EMATS: EEG based Machine or Deep Learning Algorithms for TBI & Stroke Classification

Predict New Classification from EEG

Michael Caiola (Michael.Caiola@fda.hhs.gov) and Meijun Ye (Meijun.Ye@fda.hhs.gov)

This script contains two sections:

1. Preprocess an .edf file for future classification
2. Predict the classification of a preprocessed EEG

Preprocess EDF

Input: EEG .edf file

- **>4 minutes in length**
- **>= 19 contacts**
- **>= 250 Hz sampling rate**

Requirements:

- EEGLAB with BIOSIG toolbox
- Signal Processing Toolbox

Run this section to convert your EDF into a usable preprocessed .mat file(s). The file will be cut up into 3-minute segments with the first minute discarded. Preprocessed files will be saved as .mat files.

```
LoadEDF();
```

Predict

For use with a preprocessed EEG .mat file (see above).

Requirements:

- Signal Processing Toolbox
- Deep Learning Toolbox
- EEGLAB (only necessary if using TMN)

Choose model type:

- **STFT**: Short-time Fourier Transform
- **TMN**: Topographic Map Network

- **Sensor Fusion**
- **Feature**: Deep Network using ReliefF Features
- *NOTE: Requires calculation of features: long/expensive.*
- **LDA_SVM**: SVM with LDA Features.
- *NOTE: Requires calculation of features: long/expensive. Will not provide a score*
- **ReliefF_SVM** with ReliefF Features.
- *NOTE: Requires calculation of features: long/expensive.Will not provide a score*

```
model = "STFT";
```

```
[fileLoc,path] = uigetfile(".mat","Select processed EEG file");

%Set up function
switch model
    case {"LDA","ReliefF","Feature"}
        load(fullfile(path,fileLoc))
        F = BigFeats(y);
        F = table(F','VariableNames',{'Features'});

    case "TMN"
        load("chlocs2.mat")
        ds = TopoDatastore(fullfile(path,fileLoc),[],channel_locations);
        y = read(ds);
        y = y.Predictors{1};

    case {"STFT","Fusion"}
        ds = ResampleDatastore(fullfile(path,fileLoc),100,'DataAugmentation',false);
        ds.MiniBatchSize = 1;
        reset(ds)
        y = read(ds);
        y = y.Predictors{1};

end

%Predict
switch model
    case "LDA"
        load("LDA_SVM.mat")
        F = table(F.Features(1,r_logical),'VariableNames',{'Features'});
        [yfit,score] = trainedModel.predictFcn(F);
        disp("EEG prediction: " + string(yfit));
    case "ReliefF"
        load("ReliefF_SVM.mat")
        [yfit,score] = trainedModel1.predictFcn(F);
        disp("EEG prediction: " + string(yfit));
    case "Feature"
        load("F_DL.mat","net5b")
        load("RelieffScores.mat")
        F = F.Features(1,featureIndex(1:100));
        [yfit,score] =
classify(net5b,F,"ExecutionEnvironment",'cpu',MiniBatchSize=1);
        disp("EEG prediction: " + string(yfit));
        disp("Prediction score: " + num2str(max(score)));
    case "TMN"
        load("Topo_BasicNet.mat","net3")
        [yfit,score] =
classify(net3,y,"ExecutionEnvironment",'cpu',MiniBatchSize=1);
        disp("EEG prediction: " + string(yfit));
        disp("Prediction score: " + num2str(max(score)));
    case "STFT"
        load("STFTNet.mat","net4a")
        [yfit,score] =
classify(net4a,y,"ExecutionEnvironment",'cpu',MiniBatchSize=1);
        disp("EEG prediction: " + string(yfit));
        disp("Prediction score: " + num2str(max(score)));
    case "Fusion"
        load("SFnet.mat","bnet2")
        [yfit,score] =
classify(bnet2,y,"ExecutionEnvironment",'cpu',MiniBatchSize=1);
        disp("EEG prediction: " + string(yfit));
        disp("Prediction score: " + num2str(max(score)));
end
```

```
EEG prediction: HEA
Prediction score: 0.72235
```

CAUTION: EEG classification is set using default thresholding and should be further optimized for `sensitivity/specificity/accuracy` on additional training data prior to use.

