目录

第一章　任务理解

1.1　数据集理解

数据集由德国图宾根大学计算机工程系(Rosenstiel 教授)和医学心理学与行为神经生物学研究所(Niels Birbaumer)，以及 Max-Planck-Institute for Biological Cybernetics，德国图宾根 (Bernhard Schölkopf) 和德国波恩大学癫痫学系（埃尔格教授）提供。

实验期间，受试者被要求想象执行左小指或舌头的运动，同时使用放置在大脑右侧运动皮层上的 8x8 ECoG 铂电极网格（对应 64 个不同的接收脑电信号的位置）来获取脑电活动的时间序列，假定网格完全覆盖右侧运动皮层（实际上它也部分覆盖周围的皮层区域）。

所有记录均以 1000Hz 的采样率进行。放大后，记录的电位被存储为微伏值。每个试验由想象的舌头或想象的手指运动组成，并记录 3 秒的持续时间。为避免数据反映视觉诱发电位，记录间隔在视觉提示结束后 0.5 秒开始。

1.2　数据分析

数据集由两部分组成：

- 第 1 部分：278 次试验期间的大脑活动，使用试验 x 电极通道 x 时间序列样本格式存储在名为 X 的 3D 矩阵中。
- 第 2 部分：278 次试验的标签，存储为一个名为 Y 的-1/1 值向量。

将单次试验获得脑电信号提取后分别对原矩阵和转置矩阵进行绘图，可得 64 个不同脑电接收点的信号变化情况。

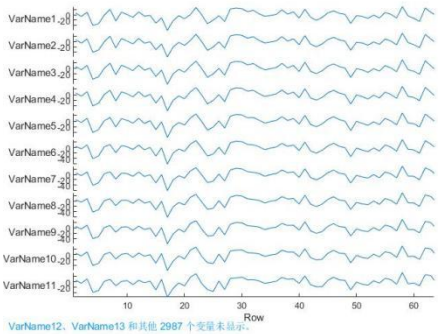

图 1.1　原矩阵 64*3000

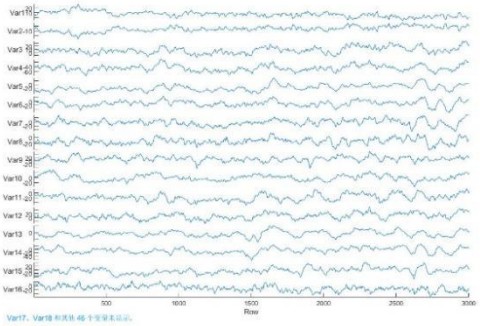

图 1.2　转置矩阵 3000*64

其中，原矩阵（图 1.1）描述的是 64 个不同电极在同一时刻的电位比较情况，转置矩阵（图 1.2）描述的是 3s 内同一电极的电位变化情况。根据我们实验要求，应当选择转置矩阵进行特征提取。

1.3 任务理解

通过查阅相关文献资料了解到：此次实验是一个典型的与运动想象脑电信号有关问题，是对已有的两类运动想象脑电信号进行二分类。小组需要根据已有的数据集和标签进行特征提取，做一个分类器，目标是向分类器输入 N 个 64×3000 的数据记录，能输出 N 个 1/-1 的分类值。

组内初步讨论，决定使用 matlab 来进行实验。首先，特征提取是最关键的一步，只有提取出好的特征，才能得到好的结果。我们采用了 3 种方法进行了特征提取，分别是：CSP(共空间模式)、DWT(小波变换)和 AR(自回归)。然后是学习分类，我们也采用了 3 种算法，分别是：Bagging 算法、Boosting 算法和 Aadboost 算法。上述算法的原理与应用下面会提及细讲。

第二章　特征提取

2.1　CSP

共模空间模式（Common Mode Space Pattern）算法的过程是在二分类任务中通过空间滤波进行特征提取的算法。

CSP 是目前公认 EEG 信号二分类中应用最广泛、精度最高的方法之一，其基本原理是寻找最优的投影方向。通过空间投影使得一种信号的方差最大的同时，另一种信号的方差达到最小，从而两种信号的特征差异最大化。

2.1.1　CSP 的实现

CSP 算法的实现关键在于求得投影矩阵。

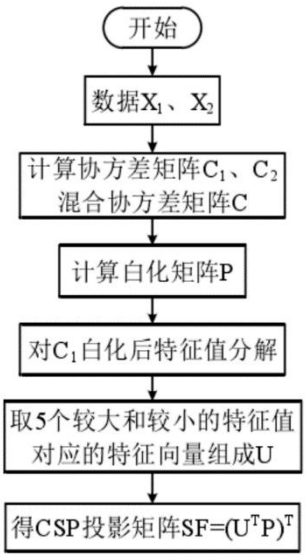

图 2-1　CSP 的实现

假设 X_1 ,X_2 分别为两类运动想象下获得的多通道（如我们做的 dataset1 是 64 通道）的响应与否矩阵，通道数显然是小于样本采集数的(一个通道至少由一个数据记录)，则 X_1 和 X_2 本身各自的协方差矩阵 C_1 ,C_2 和混合空间协方差矩阵 C 就可以计算了。

