PSYCH 734/CSE 734

Final Project

Sara Jamil
001143947
jamils2@mcmaster.ca
1/4/2017

Table of Contents

Introduction

The purpose of this project is to examine the methods used in training models to classify electroencephalography (EEG) data. EEG is a method of recording the electrophysiological activity in the brain. The EEG data used in this project was collected from participants as they watched various videos and rated their emotional response to each video. The goal is to build models which can determine a subjects overall reaction to a video from their EEG recording.

The EEG data provided consists of EEG recorded during 364 videos collected from 30 participants. Their responses to the videos were also given as a vector of two class labels consisting of either a positive overall reaction (label of 1) or negative overall reaction (label of 0) to the video. This metric was obtained by computing whether the average of the ratings for positive emotions was greater than the average for negative emotions. The information containing which EEG recording belonged to which participant was also given in order to perform different methods of classification.

There were three different methods for obtaining the classification model from the EEG data. The first task was to obtain a model using leave-one-subject-out cross-validation. The second task was to train a model that performs within-subject classification in which the model is trained and tested on data corresponding to a single participant. The final task was to obtain a model that is trained and tested on all the data to perform subject-irrespective classification. The methods used in obtaining the models as well as their accuracy rates are explained.

Task 1: Leave-One-Out Subject Classification

Methods

This task involves training a model using the leave-one-subject-out cross-validation method. In order to perform this classification, an adequate classifier must be chosen. Three different classifiers were tested in order to determine which type of classifier performed the best in terms of classification accuracy as well as computation time required for training.

The different classifiers that were tested included two Support Vector Machine (SVM) classifiers and one backpropagation neural network. The SVM classifiers were obtained from built-in Matlab functions *svmtrain* and *fitcsvm* which perform in similar ways. Both used sequential minimal optimization (SMO) as a soft-margin classifier to solve the classification problem and both used a radial basis function (Gaussian) kernel to find the nonlinear separating hyperplane.

The neural network classifier consisted of a simple 2-layer backpropagation network. The classifier was obtained from a Matlab built-in function called *train* that trains a specified neural network using one of a variety of backpropagation techniques. The method used in this case was trained using gradient descent on a neural network consisting of 10 hidden units. The model continues training until it

reaches a default value for the error or it reaches the maximum number of epochs which was specified at 10000. These parameters were chosen after several trials of various parameter values to obtain a network that performed adequately in comparison to the SVM classifiers.

The performance of the three models was assessed by recording the classification accuracy as well as the computation time required for training each of the models. The accuracy was calculated by testing the models on the test data and comparing it to the actual output values to determine the percentage of correctly classified data. This was done in order to determine which method for obtaining a model was best to use for the subsequent tasks.

The input to these models was obtained using the *FBCSP* code provided that computes Filter-Bank Common Spatial Patterns as features from the EEG data. It is common to use a feature space as input to the supervised machine learning model rather than the raw data to improve its performance. This feature space ideally includes the most relevant information to the task of classifying between different output classes.

The cross-validation method used in training and testing the data was leave-one-subject-out cross-validation, as previously mentioned. This method involves selecting the testing data as all of the data pertaining to one particular subject and excluding this test data while training on the rest of the participants' data. This means that a general model is generated based off of all participants and testing by classifying one of the participant's data which is treated as new and previously unseen data. This assesses how well the classifier is able to generalize with respect to each subject.

Results

The following table shows the classification accuracy for each participant with respect to all three models. Model 1 was trained using *svmclassify*, Model 2 was trained using *fitcsvm*, and Model 3 was trained using the neural network function *train*. The accuracies are shown as a percentage of correctly classified data.

The following figure shows the average accuracy rate with respect to each model. This value is a percentage of correctly classified data and the number in brackets shows the standard deviation.

