PCA_for_EIT_solution

December 20, 2022

1 PCA for Source Separation of Ventilation and Cardiovascular Activity in Electrical Impedance Tomography (EIT)

1.1 Introduction

In this exercise we use PCA for processing image sequences of thoracic electrical impedance tomography (EIT) signals. EIT is a non-invasive, radiation-free imaging modality which uses small alternating currents to measure bioimpedance of the thorax [1]. These measurements are then converted into image sequences of thoracic impedance changes representing ventilation (i.e., air exchange in the lungs) and cardiovascular activity (e.g., heart movement or blood volume changes in heart and lungs).

In order to analyze these data it is important to properly separate ventilation and cardiovascular activity. Besides common techniques such as frequency filtering or ECG-triggered averaging, PCA can be used for separating these two sources of signals. The present example uses the method proposed by Deibele et al. [2] for which the block diagram is shown below:

1.2 References

[1] I. Frerichs et al., "Chest electrical impedance tomography examination, data analysis, terminology, clinical use and recommendations: consensus statement of the TRanslational EIT developmeNt stuDy group," Thorax, vol. 72, no. 1, pp. 83–93, Jan. 2017, doi: 10.1136/thoraxjnl-2016-208357.

[2] J. M. Deibele, H. Luepschen, and S. Leonhardt, "Dynamic separation of pulmonary and cardiac changes in electrical impedance tomography," Physiological Measurement, vol. 29, no. 6, pp. S1–S14, Jun. 2008, doi: 10.1088/0967-3334/29/6/S01.

[1]:
```
import numpy as np
from scipy.io import loadmat
from scipy.signal import filtfilt, butter
from scipy.linalg import eigh
from biosppy.signals.ecg import hamilton_segmenter

import matplotlib.pyplot as plt
# enable interactive plotting
%matplotlib widget

import plotly.graph_objects as go
from plotly.offline import init_notebook_mode, iplot
```

```
from plotly.subplots import make_subplots
init_notebook_mode(connected=True)  # initiate notebook for offline plot
```

[2]:
```
# load data
data = loadmat('../EIT_Data.mat')
t = data['tEit'].flatten()[data['IdxRange'].flatten().astype(bool)]
fs = 1/np.median(np.diff(t))
imgs_eit = data['Imgs']
b, a = butter(4, np.asarray([0.1, 12])/(fs/2), btype='bandpass')
imgs_eit = filtfilt(b, a, imgs_eit, axis=-1)
imgs_eit *= 1E3  # adapt scaling for plotting

# ECG data to be used for bonus question
ecg = {'time': data['Ecg'][0][0][2], 'value': data['Ecg'][0][0][1], 'fs':␣
 ↪data['Ecg'][0][0][3]}
ecg_range = (ecg['time'] > t[0]) & (ecg['time'] < t[-1])
ecg['time'] = ecg['time'][ecg_range]
ecg['value'] = ecg['value'][ecg_range]

# force all timings to start at zero
t -= t[0]
ecg['time'] -= ecg['time'][0]
```

[3]:
```
# plot input data
fig = make_subplots(rows=1, cols=2, column_widths=[1, 2])
fig.update_layout(width=950, height=400)

fig.add_trace(go.Heatmap(z=np.std(imgs_eit, axis=-1), zmin=0,
                         showscale=False, colorscale='magma'), row=1, col=1)
fig.update_yaxes(title='Right', showticklabels=False, autorange="reversed",␣
 ↪row=1, col=1)
fig.update_xaxes(title='Dorsal', showticklabels=False, row=1, col=1)
fig.update_layout(title='Overall EIT Activity')

fig.add_trace(go.Scatter(x=t, y=np.nansum(imgs_eit, axis=(0, 1)),
                         name='Overall Sum Signal', line_color='black'), row=1,␣
 ↪col=2)
fig.update_xaxes(title='Time (s)', row=1, col=2)
fig.update_yaxes(title='Impedance Change \Delta Z (A.U.)', row=1, col=2)
```

