Data

July 11, 2024

Data processing.

Data from https://zenodo.org/record/2348892/files/

[43]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sys, os
import io
import json
import mne

import joblib
import re

from scipy.io import loadmat
from scipy.signal import welch

from pyriemann.embedding import SpectralEmbedding
from pyriemann.classification import MDM
from pyriemann.estimation import Covariances

from sklearn.pipeline import make_pipeline
from sklearn.model_selection import StratifiedKFold, cross_val_score,␣
↪cross_val_predict
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

from scikeras.wrappers import KerasClassifier

from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, RepeatVector,␣
↪TimeDistributed
```

[3]:
```
file_path = r'C:\Users\brian\OneDrive - University of␣
 ↪Tennessee\Desktop\Research\Python program\RTI-simulator\data_files'
files = [x for x in os.listdir(file_path) if re.search('.mat',x)]
print(files)
```

```
['subject_00.mat', 'subject_01.mat', 'subject_02.mat', 'subject_03.mat',
'subject_04.mat', 'subject_05.mat', 'subject_06.mat', 'subject_08.mat',
'subject_09.mat', 'subject_10.mat', 'subject_11.mat', 'subject_12.mat',
'subject_13.mat', 'subject_14.mat', 'subject_15.mat', 'subject_16.mat',
'subject_17.mat', 'subject_18.mat', 'subject_19.mat', 'subject_20.mat']
```

[4]:
```
len(files)
```

[4]: 20

[5]:
```
# a function to extract data from a single file
def get_data(file):
    file_path = r'C:\Users\brian\OneDrive - University of␣
 ↪Tennessee\Desktop\Research\Python program\RTI-simulator\data_files\\'
    data = loadmat(file_path + file)
    print(data)

    S = data['SIGNAL'][:, 1:17]
    stim_close = data['SIGNAL'][:, 17]
    stim_open = data['SIGNAL'][:, 18]
    stim = 1 * stim_close + 2 * stim_open

    chnames = [
        'Fp1',
        'Fp2',
        'Fc5',
        'Fz',
        'Fc6',
        'T7',
        'Cz',
        'T8',
        'P7',
        'P3',
        'Pz',
        'P4',
        'P8',
        'O1',
        'Oz',
        'O2',
        'stim']
    chtypes = ['eeg'] * 16 + ['stim']
    X = np.concatenate([S, stim[:, None]], axis=1).T
```

```
    info = mne.create_info(ch_names=chnames, sfreq=512,
                           ch_types=chtypes,
                           verbose=False)
    raw = mne.io.RawArray(data=X, info=info, verbose=False)


    # return the article as a dictionary to identify the article by the id.
    # The key format is 'articleID#'  --->> # represents integer numbers
    return raw
```

[6]:
```
files
```

[6]:
```
['subject_00.mat',
 'subject_01.mat',
 'subject_02.mat',
 'subject_03.mat',
 'subject_04.mat',
 'subject_05.mat',
 'subject_06.mat',
 'subject_08.mat',
 'subject_09.mat',
 'subject_10.mat',
 'subject_11.mat',
 'subject_12.mat',
 'subject_13.mat',
 'subject_14.mat',
 'subject_15.mat',
 'subject_16.mat',
 'subject_17.mat',
 'subject_18.mat',
 'subject_19.mat',
 'subject_20.mat']
```

[7]:
```
raw = get_data(files[0])
```

```
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
21:47:47 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00,  2.31471094e+03, -2.47386426e+03, …,
        -3.27888818e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03,  2.31207739e+03, -2.47719287e+03, …,
        -3.27667456e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03,  2.30813916e+03, -2.47775000e+03, …,
        -3.26313354e+03,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.23244141e+02,  1.92760596e+03, -2.45507520e+03, …,
        -3.27909985e+03,  0.00000000e+00,  0.00000000e+00],
```

```
       [ 1.23246094e+02,  1.93911743e+03, -2.44151099e+03, …,
        -3.25797363e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.23248047e+02,  1.94014917e+03, -2.43970483e+03, …,
        -3.24809204e+03,  0.00000000e+00,  0.00000000e+00]])}
```

[8]:
```
raw
```

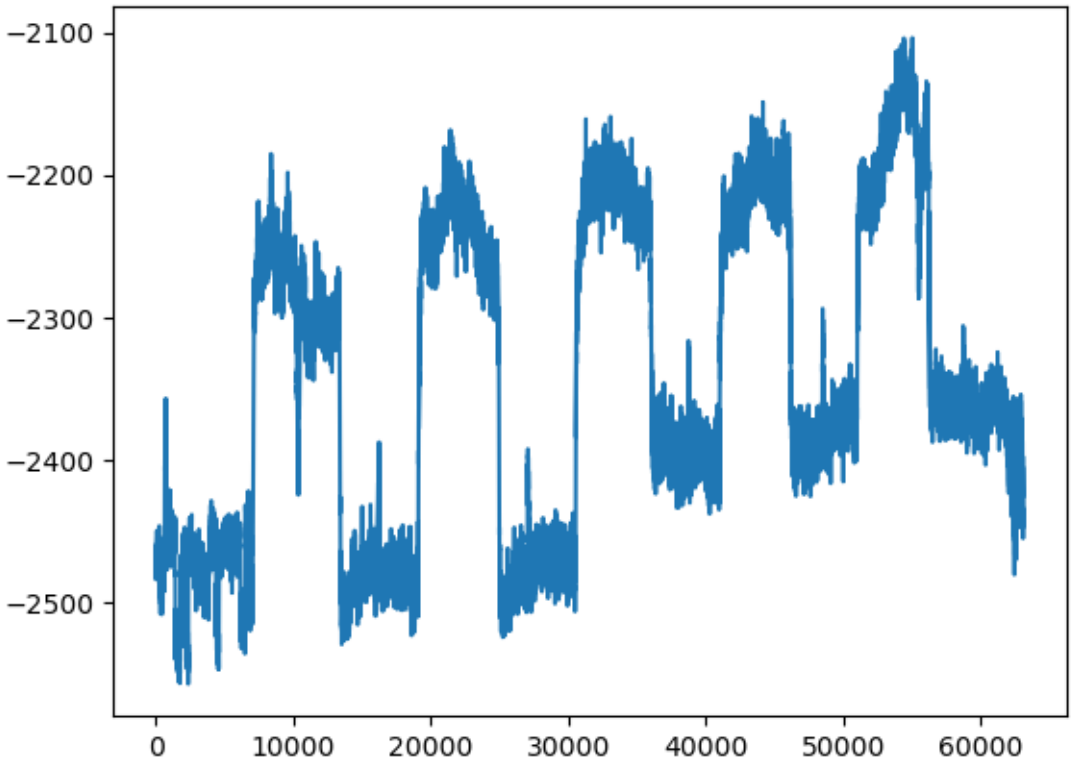
[8]:
```
<RawArray | 17 x 63104 (123.2 s), ~8.2 MB, data loaded>
```

[9]:
```
raw.describe()
```