Table 1: Task 1 Accuracy (in percent)

Task 1: LOO Subject Classification Accuracy

Participant	Model 1	Model 2	Model 3
1	100.00	100.00	100.00
2	50.00	50.00	50.00
3	85.71	85.71	85.71
4	84.62	76.92	76.92
5	60.00	60.00	60.00
6	90.91	63.64	45.45
7	66.67	66.67	44.44
8	60.00	60.00	60.00
9	100.00	100.00	75.00
10	0.00	0.00	50.00
11	80.00	80.00	80.00
12	61.54	61.54	61.54
13	71.43	76.19	76.19
14	62.50	56.25	62.50
15	73.33	73.33	60.00
16	58.82	58.82	58.82
17	52.63	52.63	47.37
18	70.00	70.00	70.00
19	55.00	50.00	55.00
20	68.75	62.50	68.75
21	100.00	100.00	100.00
22	75.00	75.00	50.00
23	85.71	85.71	85.71
24	65.00	60.00	55.00
25	60.00	53.33	60.00
26	60.00	60.00	60.00
27	75.00	83.33	91.67
28	100.00	100.00	50.00
29	72.73	72.73	72.73
30	89.47	89.47	84.21

TASK 1 MODEL 1 ACCURACY: 71.16 % (20.147)

TASK 1 MODEL 2 ACCURACY: 69.46 % (20.3798)

TASK 1 MODEL 3 ACCURACY: 66.57 % (15.9769)

Figure 1: Average Accuracy

The computation time required for training each of the models for each participant is shown in the figure below. The average computation time with respect to each model is also shown below.

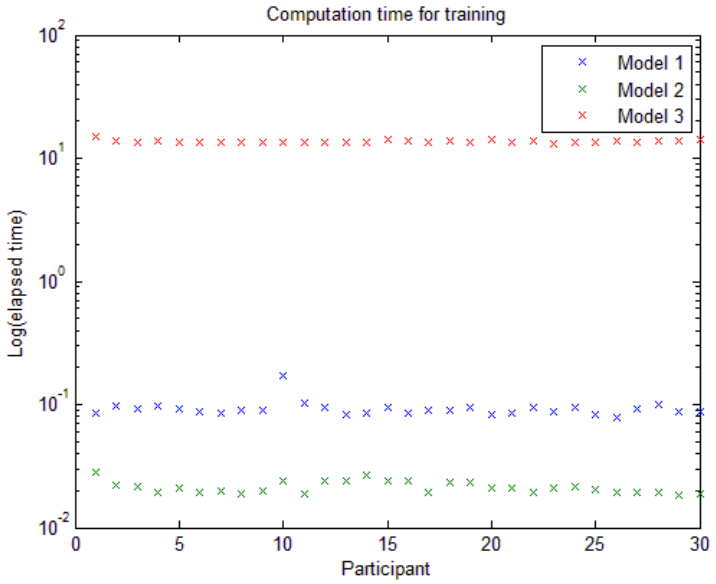

Figure 2: Computation time

```
TASK 1 AVERAGE COMPUTATION TIME:
-----
 MODEL 1: 0.093 sec
 MODEL 2: 0.021 sec
 MODEL 3: 13.638 sec
```

Figure 3: Average Computation time

It is clear that the computation time required for training the neural network model is significantly greater than that required for either of the SVM models. The average accuracy values were very similar for the SVM models and slightly worse for the neural network model. The model chosen for the rest of the subsequent tasks was chosen to be the first model trained with *svmclassify* since it provided the best classification rate while requiring very little computation time.

It is interesting to note that the standard deviation of the classification is quite large for all of the models. There is a lot of variance between in the performance of the classifier between each subject. Some were able to get 100% accuracy while other got 0%, which represents the largest range possible.

Task 2: Within-Subject Classification

Methods

This task involves performing within-subject classification using an appropriate cross-validation method and classifier. This means that for each subject, the classification accuracy is obtained when testing and training on the data belonging to that one particular subject. The cross-validation method used was leave-one-out cross-validation and the classifier was chosen to be one of the SVM classifiers that had the best performance, as shown above.

The leave-one-out cross-validation method leaves one of the samples belonging to a participant out for testing, while training on the rest of their data. Each of the participant's samples is left out and the accuracy was determined by averaging the accuracy obtained for all of these tests. This method was used because of the limited amount of data available to train on for some of the subjects. Some of the participants had very few samples to work with, so the only way to calculate the accuracy was by using the leave-one-out cross-validation method.

It is important to note that there were some exceptions where no classification accuracy could be determined for a particular subject. If all of the data pertaining to a particular subject belonged to only one of the two classes, it was not possible to train the classifier. Without samples available from both classes, the model cannot learn anything, and thus the participant must be skipped. This exception also came up in the process of cross-validation. If after excluding one of the samples, all the remaining data samples for training belonged to only one class, then it was not possible to train that model and it had to be skipped.

