**Code**

**Extract trials:**

function extracttrials(indir, outfile)

%{

extracttrials(indir, outfile)

Extracts single trials from the files in \*indir\*(input directory) and writes them to

\*outfile\*.

%}

%% scan indir(input directory) for input files

d = dir(indir);

filelist = {};

for i = 1:length(d)

if(d(i).isdir == 0)

filename = sprintf('%s\\%s',indir,d(i).name);

filelist = cat(2,filelist,{filename});

end

end

fprintf('found %i files ...\n',length(filelist));

%% initialize variables

runs = cell(1,length(filelist));

srate = 2048; % sampling rate of raw data in Hz

reference = [7 24]; %1:32; % indices of channels used as reference

filterorder = 3;

filtercutoff = [1/1024 12/1024];

[f\_b, f\_a] = butter(filterorder,filtercutoff);

decimation = 64; % downsampling factor

n\_samples = 32; % number of (temporal) samples in a trial

n\_targets = 0; % keeps track of number of target trials

n\_nontargets = 0; % keeps track of nontarget trials

%% extract features from files in filelist

for i = 1:length(filelist)

% load data

f = load(filelist{i});

fprintf('processing %s\n',filelist{i});

% rereference the data

n\_channels = size(f.data,1);

ref = repmat(mean(f.data(reference,:),1),n\_channels,1);

f.data = f.data - ref;

% drop the mastoid channels

f.data = f.data(1:32,:);

n\_channels = size(f.data,1);

% bandpass filter the data (with a forward-backward filter)

for j = 1:n\_channels

f.data(j,:) = filtfilt(f\_b,f\_a,f.data(j,:));

end

% downsample the data (from 2048 Hz to 32 Hz)

f.data = f.data(:,1:decimation:end);

% extract trials

% compute class labels

% put everything in the cell-array runs

n\_trials = size(f.events,1);

runs{i}.x = zeros(n\_channels,n\_samples,n\_trials);

for j = 1:n\_trials

pos = round(etime(f.events(j,:), ... % pos contains position of the events

f.events(1,:))\*(srate/decimation) ...

+ 1 + 0.4\*srate/decimation) ;

runs{i}.x(:,:,j) = f.data(:,pos:pos+n\_samples-1);

end

runs{i}.y = zeros(1,n\_trials);

runs{i}.y(f.stimuli == f.target) = 1;

runs{i}.y(f.stimuli ~= f.target) = -1;

runs{i}.stimuli = f.stimuli;

runs{i}.target = f.target;

% update counters

n\_targets = n\_targets + sum(runs{i}.y == 1);

n\_nontargets = n\_nontargets + sum(runs{i}.y == -1);

end

%% save results in outfile

fprintf('total target trials: %i\n',n\_targets);

fprintf('total nontarget trials %i\n',n\_nontargets);

save(outfile,'runs');

**Test classification:**

function n\_correct = testclassification(trainingfiles, testfile)

%{

testclassification(trainingfiles, testfile)

Uses the data in \*trainingfiles\* to build a classifier and tests

the classifier on the data in \*testfile\*. \*n\_correct\* contains for each

number of blocks (1-20) the number of correctly classified items. If no

output arguments are given \*n\_correct\* is plotted.

The training files and the test file have to be built with extracttrials.

Example: testclassification({'s2','s3','s4'},'s1')

%}

%% subsets of electrodes

% Fz, Cz, Pz, Oz

% channels = [31 32 13 16];

% Fz, Cz, Pz, Oz, P7, P3, P4, P8

channels = [31 32 13 16 11 12 19 20];

% Fz, Cz, Pz, Oz, P7, P3, P4, P8, O1, O2, C3, C4, FC1, FC2, CP1, CP2

% channels = [31 32 13 16 11 12 19 20 15 17 8 23 5 26 9 22];

% All electrodes

% channels = [1:32];