```
<RawArray | 17 x 63104 (123.2 s), ~8.2 MB, data loaded>
ch  name  type  unit        min         Q1     median         Q3        max
 0  Fp1   EEG   µV    1919117919.90  2181048034.68  2275398925.80  2422695739.77
2577585205.10
 1  Fp2   EEG   µV    -2557157959.00  -2456960937.47  -2362735351.55
-2231222045.88  -2104002685.50
 2  Fc5   EEG   µV    217782211.30  430085990.91  510669525.15  560539947.51
795885375.98
 3  Fz    EEG   µV    -7108479492.20  -6816313232.43  -6781700683.60
-6714244018.57  -6612576660.20
 4  Fc6   EEG   µV    -6534169433.60  -6374752075.20  -6266024902.35
-6208651123.05  -6095611328.10
 5  T7    EEG   µV    -11849524414.00  -11518110351.50  -11400988769.50
-11323991699.50  -11208399414.00
 6  Cz    EEG   µV    -2934821289.10  -2712221618.65  -2555023925.80
-2486022583.05  -2391815185.50
 7  T8    EEG   µV    -14278633789.00  -13982040039.00  -13955098633.00
-13930370361.50  -13821010742.00
 8  P7    EEG   µV    -2057885986.30  -1844913513.15  -1780144714.35
-1737875030.55  -1659822143.60
 9  P3    EEG   µV    -4158770019.50  -3908614929.20  -3739749511.70
-3644156066.90  -3515339599.60
10  Pz    EEG   µV    -6366761230.50  -6145006347.68  -5971622802.75
-5855889892.60  -5726430175.80
11  P4    EEG   µV    -12492496094.00  -12344897460.75  -12311807129.00
-12282636963.00  -12200150391.00
12  P8    EEG   µV    -7381578613.30  -7282695434.55  -7263285156.25
-7240133667.03  -7144030273.40
13  O1    EEG   µV    -59928363.80  26797467.71  52249719.62  84224887.85
241197326.66
14  Oz    EEG   µV    303357452.39  401499443.06  438953948.98  475364578.25
592525390.63
15  O2    EEG   µV    -3355604248.00  -3259574646.00  -3237737304.65
-3215229431.15  -3124399169.90
16  stim  STIM  V          0.00       0.00       0.00       0.00       2.00
```

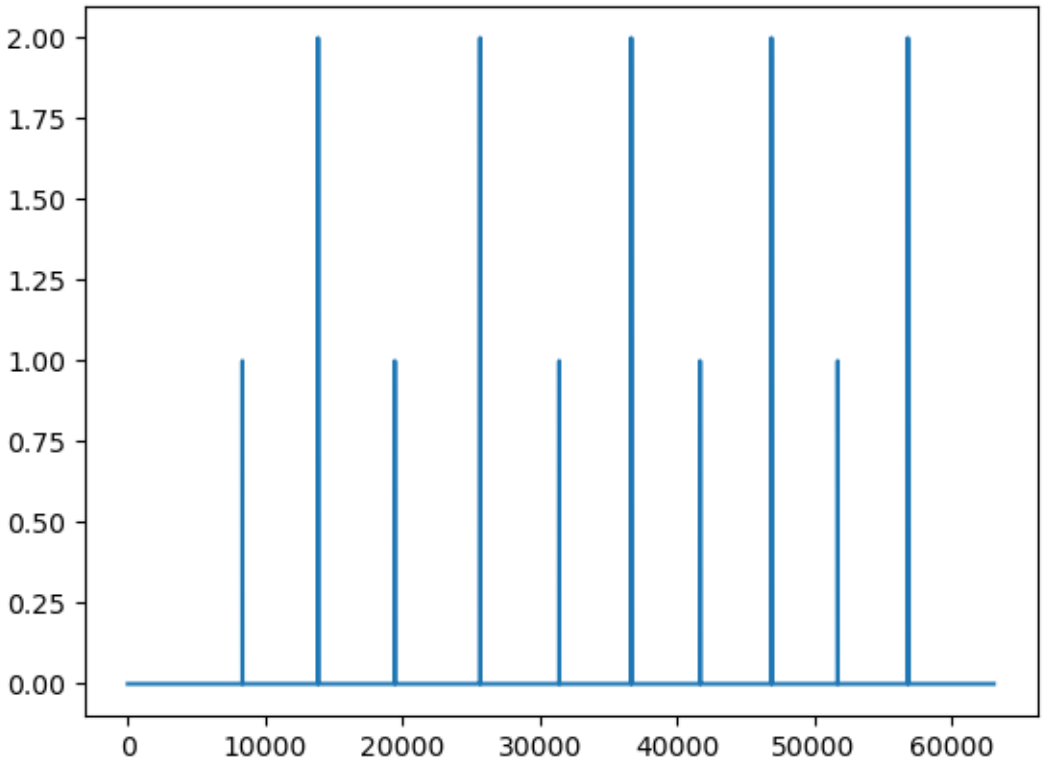
[10]:
```
plt.plot(raw.get_data()[1])
```

[10]:
```
[<matplotlib.lines.Line2D at 0x1af1de06d10>]
```

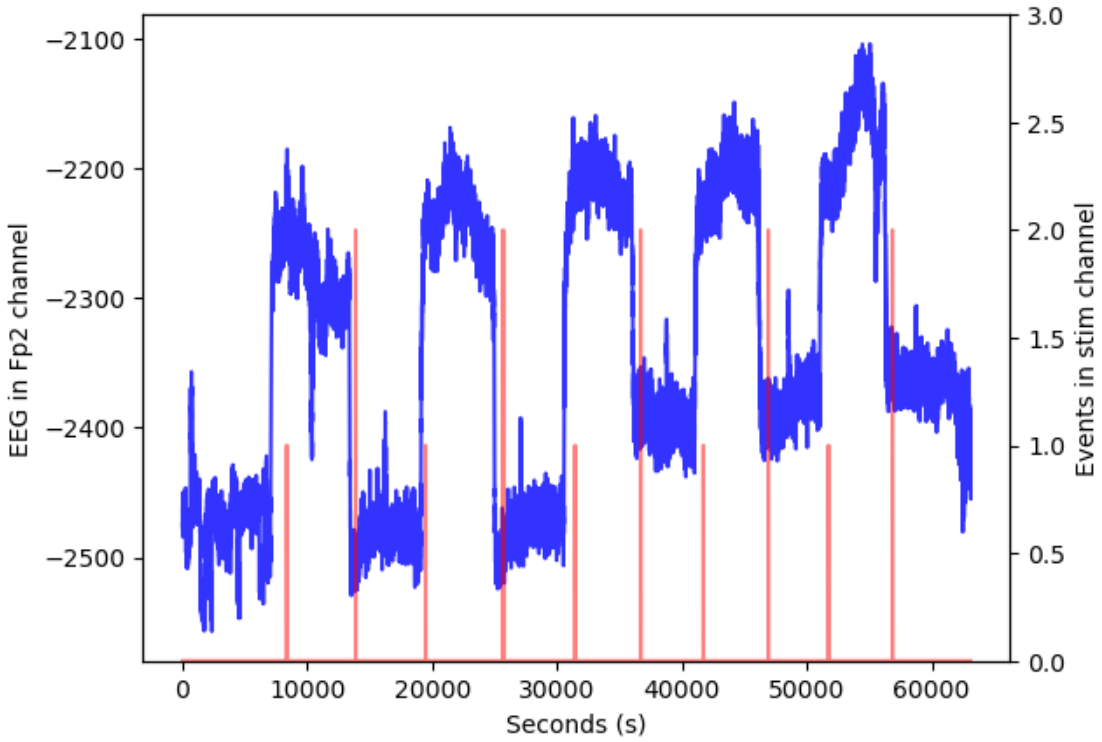


[11]:
```
plt.plot(raw.get_data()[-1])
```

[11]:
```
[<matplotlib.lines.Line2D at 0x1af1e6b9810>]
```



[12]:
```
fig = plt.figure()
ax = fig.add_subplot()
ax2 = ax.twinx()
ax.plot(raw.get_data()[1],c='b',alpha=0.8)
ax2.plot(raw.get_data()[-1],c='r',alpha=0.5)
ax.set_ylabel('EEG in Fp2 channel')
ax2.set_ylabel('Events in stim channel')
ax.set_xlabel('Seconds (s)')
ax2.set_ylim(0,3)
plt.show()
```

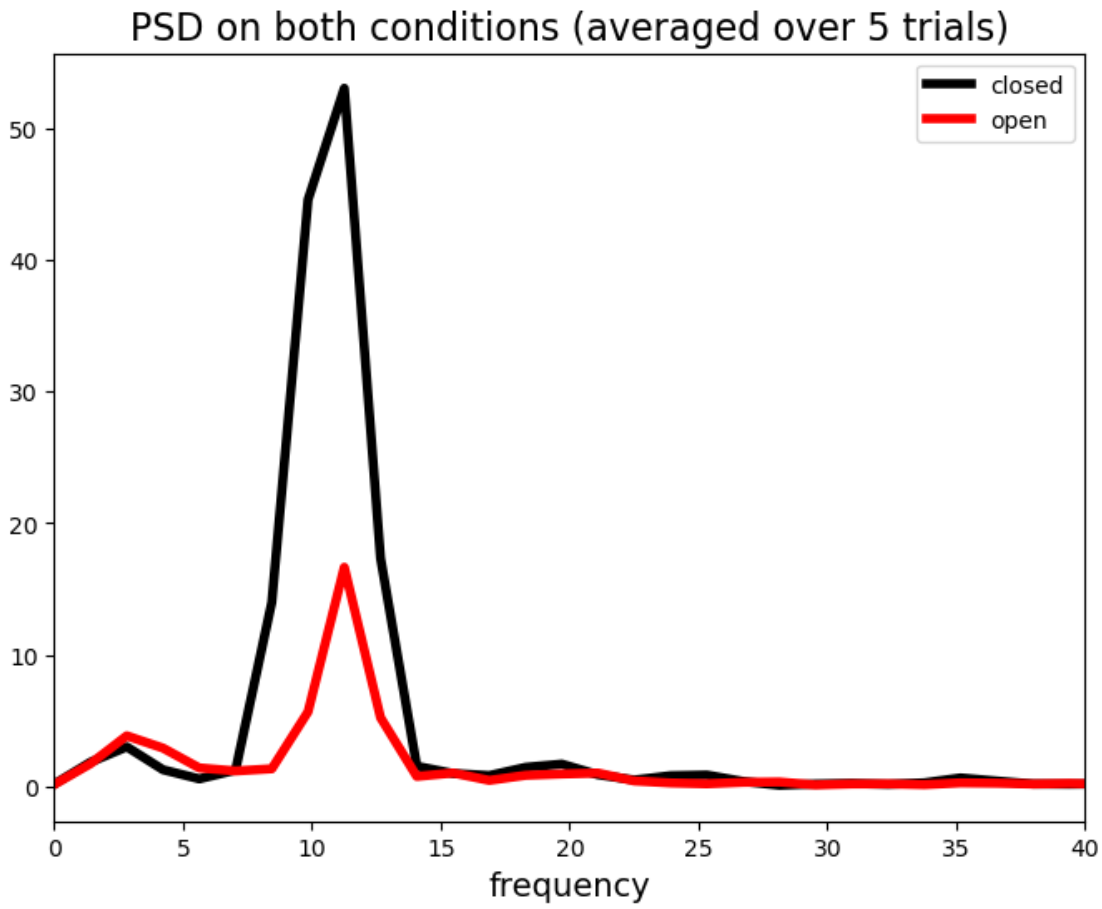


[13]:
```
def processData(raw):
    # filter data and resample
    fmin = 3
    fmax = 40
    raw.filter(fmin, fmax, verbose=False)
    raw.resample(sfreq=128, verbose=False)