Once the within-subject classification accuracy was obtained for all the participants with viable data, the average accuracy was taken for all the participants. This was done in order to assess the performance of within-subject classification in comparison to the other classification methods in tasks 1 and 3.

Results

The following table shows the within-subject classification accuracy (in percent) for each of the participants. Note that some of the participants were omitted for the reasons described above. The average accuracy is shown below with the mean and standard deviation.

Table 2: Task 2 Accuracy

Task 2: Within-Subject Classification

Participant	Model Accuracy
2	12.50
3	85.71
4	84.62
5	40.00
6	100.00
7	66.67
8	60.00
11	100.00
12	61.54
13	76.19
14	62.50
15	73.33
16	58.82
17	47.37
18	70.00
19	55.00
20	75.00
22	100.00
23	85.71
24	55.00
25	20.00
26	40.00
27	100.00
29	72.73
30	84.21

TASK 2 ACCURACY: 67.48 % (23.5976)

Figure 4: Task 2 Average Accuracy

The total number of participants used in this classification task was 25. Note that the variance is also quite large for this task as well. The average accuracy rate slightly underperformed in this task as compared to the leave-one-subject-out classification method.

Task 3: Subject-Irrespective Classification

Methods

This task involves obtaining subject-irrespective classification from the data. If all the data is treated the same (i.e. without consideration as to whom the data belongs), this gives a more general classification accuracy rate.

In this task, there are two methods of cross-validation that could be considered appropriate for measuring accuracy: k-fold or stochastic cross-validation. Either of these methods can be used since there is an adequate number of training samples to split the data into specified proportions rather than using the leave-one-out method due to lack of data. Both of these methods were implemented in order to examine the difference in classification accuracy between them. Also, the function *cvpartition* was used in both cases to partition the data into training and test sets.

For the k-fold cross-validation method, k was chosen to be 10. The data was randomly partitioned onto 10 mutually exclusive subsets that are approximately equal in size. One of these partitions is used as the test set while the rest are used in training. This is repeated 10 times, using each of these partitions as a test set, and the classification accuracy is taken as the average accuracy of all 10 iterations.

For the stochastic cross-validation method, a random x portion of the data is selected as the test set. This is repeated n times and the accuracy values are averaged in order to obtain the overall classification accuracy. The proportion chosen was $x = 1/10$ (i.e. similar to 10-fold cross-validation) and the process was repeated $n = 20$ times. The *cvpartition* method of *HoldOut* was used in order to select the random test data portion.

Results

The following results show the k-fold cross validation of the subject-irrespective classification task. The accuracy for each of the 10 folds was recorded in the table below. The average of the 10-fold cross-validation is shown in the figure below.

Table 3: Task 3 K-fold CV Accuracy

```
Task 3: Subject-Irrsepective Classification
---------------

----- K-fold cross-validation

K-fold partition     Model Accuracy
-----------------------------------------
1                        69.44
2                        72.97
3                        64.86
4                        70.27
5                        75.68
6                        72.22
7                        72.22
8                        61.11
9                        66.67
10                       77.78



-----
 TASK 3 ACCURACY: 70.32 % (5.02499)
-----
```

Figure 5: Task 3 K-fold Accuracy

The following results show the stochastic cross-validation of the subject-irrespective classification task. The accuracy for each of the 20 holdout tests was recorded in the table below. The average of accuracy is given in the figure below.

Table 4: Task 3 Stochastic CV Accuracy

```
Task 3: Subject-Irrsepective Classification
---------------

----- Stochastic cross-validation

Holdout test     Model Accuracy
-----------------------------------------
1                    66.67
2                    66.67
3                    77.78
4                    75.00
5                    75.00
6                    69.44
7                    66.67
8                    77.78
9                    72.22
10                   72.22
11                   69.44
12                   75.00
13                   72.22
14                   77.78
15                   66.67
16                   72.22
17                   69.44
18                   72.22
19                   69.44
20                   69.44



-----
 TASK 3 ACCURACY: 71.67 % (3.78087)
-----
```

Figure 6: Task 3 Stochastic Accuracy

Note that the variance in both these cross-validation methods is significantly less than that of the first two tasks. The classifiers performed much more consistently for each trial when the subject variance was removed. Although the stochastic method may provide a more accurate result of the generalization abilities of the model due to random sampling in each iteration, this is dependent on the number of iterations. Ideally, a large number of iterations would be used, but this requires more computation time. K-fold is a popular method for assessing the cross-validation accuracy of a model and has the advantage of only requiring 10 runs to give an accurate depiction of generalization ability.

