```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Seminar HCI and BCI in practice
%
% Session 7
%
% Evaluation
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% In this session the performance of the SVM classifier will be evaluated.
% You need to add the spider toolbox to your Matlab path.
% Load data files to workspace
clear all

% Load data files to workspace

load ecogStruct3.mat
load epoch2.mat
%% 1. Define the dataset for the classification based on previously found
% features (use your results from Session 3)

% ---------------------------DEFINE DATASET--------------------------------

% freqBand = [4:58 62:118 122:178]; % you can change the values here, based
on your results from Session 3 (t-value plot or relief algorithm plot)
freqBand = [62:118 122:178]; % just for faster testing
freqIdx = unique(nearly(freqBand,ecog.periodogram.centerFrequency));
nFreq = length(freqIdx);
nTrials = size(ecog.periodogram.periodogram,3);
% chan = ecog.selectedChannels; % you can change the values here, based on
your results from Session 3 (t-value plot or relief algorithm plot)
chan = [17 23 39];
nChan = length(chan);
dat = ecog.periodogram.periodogram(freqIdx,chan,:);
dat = reshape(dat,nFreq,nChan*nTrials);
dat = zscore(dat,0,2); % z-score data
dat = reshape(dat,nFreq,nChan,nTrials); % Reshape data
dat = permute(dat,[3 2 1]);
dat = reshape(dat,nTrials,nFreq*nChan);

%--------------------------------------------------------------------------


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% GRADED TASKS
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% 2.1 Classification error (0,25 P)

% ---------------------------REPEATED CLASSIFICATION-----------------------
```

```
% Cross validation steps



CV = 10;
N = 10;
accuracy_svm = [];

x = 5; % performing the same classification x times (you can change this
value)

for ii = 1:1:x
    realClassLabels = epoch.label; % Class labels
    predictedClassLabels = zeros(1,length(realClassLabels)); %preallocate
    selector =
ceil((1:length(realClassLabels))/(length(realClassLabels)/CV));
    selector = selector(randperm(length(realClassLabels)));
    complete_weights = [];

    for k = 1:CV %N


        fprintf('CV step #%i\n',k);
        testIdx = find(selector == k);
        trainIdx = setdiff(1:length(realClassLabels),testIdx);
        curTrain = dat(trainIdx,:);
        curClassLabels = realClassLabels(trainIdx);
        curTest = dat(testIdx,:);
        testClassLabels = realClassLabels(testIdx);

        % ------------------SET C-------------------------------------------
------

        optimizeC = 1; % optimized C (but 0 if not wanted)

        % ----------------SVM CLASSIFICATION---------------------------------
------

        if optimizeC == 1 % Cost parameter C iteratively optimized
            %         Determine C for each CV according to Joachims
            defaultC = ecogGetDefC(curTrain);
            %         Define starting C value based on previously computed
defaultC
            c_h(1) = defaultC + (1/3)*defaultC;
            c_l(1) = defaultC - (1/3)*defaultC;
            for i = 2:15
                c_h(i) = c_h(i-1) + (1/3)*c_h(i-1);
                c_l(i) = c_l(i-1) - (1/3)*c_l(i-1);
            end
            iterative_c(1:15) = c_l(end:-1:1);
            iterative_c(16) = defaultC;
            iterative_c(17:31) = c_h;
            res_cOpt = [];
```

```
            for m = 1:31
                % Train SVM
                R =
fitcsvm(curTrain,curClassLabels,'BoxConstraint',iterative_c(m));
%train_svm(curTrain,curClassLabels,iterative_c(m));

                weights{m} = R.Beta; %save weights
                bias{m} = R.Bias;
                Res_fit = predict(R,curTest);
                res_cOpt(m,:) = Res_fit; % Result according to each c value
                optAcc(m,:) = sum( res_cOpt(m,:) == testClassLabels ) /
length( testClassLabels ); % accuracy for each c value
                C_plus_W(m,:) = [iterative_c(m) norm(R.W)];

            end

            [max_val, max_idx] = max(optAcc); %max index for iterative c
            best_50C(ii,k) = iterative_c(max_idx);
            Res.prediction = res_cOpt(max_idx,:);
            max_indicies(ii,k) = max_idx;%save max index of CV
        else % without iteratively optimizing C
            % Default C
            defaultC = ecogGetDefC(curTrain);
            R = fitcsvm(curTrain,curClassLabels,'BoxConstraint',defaultC);
%train_svm(curTrain,curClassLabels,defaultC); % Train //
fitcsvm(curTrain,curClassLabels,'BoxConstraint',defaultC)
            %         R_fit =
fitcsvm(curTrain,curClassLabels,'BoxConstraint',1);

            %Res = test_svm(R,curTest); % Test // predict(R,curTest)
            Res_fit = predict(R,curTest);
            Res.prediction = Res_fit;

        end
        best_cv(k) = optAcc(max_idx,:); %cross validation test results saving
        bias_best(k) = bias{max_idx};
        predictedClassLabels(testIdx) = Res.prediction;
        complete_weights(k,:) = weights{max_idx};

    end

    % --------------------CALCULATE OVERALL ACCURACY-------------------------
--
    %collect weights form different CVs
    cv_acc_all{ii} = best_cv;
    bias_all{ii} = bias_best;
    weights_all{ii} = complete_weights;
    accuracy_svm(ii) = sum(predictedClassLabels == realClassLabels) /
length(realClassLabels)
    [val idx] = max(best_cv); %best cv index