    # detect the events and cut the signal into epochs
    events = mne.find_events(raw=raw, shortest_event=1, verbose=False)
    event_id = {'closed': 1, 'open': 2}
    epochs = mne.Epochs(raw, events, event_id, tmin=-0.2, tmax=0.5,␣
 ↪baseline=None,
                        verbose=False,preload=True)
    epochs.pick_types(eeg=True)
    return epochs,events

def PlotIt(epochs,ch):
    epochs.load_data().pick(ch)
    # estimate the averaged spectra for each condition
    X_closed = epochs['closed'].get_data(verbose=False)
    f, S_closed = welch(X_closed, fs=epochs.info['sfreq'], axis=2)
    S_closed = np.mean(S_closed, axis=0).squeeze()
    X_opened = epochs['open'].get_data(verbose=False)
```

```
    f, S_opened = welch(X_opened, fs=epochs.info['sfreq'], axis=2)
    S_opened = np.mean(S_opened, axis=0).squeeze()

    # plot the results
    fig = plt.figure(facecolor='white', figsize=(8, 6))
    plt.plot(f, S_closed, c='k', lw=4.0, label='closed')
    plt.plot(f, S_opened, c='r', lw=4.0, label='open')
    plt.xlim(0, 40)
    plt.xlabel('frequency', fontsize=14)
    plt.title('PSD on both conditions (averaged over 5 trials)', fontsize=16)
    plt.legend()
    plt.show()
```

[14]:
```
epochs, _ = processData(get_data(files[0]))
PlotIt(epochs,'Oz')
```

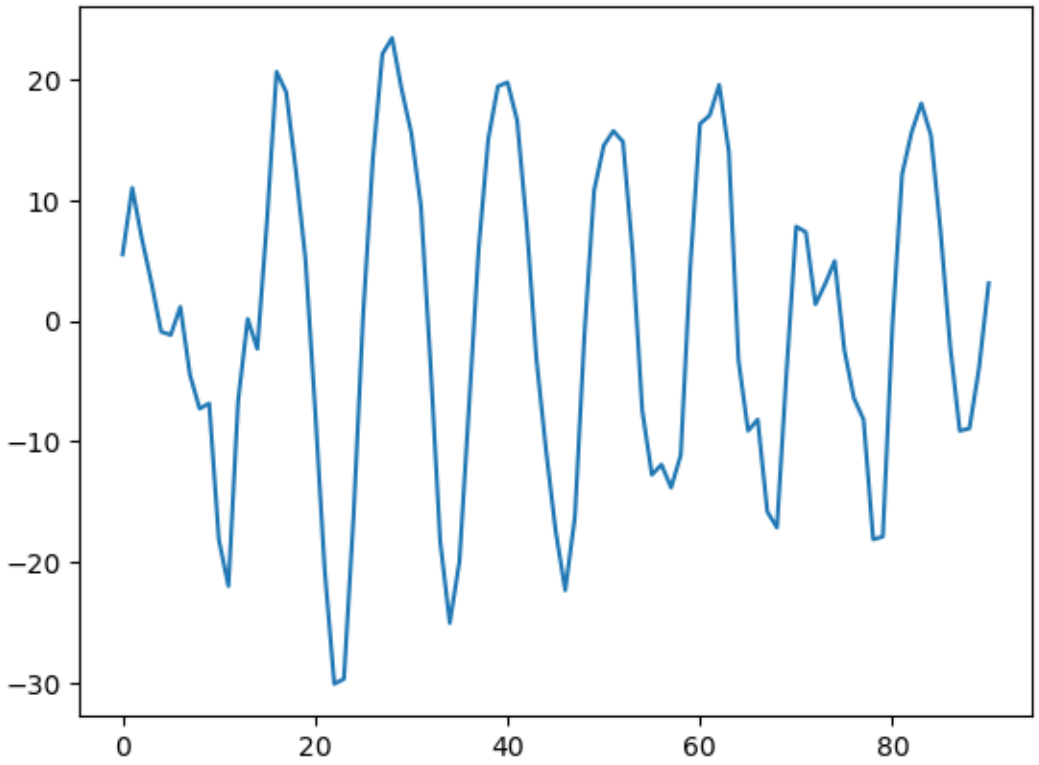
```
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
21:47:47 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00,  2.31471094e+03, -2.47386426e+03, …,
        -3.27888818e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03,  2.31207739e+03, -2.47719287e+03, …,
        -3.27667456e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03,  2.30813916e+03, -2.47775000e+03, …,
        -3.26313354e+03,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.23244141e+02,  1.92760596e+03, -2.45507520e+03, …,
        -3.27909985e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.23246094e+02,  1.93911743e+03, -2.44151099e+03, …,
        -3.25797363e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.23248047e+02,  1.94014917e+03, -2.43970483e+03, …,
        -3.24809204e+03,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).

C:\Users\brian\AppData\Local\Temp\ipykernel_10608\3535682334.py:19:
FutureWarning: The current default of copy=False will change to copy=True in
1.7. Set the value of copy explicitly to avoid this warning
  X_closed = epochs['closed'].get_data(verbose=False)
C:\Users\brian\Anaconda3\Lib\site-packages\scipy\signal\_spectral_py.py:600:
UserWarning: nperseg = 256 is greater than input length  = 91, using nperseg =
91
  freqs, _, Pxy = _spectral_helper(x, y, fs, window, nperseg, noverlap,
C:\Users\brian\AppData\Local\Temp\ipykernel_10608\3535682334.py:22:
FutureWarning: The current default of copy=False will change to copy=True in
1.7. Set the value of copy explicitly to avoid this warning
  X_opened = epochs['open'].get_data(verbose=False)
```



[15]:
```
plt.plot(epochs.get_data()[0].ravel())
```

```
C:\Users\brian\AppData\Local\Temp\ipykernel_10608\4069028416.py:1:
FutureWarning: The current default of copy=False will change to copy=True in
1.7. Set the value of copy explicitly to avoid this warning
  plt.plot(epochs.get_data()[0].ravel())
```

[15]:
```
[<matplotlib.lines.Line2D at 0x1af0fc32850>]
```



[16]:
```
import warnings
warnings.filterwarnings("ignore")

raw = get_data(files[0])

# filter data and resample
fmin = 3
fmax = 40
raw.filter(fmin, fmax, verbose=False)
raw.resample(sfreq=128, verbose=False)

# detect the events and cut the signal into epochs
events = mne.find_events(raw=raw, shortest_event=1, verbose=False)
event_id = {'closed': 1, 'open': 2}
epochs = mne.Epochs(raw, events, event_id, tmin=-0.2, tmax=0.2, baseline=None,
                    verbose=False)
epochs.load_data().pick(['Oz'])

# get trials and labels
shape_x = epochs.get_data(verbose=False).shape[0]
shape_y = epochs.get_data(verbose=False).shape[2]
X = epochs.get_data().ravel().reshape((shape_x,shape_y))
```

```
y = events[:, -1]

print('Shape of X: ', X.shape)
print('Shape of y: ', y.shape)

# cross validation
skf = StratifiedKFold(n_splits=5)
clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
scr = cross_val_score(clf, X, y, cv=skf)
preds = cross_val_predict(clf, X, y, cv=skf)