%% load training files and concatenate data and labels into two big arrays

x = [];

y = [];

for i = 1:length(trainingfiles);

fprintf('loading %s\n',trainingfiles{i});

f = load(trainingfiles{i});

n\_runs = length(f.runs);

for j = 1:n\_runs;

x = cat(3,x,f.runs{j}.x);

y = [y f.runs{j}.y];

end

end

%% select channels, windsorize, normalize, bayesian lda

x = x(channels,:,:);

w = windsor;

w = train(w,x,0.1);

x = apply(w,x);

n = normalize;

n = train(n,x,'z-score');

x = apply(n,x);

n\_channels = length(channels);

n\_samples = size(x,2);

n\_trials = size(x,3);

x = reshape(x,n\_samples\*n\_channels,n\_trials);

b = bayeslda(1);

b = train(b,x,y);

%% load testfile and do classification

f = load(testfile);

n\_runs = length(f.runs);

n\_blocks = 20;

n\_correct = zeros(1,n\_blocks);

for i = 1:n\_runs

x = f.runs{i}.x(channels,:,:);

x = apply(w,x);

x = apply(n,x);

n\_trials = size(x,3);

x = reshape(x,n\_channels\*n\_samples,n\_trials);

y = classify(b,x);

scores = zeros(1,6);

for j = 1:n\_blocks

start = (j-1)\*6+1;

stop = (j)\*6;

stimulussequence = f.runs{i}.stimuli(start:stop);

scores(stimulussequence) = scores(stimulussequence) + ...

y(start:stop);

[dummy,idx] = max(scores);

if (idx == f.runs{i}.target)

n\_correct(j) = n\_correct(j)+1;

end

end

end

%% if no output arguments plot the results

if nargout == 0

plot(n\_correct);

axis([1 20 0 6]);

% xlabel('Number of blocks');

% ylabel('Number of correct classifications');

end

**Cross validate:**

function crossvalidate(filelist)

%{

crossvalidate(filelist)

Uses n - 1 of the n files in \*filelist\* to build a classifier and tests the

classifier on the left-out file. This is done once for each file and

results are averaged. Average classification accuracy and bitrate are plotted.

The files in \*filelist\* have to be built with extracttrials.

%}

%% do the crossvalidation

n\_correct = zeros(length(filelist),20);

for i = 1:length(filelist)

trainingfiles = filelist;

trainingfiles(i) = [];

n\_correct(i,:) = testclassification(trainingfiles,filelist{i});

end

%% plot the results

accuracy = mean(n\_correct) / 6; % each file contains six runs

br = bitrate(accuracy);

x = 2.4:2.4:48;

ax = plotyy(x,100\*accuracy,x,br);

set(ax(1),'ylim',[10 100]);

set(ax(2),'ylim',[0 45]);

set(ax(1),'ytick',10:10:100);

set(ax(2),'ytick',0:5:45);

set(get(ax(1),'ylabel'),'String','Accuracy (%)')

set(get(ax(2),'ylabel'),'String','Bitrate (bits\min)')

xlabel('Time (s)');

grid;

%% function to compute bits / min from classification accuracy

function br = bitrate(accuracy)

for i = 1:length(accuracy)

if accuracy(i) > 0 && accuracy(i) < 1

br(i) = log2(6) + accuracy(i)\*log2(accuracy(i)) + ...

(1-accuracy(i))\*log2((1-accuracy(i))/5);

end

if accuracy(i) == 0

br(i) = 0;

end

if accuracy(i) == 1

br(i) = log2(6);

end

end

br = br\*60./(2.4:2.4:48);

Main Function:

% Extract trials from the EEG Raw Data

%Subject 1

extracttrials('subject1\session1','s1')

extracttrials('subject1\session2','s2')

extracttrials('subject1\session3','s3')

extracttrials('subject1\session4','s4')

% BLDA Classifier

testclassification({'s2','s3','s4'},'s1')

testclassification({'s1','s3','s4'},'s2')

testclassification({'s2','s1','s4'},'s3')

testclassification({'s2','s3','s1'},'s4')

% FLDA Classifier

crossvalidate({'s1','s2','s3','s4'})