之后，对两者的混合空间协方差矩阵进行特征值分解，可以获得相应的特征值构成的对角阵λ与特征向量矩阵 U，再通过公式 $P=\sqrt{\lambda^{-1}}\cdot U^T$ ，就可以获得

X_1 和 X_2 的协方差白化特征矩阵了（通过矩阵白化后，协方差矩阵变换为对角阵）。

这之后，构建空间滤波器，先将 C_1 ,C_2 分别白化为对角阵，再对两者主分量进行分解，获得两者的特征向量 B_1 ,B_2 以及两者特征值构成的对角阵λ_1 ,λ_2 。显然，会有 B_1 =B_2 ，λ_1 +λ_2 =E，即两者的特征值是反相关的。于是无论是λ_1 还是λ_2 ，其取得最大特征值时对应的特征向量变换方案对于矩阵方差的分离是最佳的。此时，投影矩阵对应的空间滤波器为 $W = B^T \cdot P$ 。

最后，有了空间滤波器来做特征提取，我们就只需要做投影就好了，将 X_1 ,X_2 分别与 W 进行叉乘，即可获得其特征。

2.1.2 关键代码

```
function features = extractCSP(EEGSignals, CSPMatrix, nbFilterPairs)

nbTrials = size(EEGSignals.x,3);
features = zeros(nbTrials, 2*nbFilterPairs+1);
Filter = CSPMatrix([1:nbFilterPairs (end-nbFilterPairs+1):end],:);

% 从每次试验中提取 CSP 功能
for t=1:nbTrials
   %将数据投影到 CSP 过滤器上
   projectedTrial = Filter * EEGSignals.x(:,:,t)';

   %将特征生成为投影信号的对数方差
   variances = var(projectedTrial,0,2);
   for f=1:length(variances)
      features(t,f) = log(1+variances(f));
   end
end
```

2.2 DWT

离散小波变换（Discrete Wavelet Transformation）是基本小波尺度和变换的

离散化，其中二进小波常被用作图像处理中的小波变换函数，即使用 2 的整数幂进行除法。

2.2.1 DWT 相较于 FFT

做傅里叶变换只能得到一个频谱，做小波变换可以得到一个时频谱。因为快速傅里叶变换频域和时域分辨的精确度不能兼得: 所选窗口太窄，则频域不精确；所选窗口太宽，则时域不精确。而小波变换则可以通过多解析度分析实现时域频域同时定位。

2.2.2 对 DWT 实现时频域同时定位

小波变换函数有两个很重要的因子: 伸缩因子和平移因子。

通过伸缩因子收缩和伸张小波，使得每次遍历分析实现对不同频率信号的逼近，得到频域信息。通过平移因子，使得小波能够沿信号时间轴进行遍历，得到时域信息。

2.2.3 DWT 特征提取的实现

调用 matlab 的小波提取算法包的 wavedec 函数，对信号进行 3 层小波提取；再调用 detcoef 函数进行小波细节提取；最后将一次试验的每个通道的细节进行求平均作为该次试验的特征提取值。

2.2.4 关键代码

```
function X=extractDWT(x_train,startS,endS,wStep,wRange)
FS=128;

N=size(x_train,3);
sz=floor((endS-(startS+wRange))/wStep)+1;
X=zeros(sz*223,2);
cn=0;
for i=1:N

  for sig=startS:wStep:endS-wRange

    sW=sig*FS+1;
```

```
    eW=(sig+wRange)*FS;


    C3Sig=x_train(sW:eW,5,i);

    C4Sig=x_train(sW:eW,33,i);


    waveletFunction = 'db4';

    waveletLevel=3;

    [wCoe,L] = wavedec(C3Sig,waveletLevel,waveletFunction);

    C3D3 = detcoef(wCoe,L,3);   % Mu


    [wCoe,L] = wavedec(C4Sig,waveletLevel,waveletFunction);

    C4D3 = detcoef(wCoe,L,3);  % Mu


    cn=cn+1;

    % Mean of the absolute values

    X(cn,1)=sum(C3D3.^2)/numel(C3D3);

    X(cn,2)=sum(C4D3.^2)/numel(C4D3);
  end
end
end
```