Discussion

Comparing the results for Task 1 and Task 2 shows a similar overall accuracy of about 70% with the latter performing slightly worse. In both tasks, some participants yield lower classification than others. In the leave-one-subject-out classification used in task 1, some participants had very low classification accuracies meaning that the model does a poor job at generalizing to that particular participant. In the within-subject classification used in task 2, some participants had low classification accuracies meaning that their own data wasn't able to consistently predict their reaction.

These low accuracies may be due to several factors, but one of the main factors may involve the way that the data was collected. Some of the participants had more data samples to work with. Also, the original EEG data consisted of varying lengths of EEG recordings which could introduce some variance as it may affect the features extracted. Another possible reason for the low accuracies is that the features extracted for those particular subjects do not accurately capture the differences in the two classes. The process assumes that the features chosen were relevant to the classification task for all participants. Thus, using feature selection techniques (such as minimum redundancy, maximum relevance) may provide better classification accuracies for the within-subject cases, if the available features are adequate for that participant. As seen in the first task, some participants have different accuracies when trained using different classifiers. However, it is difficult to assume that certain subjects will always perform better using one particular classifier. Any type of classifier should be able to classify the data, with the parameters being adjusted to optimize for performance.

Comparing all three tasks shows that there are slight differences between the accuracies. Although the average accuracy values did not change significantly (they were all around 70% accurate), there were large differences in variance between the first two tasks and the third task. The first task had a high variance because for some participants the classifier performs really well or really poorly depending on how different that subject is from the rest of the group. The second task also had a high variance but this was likely due to the varying amount of data that a particular subject had to work with. Some participants had very little data to train and test on, so the model may not be adequate if trained on little data as it may lead to overfitting. The third task had the smallest amount of variance because the differences between subjects were ignored (subject-irrespective classification). The classifier performed similarly overall between the different tests because the subject related variance was omitted in the cross-validation method.

Other learning models could have been used in this project which may lead to different performance. Although neural networks were evaluated in the first task, this project mainly focuses on classification using an SVM model. As seen in the first task, the results differ between the SVM and neural network models. There are many parameters to change in the neural network model, especially when considering deep neural networks. In order to obtain better results, one must try to find the best network that optimizes performance with respect to the number of hidden layers and their respective number of hidden units, the optimization method used in backpropagation, and various other parameters regarding the stopping criteria for training. It is possible that a neural network with specific

parameters would be able to outperform the SVM models used in this project, however from the tests in task 1 it is apparent that it is difficult to obtain a better classifier.

There are several ways in which the classification accuracy may be improved. First method to try would be to use different features. Feature extraction can be considered one of the most important parts in determining the accuracy. One must try to identify the features that best explain the differences between the classes (i.e. using various time/frequency signal processing analysis techniques). Another improvement may be to use different feature selection techniques. An example of feature selection would be an algorithm that chooses a subset of features that are maximally relevant and minimally redundant (mRMR). This may improve the accuracy or may at least provide performance improvement by reducing computation time. Another method that may improve classification is to consider changing the output classes. Classification is highly dependent on what is chosen for the different classes. In this case it was the average of positive and negative emotions transformed into a binary classification. However, this does not take into consideration how positive or negative the experience was (i.e. the magnitude). Consider incorporating this information and using a regression technique. The output used in this project assumes that the grouping of positive and negative emotions is adequate for representing their overall reaction. However, the actual experience may be more complex and not represented well using this scheme.