Disclaimer

This software and documentation (the "Software") were developed at the Food and Drug Administration (FDA) by employees of the Federal Government in the course of their official duties. Pursuant to Title 17, Section 105 of the United States Code, this work is not subject to copyright protection and is in the public domain. Permission is hereby granted, free of charge, to any person obtaining a copy of the Software, to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, or sell copies of the Software or derivatives, and to permit persons to whom the Software is furnished to do so. FDA assumes no responsibility whatsoever for use by other parties of the Software, its source code, documentation or compiled executables, and makes no guarantees, expressed or implied, about its quality, reliability, or any other characteristic. Further, use of this code in no way implies endorsement by the FDA or confers any advantage in regulatory decisions. Although this software can be redistributed and/or modified freely, we ask that any derivative works bear some notice that they are derived from it, and any modified versions bear some notice that they have been modified.

Supporting Functions

Loading and Cleaning

```
function LoadEDF(savefolder)
N=3;
[edfFile,path] = uigetfile("*.edf","Choose .edf File");
if nargin < 1
    savefolder= uigetdir([],"Choose Save Folder");
end
loadEEGLAB();
rawEEG=joinEEG(fullfile(path,edfFile));
if isempty(rawEEG.data)
    warning("Issue finding Channels.")
    return
end
data=cutEEG(rawEEG,N);
if isempty(data)
    warning("Issue splitting EEG.")
    return
end
try
    clndata=filtEEG(rawEEG,data);
catch
    warning("Could not clean EEG.")
    return
end
for kk=1:size(clndata,3)
    y=clndata(:,:,kk);
    if or(isempty(y),y==zeros(size(y)))
        continue
    end
    try
        save(fullfile(savefolder,"ProcessedEEG",erase(string(edfFile),".edf")
+"-"+string(kk)+".mat"),'y')
    catch
        mkdir(fullfile(savefolder,"ProcessedEEG"))
        save(fullfile(savefolder,"ProcessedEEG",erase(string(edfFile),".edf")
+"-"+string(kk)+".mat"),'y')
    end
end
disp("EEG successfully exported!")
end
```

```
function rawEEG=joinEEG(names)
out=0;
cnt = 0;
```

```
try
    [rawEEG] = FindCh(names);
    param_flag = 0;
catch
    rawEEG.comments=[];
    rawEEG.data=[];
    param_flag=1;
end

if param_flag
    rawEGG=[];
else
    if rawEEG.xmax < 3*60
        error(".edf is too short.")
    end
    rawEEG.data=rawEEG.data(:,1:250*60*(floor(length(rawEEG.data)/250/60)-1));
    rawEEG.xmax=length(rawEEG.data)/250;
    rawEEG.pnts=length(rawEEG.data);
    rawEEG.times=0:4:4*(rawEEG.pnts-1);
end
end
```

```
function data=cutEEG(EEG,n)
cuts=floor(EEG.xmax/60/n);
data=zeros(19,n*60*250,cuts);
for i=1:cuts
    data(:,:,i)=EEG.data(:,1+(i-1)*n*60*250:i*n*60*250);
end
end
```

```
function [clndata]=filtEEG(EEG,data)
for i=1:size(data,3)
    rawEEG=EEG;
    rawEEG.data=data(:,:,i);
    rawEEG.pnts=length(rawEEG.data);
    filtEEG = pop_eegfiltnew(rawEEG,1,100);
    EEG = pop_reref(filtEEG,[]);
    [~,W] = fastica(EEG.data,'verbose','off');
    EEG = pop_editset(EEG,'icaweights',W);
    EEG = iclabel(EEG);
    [~,ictype] = max(EEG.etc.ic_classification.ICLabel.classifications,[],2);
    icreject = find(ictype~=1);
    % EEG.reject.gcompreject(1,icreject') = 1;
    if ~(length(icreject) == size(EEG.icaweights,1))
        clean_EEG = pop_subcomp(EEG,icreject',0,0);
        %rawdata(:,:,i) = EEG.data;
        clndata(:,:,i) = clean_EEG.data;
    end
end
end
```

```
function [rawEEG,cnt] = FindCh(file)
cnt = 0;
load('chlocs2.mat','channel_locations');
ch_locs = struct2table(channel_locations);
ch_locs = string(ch_locs.labels);
rawEEG = pop_biosig(file,'blockrange',[60
Inf],'importevent','off','importannot','off');
chs = struct2table(rawEEG.chanlocs);
chs = string(chs.labels);
chs = erase(chs,"-REF"|"-LE");
rawEEG = pop_resample(rawEEG,250);
in = [];
for i = 1:length(ch_locs)
    in1 = find(strcmpi(chs,ch_locs(i)));
    in = [in in1];
    if and(isempty(in1),ismember(ch_locs(i),["T3";"T4";"T5";"T6"]))
        switch ch_locs(i)
            case "T3"
                in2 = find(strcmpi(chs,"T7"));
            case "T4"
                in2 = find(strcmpi(chs,"T8"));
            case "T5"
                in2 = find(strcmpi(chs,"P7"));
            case "T6"
                in2 = find(strcmpi(chs,"P8"));
        end
        in = [in in2];
    end
end
if length(in)~=19
    error("Could not locate all channels!");
end
rawEEG.data = rawEEG.data(in,:);
rawEEG.chanlocs = channel_locations;
rawEEG.nbchan = 19;
if ~isequal(in,[1:16,19:21])
    cnt = 1;
    if ~isequal(in,1:19)
        cnt = 2;
    end
end
end
```

```
function loadEEGLAB()
if exist("pop_biosig")~=2
    path = pwd;
    try
        cd(fullfile("..","eeglab"))
    catch
        disp("EEGLAB not found.")
        eeglabpath = uigetdir([],"Locate EEGLAB Directory");
        cd(eeglabpath)
    end
    eeglab
    close
    cd(path);
end
end
```