[4]:
```
def compute_principal_components(X):
    # perform PCA and compute principal components (pc) and eigenvalues of X
    A = np.dot(X.transpose(), X)  # covariance matrix
    [eigenvalues, eigenvectors] = eigh(A)
    pc = np.dot(X, eigenvectors)
    return pc, eigenvalues
```

```
def estimate_cardiac_frequency(s, fs):
    sign_cardiac = np.sign(s)
    # find where zero crossings occured
    index = np.where(np.diff(sign_cardiac) == 2)[0]
    # interpolate to a sub-sample resolution
    pos = index + s[index + 1] / (s[index + 1] - s[index])
    # interpolate RR interval values
    rr = np.diff(pos) / fs
    return 1/np.median(rr)

def lms(A, B):
    if B.ndim == 1:
        B = np.asmatrix(B).transpose()
    tmp = np.linalg.solve(np.dot(B.transpose(), B), B.transpose())
    return np.dot(np.dot(B, tmp), A)

# separate ventilation and cardiovascular activity using PCA
# according to the algorithm by Deibele et al., PhysMeas, 2008
# https://dx.doi.org/10.1088/0967-3334/29/6/S01
imgs_tmp = np.reshape(imgs_eit, [-1, imgs_eit.shape[-1]])
are_valid_pixels = np.all(~np.isnan(imgs_tmp), 1)
X = imgs_tmp[are_valid_pixels, :].transpose()

# first approximation (see block diagram)
X = X - np.repeat(np.reshape(np.mean(X, 0), [1, -1]), X.shape[0], 0)
PC1, lambda1 = compute_principal_components(X)
Bv = PC1[:, -1]
Xv_ = lms(X, Bv)
Xc_ = X - Xv_

# second approximation (see block diagram)
Xc_ = Xc_ - np.repeat(np.reshape(np.mean(Xc_, 0), [1, -1]), Xc_.shape[0], 0)
b, a = butter(6, np.asarray([0.92, 4.6])/(fs/2), btype='bandpass')
Xc_bp = filtfilt(b, a, Xc_, axis=0)
PC2, lambda2 = compute_principal_components(Xc_bp)
Bc_ = PC2[:, -2:]
fc = estimate_cardiac_frequency(Bc_[:, 0], fs)

# create cardiac template functions
Bc = np.hstack((Bc_, np.roll(Bc_, int(fs/fc/3), 0),
                np.roll(Bc_, -int(fs/fc/3), 0)))
Xc1 = lms(Xc_, Bc)
Xc2 = lms(Xv_, Bc)
Xc = (Xc1 + Xc2).transpose()
Xv = (Xv_ - Xc2).transpose()

# cardiovascular activity
```

```
imgs_card = np.full(imgs_tmp.shape, np.nan)
imgs_card[are_valid_pixels, :] = Xc
imgs_card = imgs_card.reshape(imgs_eit.shape)
# ventilation activity
imgs_vent = np.full(imgs_tmp.shape, np.nan)
imgs_vent[are_valid_pixels, :] = Xv
imgs_vent = imgs_vent.reshape(imgs_eit.shape)
```

[5]:
```
# plot ventilation acticity
fig = make_subplots(rows=1, cols=2, column_widths=[1, 2])
fig.update_layout(width=950, height=400)

fig.add_trace(go.Heatmap(z=np.std(imgs_vent, axis=-1), zmin=0,
                         showscale=False, colorscale='magma'), row=1, col=1)
fig.update_yaxes(title='Right', showticklabels=False, autorange="reversed",␣
 ↪row=1, col=1)
fig.update_xaxes(title='Dorsal', showticklabels=False, row=1, col=1)
fig.update_layout(title='Ventilation Activity')

fig.add_trace(go.Scatter(x=t, y=np.nanmean(imgs_vent, axis=(0, 1)),
                         name='Mean Signal', line_color='black'), row=1, col=2)
fig.update_xaxes(title='Time (s)', row=1, col=2)
fig.update_yaxes(title='Impedance Change \Delta Z (A.U.)', row=1, col=2)

# add example signals in ??? regions
regions = {'Region RL': ([10, 14], 'magenta'), 'Region LL': ([22, 14],␣
 ↪'orange')}
for reg, tmp in regions.items():
    fig.add_scatter(x=[tmp[0][0]], y=[tmp[0][1]], mode='markers',␣
 ↪marker_symbol='square-open',
                    marker_size=10, legendgroup=reg, marker_color=tmp[1],␣
 ↪name=reg, row=1, col=1)
    fig.add_trace(go.Scatter(x=t, y=imgs_vent[tmp[0][1], tmp[0][0], :],␣
 ↪legendgroup=reg,
                             showlegend=False, name=reg, line_color=tmp[1]),␣
 ↪row=1, col=2)

fig.show()
```