Appendix

FinalProjectCode.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Here's your starter code for your final project. Fill in the sections
% indicated by the comments. Make sure I can check all of your results by
% running this m-file.
%
% Good luck!
% Kiret
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %
% % PSYCH/CSE 734 Project
% % Sara Jamil
% %
% % Please run the code one section at a time
% %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% TASK 1: LOO subject classification
fprintf('\n---------------\nTask 1: LOO Subject Classification\n-------------
--\n');

prelimSetup % Preliminary Setup

% Some arrays in which to store data
Accuracy = zeros(nSubj,3);  % The classification accuracy for each subject
elt = zeros(nSubj,3);   % Computation time for training

for i = 1:nSubj
    fprintf('Obtaining Results for Participant %d/%d\n',i,nSubj);

    % Partition training and test set so the test set is all the trials for
    % participant i and the training set is all trials for the other
    % participants put together
    trainidx = SubjectID~=i;
    testidx = SubjectID==i;

    % Compute FBCSP features (see 'help FBCSPfeatures' for details on
    % inputs and outputs). FBCSPfeatures.m is already set up to construct
    % the CSP filters using only the training data and to give you the
    % features for the training data and the test data together.
    [EEGcsp,Features,cspftrs,Psel] = FBCSPfeatures(EEG,Y,trainidx,opt);

    % For input into SVMTRAIN
    Features_train = Features(:,trainidx)';
    Features_test = Features(:,testidx)';
    Y_train = Y(1,trainidx)';
    Y_test = Y(1,testidx)';

    % If you wish to use feature selection, that should go here

    % Classification - put your classifier here (get Yhat from your
```

```
    % classifier). If your classifier instead just outputs a classification
    % accuracy, you can remove the variables 'Yhat' and 'Correct' and fill
    % the elements of Accuracy directly from your classifier function

    % Training SVM Classifiers
    tic;
    model1 =
svmtrain(Features_train,Y_train,'kernel_function','rbf','method','SMO');
    elt(i,1) = toc;
    tic;
    model2 =
fitcsvm(Features_train,Y_train,'KernelFunction','gaussian','Solver','SMO','Ve
rbose',0,'CrossVal','off');
    elt(i,2) = toc;

    % Training neural network
    net = feedforwardnet(10,'traingd');
    net.divideParam.trainRatio = 1;
    net.divideParam.valRatio = 0;
    net.divideParam.testRatio = 0;
    net.trainParam.epochs = 10000;
    net.trainParam.showWindow = 0;
    tic;
    [net,tr] = train(net,Features_train',Y_train');
    elt(i,3) = toc;


    % Compile results
    C1 = svmclassify(model1,Features_test);
    acc1 = Y_test==C1;  % number of correct classifications
    Accuracy(i,1) = 100*(mean(acc1));     % Plug in the accuracy you get from
your classifier here
    C2 = predict(model2,Features_test);
    acc2 = Y_test==C2;
    Accuracy(i,2) = 100*(mean(acc2));
    C3 = net(Features_test');
    C3(C3>=0.5) = 1; C3(C3<0.5) = 0; C3 = C3';
    acc3 = Y_test==C3;
    Accuracy(i,3) = 100*(mean(acc3));

    fprintf('----MODEL 1 Accuracy: %3.2f %%\n',Accuracy(i,1)); % You could
display your accuracy here in case you want to see it as the code runs
    fprintf('----MODEL 2 Accuracy: %3.2f %%\n',Accuracy(i,2)); % You could
display your accuracy here in case you want to see it as the code runs
    fprintf('----MODEL 3 Accuracy: %3.2f %%\n',Accuracy(i,3)); % You could
display your accuracy here in case you want to see it as the code runs
end

% Compute the mean and std accuracy across all participants
Task1_meanAccuracy = mean(Accuracy);
Task1_stdAccuracy = std(Accuracy);

fprintf('-----\n TASK 1 MODEL 1 ACCURACY: %3.2f %% (%g)\n-----
\n',Task1_meanAccuracy(1),Task1_stdAccuracy(1));
```

```
fprintf('-----\n TASK 1 MODEL 2 ACCURACY: %3.2f %% (%g)\n-----
\n',Task1_meanAccuracy(2),Task1_stdAccuracy(2));
fprintf('-----\n TASK 1 MODEL 3 ACCURACY: %3.2f %% (%g)\n-----
\n',Task1_meanAccuracy(3),Task1_stdAccuracy(3));