2.3 AR

自回归模型（Autoregressive Model）是最常见的平稳时间序列模型之一，简单理解起来就是使用前面几个数据来预测后面的数据。

2.3.1 AR 的原理

将 AR 模型应用于脑电分析的基本思想是假设 AR 过程可以用来逼近真实的脑电信号，并且脑电信号可以用线性滤波器来描述其产生过程，输出的 系统可以由输出的当前状态决定，由过去状态决定。 基于这个假设，根据实际脑电信号选择合适的阶次和参数，可以使 AR 模型对应的 AR 过程尽可能接近脑电信号。

常用的估计 AR 系数的方法有自相关法或求解 Yule-Walker 方程，如 Burg 算法、Levinson-Durbin 递归算法来估计反射系数和确定 AR 模型参数。

2.3.2 关键代码

```
function X=extractAR(x_train,AROrder,startS,endS,wStep,wRange)


FS=128;


N=size(x_train,3);
sz=floor((endS-(startS+wRange))/wStep)+1;
X=zeros(sz*FS,2);
cn=0;
for i=1:N


  for sig=startS:wStep:endS-wRange
    sW=sig*FS+1;
    eW=(sig+wRange)*FS;


    C3Sig=x_train(sW:eW,1,i);
    C4Sig=x_train(sW:eW,3,i);
    c3= arburg(C3Sig, AROrder);
    c4= arburg(C4Sig, AROrder);
    cn=cn+1;
    X(cn,1)=sum(c3.^2)/numel(c3);
    X(cn,2)=sum(c4.^2)/numel(c4);


  end
end
```

第三章　学习分类

3.1　Bagging

3.1.1　Bagging 的基本思路

装袋算法（Bagging）的基本思路是先将训练集分离成多个子集，然后通过各个子集训练多个模型，这是一种提高分类准确率的算法，通过给定组合投票的方式获得最优解。举个例子，比方说你生病了，去 n 个医院看了 n 个医生，每个医生都给你开了药方，最后哪个药方出现次数多，就说明这个药方越有可能是最优解。

装袋模型有三种：装袋决策树、随机森林和极端随机树。

3.1.2　关键代码

```
function [prediction, err] = bagging_predict (model, X, y)
  num_iterations = length(model.models);
  no_objects = length(y);
  pred_prediction = zeros([no_objects num_iterations]);
  err = zeros([num_iterations 1]);
  if strcmp(model.algorithm, 'svm')
    for alg = 1 : num_iterations
      pred_prediction(:,alg) = svmpredict(y, X, model.models{alg});
      func = @(i) find_max(pred_prediction(i,:));
      prediction = arrayfun(func, 1 : no_objects)';
      err(alg) = 1-sum(prediction ~= y) / no_objects;
    end
  elseif strcmp(model.algorithm, 'classification_tree')
    for alg = 1 : num_iterations
      pred_prediction(:,alg) = predict(model.models{alg}, X);
      func = @(i) find_max(pred_prediction(i,:));
      prediction = arrayfun(func, 1 : no_objects)';
```

```
        err(alg) = 1-sum(prediction ~= y) / no_objects;
    end
  else
    error('Incorrect type of algorithm!');
  end
end
function [result] = find_max (vector)

  if sum(vector == -1) > sum(vector == +1)
    result = -1;
  else
    result = +1;
  end
end
```

3.2 Boosting

3.2.1 Boosting 的基本思路

Boosting 是一族可将弱学习器提升为强学习器的算法。

这族算法的工作机制是先从初始训练集训练出一个基学习器，再根据基学习器的表现对训练样本分布进行调整，使得先前基学习器做错的训练样本在后续受到更多关注，然后基于调整后的样本分布来训练下一个基学习器；如此重复进行，直至基学习器数目达到事先指定的值 T，最终将这 T 个基学习器进行加权结合。

3.2.2 关键代码

```
function [prediction, err] = gradient_boosting_predict (model, X, y)
  num_iterations = length(model.weights);
  no_objects = length(y);
  pred_prediction = zeros([no_objects num_iterations]);

  for alg = 1 : num_iterations
    value = zeros([no_objects 1]) + model.b_0;
```

```
    for i = 1 : alg
        if strcmp(model.algorithm, 'epsilon_svr')
            value = value + svmpredict(y, X, model.models{i}) * model.weights(i);
        elseif strcmp(model.algorithm, 'regression_tree')
            value = value + predict(model.models{i}, X) * model.weights(i);
        end
    end
    pred_prediction(:,alg) = value;
  end
  prediction = pred_prediction(:,end);
  err = zeros([num_iterations 1]);
  if strcmp(model.loss, 'absolute')
    temp = (bsxfun(@minus, pred_prediction, y));
    err = 1-abs(sum(temp)) / no_objects;
  elseif strcmp(model.loss, 'logistic')
    prediction = sign(prediction);
    temp = (bsxfun(@eq, sign(pred_prediction), y));
    err = 1-sum(temp == 0) / no_objects;
  end
  if size(err, 1) == 1
    err = err';
  end
end
```