[6]:
```
# plot cardiovascular acticity
fig = make_subplots(rows=1, cols=2, column_widths=[1, 2])
fig.update_layout(width=950, height=400)

fig.add_trace(go.Heatmap(z=np.std(imgs_card, axis=-1), zmin=0,
                         showscale=False, colorscale='magma'), row=1, col=1)
fig.update_yaxes(title='Right', showticklabels=False, autorange="reversed",␣
 ↪row=1, col=1)
```

```
fig.update_xaxes(title='Dorsal', showticklabels=False, row=1, col=1)
fig.update_layout(title='Cardiovascular Activity')

fig.add_trace(go.Scatter(x=t, y=np.nanmean(imgs_card, axis=(0, 1)),
                         name='Mean Signal', line_color='black'), row=1, col=2)
fig.update_xaxes(title='Time (s)', row=1, col=2)
fig.update_yaxes(title='Impedance Change \Delta Z (A.U.)', row=1, col=2)

# add three example signals in ??? regions
regions = {'Region A': ([18, 9], 'green'), 'Region B': ([10, 15], 'blue'),␣
 ↪'Region C': ([23, 15], 'red')}
for reg, tmp in regions.items():
    fig.add_scatter(x=[tmp[0][0]], y=[tmp[0][1]], mode='markers',␣
 ↪marker_symbol='square-open',
                    marker_size=10, legendgroup=reg, marker_color=tmp[1],␣
 ↪name=reg, row=1, col=1)
    fig.add_trace(go.Scatter(x=t, y=imgs_card[tmp[0][1], tmp[0][0], :],␣
 ↪legendgroup=reg,
                             showlegend=False, name=reg, line_color=tmp[1]),␣
 ↪row=1, col=2)
fig.show()
```

[1]:
```
from IPython.display import display, Math, Latex
display(Latex(r"\newpage"))
```

2 Exercise Questions

Please provide your answers directly below each question.

2.1 Question 1

Determine the frequency of the ventilation activity (i.e., the respiratory rate), both expressed in `Hz` and `respirations/min`.

[7]:
```
# respiratory rate
from scipy.signal import filtfilt, butter
b, a = butter(4, 1.0/(fs/2))
vent_sum_signal = np.nanmean(imgs_vent, axis=(0, 1))
vent_sum_signal_filt = filtfilt(b, a, vent_sum_signal)
zc_vent = (np.diff(vent_sum_signal_filt)[1:] >= 0) & (np.
 ↪diff(vent_sum_signal_filt)[:-1] < 0)
t_zc_vent = np.where(zc_vent)[0]/fs

rr_mean = 60 / np.mean(np.diff(t_zc_vent))
rr_median = 60 / np.median(np.diff(t_zc_vent))

print('Respiratory rate (RR): mean={:.1f} rpm; median={:.1f} rpm'.
 ↪format(rr_mean, rr_median))
print('Respiratory rate (RR): mean={:.2f} Hz; median={:.2f} Hz\n'.
 ↪format(rr_mean/60, rr_median/60))
```

```
Respiratory rate (RR): mean=19.9 rpm; median=19.6 rpm
Respiratory rate (RR): mean=0.33 Hz; median=0.33 Hz
```

2.2 Question 2

Determine the frequency of the cardiovascular activity, both expressed in `Hz` and `beats/min`.

