Seminar Hands-on BCI implementation

Session 6: Classification

In this session the antagonistic finger movements are finally discriminated by means of three different classification algorithms.
We will use the spider toolbox, so unpack it and add it to your Matlab path!

```
clear all
```

Load data files to workspace

```
load ecogStruct3.mat
load epoch2.mat
```

Define dataset based on previously defined features (use your results from Session 5)

Number of trials with finger movement

```
nTrials = size(ecog.periodogram.periodogram,3);
```

Frequency features

```
freqBand =  % base your values here on your results from Session 5 (t-value plot or relief algorithm plot)
freqIdx = unique(nearly(freqBand,ecog.periodogram.centerFrequency));
nFreq = length(freqIdx);
```

Channel features

```
chan =  % base your values here on your results from Session 5 (t-value plot or relief algorithm plot)
nChan = length(chan);
```

Prepare date for z-scoring (same as in Session 4)

```
dat = reshape(ecog.periodogram.periodogram(freqIdx,chan,:),nFreq,nChan*nTrials);
% z-score data
dat = zscore(dat,0,2);
% Reshape data
dat = reshape(dat,nFreq,nChan,nTrials);
dat = permute(dat,[3 2 1]);
dat = reshape(dat,nTrials,nFreq*nChan);
```

Create subsets for cross-validation

```
realClassLabels = epoch.label; % Class labels
N = 10; % CV steps

selector = ceil((1:length(realClassLabels))/(length(realClassLabels)/N));
selector = selector(randperm(length(realClassLabels)));
```

TASK 1 (2 pt):

What does the variable selector contain?
What is the purpose of cross-validation? Describe this method and its advantages/disadvantages.

Classification

Define classification algorithm: 'bayes','lda','svm'

```
alg = 'svm';

balance = 0; % balance datasets 1 = yes / 0 = no

optimizeC = 1; % only important if alg = 'svm'

predictedClassLabels = zeros(1,length(realClassLabels)); % creating a vector, where later the class predicted for each trial will be stored
```

Cross validation

```
for k = 1:N
    fprintf('CV step #%i\n',k);
    testIdx = find(selector == k);
    trainIdx = setdiff(1:length(realClassLabels),testIdx);

    if balance == 1
        labelIdx = getBalancedTrainset(realClassLabels(trainIdx));
        trainIdx(~labelIdx) = [];
    end

    TRAIN classifier (LDA, bayes)
    curTrain = dat(trainIdx,:);
    curClassLabels = realClassLabels(trainIdx);

    if strcmpi(alg,'bayes')
        R = train_bayes(curTrain,curClassLabels);
    elseif strcmpi(alg,'lda')
        R = train_lda(curTrain,curClassLabels);
    end
```

TASK 2 (2 pt):

What does it mean to balance datasets? Why is this sometimes done?
Have a look at the train_lda function.
What kind of information does it store in the structure R?

```
    TEST
    curTest = dat(testIdx,:);

    if strcmpi(alg,'bayes')
        Res = test_bayes(R,curTest);
    elseif strcmpi(alg,'lda')
        Res = test_lda(R,curTest);
    end
```

TASK 3 (1 pt):

Have a look at the test_lda function.
What information is stored in Res.prediction?

```
Special case 'SVM'

    if strcmpi(alg,'svm')

    testClassLabels = realClassLabels(testIdx);

        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 default value
            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 = train_svm(curTrain,curClassLabels,iterative_c(m));

                % Test
                Res = test_svm(R,curTest);

                % Result according to each c value
                res_cOpt(m,:) = Res.prediction;

                % accuracy for each c value
                   optAcc(m) = sum( res_cOpt(m,:) == testClassLabels ) / length(
                  testClassLabels );
            end

            [max_val, max_idx] = max(optAcc);

            Res.prediction = res_cOpt(max_idx,:);
```

TASK 4 (1 pt):

What is the cost parameter C and why is it iteratively optimized?

```
        else % without iteratively optimizing C
            % Default C
            defaultC = ecogGetDefC(curTrain);
            % Train
            R = train_svm(curTrain,curClassLabels,defaultC);
            % Test
            Res = test_svm(R,curTest);

        end
    end

    predictedClassLabels(testIdx) = Res.prediction;

end

accuracy = sum( predictedClassLabels == realClassLabels ) / length(
realClassLabels )
```

TASK 5 (1 pt):

Compare the results of the 3 different algorithms (change 'alg' in line 64). Which one produces the best results (highest accuracy)?
Then also change your selected features (channels/frequencies, line 27 and 32), always keeping in mind
the results from last week's t-values/relief algorithm.
Which features lead to the highest accuracy? (Also keep in mind that the more features you use the longer the calculation time is, so try to reduce your number of features without this resulting in a lower accuracy.)

TASK 6

```
% If you have time left, you can try to use the data from the PCA (load
% resultsPCA.mat - data saved in xPCA) to perform the classification.
% Use the information you got last week from the t-values and relief
% algorithm to choose the best features.
```