Table of Contents

Loading....1
Valuable Channels for Preprocessing....1
Main....1
Splitting to validation and train....2
CSP/LDA....2
Concatinating Train_set and Valid_set....3
Predicting using WLDA's and Threshold's....3
Accuracy....4
Functions....4

Loading

```
rng(1);
sub1 = load('E:\university\Term_8\BSS\HWs\HW78\dataset\subj_1.mat');
sub1 = sub1.data;
Fs = 2400;

Data_class4 = cell2mat(sub1(4));
size_class4 = size(Data_class4,3);
Data_class3 = cell2mat(sub1(3));
size_class3 = size(Data_class3,3);
Data_class2 = cell2mat(sub1(2));
size_class2 = size(Data_class2,3);
Data_class1 = cell2mat(sub1(1));
size_class1 = size(Data_class1,3);
```

Valuable Channels for Preprocessing

```
channels_first_classifier =
[39,15,48,24,57,31,60,32,59,28,52,19,43,47,23,...
                     56,30,58,29,53,20,44,49,21,50,22,51,25,54,26,55,27];
% channels_second_classifier = 1:63;
% channels_third_classifier = 1:63;
```

Main

```
confusion_matrix_training = zeros(4,4);
confusion_matrix_validation = zeros(4,4);

for leave_one_out=1:5

    r1 = randi(size_class1,1);
    r2 = randi(size_class2,1);
    r3 = randi(size_class3,1);
    r4 = randi(size_class4,1);
```

Splitting to validation and train

```
    ValidData_class4 = Data_class4(:,:,r4);
    ValidData_class3 = Data_class3(:,:,r3);
    ValidData_class2 = Data_class2(:,:,r2);
    ValidData_class1 = Data_class1(:,:,r1);

    ValidData_all = cat(3,ValidData_class4,ValidData_class3 ...
                        ,ValidData_class2,ValidData_class1);
    ValidData_Labels = [4,3,2,1];

    TrainData_class4_trials = setdiff(1:size_class4,r4);
    TrainData_class3_trials = setdiff(1:size_class3,r3);
    TrainData_class2_trials = setdiff(1:size_class2,r2);
    TrainData_class1_trials = setdiff(1:size_class1,r1);

    TrainData_class4 = Data_class4(:,:,TrainData_class4_trials);
    TrainLabel_class4 = zeros(1,size(TrainData_class4,3)) + 4;

    TrainData_class3 = Data_class3(:,:,TrainData_class3_trials);
    TrainLabel_class3 = zeros(1,size(TrainData_class3,3)) + 3;

    TrainData_class2 = Data_class2(:,:,TrainData_class2_trials);
    TrainLabel_class2 = zeros(1,size(TrainData_class2,3)) + 2;

    TrainData_class1 = Data_class1(:,:,TrainData_class1_trials);
    TrainLabel_class1 = zeros(1,size(TrainData_class1,3)) + 1;

    TrainData_Labels = [TrainLabel_class4,TrainLabel_class3,...
                            TrainLabel_class2,TrainLabel_class1];

    TrainData_class123 =
cat(3,TrainData_class1,TrainData_class2,TrainData_class3);
    TrainData_class12 = cat(3,TrainData_class1,TrainData_class2);
```

CSP/LDA

```

    [WLDA4, c4, Wcsp4] = ...
            csp_lda(TrainData_class4,TrainData_class123, ...
            channels_first_classifier,1);

    [WLDA3, c3, Wcsp3] = ...
            csp_lda(TrainData_class3,TrainData_class12, ...
            channels_first_classifier,1);

    [WLDA2, c2, Wcsp2] = ...
            csp_lda(TrainData_class2,TrainData_class1, ...
            channels_first_classifier,1);
```