3.3 Aadboost

Aadboost 算法系统具有较高的检测速率，且不易出现过适应现象，在实现过程中根据训练集的大小初始化样本权值，使其满足均匀分布，在后续操作中通过公式来改变和规范化算法迭代后样本的权值。

3.3.1 AdaBoost 的实现

- 先通过对 N 个训练样本的学习得到第一个弱分类器；

- 将分错的样本和其他的新数据一起构成一个新的 N 个的训练样本，通过对这个样本的学习得到第二个弱分类器 ；
- 将 1 和 2 都分错了的样本加上其他的新样本构成另一个新的 N 个的训练样本，通过对这个样本的学习得到第三个弱分类器；
- 最终经过提升的强分类器。即某个数据被分为哪一类要由各分类器权值决定。

3.3.2 关键代码

```
function [Label, Err] = predAdaBoost(abClassifier, X, Y)
N = size(X, 1);
if nargin < 3
   Y = [];
end
M = abClassifier.nWC;
LabM = zeros(N, M);
for i = 1:M
   LabM(:,i) = abClassifier.Weight(i)*predStump(X, abClassifier.WeakClas{i});
end
Label = zeros(N, 1);
LabM = sum(LabM, 2);
idx = logical(LabM > 0);
Label(idx) = 1;
Label(~idx) = -1;
if ~isempty(Y)
   Err = logical(Label ~= Y);
   Err = 1-sum(Err)/N;
end
end
```

第四章　实验结果分析

4.1　实验结果

通过对三种特征提取算法（CSP,DWT,AR）与三种学习分类算法（Bagging, Boosting, AdaBoost）两两组合，我们可以得到九种组合方法，下面我们用这九种方法分别对拟合过的数据集与未拟合过的数据集进行分类标签，所得预测准确率结果如下。