%% TASK 2: Within-Subject Classification
fprintf('\n---------------\nTask 2: Within-Subject Classification\n----------
-----\n');

prelimSetup % Preliminary Setup

Accuracy = [];  % The classification accuracy for each subject

for i = 1:nSubj
    fprintf('Obtaining Results for Participant %d/%d\n',i,nSubj);

    % Choosing the participant to obtain training and test data
    partidx = SubjectID==i;
    nTrials_p = sum(partidx);
    EEGp = EEG(1,partidx);
    Yp = Y(1,partidx);
    tempidx = true(1,nTrials_p);

    if (length(unique(Yp))<2)
        % Don't train on this participant
        % because all of the EEG data belongs to one of the two classes
        fprintf('----Skipping this participant\n');
    else
        % Compute FBCSP features for training and test data
        % of participant i
        [EEGcsp,Features,cspftrs,Psel] = FBCSPfeatures(EEGp,Yp,tempidx,opt);

        acc = [];

        for j = 1:nTrials_p
            % Creating training and test indices
            t = tempidx; t(j) = false;
            trainidx = t;
            testidx = ~t;

            % For input into SVMTRAIN
            Features_train = Features(:,trainidx)';
            Features_test = Features(:,testidx)';
            Y_train = Yp(1,trainidx)';
            Y_test = Yp(1,testidx)';

            if (length(unique(Y_train))<2)
                % Don't train on this trial
                % because training set contains only one class
            else
                % Classification
                model =
svmtrain(Features_train,Y_train,'kernel_function','rbf','method','SMO');

                % Compile results for each test
```

```
                C = svmclassify(model,Features_test);
                acc = [acc; Y_test==C];
            end
        end

        % Compile results
        Accuracy = [Accuracy; i, 100*mean(acc)]; %include participant #
        fprintf('----Accuracy: %3.2f %%\n',Accuracy(end));
    end

end

% Compute the mean and std accuracy across all participants
Task2_meanAccuracy = mean(Accuracy);
Task2_stdAccuracy = std(Accuracy);

fprintf('-----\n TASK 2 ACCURACY: %3.2f %% (%g)\n-----
\n',Task2_meanAccuracy(2),Task2_stdAccuracy(2));

%% TASK 3: Subject-Irrespective Classification
fprintf('\n---------------\nTask 3: Subject-Irrsepective Classification\n----
-----------\n');

prelimSetup % Preliminary Setup

%%%% K-fold cross-validation
fprintf('\n----- K-fold cross-validation \n \n');

% Choosing the training and testing data
% 10-fold cross-validation
CVO = cvpartition(Y','Kfold',10);

% Compute all FBCSP features for training and test data
tempidx = true(1,nTrials); %precompute features (all data)
[EEGcsp,Features,cspftrs,Psel] = FBCSPfeatures(EEG,Y,tempidx,opt);

Accuracy = zeros(CVO.NumTestSets,1);  % The classification accuracy for each
partition

for i = 1:CVO.NumTestSets
    fprintf('Obtaining Results for K-fold partition
%d/%d\n',i,CVO.NumTestSets);
    % For input into SVMTRAIN
    trainidx = CVO.training(i);
    testidx = CVO.test(i);

    Features_train = Features(:,trainidx)';
    Features_test = Features(:,testidx)';
    Y_train = Y(1,trainidx)';
    Y_test = Y(1,testidx)';

    % Classification
    model1 =
svmtrain(Features_train,Y_train,'kernel_function','rbf','method','SMO');
```

```
%     % Testing a second model (not necessary)
%     model2 =
fitcsvm(Features_train,Y_train,'KernelFunction','gaussian','Solver','SMO','Ve
rbose',0,'CrossVal','off');

    % Compile results for each test
    C1 = svmclassify(model1,Features_test);
    acc1 = Y_test==C1;
    Accuracy(i,1) = 100*mean(acc1);
%     C2 = predict(model2,Features_test);
%     acc2 = Y_test==C2;
%     Accuracy(i,2) = 100*mean(acc2);

    fprintf('----Accuracy: %3.2f %%\n',Accuracy(i,1));
%     fprintf('----Accuracy: %3.2f %%\n',Accuracy(i,2));
end