Concatinating Train_set and Valid_set

```
    TrainData_class1=preprocess(TrainData_class1,channels_first_classifier);
    TrainData_class2=preprocess(TrainData_class2,channels_first_classifier);
    TrainData_class3=preprocess(TrainData_class3,channels_first_classifier);
    TrainData_class4=preprocess(TrainData_class4,channels_first_classifier);

    ValidData_class1=preprocess(ValidData_class1,channels_first_classifier);
    ValidData_class2=preprocess(ValidData_class2,channels_first_classifier);
    ValidData_class3=preprocess(ValidData_class3,channels_first_classifier);
    ValidData_class4=preprocess(ValidData_class4,channels_first_classifier);

    train_set_all = cat(3,TrainData_class4,TrainData_class3,...
                          TrainData_class2,TrainData_class1);

    valid_set_all = cat(3,ValidData_class4,ValidData_class3,...
                          ValidData_class2,ValidData_class1);
```

Predicting using WLDA's and Threshold's

```
    prediction_training = predict(train_set_all,Wcsp4,Wcsp3,Wcsp2,...
                                  WLDA4,WLDA3,WLDA2,c4,c3,c2);
    display('injam1');
    prediction_validation = predict(valid_set_all,Wcsp4,Wcsp3,Wcsp2,...
                                  WLDA4,WLDA3,WLDA2,c4,c3,c2);
    display('injam2');
    cm_training = confusionmat(TrainData_Labels,prediction_training);
    confusion_matrix_training = confusion_matrix_training + cm_training;

    display('injam3');
    cm_validation = confusionmat(ValidData_Labels,prediction_validation);
    confusion_matrix_validation = confusion_matrix_validation +
cm_validation;
end
```

```
injam1
injam2
injam3
injam1
injam2
injam3
injam1
injam2
injam3
injam1
injam2
injam3
injam1
injam2
injam3
```

```
confusion_matrix_training
```

```
confusion_matrix_training = 4×4
    82     1     2     0
     0    79     9     2
     0     1    93     1
     0     0     0    95
```

```
confusion_matrix_validation
```

```
confusion_matrix_validation = 4×4
     0     4     1     0
     2     1     0     2
     2     0     2     1
     1     0     0     4
```

Accuracy

```
accu = trace(confusion_matrix_training)/(sum(confusion_matrix_training(:)))
```

```
accu = 0.9562
```