4.1.1　拟合过的数据预测准确率

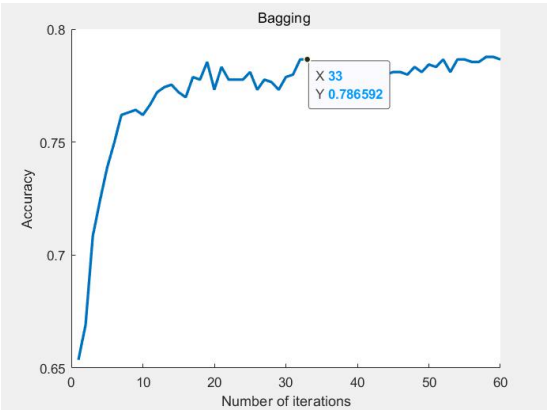

图 4-1　AR + bagging

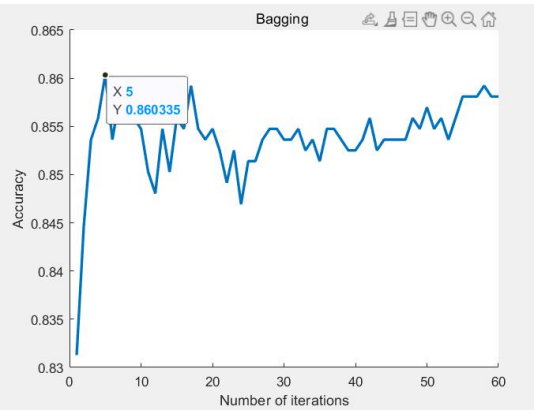

图 4-2　CSP + bagging

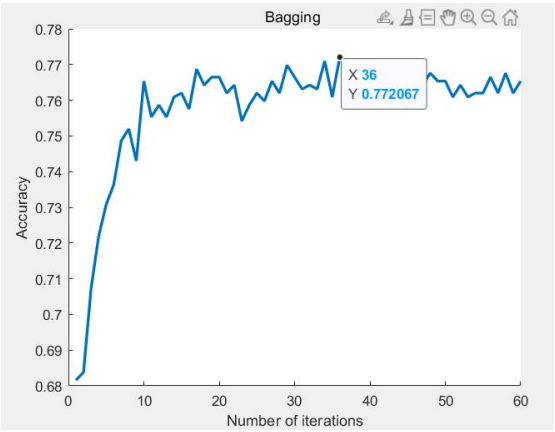

图 4-3　DWT + bagging

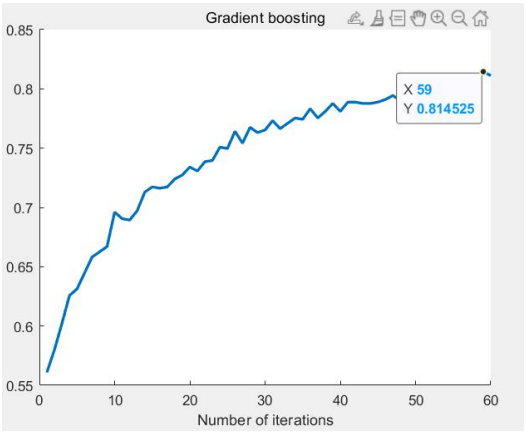

图 4-4　AR + boosting

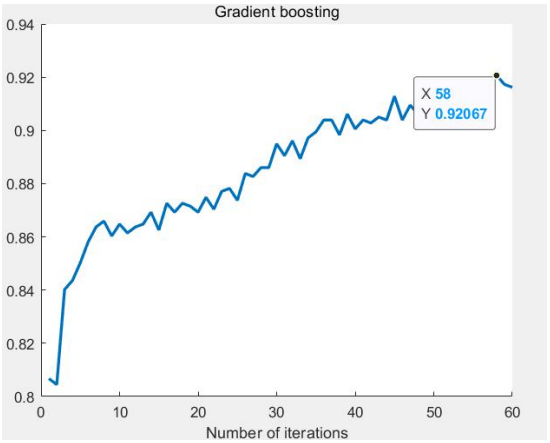

图 4-5　CSP + boosting

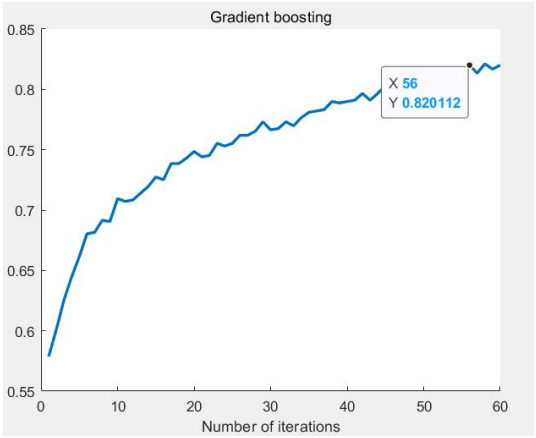

图 4-6　DWT + boosting

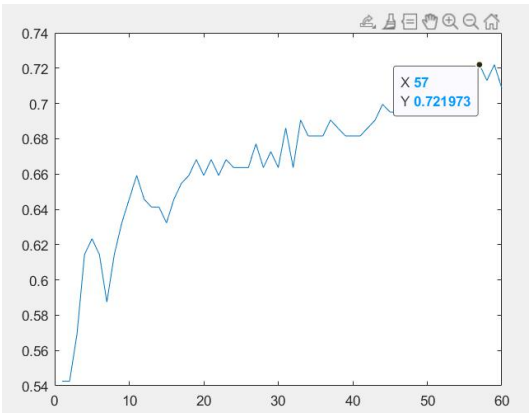

图 4-7　AR + adaboost