# print results of classification
print('subject', files[0])
print('mean SVC accuracy :', scr.mean())
```

```
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
21:47:47 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00,  2.31471094e+03, -2.47386426e+03, …,
        -3.27888818e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03,  2.31207739e+03, -2.47719287e+03, …,
        -3.27667456e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03,  2.30813916e+03, -2.47775000e+03, …,
        -3.26313354e+03,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.23244141e+02,  1.92760596e+03, -2.45507520e+03, …,
        -3.27909985e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.23246094e+02,  1.93911743e+03, -2.44151099e+03, …,
        -3.25797363e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.23248047e+02,  1.94014917e+03, -2.43970483e+03, …,
        -3.24809204e+03,  0.00000000e+00,  0.00000000e+00]])}
Using data from preloaded Raw for 10 events and 53 original time points …
0 bad epochs dropped
Shape of X:  (10, 53)
Shape of y:  (10,)
subject subject_00.mat
mean SVC accuracy : 0.9
```

[21]:
```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪8,shuffle=True)
model = SVC(gamma='auto')
SS = StandardScaler().fit(X_train)
X_train = SS.transform(X_train)
model.fit(X_train,y_train)
y_pred = model.predict(SS.transform(X_test))
print('SCV accuracy score: ', accuracy_score(y_test, y_pred))
```

```
SCV accuracy score:  1.0
```

[22]:
```
model.get_params()
```

[22]:
```
{'C': 1.0,
 'break_ties': False,
 'cache_size': 200,
 'class_weight': None,
 'coef0': 0.0,
 'decision_function_shape': 'ovr',
 'degree': 3,
 'gamma': 'auto',
 'kernel': 'rbf',
 'max_iter': -1,
 'probability': False,
 'random_state': None,
 'shrinking': True,
 'tol': 0.001,
 'verbose': False}
```

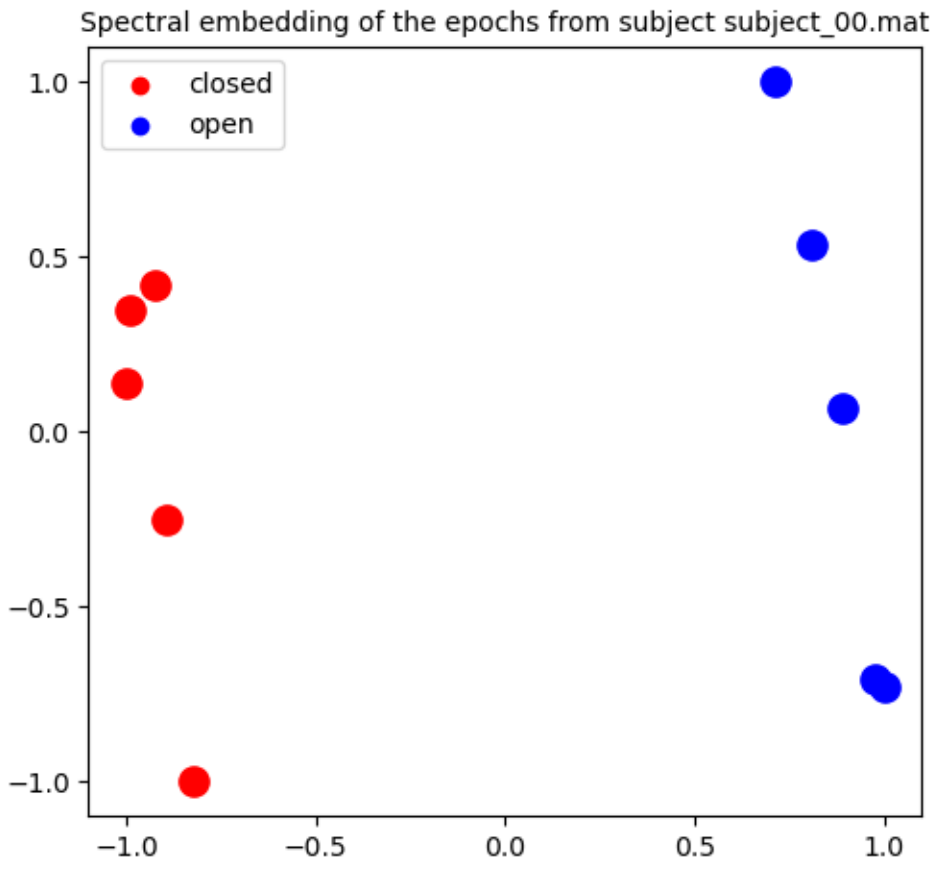
[23]:
```
subject = files[0]

# filter data and resample
fmin = 3
fmax = 40
raw.filter(fmin, fmax, verbose=False)
raw.resample(sfreq=128, verbose=False)

# detect the events and cut the signal into epochs
events = mne.find_events(raw=raw, shortest_event=1, verbose=False)
event_id = {'closed': 1, 'open': 2}
epochs = mne.Epochs(raw, events, event_id, tmin=2.0, tmax=8.0, baseline=None,
                    verbose=False, preload=True)
epochs.pick_types(eeg=True)

# get trials and labels
X = epochs.get_data()
y = events[:, -1]
```

```
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
```

[24]:
```
# cross validation
skf = StratifiedKFold(n_splits=5)
clf = make_pipeline(Covariances(estimator='lwf'), MDM())
scr = cross_val_score(clf, X, y, cv=skf)

# print results of classification
```

```
print('subject', subject)
print('mean accuracy :', scr.mean())

# get the spectral embedding of the epochs
C = Covariances(estimator='lwf').fit_transform(X)
emb = SpectralEmbedding(metric='riemann').fit_transform(C)

# scatter plot of the embedded points
fig = plt.figure(facecolor='white', figsize=(5.6, 5.2))
colors = {1: 'r', 2: 'b'}
for embi, yi in zip(emb, y):
    plt.scatter(embi[0], embi[1], s=120, c=colors[yi])
labels = {1: 'closed', 2: 'open'}
for yi in np.unique(y):
    plt.scatter([], [], c=colors[yi], label=labels[yi])
plt.xticks([-1, -0.5, 0.0, +0.5, 1.0])
plt.yticks([-1, -0.5, 0.0, +0.5, 1.0])
plt.legend()
plt.title(
    'Spectral embedding of the epochs from subject ' +
    str(subject),
    fontsize=10)

plt.show()
```

```
subject subject_00.mat
mean accuracy : 1.0
```



[25]:
```
chnames = [
        'Fp1',
        'Fp2',
        'Fc5',
        'Fz',
        'Fc6',
        'T7',
        'Cz',
        'T8',
        'P7',
        'P3',
        'Pz',
        'P4',
        'P8',
        'O1',
        'Oz',
        'O2',
        'stim']
```

```
n_channels = len(chnames)
n_channels
```

[25]: 17

[26]:
```
epochs.get_data()
```

[26]:
```
array([[[-13.6555626 ,  -8.43792647,   0.4327612 , …,  -2.0590503 ,
           3.62017393,   4.05957531],
         [-10.48512907,  -7.89960328,  -1.62631367, …,   1.7644762 ,
           7.90762893,   6.49737362],
         [-13.90599708,  -6.97197255,  -0.16718558, …,  -8.07524231,
          -10.16628791,  -9.80080508],
         …,
         [  0.67555152,  -9.44551574, -22.27072963, …,  20.36382956,
          21.77589502,  15.82469824],
         [ -4.16323738, -14.96201388, -26.33704855, …,  16.62561418,
          23.34160815,  21.40371071],
         [ -0.54926781, -13.0079802 , -27.22551531, …,  17.9207628 ,
          25.05806864,  24.23546433]],

        [[  9.08259479,   1.42127325,  -4.74131195, …,   8.44791783,
          17.52587207,  13.46316649],
         [  5.19883551,  -0.86217908,  -5.96154346, …,   6.26591319,
          13.34935061,  12.38799764],
         [  6.45401865,   2.71203585,  -4.01962687, …,   4.34449445,
          10.20105637,   6.67852865],
         …,
         [  5.04639053,   2.92610321,  -2.4069871 , …,  -6.12085808,
          -2.83078385,  -6.14671001],
         [  4.98113555,   0.88841201,  -3.45241793, …,  -4.53116472,
          -3.32835817,  -7.10035041],
         [  7.58278984,   2.03818742,  -2.3308292 , …,  -6.37556097,
          -6.28058307,  -9.61317534]],

        [[-11.86244747, -10.75712284,   2.30726066, …, -14.17802078,
          -8.35864349,   0.852987  ],
         [-10.79943899,  -9.13574389,   4.46393184, …, -17.06846499,
          -9.33469447,  -0.07023168],
         [ -9.07556948,  -4.44429625,   4.82379401, …, -12.98168155,
          -5.84712595,  -1.46866741],
         …,
         [ -0.91762697,  -1.48379098,  -7.39504874, …, -10.12738497,
          -6.05181872, -10.9871218 ],
         [  5.06838931,   5.43640206,  -3.45162226, …,  -5.70508822,
           1.76535531,  -0.40568202],
         [  4.52786474,   3.73019228,  -4.45365841, …,  -3.9429515 ,
```