    [test_ROC] =
plot_ROC(dat,realClassLabels,complete_weights(idx,:),bias_all{ii}(idx),'true'
);
```

```
end



% --------------------------EXPLAIN OUTPUTS--------------------------------
% 2.1.1 What information do these variables give you? (0,25 P)

%accuracy_svm

%Res_fit




% --------------------------PLOT ACCURACIES-----------------------

% Plot mean and standard error of accuracy across X repetitive
classifications
figure;
errorbar(mean(accuracy_svm),std(accuracy_svm)/sqrt(length(accuracy_svm)),'xr'
)


%--------------------------------------------------------------------------

%% 3. Get best C value by averaging across all results

% ---------------------------GET BEST C------------------------------------

%bring best_50C into 1d
best_50C = best_50C(:);
bestC = median(best_50C);
% Plot histogram
figure; hist(best_50C);

%--------------------------------------------------------------------------

%% 2.2 SVM weights (0,5 P)

%--------------------------FIND BEST CROSS VALIDATION----------------------

% 2.2.1 Find best CV-Step Code (0,25 P):

best_classification = 8; % Choose the x. classification (containing the
cross-validation step with the highest accuracy)
best_cv = 5; % Choose the best cross-validation step form cv_acc_all based on
best classificatoin

best_w = weights_all{best_classification}(best_cv,:);

%--------------------------PLOT BEST W VECTOR------------------------------
%  2.2.2 Plot the weights for best result (0,125 P):
```

```matlab
plotSvmWeights(abs(best_w),[],nFreq,nChan,freqBand,chan)% FIRST PLOT (BEST W)

%--------------------------PLOT AVERAGE W VECTOR---------------------------
% 2.2.3 Plot AVERAGE w vector (0,125 P):

% Average W vector
weights_all_matrix = [];
for ii = 1:1:x
    weights_all_matrix = [weights_all_matrix ; weights_all{ii}]
end

avW = mean(abs(weights_all_matrix));

% Plot the average weights
plotSvmWeights(avW,[],nFreq,nChan,freqBand,chan) % SECOND PLOT (AVERAGE W)

%-------------------------EXPLAIN PLOT DIFFERENCES-------------------------

% Explain what you see in this plot?




%--------------------------------------------------------------------------

%% 2.3 Area under curve (ROC: receiver operation characteristics) - (1 P)

% questions in regards to plots created by the function plot_ROC above

% What does the first plot (distances to the hyperplane) portray. How would
% this plot ideally look like?
% What information can you get from the second plot? What is a ROC? And
% what does the Area under the ROC mean? How would this plot ideally look
like?




%--------------------------------------------------------------------------

%% 2.4 Estimation of the chance level (0,25 P) (works in matlab 2014)

%-----------------------Estimating Chance Level & CV-------------------------
--------

% we want to bootstrap the data
num_estimates = 40; % #repetitions/estimates


for ii = 1:1:num_estimates
    realClassLabels = epoch.label; % Class labels
```

```
    predictedClassLabels = zeros(1,length(realClassLabels)); %preallocate
    selector =
ceil((1:length(realClassLabels))/(length(realClassLabels)/CV));
    selector = selector(randperm(length(realClassLabels)));
    complete_weights = [];

    for k = 1:CV %N


        fprintf('CV step #%i\n',k);
        testIdx = find(selector == k);
        trainIdx = setdiff(1:length(realClassLabels),testIdx);
        curTrain = dat(trainIdx,:);
        curClassLabels = realClassLabels(trainIdx);
        curTest = dat(testIdx,:);
        testClassLabels = realClassLabels(testIdx);




        % Chance level estimation by randomizing the training data
        bsTable=[];
        fingerFlex = find(curClassLabels' == 20); % flexion
        fingerExtend = find(curClassLabels' == 21); % extension


        b = randperm(length(curClassLabels')); %randomising data
        classOneSamples = b(1:length(fingerFlex));
        classTwoSamples = b(length(fingerFlex)+1:end);
        bsTable(fingerFlex) =
classOneSamples(ceil(rand(length(fingerFlex),1)*length(fingerFlex)));
        bsTable(fingerExtend) =
classTwoSamples(ceil(rand(length(fingerExtend),1)*length(fingerExtend)));

        curTrain = curTrain(bsTable,:);

        R = fitcsvm(curTrain,curClassLabels,'BoxConstraint',bestC);
%train_svm(curTrain,curClassLabels,defaultC); % Train //
fitcsvm(curTrain,curClassLabels,'BoxConstraint',defaultC)
        %         R_fit = fitcsvm(curTrain,curClassLabels,'BoxConstraint',1);

        %Res = test_svm(R,curTest); % Test // predict(R,curTest)
        Res_fit = predict(R,curTest);
        predictedClassLabels(testIdx) = Res_fit;


    end

    accuracy_CL(ii) = sum(predictedClassLabels == realClassLabels) /
length(realClassLabels); %CL - chance level estimates
end
%-----------------------PLOT RESULTS---------------------------------------

%-Plot results with standard error
figure;
```

```
errorbar(mean(accuracy_svm),1.96*std(accuracy_svm),'xr') % These are the
accuracy values from the first part of the Session (in red)
hold on
errorbar(mean(accuracy_CL),1.96*std(accuracy_CL),'xb') % These are the
accuracy values from the bootstrapping (in blue)
xlim([0.97 1.03])
ylim([0.35 0.9])



%-----------------------ANSWER QUESTIONS----------------------------------
% 2.4 How is the chance level estimated and for what do you use it here?
% Why is this an estimation of the chance level? (0,25 P):
```