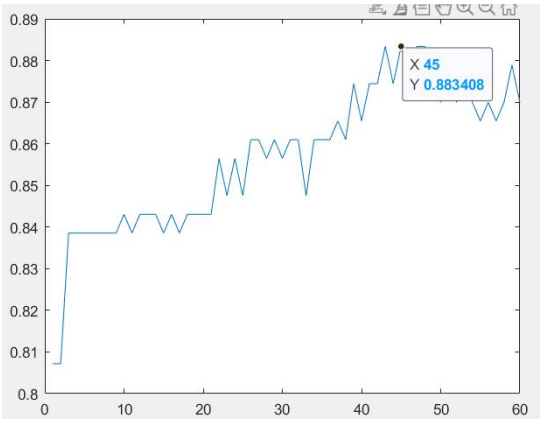

图 4-8　CSP + adaboost

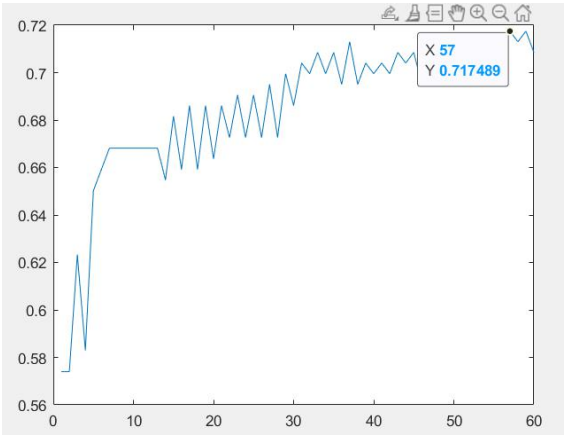

图 4-9　DWT + adaboost

4.1.2　未拟合过的数据（测试集）预测准确率

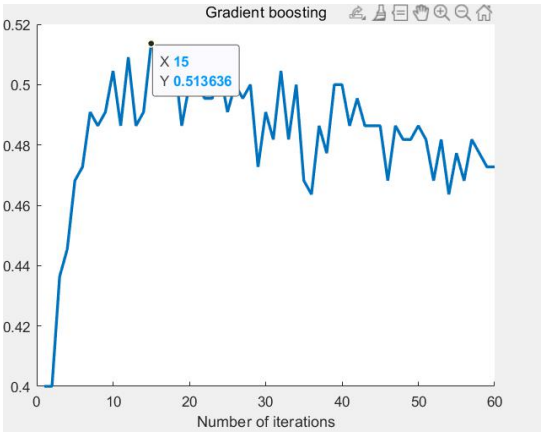

图 4-10　AR + bagging

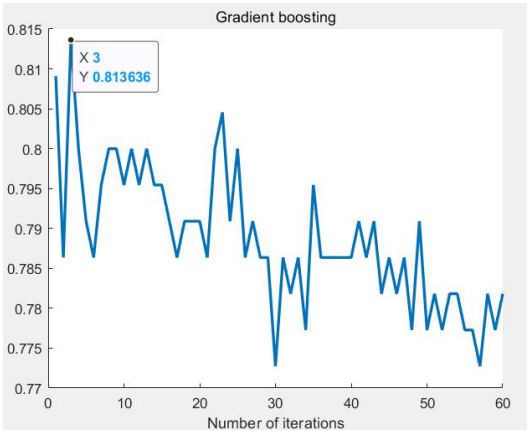

图 4-11　CSP + bagging

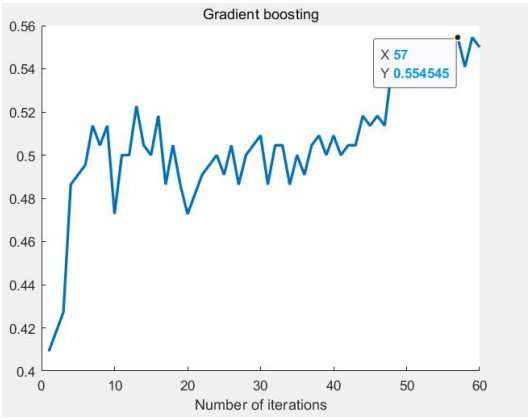

图 4-12　DWT + bagging

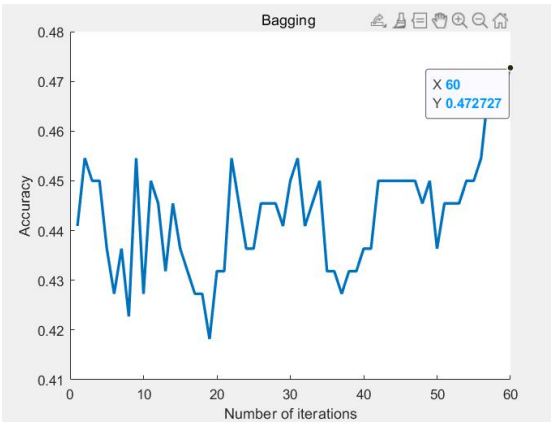

图 4-13　AR + boosting

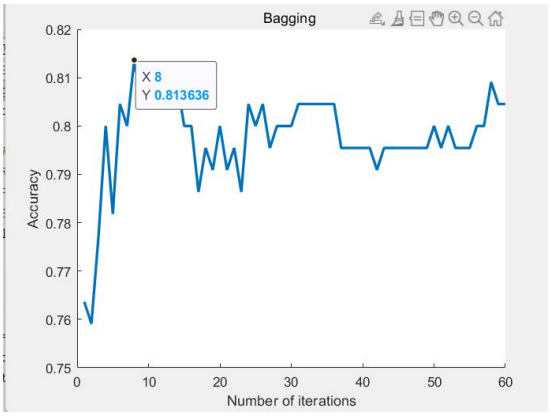

图 4-14　CSP + boosting