% Compute the mean and std accuracy across all participants
Task3_meanAccuracy = mean(Accuracy);
Task3_stdAccuracy = std(Accuracy);

fprintf('-----\n TASK 3 MODEL 1 ACCURACY: %3.2f %% (%g)\n-----
\n',Task3_meanAccuracy(1),Task3_stdAccuracy(1));
% fprintf('-----\n TASK 3 MODEL 2 ACCURACY: %3.2f %% (%g)\n-----
\n',Task3_meanAccuracy(2),Task3_stdAccuracy(2));


%% TASK 3: Subject-Irrespective Classification
fprintf('\n---------------\nTask 3: Subject-Irrsepective Classification\n----
----------\n');

prelimSetup % Preliminary Setup

%%%% Stochastic cross-validation
fprintf('\n ----- Stochastic cross-validation \n \n');

% Compute all FBCSP features for training and test data
tempidx = true(1,nTrials);
[EEGcsp,Features,cspftrs,Psel] = FBCSPfeatures(EEG,Y,tempidx,opt);

nTests = 20; % Number of tests for stochastic cv

Accuracy = zeros(nTests,1);  % The classification accuracy for each partition

for i = 1:nTests
    fprintf('Obtaining Results for Holdout test %d/%d\n',i,nTests);

    % Choosing the training and testing data
    % Stochastic/Holdout cross-validation on 10% of the data
    CVO = cvpartition(Y','Holdout',0.1);

    % For input into SVMTRAIN
    trainidx = CVO.training;
    testidx = CVO.test;
```

```
    Features_train = Features(:,trainidx)';
    Features_test = Features(:,testidx)';
    Y_train = Y(1,trainidx)';
    Y_test = Y(1,testidx)';

    % Classification
    model1 =
svmtrain(Features_train,Y_train,'kernel_function','rbf','method','SMO');
%     % Testing a second model (not necessary)
%     model2 =
fitcsvm(Features_train,Y_train,'KernelFunction','gaussian','Solver','SMO','Ve
rbose',0,'CrossVal','off');

    % Compile results for each test
    C1 = svmclassify(model1,Features_test);
    acc1 = Y_test==C1;
    Accuracy(i,1) = 100*mean(acc1);
%     C2 = predict(model2,Features_test);
%     acc2 = Y_test==C2;
%     Accuracy(i,2) = 100*mean(acc2);

    fprintf('----Accuracy: %3.2f %%\n',Accuracy(i,1));
%     fprintf('----Accuracy: %3.2f %%\n',Accuracy(i,2));

end

Task3_meanAccuracy = mean(Accuracy);
Task3_stdAccuracy = std(Accuracy);

fprintf('-----\n TASK 3 MODEL 1 ACCURACY: %3.2f %% (%g)\n-----
\n',Task3_meanAccuracy(1),Task3_stdAccuracy(1));
% fprintf('-----\n TASK 3 MODEL 2 ACCURACY: %3.2f %% (%g)\n-----
\n',Task3_meanAccuracy(2),Task3_stdAccuracy(2));
```

prelimSetup.m

```
% Preliminary setup
clear; close all; clc;

% Loading the data
load FinalProjectData

% Add the FBCSP folder to the path
addpath FBCSP

% Get some informatino about data size
nSubj = length(unique(SubjectID));  % Number of Subjects
nTrials = length(Y);                % Total number of trials
```

```
% Example: If you want to compute how many trials belong to Participant 5,
% you could use 'nTrials_p5 = sum(SubjectID==5);

% Options for FBCSP
opt.features.cspcomps = 2;      % This is K from the lecture slides (you
should have 2*K features per filter in the filter bank)
opt.features.cspband = [4,32];  % The filter bank will be constructed within
this frequency range
opt.features.cspstep = 2;       % Adjacent filters will having initial
frequency shifted by 2 Hz
opt.features.cspwidth = 4;      % Each bandpass filter will cover a 4 Hz band
opt.mode = 'concat';            % CSP can actually be computed according to
about 4 mathematically equivalent formulations. I've selected one for you to
use here.
opt.filter.type = 'butter';     % The filter bank will be constructed of
Butterworth bandpass filters
opt.filter.order = 2;           % Each filter will be second-order
opt.filter.sr = 220;            % This is just the sampling rate of the EEG
signal (needed to compute the coefficients of each bandpass filters)
```