Feature Calculation

```
function F = BigFeats(data)
srate=250;
numFeats = 19*2 + 171*5;
idx = 0;
h = waitbar(0,"Calculating Features");
tflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',4,'HalfPowerFrequency2',8, ...
    'SampleRate',srate);
aflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',8,'HalfPowerFrequency2',12, ...
    'SampleRate',srate);
gflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',25,'HalfPowerFrequency2',40, ...
    'SampleRate',srate);
F=zeros(380,size(data,3));
for j=1:size(data,3)
    stats = []; %mean, max, min, std 76 values
    spectent = []; % 19 Values
    PACag = []; % 19 Values
    PACtg = []; % 19 Values
    PACta = []; % 19 Values
    [abs_psd,rel_psd]=getPSD(data(:,:,j),srate);
    for i=1:19
        %% Spectral Entropy of each channel
        X = fft(data(i,:,j));
        S = abs(X).^2;
        P = S./sum(S);
        H = 0;
        for m = 1:length(P)
            H = H + P(m)*log2(P(m));
        end
        spectent = [spectent,-H];
        PACag = [PACag, getPAC(data(i,:,j),aflt,gflt)];
        PACtg = [PACtg, getPAC(data(i,:,j),tflt,gflt)];
        PACta = [PACta, getPAC(data(i,:,j),tflt,aflt)];
        idx = idx + 1;
        waitbar(idx/(numFeats),h)
    end
    stats=[mean(data(:,:,j),2)',max(data(:,:,j),[],2)',min(data(:,:,j),
[],2)',std(data(:,:,j),[],2)'];
    F(:,j)=[stats,reshape(abs_psd,[1,19*6]),reshape(rel_psd,
[1,19*6]),spectent,PACag,PACtg,PACta];
end
tflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',4,'HalfPowerFrequency2',8, ...
    'SampleRate',srate);
aflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',8,'HalfPowerFrequency2',12, ...
    'SampleRate',srate);
gflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',25,'HalfPowerFrequency2',40, ...
    'SampleRate',srate);
dflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',1,'HalfPowerFrequency2',4, ...
    'SampleRate',srate);
mflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',12,'HalfPowerFrequency2',16, ...
    'SampleRate',srate);
bflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',16,'HalfPowerFrequency2',20, ...
    'SampleRate',srate);
RCH=zeros(size(data,3),171*6);
for j=1:size(data,3)
    tdata = zeros(size(data,1),size(data,2));
    adata = zeros(size(data,1),size(data,2));
    mdata = zeros(size(data,1),size(data,2));
    bdata = zeros(size(data,1),size(data,2));
    gdata = zeros(size(data,1),size(data,2));
    ddata = zeros(size(data,1),size(data,2));

    for ch = 1:19
        tdata(ch,:) = filter(tflt,data(ch,:,j));
        adata(ch,:) = filter(aflt,data(ch,:,j));
        mdata(ch,:) = filter(mflt,data(ch,:,j));
        bdata(ch,:) = filter(bflt,data(ch,:,j));
        gdata(ch,:) = filter(gflt,data(ch,:,j));
        ddata(ch,:) = filter(dflt,data(ch,:,j));
        idx = idx + 1;
        waitbar(idx/(numFeats),h)
    end

    [tCH,idx] = getCOH(tdata,srate,h,idx,numFeats);
    [aCH,idx] = getCOH(adata,srate,h,idx,numFeats);
    [mCH,idx] = getCOH(mdata,srate,h,idx,numFeats);