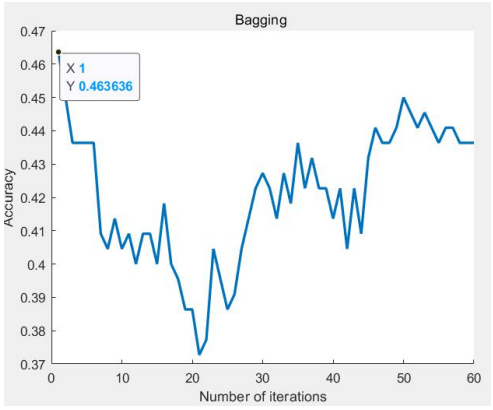

图 4-15　DWT + boosting

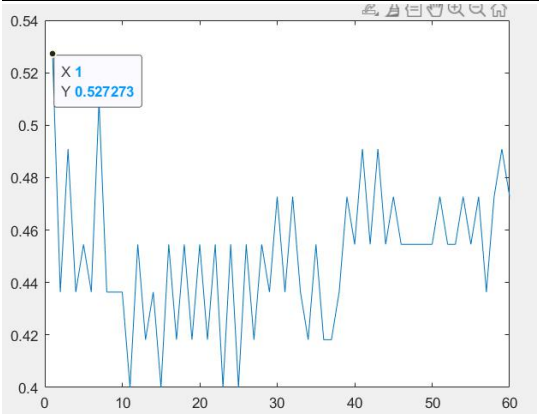

图 4-16　AR + adaboost

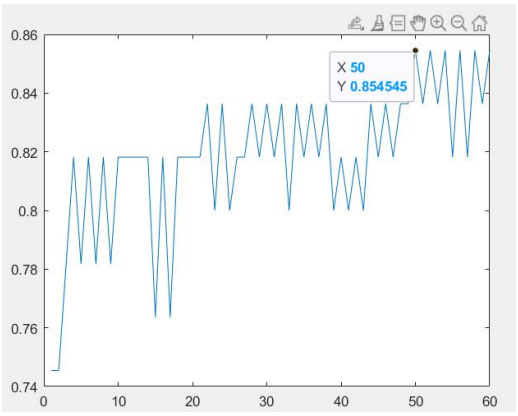

图 4-17　CSP + adaboost

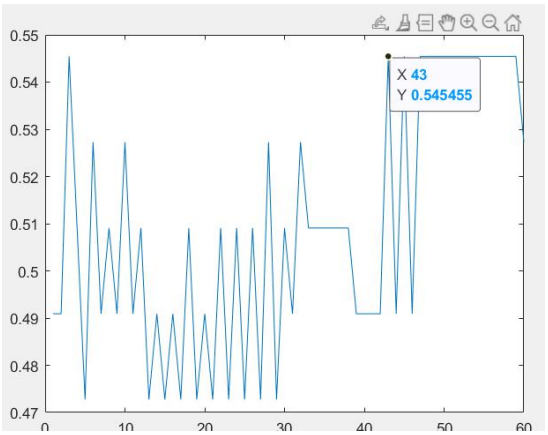

图 4-18　DWT + adaboost

4.2　结果分析

对于拟合过的数据，三种特征提取算法都有很好的分类效果。而对于测试数据，只有使用 CSP 算法进行特征提取的分类效果很好，但是用 AR 算法和 DWT 算法进行特征提取的分类效果差强人意。

通过进一步对准确度比对分析，我们能发现，这不是分类算法的问题，因为三种分类算法对于用 CSP 进行特征提取的测试数据的分类效果依然很好，而采取不同特征提取算法时，准确度的差异明显，问题出在特征提取上。

