83
84  def train_bayes(train_content, train_label):
85      bayes = [BernoulliNB()]
86      clfs = []
87      for baye in bayes:
88          clf = clf_train(baye, train_content, train_label)
89          clfs.append(clf)
90      return clfs
91
```

1. 主程序如下：

classer.py

```
def main():
    root_dir = "./data2"
    train_dict_file = open(os.path.join(root_dir, "train_data.pkl"), 'rb')
    train_data = pickle.load(train_dict_file)

    data_x = {}
    data_y = {}
    #sep_dir = ['S1', 'S2', 'S3', 'S4', 'S5']
    sep_dir = ['S1']
    # sep_dir = ['S1']
    for user in sep_dir:
        print("\n\n current user:", user)
        x, y = [], []
        user_data = train_data[user]
        for target_char, char_data in user_data.items():
            # print(target_char)
            for cols, fea_la in char_data.items():
                for idx, features in fea_la.items():
                    x.append(features[0])
                    y.append(features[-1])
        x = np.array(x)
        y = np.array(y)
        data_x[user] = x
        data_y[user] = y

        train_content, test_content, train_label, test_label = train_test_split(
            x, y, test_size=0.2, random_state=1, shuffle=True)

        scores = []

        svms = train_svm(train_content, train_label)
        for clf in svms:
            score = clf_pred(clf, test_content, test_label)
            scores.append(score)
        #plot_result(scores)
        bayes = train_bayes(train_content, train_label)
        for clf in bayes:
            score = clf_pred(clf, test_content, test_label)
            scores.append(score)
        classifier = LogisticRegression()
        classifier.fit(train_content, train_label)
        yp = classifier.predict(test_content)
        score = elevate_result(yp, test_label)
        scores.append(score)
        plot_result(scores)


if __name__ == "__main__":
    #     pass
    main()
```

附录 E　问题二

```
39
40  def load_data_from_file(content_path,label_path):
41      f = open(content_path, 'r')
42      reader_c = csv.reader(f)
43      content = list(reader_c)
44      print('content', content)
45      #content = io.mmread(content_path)
46      fl = open(label_path, 'r')
47      reader_l = csv.reader(fl)
48      label = list(reader_l)
49      #label = io.mmread(label_path)
50      print('label',label)
51      # with open(label_path, 'r') as f:
52      #     label = json.load(f)
53      return content,label
54
55  def split_data(content,label):
56      train_content,test_content,train_label,test_label = train_test_split(content,label,test_size=0.2)
57      return train_content,test_content,train_label,test_label
58
59   @logtime
60  def dimensionality_reduction(content):
61      n_components = 1000
62      pca = PCA(n_components=n_components, svd_solver='auto')
63      pca.fit(content)
64      content = sparse.csr_matrix(pca.transform(content))
65      return content
66
```

附录 F　问题三

S3VM.py

```
import copy as cp
import numpy as np
from scipy import optimize
from scipy import sparse
import time


class linear_k:

    def __init__(self, issparse):
        self._sparse = issparse

    def compute(self, data1, data2):
        if self._sparse:
            return data1 * data2.T
        else:
            return np.mat(data1) * np.mat(data2).T


class rbf_k:

    def __init__(self, sigma, issparse):
        self._sparse = issparse
        self._sigma = sigma

    def compute(self, mat1, mat2):
        mat1 = np.mat(mat1)
        mat2 = np.mat(mat2)
        mat1T_mat1 = np.mat([(v * v.T)[0, 0] for v in mat1]).T
        mat2T_mat2 = np.mat([(v * v.T)[0, 0] for v in mat2]).T
        mat1T_mat1 = mat1T_mat1 * np.mat(np.ones((mat2.shape[0], 1), dtype=np.float64)).T
        mat2T_mat2 = np.mat(np.ones((mat1.shape[0], 1), dtype=np.float64)) * mat2T_mat2.T
        k = mat1T_mat1 + mat2T_mat2
        k -= 2 * mat1 * mat2.T
        k *= - 1. / (2 * np.power(self._sigma, 2))
        return np.exp(k)

class Quasi_Newton_S3VM:
```

附录 G　问题四

K_means_q4.ipynb　sleep_data.py

Managed Jupyter server: auto-start　Python 3