```
    7.1824958 ,   8.7959712 ]],

…,

[[ -7.06715902,  -0.52234393,   4.72398417, …,  -4.39060246,
   -3.03674347, -10.50143253],
 [ -8.82417874,  -7.10107015,  -3.76516207, …,  -3.01441733,
   -0.04407106,  -5.860927  ],
 [  2.11008707,   7.13523497,  10.35324804, …,  -8.3767487 ,
   -7.68127331,  -6.24687753],
 …,
 [ -7.61232438, -10.95782846,  -9.00750419, …, -15.17733387,
    0.6914643 ,  10.05859154],
 [ -2.28891232,  -9.84693475, -10.58919842, …, -14.80438838,
    0.72807185,   7.85226094],
 [ -3.38688211,  -9.52510448, -10.27788599, …, -11.31535441,
    2.53959133,   6.75854273]],

[[-12.80041812, -14.74161471,  -8.90277784, …,  11.70500733,
   13.23699829,   5.67738692],
 [-11.47658181, -15.10948623, -10.04814194, …,  12.97422089,
   13.99446237,   7.41266786],
 [-12.6164026 , -12.29179376,  -7.92436024, …,  10.0523497 ,
    7.89240907,  -1.60708645],
 …,
 [  3.57438491,   0.1407641 ,  -6.91930374, …,   0.4271212 ,
   -3.40261315, -13.80128836],
 [  8.66381758,   5.40048246,  -0.20898646, …,  -2.2513894 ,
   -4.79887071, -12.88192021],
 [ 23.13010365,  19.58619838,   9.06891696, …, -11.01830444,
   -5.78442092,  -6.09497366]],

[[  3.09568992,  -2.46014788,  -0.42801347, …,  -7.33413876,
   -4.77736963,  -2.59912141],
 [ -1.16399306,  -3.57570039,   0.73842446, …,  -2.4619443 ,
    0.456143  ,   1.74138737],
 [  3.66284584,   0.02936647,   1.59045446, …,  -7.54957996,
   -3.6326316 ,   0.61129584],
 …,
 [ -2.11252461,  -0.86398252,   1.38224153, …, -12.20384182,
   -7.25631241,   0.92016556],
 [ -4.69130621,  -4.72061322,  -2.16445153, …, -15.91056204,
  -11.11016716,  -2.91553307],
 [ -3.56166695,  -7.34666083, -10.20057476, …, -15.02658337,
  -12.50822543,  -4.91150708]]])
```

[27]:
```
files
```

[27]:
```
['subject_00.mat',
 'subject_01.mat',
 'subject_02.mat',
 'subject_03.mat',
 'subject_04.mat',
 'subject_05.mat',
 'subject_06.mat',
 'subject_08.mat',
 'subject_09.mat',
 'subject_10.mat',
 'subject_11.mat',
 'subject_12.mat',
 'subject_13.mat',
 'subject_14.mat',
 'subject_15.mat',
 'subject_16.mat',
 'subject_17.mat',
 'subject_18.mat',
 'subject_19.mat',
 'subject_20.mat']
```

[28]:
```
events.shape
```

[28]:
```
(10, 3)
```

[29]:
```
np.array(chnames).shape
```

[29]:
```
(17,)
```

[30]:
```
X = []
y = []
for file in files:
    data = get_data(file)
    processed,events = processData(data)
    for processed_one, event in zip(processed,events):
        X.append(processed_one)
        y.append(event[-1])
```

```
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
21:47:47 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00,  2.31471094e+03, -2.47386426e+03, …,
        -3.27888818e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03,  2.31207739e+03, -2.47719287e+03, …,
        -3.27667456e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03,  2.30813916e+03, -2.47775000e+03, …,
        -3.26313354e+03,  0.00000000e+00,  0.00000000e+00],
       …,
```

```
       [ 1.23244141e+02,  1.92760596e+03, -2.45507520e+03, …,
        -3.27909985e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.23246094e+02,  1.93911743e+03, -2.44151099e+03, …,
        -3.25797363e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.23248047e+02,  1.94014917e+03, -2.43970483e+03, …,
        -3.24809204e+03,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
22:14:22 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00, -1.66397969e+04,  1.07526416e+03, …,
        -1.97763594e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03, -1.66351543e+04,  1.07932727e+03, …,
        -1.97789062e+04,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03, -1.66370605e+04,  1.07823169e+03, …,
        -1.97734941e+04,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 2.33994141e+02, -1.83469707e+04, -4.33083130e+02, …,
        -2.30812480e+04,  0.00000000e+00,  0.00000000e+00],
       [ 2.33996094e+02, -1.83304590e+04, -4.15464417e+02, …,
        -2.30606270e+04,  0.00000000e+00,  0.00000000e+00],
       [ 2.33998047e+02, -1.83295391e+04, -4.17637817e+02, …,
        -2.30502891e+04,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Fri Dec 14
11:04:14 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00, -1.43192041e+04, -1.19112295e+04, …,
        -1.20157383e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03, -1.43258682e+04, -1.19135254e+04, …,
        -1.20199561e+04,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03, -1.43321328e+04, -1.19256230e+04, …,
        -1.20203779e+04,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 3.58681641e+02, -1.49280137e+04, -1.25770703e+04, …,
        -1.29394561e+04,  0.00000000e+00,  0.00000000e+00],
       [ 3.58683594e+02, -1.49185469e+04, -1.25556514e+04, …,
        -1.29132471e+04,  0.00000000e+00,  0.00000000e+00],
       [ 3.58685547e+02, -1.49115693e+04, -1.25713750e+04, …,
        -1.29206279e+04,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
22:41:32 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL':
array([[0.00000000e+00, 2.09764297e+04, 3.29868604e+03, …,
        2.09602559e+04, 0.00000000e+00, 0.00000000e+00],
       [1.95312500e-03, 2.09743809e+04, 3.29208496e+03, …,
        2.09537383e+04, 0.00000000e+00, 0.00000000e+00],
       [3.90625000e-03, 2.09910020e+04, 3.31095776e+03, …,
        2.09655527e+04, 0.00000000e+00, 0.00000000e+00],
       …,
```

```
       [1.77869141e+02, 1.99884473e+04, 3.35226147e+03, …,
        2.05186797e+04, 0.00000000e+00, 0.00000000e+00],
       [1.77871094e+02, 1.99924277e+04, 3.35513647e+03, …,
        2.05217227e+04, 0.00000000e+00, 0.00000000e+00],
       [1.77873047e+02, 1.99918926e+04, 3.35940601e+03, …,
        2.05249844e+04, 0.00000000e+00, 0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
22:43:52 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00, -1.16675029e+04, -1.09190381e+04, …,
       -4.41920801e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03, -1.16667725e+04, -1.09165732e+04, …,
        -4.40579785e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03, -1.16702510e+04, -1.09149277e+04, …,
        -4.41653662e+03,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.90806641e+02, -1.04315381e+04, -1.00058809e+04, …,
        -4.31797070e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.90808594e+02, -1.04291855e+04, -9.99989355e+03, …,
        -4.31083057e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.90810547e+02, -1.04298799e+04, -1.00003281e+04, …,
        -4.31381836e+03,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
22:45:01 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00, -8.67503711e+03, -7.47208545e+03, …,
         9.75596973e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03, -8.68208496e+03, -7.48723438e+03, …,
         9.73640625e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03, -8.67931934e+03, -7.47989355e+03, …,
         9.74194336e+03,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.85806641e+02, -8.47913477e+03, -7.63415430e+03, …,
         1.04561074e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.85808594e+02, -8.47683789e+03, -7.63309912e+03, …,
         1.04549609e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.85810547e+02, -8.48259961e+03, -7.64177637e+03, …,
         1.04554639e+04,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Fri Dec 14
11:16:59 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00, -1.32752217e+04,  9.74668945e+03, …,
         5.31288184e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03, -1.32811973e+04,  9.74449707e+03, …,
         5.30222412e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03, -1.32782285e+04,  9.74534277e+03, …,
         5.30332812e+03,  0.00000000e+00,  0.00000000e+00],
       …,
```

```
       [ 1.89681641e+02, -1.26332246e+04,  8.52971094e+03, …,
         5.86034424e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.89683594e+02, -1.26366846e+04,  8.51386328e+03, …,
         5.86067041e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.89685547e+02, -1.26396006e+04,  8.51233398e+03, …,
         5.85458105e+03,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
23:19:30 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00, -4.67942871e+03, -1.23046592e+04, …,
        -9.56047168e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03, -4.68185889e+03, -1.23147100e+04, …,
        -9.56731836e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03, -4.69359814e+03, -1.23193418e+04, …,
        -9.56896777e+03,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.53931641e+02, -5.60968701e+03, -1.24945840e+04, …,
        -1.07021465e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.53933594e+02, -5.59143115e+03, -1.24841650e+04, …,
        -1.07201289e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.53935547e+02, -5.58993701e+03, -1.24699414e+04, …,
        -1.07867490e+04,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
23:32:48 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00, -6.26028418e+03,  1.40598828e+04, …,
         1.31332812e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03, -6.26097021e+03,  1.40629668e+04, …,
         1.30697295e+04,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03, -6.26174854e+03,  1.40487725e+04, …,
         1.30442979e+04,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.44744141e+02, -5.40137256e+03,  1.66278926e+04, …,
         1.41594746e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.44746094e+02, -5.40583398e+03,  1.66126035e+04, …,
         1.41147744e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.44748047e+02, -5.41339014e+03,  1.66347832e+04, …,
         1.41448164e+04,  0.00000000e+00,  0.00000000e+00]])}

NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Fri Dec 14
09:53:31 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00, -2.01390488e+04, -9.87133984e+03, …,
         7.71173486e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03, -2.01402305e+04, -9.87845508e+03, …,
         7.70625684e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03, -2.01317832e+04, -9.87711328e+03, …,
         7.70870264e+03,  0.00000000e+00,  0.00000000e+00],
```

```
       …,
       [ 1.54806641e+02, -2.20300117e+04, -8.58470508e+03, …,
         7.34203027e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.54808594e+02, -2.20374902e+04, -8.59333301e+03, …,
         7.37336182e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.54810547e+02, -2.20344863e+04, -8.59097168e+03, …,
         7.40634424e+03,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
23:35:08 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00, -7.00834961e+03, -1.63120527e+04, …,
        -1.13427979e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03, -7.00008691e+03, -1.62987773e+04, …,
        -1.13067783e+04,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03, -6.99757275e+03, -1.62960986e+04, …,
        -1.12903047e+04,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.74681641e+02, -7.08215479e+03, -1.65250000e+04, …,
        -1.17324180e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.74683594e+02, -7.08532129e+03, -1.65265312e+04, …,
        -1.16824404e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.74685547e+02, -7.08412207e+03, -1.65231641e+04, …,
        -1.16614248e+04,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
23:38:05 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00, -1.85819946e+03,  6.92476807e+03, …,
         4.49447510e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03, -1.85853528e+03,  6.92343701e+03, …,
         4.49389111e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03, -1.85632800e+03,  6.92651221e+03, …,
         4.49802930e+03,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.70931641e+02, -1.85728967e+03,  5.70990039e+03, …,
         3.24578223e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.70933594e+02, -1.86786768e+03,  5.70083496e+03, …,
         3.22943213e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.70935547e+02, -1.85912231e+03,  5.70958350e+03, …,
         3.24362134e+03,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
23:39:13 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00, -2.97488940e+03, -2.99532715e+03, …,
         1.52550371e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03, -2.97282837e+03, -2.99205396e+03, …,
         1.52665391e+04,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03, -2.97443359e+03, -2.99176294e+03, …,
         1.52600264e+04,  0.00000000e+00,  0.00000000e+00],
```

```
       …,
       [ 1.67056641e+02, -2.70361426e+03, -4.12027979e+03, …,
         1.44794385e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.67058594e+02, -2.70550684e+03, -4.11976807e+03, …,
         1.44671611e+04,  0.00000000e+00,  0.00000000e+00],
       [ 1.67060547e+02, -2.71769385e+03, -4.13039307e+03, …,
         1.44537490e+04,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Thu Dec 13
23:40:59 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00,  6.66329785e+03,  1.35169512e+04, …,
        -9.80967346e+02,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03,  6.65918994e+03,  1.35172695e+04, …,
        -9.78917603e+02,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03,  6.65212451e+03,  1.35181104e+04, …,
        -9.84906372e+02,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.51369141e+02,  5.53879395e+03,  1.35083740e+04, …,
        -3.37770825e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.51371094e+02,  5.53934961e+03,  1.35063486e+04, …,
        -3.38266895e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.51373047e+02,  5.55110059e+03,  1.35109824e+04, …,
        -3.38012964e+03,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Fri Dec 14
08:19:58 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00,  6.50966187e+02,  1.31349014e+04, …,
        -1.16189758e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03,  6.59100220e+02,  1.31437432e+04, …,
        -1.14486267e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03,  6.72222290e+02,  1.31551523e+04, …,
        -1.13096497e+03,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.51806641e+02,  1.13935699e+02,  1.32290088e+04, …,
        -2.47190210e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.51808594e+02,  1.13131226e+02,  1.32268984e+04, …,
        -2.46155566e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.51810547e+02,  1.01362976e+02,  1.32233242e+04, …,
        -2.46410571e+03,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Fri Dec 14
09:28:35 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL':
array([[0.00000000e+00, 8.45403516e+03, 5.90149072e+03, …,
        8.03112012e+03, 0.00000000e+00, 0.00000000e+00],
       [1.95312500e-03, 8.46168359e+03, 5.91022607e+03, …,
        8.03578320e+03, 0.00000000e+00, 0.00000000e+00],
       [3.90625000e-03, 8.45612598e+03, 5.90156592e+03, …,
        8.02327344e+03, 0.00000000e+00, 0.00000000e+00],
```

```
       …,
       [1.78556641e+02, 5.81560938e+03, 2.92037036e+03, …,
        6.82006885e+03, 0.00000000e+00, 0.00000000e+00],
       [1.78558594e+02, 5.81003320e+03, 2.92939258e+03, …,
        6.81040771e+03, 0.00000000e+00, 0.00000000e+00],
       [1.78560547e+02, 5.80005176e+03, 2.92478735e+03, …,
        6.79930469e+03, 0.00000000e+00, 0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Fri Dec 14
09:31:45 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00, -3.98915967e+03, -5.20346338e+03, …,
         2.04544324e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03, -3.97775830e+03, -5.19722900e+03, …,
         2.05376196e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03, -3.98936279e+03, -5.20853809e+03, …,
         2.06386157e+03,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.65306641e+02, -3.38122339e+03, -3.77437915e+03, …,
         3.38761597e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.65308594e+02, -3.39617944e+03, -3.78047290e+03, …,
         3.37003516e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.65310547e+02, -3.39115747e+03, -3.78438892e+03, …,
         3.37148560e+03,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Fri Dec 14
09:33:23 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00, -3.25018799e+03, -1.16482900e+04, …,
        -5.19501904e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03, -3.26420630e+03, -1.16450293e+04, …,
        -5.19266064e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03, -3.24926611e+03, -1.16278750e+04, …,
        -5.22086572e+03,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.75431641e+02, -4.45101709e+03, -1.20219932e+04, …,
        -5.88204785e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.75433594e+02, -4.41144043e+03, -1.19993711e+04, …,
        -5.85838770e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.75435547e+02, -4.41900000e+03, -1.19973506e+04, …,
        -5.85413867e+03,  0.00000000e+00,  0.00000000e+00]])}

NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Fri Dec 14
09:36:22 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL':
array([[0.00000000e+00, 2.34733438e+05, 2.36993516e+05, …,
        2.33703922e+05, 0.00000000e+00, 0.00000000e+00],
       [1.95312500e-03, 2.34741844e+05, 2.37002516e+05, …,
        2.33717125e+05, 0.00000000e+00, 0.00000000e+00],
       [3.90625000e-03, 2.34771469e+05, 2.37035047e+05, …,
```

```
        2.33748859e+05, 0.00000000e+00, 0.00000000e+00],
       …,
       [1.77556641e+02, 2.34507750e+05, 2.36753484e+05, …,
        2.33383859e+05, 0.00000000e+00, 0.00000000e+00],
       [1.77558594e+02, 2.34579906e+05, 2.36819281e+05, …,
        2.33468031e+05, 0.00000000e+00, 0.00000000e+00],
       [1.77560547e+02, 2.34606703e+05, 2.36851562e+05, …,
        2.33487109e+05, 0.00000000e+00, 0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Fri Dec 14
09:37:33 2018', '__version__': '1.0', '__globals__': [], 'SIGNAL': array([[
0.00000000e+00,  3.07348340e+03,  3.46623901e+03, …,
       -1.47142151e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.95312500e-03,  3.07292627e+03,  3.46863574e+03, …,
        -1.47281995e+03,  0.00000000e+00,  0.00000000e+00],
       [ 3.90625000e-03,  3.07927344e+03,  3.47361621e+03, …,
        -1.46125806e+03,  0.00000000e+00,  0.00000000e+00],
       …,
       [ 1.59681641e+02,  3.65135474e+03,  3.65933667e+03, …,
        -2.03263623e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.59683594e+02,  3.64728296e+03,  3.64336548e+03, …,
        -2.04652637e+03,  0.00000000e+00,  0.00000000e+00],
       [ 1.59685547e+02,  3.65707983e+03,  3.64766870e+03, …,
        -2.03040259e+03,  0.00000000e+00,  0.00000000e+00]])}
NOTE: pick_types() is a legacy function. New code should use inst.pick(…).
```

[31]:
```
X = np.array(X)
print('X shape: ', X.shape)
y = np.array(y)
print('y shape: ', y.shape)
```