为了方便进一步分析，我们画出了用 CSP、AR 和 DWT 进行特征提取后的数据图。

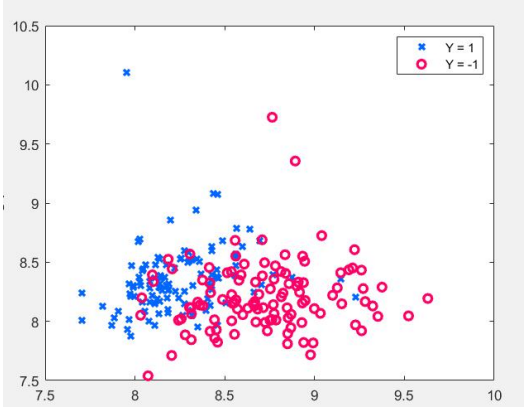

图 4-19　CSP 特征提取

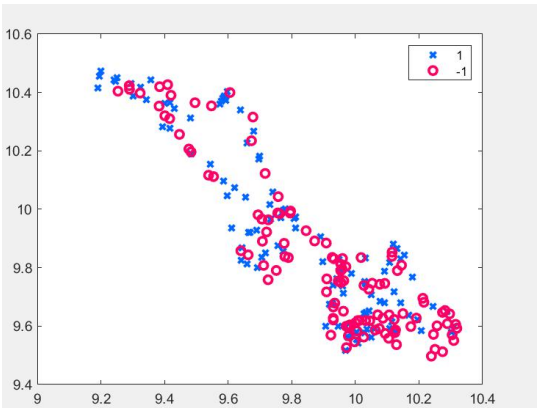

图 4-20　AR 的特征提取

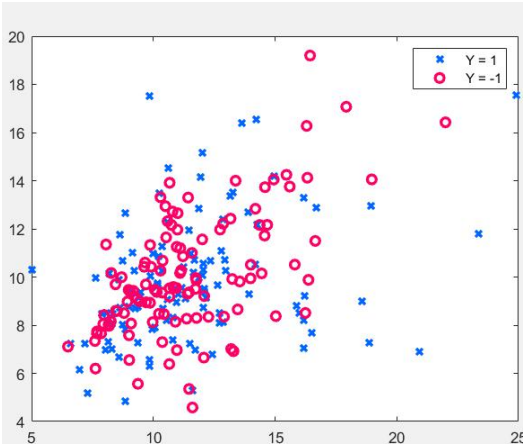

图 4-21　DWT 特征提取

从图中我们可以明显看出 CSP 特征提取后的结果有很明晰的分界线，而 DWT 与 AR 提取后的结果近乎没有明显的分界线可言。

对于训练集有很好的分类效果则是分类模型过拟合的原因。

进一步查阅资料，我们发现对于 DWT 和 AR 不能很好的进行特征提取这一结果，在其他学者的研究中也同样存在，我们的猜想得到印证。

	SVM					LDA				
	本文	CSP	AR	PSD	DWT	本文	CSP	AR	PSD	DWT
LH/RH	76.13	70.39	55.57	49.72	49.88	73.42	71.80	55.46	54.98	53.71
LH/FO	78.37	72.28	58.16	49.74	50.50	77.00	77.33	54.14	58.48	56.82
LH/TO	82.13	73.40	61.37	50.35	50.14	78.41	75.10	56.99	64.14	63.34
RH/FO	82.52	75.12	58.07	50.18	50.18	79.23	75.67	55.77	59.37	57.81
RH/TO	84.03	76.26	61.13	49.54	49.54	81.11	76.32	60.93	66.20	63.44
FO/TO	76.36	64.23	54.54	50.09	50.09	74.20	67.61	52.99	57.89	58.18
Mean	79.92	71.95	58.14	49.93	50.05	77.22	73.98	56.04	60.17	58.88
运行时间/s	2.39	4.05	12.43	740.31	140.18	2.26	3.57	13.06	739.56	141.28

图 4-22　其他学者对不同特征提取算法分类准确率比对

第五章　实验心得

从寻找队友组队，到一起对题目进行分析，一起查找相关资料，一起修改代码，再到准备答辩。此次实验给我们带来了很多不一样的体验，不同于以往的个人编写系统比如 C 语言、C++的课程设计，此次实验任务需要我们小组合作，需要组内齐心协力，一起完成。在这一过程中，我们也曾遇到许多困难，一开始大家在线上讨论，积极性不是很高，但当我们开了几次线下会议后，一切都变得明朗了起来，从实验思路再到各种问题的处理，都得到了很好的解决。令我印象深刻的是，在对未拟合的数据（测试集）进行分类时，使用 DWT 和 AR 特征提取的方法得到的分类准确率只能略高于 50%，与 CSP 得到的准确率差距蛮大，但是我们并不是简单放弃这两种方案。通过组内讨论并查阅相关文献资料，我们很快发现原来这是 DWT 和 AR 这两种特征提取算法本身的原因导致的，我们的实验代码思路是没问题的。由于实验的训练集中有很多无关因子，DWT 和 AR 的提取未能很好处理无关因子与相关变量的关系，与此相对，CSP 由于是对空域滤波，所以恰好能解决这一问题。像这样的例子还有很多，但组内精诚团结，共同克服了难关，我们也就顺利地完成了实验任务。

很高兴能进行此次脑科学实验，我们收获到的不仅仅是二分类算法和脑科学的相关知识，相信此次经历能让我在今后的学习和生活中受到启迪。

参考文献

[1]白小平. 四类运动想象脑电信号特征分析与提取[D].北京理工大学,2015.

[2]Blankertz B, Müller KR. The BCI competition. III Validating alternative approaches to actual BCI problems. IEEE Trans Neural Syst Rehabil Eng. 2006 Jun;14(2)153-9. doi 10.1109TNSRE.2006.875642. PMID 16792282.

[3]尹春辉. 基于两类运动想象脑电信号的特征提取算法研究[D].南昌大学,2020.DOI:10.27232/d.cnki.gnchu.2020.003062.

[4]郭闽榕.基于运动想象的脑电信号特征提取研究[J].信息技术与网络安全,2021,40(01):62-66.DOI:10.19358/j.issn.2096-5133.2021.01.011.

[5]刘宝,蔡梦迪,薄迎春,张欣.一种基于 PSO-CSP-SVM 的运动想象脑电信号特征提取及分类算法[J].中南大学学报(自然科学版),2020,51(10):2855-2866.