2019ws-BCIIL-Sheet02-Solution

November 5, 2019

1 BCI-IL - Exercise Sheet #02

Sample solution

[1]:
```
% matplotlib inline

import numpy as np
import scipy as sp
from matplotlib import pyplot as plt

import bci_minitoolbox as bci
```

1.1 Preparation: Loading Data

[2]:
```
fname = 'erp_hexVPsag.npz'
cnt, fs, clab, mnt, mrk_pos, mrk_class, mrk_className = bci.load_data(fname)
```

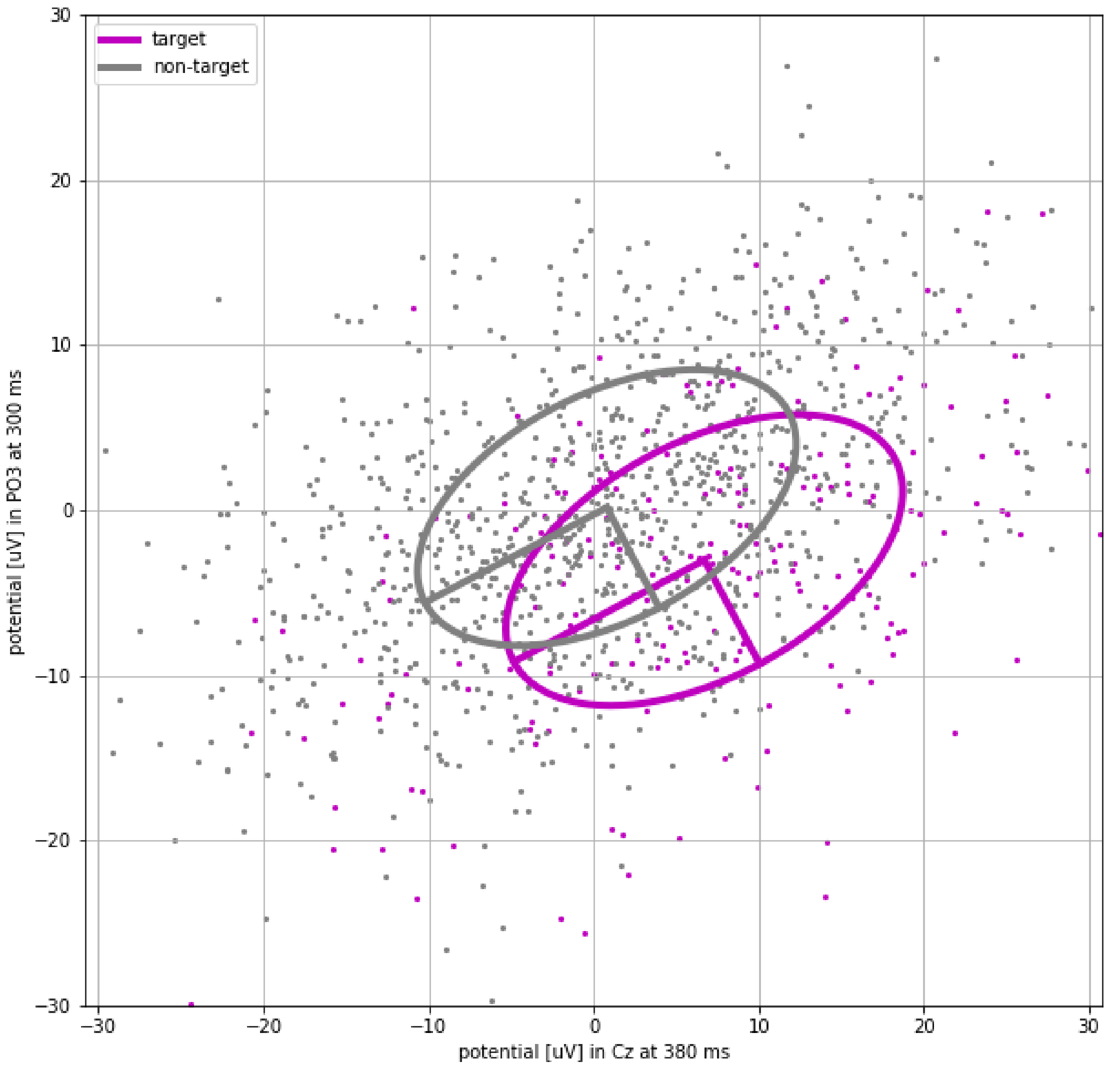
1.2 Exercise 1: Scatter Plot of 2D Features (8 points)