[8]:
```
# heart rate
from scipy.signal import filtfilt, butter
b, a = butter(4, 1.0/(fs/2))
card_sum_signal = np.nanmean(imgs_card, axis=(0, 1))
card_sum_signal_filt = filtfilt(b, a, card_sum_signal)
zc_card = (np.diff(card_sum_signal_filt)[1:] >= 0) & (np.
 ↪diff(card_sum_signal_filt)[:-1] < 0)
t_zc_card = np.where(zc_card)[0]/fs

hr_mean = 60 / np.mean(np.diff(t_zc_card))
hr_median = 60 / np.median(np.diff(t_zc_card))
```

```
print('Heart rate (HR): mean={:.1f} bpm; median={:.1f} bpm'.format(hr_mean,␣
 ↪hr_median))
print('Heart rate (HR): mean={:.2f} Hz; median={:.2f} Hz\n'.format(hr_mean/60,␣
 ↪hr_median/60))
```

```
Heart rate (HR): mean=57.6 bpm; median=58.3 bpm
Heart rate (HR): mean=0.96 Hz; median=0.97 Hz
```

2.3 Question 3

Determine the follwing three values:

- i) the maximal amplitude of ventilation activity;
- ii) the maximal amplitude of cardiovascular activity; and
- iii) the ratio between i) and ii), i.e., ventilation vs cardiovascular activity.

[9]:
```
# ventilation vs cardio activity
vent_activity = np.std(imgs_vent, axis=-1)
card_activity = np.std(imgs_card, axis=-1)
vent_amplitude = np.nanmax(vent_activity.reshape(-1, 1))
card_amplitude = np.nanmax(card_activity.reshape(-1, 1))

print('i) Ventilation Amplitude: {:.4f}'.format(vent_amplitude))
print('ii) Cardiovascular Amplitude: {:.4f}'.format(card_amplitude))
print('iii) Ratio Ventilation/Cardiovascular Amplitudes = {:.2f}'.
 ↪format(vent_amplitude / card_amplitude))
print('iii) Ratio Cardiovascular/Ventilation Amplitudes = {:.3f}'.
 ↪format(card_amplitude / vent_amplitude))
```

```
i) Ventilation Amplitude: 0.0282
ii) Cardiovascular Amplitude: 0.0032
iii) Ratio Ventilation/Cardiovascular Amplitudes = 8.94
iii) Ratio Cardiovascular/Ventilation Amplitudes = 0.112
```

2.4 Question 4

Determine the eigenvalues of the first three principal components resulting from the first PCA (see variable `PC1`).

[10]:
```
# eigenvalues first PCA
np.flip(lambda1[-3:])
```

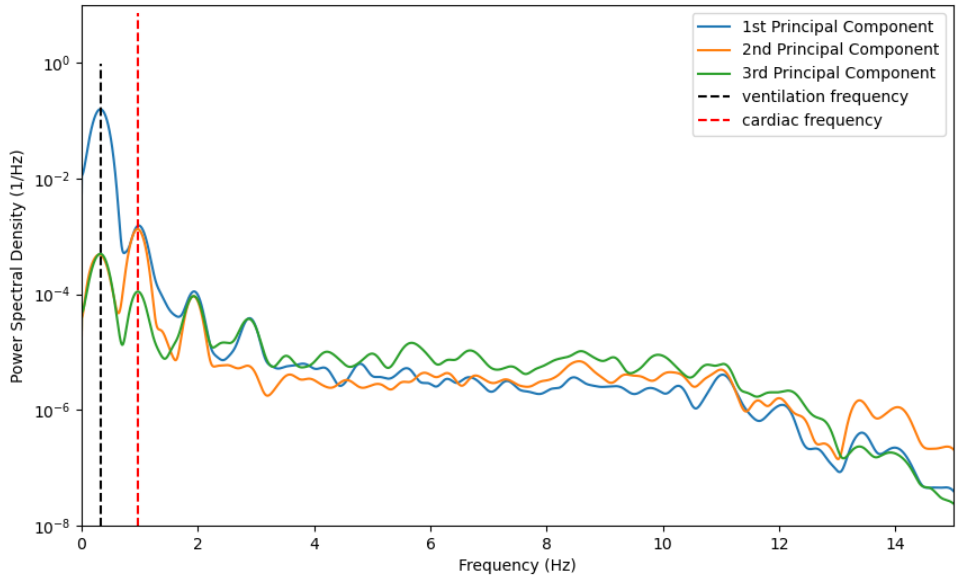
[10]:
```
array([107.58688656,   2.11148888,   0.80315244])
```