```
#%%

data = pd.read_excel("p4.xlsx", sheet_name=None)
X_list, y_list = [], []
for key, sheet in data.items():
    _x = np.asarray(sheet[["Alpha", "Beta", "Theta", "Delta"]])
    _y = int(key[-2]) * np.ones(_x.shape[0], dtype=int)
    X_list.append(_x)
    y_list.append(_y)
X, y = np.concatenate(X_list), np.concatenate(y_list)
X = pd.DataFrame(X,columns=["Alpha", "Beta", "Theta", "Delta"])
y = pd.DataFrame(y,columns=['Target'])
y = pd.DataFrame(y['Target'] - 2)
X.head()
y.head()

from mpl_toolkits.mplot3d import Axes3D
plt.figure(figsize=(24,6))
colors = np.array(['red', 'green', 'blue','yellow','purple'])
sleep_targets_legend = np.array(["deep sleep","sleep II","sleep I","eye move","clear"])
red_patch = mpatches.Patch(color='red', label='deep sleep')
green_patch = mpatches.Patch(color='green', label='sleep II')
blue_patch = mpatches.Patch(color='blue', label='sleep I')
yellow_patch = mpatches.Patch(color='yellow', label='eye move')
purple_patch = mpatches.Patch(color='purple', label='clear')

ax1 = plt.axes(projection='3d')
# "Alpha", "Beta", "Theta", "Delta"
```

Managed Jupyter server: auto-start　Python 3

```
plt.title('Alpha vs Theta')
plt.legend(handles=[red_patch, green_patch, blue_patch, yellow_patch, purple_patch])

plt.subplot(2, 3, 3)
plt.scatter(X['Alpha'], X['Delta'], c=colors[y['Target']])
plt.title('Alpha vs Delta')
plt.legend(handles=[red_patch, green_patch, blue_patch, yellow_patch, purple_patch])

plt.subplot(2, 3, 4)
ax1.scatter3D(X['Beta'], X['Theta'], X['Delta'], c=colors[y['Target']])
plt.title('Beta vs Theta')
plt.legend(handles=[red_patch, green_patch, blue_patch, yellow_patch, purple_patch])
plt.show()

plt.subplot(2, 3, 5)
plt.scatter(X['Beta'], X['Delta'], c=colors[y['Target']])
plt.title('Beta vs Delta')
plt.legend(handles=[red_patch, green_patch, blue_patch, yellow_patch, purple_patch])

plt.subplot(2, 3, 6)
plt.scatter(X['Theta'], X['Delta'], c=colors[y['Target']])
plt.title('Theta vs Delta')
plt.legend(handles=[red_patch, green_patch, blue_patch, yellow_patch, purple_patch])

iris_k_mean_model = KMeans(n_clusters=5)
iris_k_mean_model.fit(X)

iris_k_mean_model.labels_
```

```
#%%

iris_k_mean_model.cluster_centers_

#%%

plt.figure(figsize=(12,3))
colors = np.array(['red', 'green', 'blue', 'yellow', 'purple'])
predictedY = np.choose(iris_k_mean_model.labels_, [1, 0, 4, 3, 2]).astype(np.int64)

#%%

sm.accuracy_score(predictedY, y['Target'])

#%% md

## Interpretation of Confusion Matrix
Correctly identifed all 0 classes as 0's
correctly classified 48 class 1's but miss-classified 2 class 1's as class 2
correctly classified 36 class 2's but miss-classified 14 class 2's as class 1

#%%

sm.confusion_matrix(predictedY, y['Target'])

#%%
```