Make a scatter plot of the two distributions - *targets* and *nontargets*, one dot for each trial. On the x-axis, plot the value of channel Cz at `t` = 380 ms, and at the y-axis the value of PO3 at `t` = 300 ms. *(You may refer to the results of sheet 01 for the reason of this choice.)* Draw for both distributions the two principal axes, with the lengths of the axes being the standard deviation of the data in that direction. Draw also the corresponding ellipses. **Hint:** You can get that with a transformation of the unit circle as on the slide *Illustration of Multiplication* . . . .

[3]:
```
chans = ['Cz', 'PO3']
time_points = [380, 300]
ref_ival= [-100, 0]
ival= [ref_ival[0], max(time_points)]

epo, epo_t= bci.makeepochs(cnt, fs, mrk_pos, ival)
epo= bci.baseline(epo, epo_t, ref_ival)

c0 = clab.index(chans[0])
c1 = clab.index(chans[1])
t0 = np.argmin(np.abs(epo_t - time_points[0]))
t1 = np.argmin(np.abs(epo_t - time_points[1]))
```

```python
fv = epo[[t0, t1], [c0, c1], :]

hf = plt.figure(figsize=(10,10))
col = ['m','0.5']
for ci in range(len(mrk_className)):
    X = fv[:, mrk_class==ci ]
    plt.scatter(X[0, :], X[1, :], s=10, c=col[ci], linewidths=0)
    mu = np.mean(X, axis=1, keepdims=True)
    C = np.cov(X)
    d, V = np.linalg.eigh(C)
    # draw the axes corresponding to the two Eigenvectors
    L = np.append(mu, mu + V[:, [0]] * np.sqrt(d[0]), axis=1)
    plt.plot(L[0, :], L[1, :], '-', c=col[ci], linewidth=4,␣
 ↪label=mrk_className[ci])
    L = np.append(mu, mu + V[:, [1]] * np.sqrt(d[1]), axis=1)
    plt.plot(L[0, :], L[1, :], '-', c=col[ci], linewidth=4)
    # draw ellipse
    tline = np.linspace(0, 2 * np.pi, 100)
    sphere = np.vstack((np.sin([tline]), np.cos([tline])))
    elli = sp.linalg.sqrtm(C).dot(sphere)
    plt.plot(mu[0] + elli[0, :], mu[1] + elli[1, :], c=col[ci], linewidth=4)

plt.grid(True)
plt.axis('equal')
plt.xlim((-30,30))
plt.ylim((-30,30))
plt.xlabel('potential [uV] in {} at {} ms'.format(chans[0], time_points[0]))
plt.ylabel('potential [uV] in {} at {} ms'.format(chans[1], time_points[1]))
plt.legend()
```