    [bCH,idx] = getCOH(bdata,srate,h,idx,numFeats);
    [gCH,idx] = getCOH(gdata,srate,h,idx,numFeats);
    [dCH,idx] = getCOH(ddata,srate,h,idx,numFeats);

    close(h)

    RCH(j,:) = [tCH,aCH,mCH,bCH,gCH,dCH];
end
F = [F; RCH'];
end
```

```
function [abs_psd,rel_psd]=getPSD(data,srate)
% data: sample X channel
data=data';
fband=[1 4; 4 8; 8 12; 12 16; 16 20; 25 40];
tot=bandpower(data,srate,[1 srate/2]);

n=size(fband,1);

for i=1:n
    abs_psd(:,i)=bandpower(data,srate,fband(i,:));
end

ch=max(size(abs_psd,1));

for j=1:ch
    rel_psd(j,:)=abs_psd(j,:)./tot(j);
end
end
```

```
function PAC = getPAC(data,pflt,aflt)

fp = filter(pflt,data);
fp = hilbert(fp);
phi = angle(fp).*(180/pi);

fa = filter(aflt,data);
fa = hilbert(fa);
A = abs(fa);
```

```
edges = -180:20:180;
bin_phi = discretize(phi,edges);
A_mean = zeros(1,18);
for b = 1:18
    A_mean(b) = mean(A(bin_phi==b));
end


Pj = zeros(1,18);
for j = 1:18
    Pj(j) = A_mean(j)/sum(A_mean);
end
Dkl = zeros(1,18);
for k = 1:18
    Dkl(k) = log(Pj(k)/(1/18))*Pj(k);
end
PAC = sum(Dkl)/log(18);
end
```

```
function [C,idx] = getCOH(data,srate,h,idx,numFeats)
    ind = [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,...
        0,  0,  0,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,...
        1,  1,  1,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,...
        2,  2,  2,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,...
        3,  3,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,...
        5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  6,  6,  6,...
        6,  6,  6,  6,  6,  6,  6,  6,  6,  7,  7,  7,  7,  7,  7,  7,...
        7,  7,  7,  7,  8,  8,  8,  8,  8,  8,  8,  8,  8,  8,  9,  9,...
        9,  9,  9,  9,  9,  9,  9, 10, 10, 10, 10, 10, 10, 10, 10, 11,...
       11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13,...
       13, 14, 14, 14, 14, 15, 15, 15, 16, 16, 17; 1,  2,  3,  4,  5,...
        6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,...
       17, 18,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,...
       16, 17, 18,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,...
       16, 17, 18,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,...
       17, 18,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,...
        6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,  7,  8,  9,...
       10, 11, 12, 13, 14, 15, 16, 17, 18,  8,  9, 10, 11, 12, 13, 14,...
       15, 16, 17, 18,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 10, 11,...
       12, 13, 14, 15, 16, 17, 18, 11, 12, 13, 14, 15, 16, 17, 18, 12,...
       13, 14, 15, 16, 17, 18, 13, 14, 15, 16, 17, 18, 14, 15, 16, 17,...
       18, 15, 16, 17, 18, 16, 17, 18, 17, 18, 18]+1;
   C = zeros(1,length(ind));

    for c = 1:length(ind)
        y = mscohere(data(ind(1,c),:),data(ind(2,c),:),...
                          srate*30,0,1:0.1:100,srate);
        C(c) = mean(y);
        idx = idx + 1;
        waitbar(idx/numFeats,h)
    end

end
```