Functions.

```
function zm_data = zeromean(data)
data_size = size(data);
zm_data = zeros(data_size);
for k = 1:data_size(3)
    avg = mean(data(:,:,k),2);
    zm_data(:,:,k) = data(:,:,k)-avg;
end
end


function filtered_set = cspfilter(W,Data)
data_size = size(Data);
is3D = size(data_size);
if is3D(2) == 3
    d3 = data_size(3);
    for k = 1:d3
    filtered = W*Data(:,:,k);
    avg = mean(filtered,2);
    variance = sum((filtered-avg).^2,2)/data_size(2);
    filtered_set(k,:) = variance.';
    end
else
    filtered = W*Data;
    avg = mean(filtered,2);
    variance = sum((filtered-avg).^2,2)/data_size(2);
    filtered_set = variance.';
end
end
```

```
function label = predict(Data,Wcsp4,Wcsp3,Wcsp2,...
                            WLDA4,WLDA3,WLDA2,c4,c3,c2)
Data_size = size(Data,3);
label = zeros(1,Data_size);
for i = 1:Data_size
    test4 = cspfilter(Wcsp4, Data);
    X4 = test4(i,:).';
    if  WLDA4.'*X4 > c4
        label(i) = 4;
    else
        %temp3 = Filter(Data,2,7);
        test3 = cspfilter(Wcsp3, Data);
        X3 = test3(i,:).';
        if WLDA3.'*X3 < c3
            label(i) = 3;
        else
            %temp2 = Filter(Data,1/3,3);
            test2 = cspfilter(Wcsp2, Data);
            X2 = test2(i,:).';
            if WLDA2.'*X2 < c2
                label(i) = 2;
            else
                label(i) = 1;
            end
        end
    end
end
end

function  [WLDA, c, W_valuable_channels_csp] = ...
                csp_lda(TrainData_class0,TrainData_class1, ...
                        channels,prep_on)
%Implementing the csp algorithm
    if prep_on
        TrainData_class0=preprocess(TrainData_class0,channels);
        TrainData_class1=preprocess(TrainData_class1,channels);
    end


    trainC0_size = size(TrainData_class0);
    trainC1_size = size(TrainData_class1);

    Rx0 = zeros(trainC0_size(1),trainC0_size(1));
    Rx1 = zeros(trainC1_size(1),trainC1_size(1));

%For loop over each trial %Class1
    for k = 1:trainC0_size(3)
        X = TrainData_class0(:,:,k);
```

```
        Rx0 = Rx0+X*X.';
    end

%For loop over each trial %Class2
    for k = 1:trainC1_size(3)
        X = TrainData_class1(:,:,k);
        Rx1 = Rx1+X*X.';
    end

    Rx0 = (1/trainC0_size(3))*Rx0;
    Rx1 = (1/trainC1_size(3))*Rx1;

    [W,lambda] = eig(Rx0,Rx1);
    [~,series] = sort(diag(lambda),'descend');

%Sorting from the highest eigenvalue to the lowest
    W = W(:,series);

% Normalizing
    for i=1:size(W,2)
    W(:,i) = W(:,i)/norm(W(:,i));
    end

%Implementing the LDA algorithm
    W_valuable_channels_csp = [W(:,1:10) W(:,21:30)].';
    train_set0 = cspfilter(W_valuable_channels_csp,TrainData_class0);
    train_set1 = cspfilter(W_valuable_channels_csp,TrainData_class1);

    mean0 = mean(train_set0).';
    mean1 = mean(train_set1).';

    cov0 = cov(train_set0);
    cov1 = cov(train_set1);

    [WLDA,landa] = eig((mean0-mean1)*(mean0-mean1).',cov0+cov1);

%Sorting from the highest eigenvalue to the lowest
    [~,seriesLDA] = sort(diag(landa),'descend');
    WLDA = WLDA(:,seriesLDA(1));

%Normalizing WLDA
    WLDA = WLDA/norm(WLDA);
%Threshold
    c = (WLDA.'*mean0 + WLDA.'*mean1)/2;

end


function Data = preprocess(Data,channels)
%Preprocessing /// valuable channels
```

```
Data = Data(channels,:,:);

%Preprocessing /// bandpass filteration
%Data_Allclasses = Filter(Data_Allclasses,20,40);
end

% function [x_fil] = Filter(x,bf)
%     x_fil = zeros(size(x));
%     L0 = size(x,2)/2;
%     Xfil = zeros(1,2*L0);
%     for i = 1:size(x,3)
%         for j = 1:size(x,1)
%             F_x = fftshift(fft(x(j,:,i)));
%             Xfil(L0-bf(2)*3+1:L0-bf(1)*3) = F_x(L0-bf(2)*3+1:L0-bf(1)*3);
%             Xfil(L0+bf(1)*3+1:L0+bf(2)*3) = F_x(L0+bf(1)*3+1:L0+bf(2)*3);
%             x_fil(j,:,i) = ifft(ifftshift(Xfil));
%         end
%     end
% end
%

% function x_filtered = Filter(x,bf)
%     Fs = 2400;
%     temp = zeros(size(x));
%     for i = 1:size(x,3)
%         for j = 1:size(x,1)
%             temp(j,:,i) = bandpass(x(j,:,i),[20,40],Fs);
%         end
%     end
%     x_filtered = temp;
% end

function x_filtered = Filter(x,f1,f2)
fs = 2400;
filterorder = 5;
[z,p,k] = butter(filterorder,[f1/(fs/2) f2/(fs/2)],'bandpass');
sos = zp2sos(z,p,k);
temp = zeros(size(x));
     for i = 1:size(x,3)
         for j = 1:size(x,1)
            temp(j,:,i)  = sosfilt(sos,x(j,:,i));
         end
     end
x_filtered = temp;
end
```