[3]: <matplotlib.legend.Legend at 0x86532dff98>

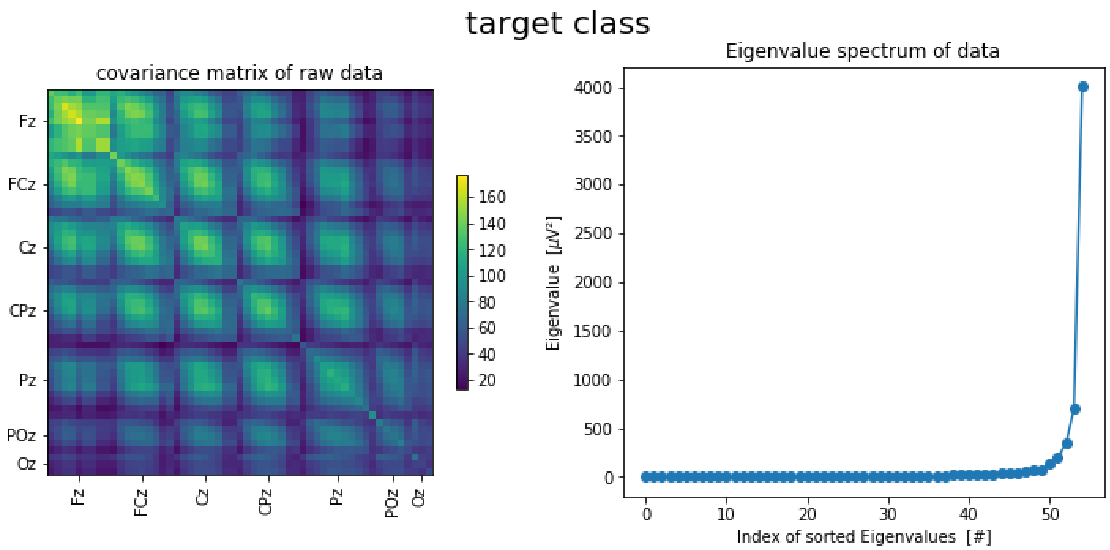


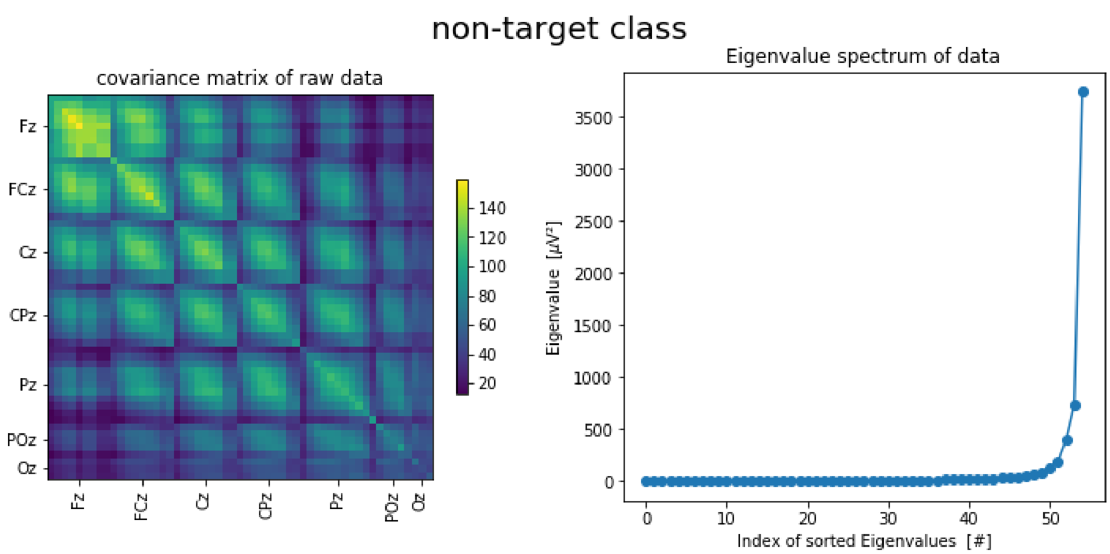
1.3 Exercise 2: Covariances and Eigenvalues (7 points)