```
X shape:  (200, 16, 91)
y shape:  (200,)
```

[32]:
```
X_trans = Covariances(estimator='lwf').fit_transform(X,y)
```

[33]:
```
X_trans = np.array(X_trans)
print('X shape: ', X_trans.shape)
```

```
X shape:  (200, 16, 16)
```

[34]:
```
X_trans_picked = X_trans[:,8,:]
```

[35]:
```
X_trans_picked.shape
```

[35]:
```
(200, 16)
```

[36]:
```
#X_trans_picked = X_trans.reshape(X_trans.shape[0],-1)
```

[37]:
```
# cross validation
skf = StratifiedKFold(n_splits=5)
model = SVC(kernel='rbf', gamma='auto')
scr = cross_val_score(model, X_trans_picked, y, cv=skf)
preds = cross_val_predict(model, X_trans_picked, y, cv=skf)

# print results of classification
print('mean accuracy :', scr.mean())
```

```
mean accuracy : 0.53
```

[38]:
```
X_trans_picked.shape
```

[38]:
```
(200, 16)
```

[39]:
```
len(chnames)
```

[39]:
```
17
```

[40]:
```
acc_test = {}
for i,ch in zip(range(X_trans.shape[1]),chnames):
    print('-'*100)
    print(ch)
    X_trans_picked = X_trans[:,i,:]
    # cross validation
    skf = StratifiedKFold(n_splits=5)
    svc = SVC(kernel='rbf', gamma='auto')
    scr = cross_val_score(svc, X_trans_picked, y, cv=skf)
    preds = cross_val_predict(svc, X_trans_picked, y, cv=skf)

    # print results of classification
    print('mean accuracy :', scr.mean())
    acc_test[ch] = scr.mean()

    grid_SVC ={'C':list(range(1,5)),
                'kernel':['linear','poly','rbf','sigmoid'],
                'degree':list(range(1,5)),
                'gamma':['auto','scale'],

            }
    grid_svc = GridSearchCV(svc, grid_SVC, cv=skf, scoring='accuracy')
    grid_svc.fit(X_trans_picked, y)
    print("Best parameters set found on development set:")
    print(grid_svc.best_params_)
    print()
    print("Grid best score:")
    print (grid_svc.best_score_)
```

```
--------------------------------------------------------------------------------
--------------------
Fp1
mean accuracy : 0.515
Best parameters set found on development set:
{'C': 3, 'degree': 1, 'gamma': 'scale', 'kernel': 'rbf'}

Grid best score:
0.8099999999999999
--------------------------------------------------------------------------------
--------------------
Fp2
mean accuracy : 0.52
Best parameters set found on development set:
{'C': 1, 'degree': 1, 'gamma': 'auto', 'kernel': 'poly'}

Grid best score:
0.7700000000000001
--------------------------------------------------------------------------------
--------------------
Fc5
mean accuracy : 0.5349999999999999
Best parameters set found on development set:
{'C': 1, 'degree': 1, 'gamma': 'auto', 'kernel': 'linear'}

Grid best score:
0.7799999999999999
--------------------------------------------------------------------------------
--------------------
Fz
mean accuracy : 0.5399999999999999
Best parameters set found on development set:
{'C': 1, 'degree': 2, 'gamma': 'auto', 'kernel': 'poly'}

Grid best score:
0.8099999999999999
--------------------------------------------------------------------------------
--------------------
Fc6
mean accuracy : 0.52
Best parameters set found on development set:
{'C': 4, 'degree': 1, 'gamma': 'scale', 'kernel': 'poly'}

Grid best score:
0.765
--------------------------------------------------------------------------------
--------------------
T7
```

```
mean accuracy : 0.5
Best parameters set found on development set:
{'C': 4, 'degree': 2, 'gamma': 'scale', 'kernel': 'poly'}

Grid best score:
0.715
--------------------------------------------------------------------------------
--------------------
Cz
mean accuracy : 0.5249999999999999
Best parameters set found on development set:
{'C': 2, 'degree': 1, 'gamma': 'auto', 'kernel': 'poly'}

Grid best score:
0.78
--------------------------------------------------------------------------------
--------------------
T8
mean accuracy : 0.505
Best parameters set found on development set:
{'C': 3, 'degree': 1, 'gamma': 'scale', 'kernel': 'rbf'}

Grid best score:
0.695
--------------------------------------------------------------------------------
--------------------
P7
mean accuracy : 0.53
Best parameters set found on development set:
{'C': 1, 'degree': 4, 'gamma': 'auto', 'kernel': 'poly'}

Grid best score:
0.775
--------------------------------------------------------------------------------
--------------------
P3
mean accuracy : 0.51
Best parameters set found on development set:
{'C': 1, 'degree': 1, 'gamma': 'auto', 'kernel': 'linear'}

Grid best score:
0.7849999999999999
--------------------------------------------------------------------------------
--------------------
Pz
mean accuracy : 0.51
Best parameters set found on development set:
{'C': 3, 'degree': 1, 'gamma': 'scale', 'kernel': 'rbf'}
```

```
Grid best score:
0.77
--------------------------------------------------------------------------------
--------------------
P4
mean accuracy : 0.545
Best parameters set found on development set:
{'C': 1, 'degree': 1, 'gamma': 'auto', 'kernel': 'poly'}

Grid best score:
0.76
--------------------------------------------------------------------------------
--------------------
P8
mean accuracy : 0.515
Best parameters set found on development set:
{'C': 2, 'degree': 1, 'gamma': 'auto', 'kernel': 'linear'}

Grid best score:
0.8150000000000001
--------------------------------------------------------------------------------
--------------------
O1
mean accuracy : 0.51
Best parameters set found on development set:
{'C': 1, 'degree': 1, 'gamma': 'auto', 'kernel': 'poly'}

Grid best score:
0.86
--------------------------------------------------------------------------------
--------------------
Oz
mean accuracy : 0.52
Best parameters set found on development set:
{'C': 3, 'degree': 1, 'gamma': 'auto', 'kernel': 'linear'}

Grid best score:
0.8100000000000002
--------------------------------------------------------------------------------
--------------------
O2
mean accuracy : 0.51
Best parameters set found on development set:
{'C': 1, 'degree': 2, 'gamma': 'auto', 'kernel': 'poly'}

Grid best score:
0.8200000000000001
```

[38]:
```
np.max(list(acc_test.values()))
```

[38]:
```
0.545
```

[39]:
```
acc_test = {}
for i,ch in zip(range(X_trans.shape[1]),chnames):
    print('-'*100)
    print(ch)
    X_trans_picked = X_trans[:,i,:]
    # cross validation
    skf = StratifiedKFold(n_splits=8)
    rf = RandomForestClassifier()
    scr = cross_val_score(rf, X_trans_picked, y, cv=skf)
    preds = cross_val_predict(rf, X_trans_picked, y, cv=skf)

    # print results of classification
    print('mean accuracy :', scr.mean())
    acc_test[ch] = scr.mean()

    grid_RF ={'max_depth':[3,5,10,None],
                'n_estimators':[100,120,140,200],
                'max_features':[1,3,5,7],
                'min_samples_leaf':[1,2,3],
                'min_samples_split':[1,2,3]}
    grid_rf = GridSearchCV(rf, grid_RF, cv=skf, scoring='accuracy')
    grid_rf.fit(X_trans_picked, y)
    print("Best parameters set found on development set:")
    print(grid_rf.best_params_)
    print()
    print("Grid best score:")
    print (grid_rf.best_score_)
```

```
----------------------------------------------------------------------------------
--------------------
Fp1
mean accuracy : 0.74
Best parameters set found on development set:
{'max_depth': 3, 'max_features': 3, 'min_samples_leaf': 1, 'min_samples_split':
3, 'n_estimators': 120}

Grid best score:
0.765
----------------------------------------------------------------------------------
--------------------
Fp2
mean accuracy : 0.765
Best parameters set found on development set:
```

```
{'max_depth': None, 'max_features': 5, 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 140}

Grid best score:
0.79
--------------------------------------------------------------------------------
--------------------
Fc5
mean accuracy : 0.775
Best parameters set found on development set:
{'max_depth': 5, 'max_features': 7, 'min_samples_leaf': 1, 'min_samples_split':
2, 'n_estimators': 100}

Grid best score:
0.805
--------------------------------------------------------------------------------
--------------------
Fz
mean accuracy : 0.8
Best parameters set found on development set:
{'max_depth': 5, 'max_features': 3, 'min_samples_leaf': 1, 'min_samples_split':
3, 'n_estimators': 100}

Grid best score:
0.8200000000000001
--------------------------------------------------------------------------------
--------------------
Fc6
mean accuracy : 0.725
Best parameters set found on development set:
{'max_depth': None, 'max_features': 3, 'min_samples_leaf': 1,
'min_samples_split': 3, 'n_estimators': 120}

