Hands-on tutorlal for fNIRS dataset: Classification

This MATLAB script is a hands-on tutorial to classify task-related temporal hemodynamic responses.

written by Jaeyoung Shin (jyshin34@wku.ac.kr) 08. Sep. 2019

1. Run BBCI toolbox

```
format compact
disp('1. run BBCI toobox')

% set paths
WorkingDir = pwd;
MyToolboxDir = fullfile(WorkingDir,'BBCI'); % toolbox path
MyDatDir  = fullfile(WorkingDir,'data');
MyMatDir  = fullfile(MyDatDir,'matdata'); % fNIRS dataset path

% run BBCI toolbox
addpath(MyToolboxDir);
startup_bbci_toolbox('DataDir', MyDatDir, 'MatDir', MyMatDir,...
    'TmpDir','/tmp/', 'History', 0);

disp(BTB.MatDir);
```

2. Load example fNIRS dataset

```
disp('2. load example fNIRS dataset')

% file name: e.g.) fNIRS 04.mat
file = 'fNIRS 04.mat';

% load fNIRS dataset file to MATLAB workspace
[cntHb, mrk, mnt] = file_loadMatlab(file);
```

3. Band-pass filtering to eliminate physiological noises

```
disp('3. band-pass filtering')

% zero-order band-pass filtering using [ord]-order Butterworth IIR filter
% with passpand of [band]
ord = 3;
band = [0.01 0.1]/cntHb.fs*2;

[b, a] = butter(ord, band, 'bandpass');
cntHb = proc_filtfilt(cntHb, b, a); % zero-order filtering
```

4. Segmentation and baseline correction

```
disp('4-1. segmentation')
```

```
% segment cntHb into epochs ranging [ival_epo]
ival_epo = [-1 25]*1000; % msec
epo = proc_segmentation(cntHb, mrk, ival_epo);

disp('4-2. baseline correction')

% baseline correction using reference interval of [ival_base]
ival_base = [-1 0]*1000; % msec
epo = proc_baseline(epo, ival_base);
```

5. Feature extraction

```
disp('5. feature extraction')

% feature extraction using time windows with [ival_fv] intervals
ival_fv = [0 5; 5 10; 10 15]*1000;
fv = proc_jumpingMeans(epo, ival_fv); % compute mean values of epo

% reshape [fv] to fit 'fitcecoc' (or 'fitcsvm')
xsz = size(fv.x);
X = reshape(fv.x, [prod(xsz(1:2)), xsz(3)]);
X = X'; % feature vector
Y = vec2ind(fv.y)'; % label - 1: RHT / 2: LHT / 3: FT
```

6. Leav-one-out cross-validation

```
disp('6. leave-one-out cross-validation')

% type of learner: linear SVM with feature vector standardization
t = templateSVM('Standardize',true, 'KernelFunction', 'linear');

% three-class leave-one-out cross-validation with 'one versus one' coding
mdl = fitcecoc(X,Y,'Learner', t, 'Leaveout','on','Coding','onevsone');

% in case of binary classification, use 'fitcsvm' with 'ClassNames' option
% insted of 'fitcecoc'
% e.g.) mdl = fitcsvm(X,Y,'Standardize',true,'Leaveout','on',...
%                     'ClassNames',{'1','2'}); % 1: RHT / 2: LHT

% cross-validation performance
acc = 1 - kfoldLoss(mdl);

fprintf("LOOCV performance = %f\n",acc);
disp('Done.')
```

```
% remove toolbox path
rmpath(MyToolboxDir);
```