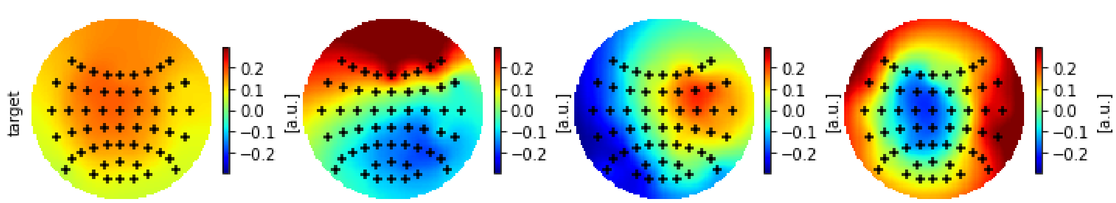
Calculate the channelwise covariance matrices (channel x channel) of the data for time point 380 ms for both classes (`np.cov`) and visualize them (`pl.imshow`). Perform an Eigenvalue decomposition (`np.linalg.eigh`) of the covariance matrices and plot (again class-wise) the eigenvalue spectrum. Then determine the four principle components (Eigenvectors) for each class that correspond to the largest Eigenvalues and display them as scalp maps (function `scalpmap` provided in the `bci_minitoolbox`).

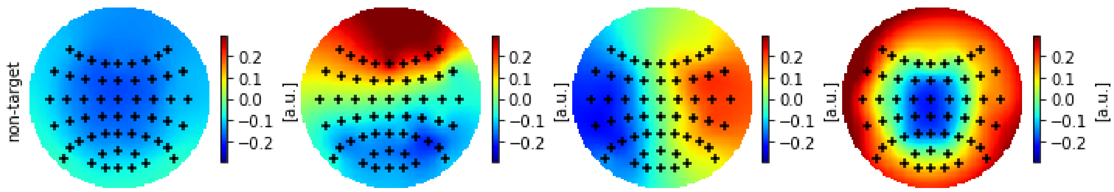
[4]:
```
time_point = 380

# tick marks for the covariance plot
selected_channels = ['Fz', 'FCz', 'Cz', 'CPz', 'Pz', 'POz', 'Oz']
idx = [clab.index(x) for x in selected_channels]
```

```
t0 = np.argmin(np.abs(epo_t - time_point))
fv = epo[t0, :, :]
EVs=np.zeros((epo.shape[1],4,2))

for ci in range(len(mrk_className)):
    Cf= np.cov(fv[:,mrk_class==ci])
    df, Vf= np.linalg.eigh(Cf)
    plt.figure(figsize=(12, 5))
    plt.suptitle(mrk_className[ci]+' class', fontsize=20)
    plt.subplot(1, 2, 1)
    plt.imshow(Cf)
    plt.title('covariance matrix of raw data')
    plt.colorbar(shrink=.5)
    plt.xticks(idx, selected_channels, rotation='vertical')
    plt.yticks(idx, selected_channels)

    plt.subplot(1, 2, 2)
    plt.plot(df, '-o' )
    plt.title('Eigenvalue spectrum of data')
    plt.xlabel('Index of sorted Eigenvalues  [#]')
    plt.ylabel('Eigenvalue  [$\mu$Vš]')
    EVs[:,:,ci]=Vf[:,-4:]

maxamp = abs(EVs).max()

for ci in range(len(mrk_className)):
    plt.figure(figsize=(12, 3))
    for k in range(4):
        plt.subplot(1, 4, k+1)
        bci.scalpmap(mnt, EVs[:,-1-k,ci], clim=(-maxamp,maxamp), cb_label='[a.u.
↪]')
        if k==0:
            plt.ylabel(mrk_className[ci])
            plt.axis('on')
            plt.yticks([])
            plt.xticks([])
            plt.box('off')
```


target class
covariance matrix of raw data
Fz
FCz
Cz
CPz
Pz
POz
Oz
160
140
120
100
80
60
40
20
Eigenvalue spectrum of data
Eigenvalue [μV^2]
Index of sorted Eigenvalues [#]
0
500
1000
1500
2000
2500
3000
3500
4000
10
30
50


non-target class
covariance matrix of raw data
Fz
FCz
Cz
CPz
Pz
POz
Oz
140
120
100
80
60
40
20
Eigenvalue spectrum of data
Eigenvalue [μV^2]
Index of sorted Eigenvalues [#]
0
500
1000
1500
2000
2500
3000
3500
10
20
30
40
50


target
[a.u.]
0.2
0.1
0.0
−0.1
−0.2


non-target
0.2
0.1
0.0
−0.1
−0.2
[a.u.]