2.5 Question 5

Similar to Question 4, determine the two most dominant frequencies for the first three principal components. Note that this would lead to a total of 6 values, 2 frequencies for each of the 3 PCA

components. However, for some components only one dominant frequency might be present. For all 3 PCA components, which one is rather related to ventilation or cardiovascular activity?

[11]:
```
# spectral estimation of dominant frequencies
from scipy.signal import welch
plt.figure(figsize=(10, 6))
f, Pxxf = welch(PC1[:, -1], fs, return_onesided=True, nfft=2**14)
plt.semilogy(f, Pxxf, label='1st Principal Component')
f, Pxxf = welch(PC1[:, -2], fs, return_onesided=True, nfft=2**14)
plt.semilogy(f, Pxxf, label='2nd Principal Component')
f, Pxxf = welch(PC1[:, -3], fs, return_onesided=True, nfft=2**14)
plt.semilogy(f, Pxxf, label='3rd Principal Component')
plt.vlines(rr_median/60, plt.ylim()[0], plt.ylim()[-1], linestyle='dashed',␣
 ↪color='black', label='ventilation frequency')
plt.vlines(hr_median/60, plt.ylim()[0], plt.ylim()[-1], linestyle='dashed',␣
 ↪color='red', label='cardiac frequency')
plt.legend()
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power Spectral Density (1/Hz)')
plt.xlim([0, 15])
plt.ylim([10**(-8), 10])
plt.show()
```



- 1st PC: 0.33 Hz (ventilation), less strong 0.97 Hz (cardiovascular)
- 2nd PC: 0.97 Hz (cardiovascular), less strong 0.33 Hz (ventilation)
- 3rd PC: 0.33 Hz (ventilation), less strong 2*0.97 Hz (2nd harmonic of cardiovascular), 1st

cardiovascular harmonic less strong

2.6 Question 6

The three example signals (Regions A to C: **green, blue, red**) of cardiovascular activity show the impedance change over time. Can you guess the underlying anatomical structure for each of the three example signals (Regions A to C: **green, blue, red**).

- green: heart
- blue: right lung
- red: left lung

2.7 Question 7 - *Bonus Question* (not graded)

Can you detect the QRS peaks (e.g., using `biosppy.signals.ecg.hamilton_segmenter`) on the ECG signal (see variable `ecg`) and plot them together with the cardiovascular EIT activity?

[12]:
```
## detect R-peaks in ECG
rpeaks = hamilton_segmenter(ecg['value'], ecg['fs'])
tr = ecg['time'][rpeaks]

# plot
fig = make_subplots(rows=2, cols=1, shared_xaxes=True)
fig.update_layout(width=950, height=800)

fig.add_trace(go.Scatter(x=t, y=np.nanmean(imgs_card, axis=(0, 1)),
                         name='Cardiovascular Mean Signal'), row=1, col=1)
fig.update_yaxes(title='Impedance Change \DeltaZ (A.U.)', row=1, col=1)
fig.update_xaxes(title='Time (s)', row=1, col=1)
for tr_peak in tr:
    fig.add_vline(x=tr_peak, line_color='grey')

fig.add_trace(go.Scatter(x=ecg['time'], y=ecg['value'], name='ECG',␣
 ↪line_color='black'), row=2, col=1)
fig.add_trace(go.Scatter(x=ecg['time'][rpeaks], y=ecg['value'][rpeaks],␣
 ↪mode='markers',
                          marker_color='red', marker_symbol='circle-open',␣
 ↪name='QRS Peaks'), row=2, col=1)
fig.update_yaxes(title='ECG (A.U.)', row=2, col=1)
fig.update_xaxes(title='Time (s)', row=2, col=1)
```