Grid best score:
0.78
--------------------------------------------------------------------------------
--------------------
T7
mean accuracy : 0.7
Best parameters set found on development set:
{'max_depth': 3, 'max_features': 3, 'min_samples_leaf': 1, 'min_samples_split':
3, 'n_estimators': 140}

Grid best score:
0.74
--------------------------------------------------------------------------------
--------------------
Cz
```

```
mean accuracy : 0.765
Best parameters set found on development set:
{'max_depth': 10, 'max_features': 1, 'min_samples_leaf': 1, 'min_samples_split':
2, 'n_estimators': 140}

Grid best score:
0.79
--------------------------------------------------------------------------------
--------------------
T8
mean accuracy : 0.65
Best parameters set found on development set:
{'max_depth': 10, 'max_features': 1, 'min_samples_leaf': 1, 'min_samples_split':
2, 'n_estimators': 140}

Grid best score:
0.715
--------------------------------------------------------------------------------
--------------------
P7
mean accuracy : 0.755
Best parameters set found on development set:
{'max_depth': 5, 'max_features': 5, 'min_samples_leaf': 1, 'min_samples_split':
2, 'n_estimators': 200}

Grid best score:
0.78
--------------------------------------------------------------------------------
--------------------
P3
mean accuracy : 0.725
Best parameters set found on development set:
{'max_depth': 10, 'max_features': 3, 'min_samples_leaf': 2, 'min_samples_split':
2, 'n_estimators': 140}

Grid best score:
0.77
--------------------------------------------------------------------------------
--------------------
Pz
mean accuracy : 0.72
Best parameters set found on development set:
{'max_depth': 5, 'max_features': 7, 'min_samples_leaf': 3, 'min_samples_split':
2, 'n_estimators': 140}

Grid best score:
0.765
--------------------------------------------------------------------------------
```

```
--------------------
P4
mean accuracy : 0.77
Best parameters set found on development set:
{'max_depth': 10, 'max_features': 5, 'min_samples_leaf': 3, 'min_samples_split':
2, 'n_estimators': 100}

Grid best score:
0.8
--------------------------------------------------------------------------------
--------------------
P8
mean accuracy : 0.805
Best parameters set found on development set:
{'max_depth': 10, 'max_features': 7, 'min_samples_leaf': 2, 'min_samples_split':
3, 'n_estimators': 140}

Grid best score:
0.8350000000000001
--------------------------------------------------------------------------------
--------------------
O1
mean accuracy : 0.75
Best parameters set found on development set:
{'max_depth': 5, 'max_features': 3, 'min_samples_leaf': 1, 'min_samples_split':
3, 'n_estimators': 140}

Grid best score:
0.79
--------------------------------------------------------------------------------
--------------------
Oz
mean accuracy : 0.755
Best parameters set found on development set:
{'max_depth': None, 'max_features': 1, 'min_samples_leaf': 2,
'min_samples_split': 2, 'n_estimators': 100}

Grid best score:
0.78
--------------------------------------------------------------------------------
--------------------
O2
mean accuracy : 0.75
Best parameters set found on development set:
{'max_depth': 5, 'max_features': 1, 'min_samples_leaf': 1, 'min_samples_split':
2, 'n_estimators': 200}

Grid best score:
```

```
0.78
```

[40]:
```
np.max(list(acc_test.values()))
```

[40]:
```
0.805
```

It seems like O1 channel seems to be the best for SVC (86% accuracy) and P8 channel (83.6% accuracy) for RF.

[36]:
```
chnames.index('O1')
```

[36]:
```
13
```

[45]:
```
print('-'*100)
print('O1 channel')
i = chnames.index('O1')
X_trans_picked = X_trans[:,i,:]
# cross validation
skf = StratifiedKFold(n_splits=5)
svc = SVC(C=1,
          degree=1,
          gamma='auto',
          kernel='poly')
scr = cross_val_score(svc, X_trans_picked, y, cv=skf)
preds = cross_val_predict(svc, X_trans_picked, y, cv=skf)

# print results of classification
print('mean accuracy :', scr.mean())

svc.fit(X_trans_picked, y)
```

```
----------------------------------------------------------------------------------------
--------------------
O1 channel
mean accuracy : 0.86
```

[45]:
```
SVC(C=1, degree=1, gamma='auto', kernel='poly')
```

[48]:
```
print('-'*100)
print('P8 channel')
i = chnames.index('P8')
X_trans_picked = X_trans[:,i,:]
# cross validation
skf = StratifiedKFold(n_splits=8)
rf = RandomForestClassifier(max_depth= 10,
                            max_features= 7,
                            min_samples_leaf= 2,
                            min_samples_split= 3,
```

```
                              n_estimators= 140)
scr = cross_val_score(rf, X_trans_picked, y, cv=skf)
preds = cross_val_predict(rf, X_trans_picked, y, cv=skf)

# print results of classification
print('mean accuracy :', scr.mean())


rf.fit(X_trans_picked, y)
```

```
--------------------------------------------------------------------------------
--------------------
P8 channel
mean accuracy : 0.8200000000000001
```

[48]:
```
RandomForestClassifier(max_depth=10, max_features=7, min_samples_leaf=2,
                       min_samples_split=3, n_estimators=140)
```

[74]:
```
def build_LSTM(hidden=16, window=16):
    # Define the model
    LSTM_model = Sequential()
    LSTM_model.name="LSTM"
    LSTM_model.add(LSTM(hidden,␣
 ↪input_shape=(window,window),name='LSTM_hidden1'))
    LSTM_model.add(Dense(32,name='Dense_hidden1'))
    LSTM_model.add(Dense(1,name='Output'))
    LSTM_model.compile(optimizer='adam', loss='mae')
    #LSTM_model.summary()
    return LSTM_model

LSTM_model = KerasClassifier(build_fn=build_LSTM(32,16))
LSTM_model
```

[74]:
```
KerasClassifier(
        model=None
        build_fn=<Sequential name=LSTM, built=True>
        warm_start=False
        random_state=None
        optimizer=rmsprop
        loss=None
        metrics=None
        batch_size=None
        validation_batch_size=None
        verbose=1
        callbacks=None
        validation_split=0.0
        shuffle=True
```

```
        run_eagerly=False
        epochs=1
        class_weight=None
)
```

[76]:
```
# cross validation
skf = StratifiedKFold(n_splits=5)
scr = cross_val_score(LSTM_model, X_trans, y, cv=skf)
preds = cross_val_predict(LSTM_model, X_trans, y, cv=skf)

# print results of classification
print('mean accuracy :', scr.mean())
acc_test[ch] = scr.mean()

LSTM_model.fit(X_trans,y,epochs=100,verbose=0)
```

```
5/5             2s 6ms/step - loss:
0.1094
2/2             1s 349ms/step
5/5             2s 5ms/step - loss:
0.0874
2/2             1s 442ms/step
5/5             2s 5ms/step - loss:
0.0872
2/2             1s 310ms/step
5/5             2s 5ms/step - loss:
0.1173
2/2             1s 278ms/step
5/5             1s 5ms/step - loss:
0.1012
2/2             1s 279ms/step
5/5             1s 5ms/step - loss:
0.1255
2/2             1s 303ms/step
5/5             2s 6ms/step - loss:
0.1028
2/2             1s 498ms/step
5/5             2s 5ms/step - loss:
0.0909
2/2             1s 390ms/step
5/5             1s 8ms/step - loss:
0.1026
2/2             1s 286ms/step
5/5             2s 12ms/step - loss:
0.1170
2/2             1s 303ms/step
mean accuracy : 0.97
```

[76]:
```
KerasClassifier(
        model=None
        build_fn=<Sequential name=LSTM, built=True>
        warm_start=False
        random_state=None
        optimizer=rmsprop
        loss=None
        metrics=None
        batch_size=None
        validation_batch_size=None
        verbose=1
        callbacks=None
        validation_split=0.0
        shuffle=True
        run_eagerly=False
        epochs=1
        class_weight=None
)
```

[50]:
```
X_trans.shape
```

[50]:
```
(200, 16, 16)
```

[ ]:
```
LSTM_mode.predict()
```

[ ]:

[ ]:

[ ]:

[ ]:

[49]:
```
#joblib.dump(svc,'SVC_model.pkl')
```

[49]:
```
['SVC_model.pkl']
```

[50]:
```
#joblib.dump(rf,'RF_model.pkl')
```

[50]:
```
['RF_model.pkl']
```

[48]:
```
#joblib.dump(grid_rf.best_params_,'RF_params.pkl')
```

[48]:
```
['RF_params.pkl']
```

[49]:
```
#joblib.dump(grid_rf.best_score_,'RF_score.pkl')
```

[49]: ['RF_score.pkl']

[77]:
```
#joblib.dump(LSTM_model,'LSTM_model.pkl')
```

[77]: ['LSTM_model.pkl']

[ ]:
```

```