Assignment 3 - MAD

Mikkel Willén

November 2021

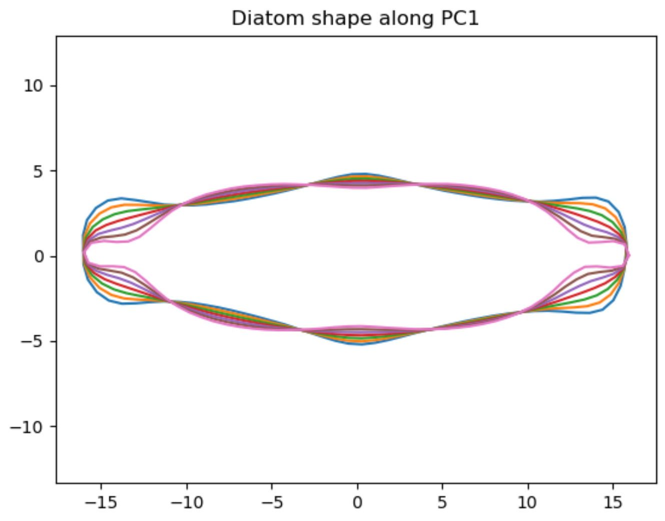
Exercise 1

a)

Varians for de første 10 komponenter:
Proportion of variance explained by the first 1 principal components: 0.7718721493017527
Proportion of variance explained by the first 2 principal components: 0.9276996293043025
Proportion of variance explained by the first 3 principal components: 0.9521198453942007
Proportion of variance explained by the first 4 principal components: 0.9637878603999529
Proportion of variance explained by the first 5 principal components: 0.9739084497954094
Proportion of variance explained by the first 6 principal components: 0.98236065164916
Proportion of variance explained by the first 7 principal components: 0.9889975933245944
Proportion of variance explained by the first 8 principal components: 0.9910287023941854
Proportion of variance explained by the first 9 principal components: 0.9926692113360289
Proportion of variance explained by the first 10 principal components: 0.9939926229665051

b)

Herunder er plot af den fjerde komponent:



Exercise 2

Exercise 3

a)

b)

c)

Exercise 4

a)

Min nulhypotese er $\theta = \theta_0$

b)

Step 1:
Fra opgave teskten, kan vi se, at dataen er normalfordelt med mean μ og varians σ^2

Step 2:
Min nulhypotese er $\theta = \theta_0$ og min alternative hypotese er $\theta \neq \theta_0$

Step 3: Forskel på X_i og Y_i

1	2	3	4	5
1	0.5	- 0.5	1.5	0.5

Step 4:
Fra opgaveteksten er $\alpha = 5\%$

Step 5:
Får c_1 og c_2 fra python med kommandoen

```
gamma = 0.95
c = scipy.stats.t.ppf((1+gamma)/2)
```

Som giver $c_1 = -2.776$ og $c_2 = 2.776$

Step 6:
Udregner sample mean

$$\bar{x} = \frac{\sum(X_i - Y_i)}{N} = \frac{1 + 0.5 - 0.5 + 1.5 + 0.5}{5} = 0.6$$

Variance

$$\begin{aligned} std^2 &= \frac{\sum(X_i - \bar{x})^2}{N-1} \\ &= \frac{(1-0.6)^2 + (0.5-0.6)^2 + (-0.5-0.6)^2 + (1.5-0.6)^2 + (0.5-0.6)^2}{4} \\ &= 0.55 \\ std &= 0.742 \end{aligned}$$

Udregner t-test

$$t = \frac{\bar{x} - \mu_0}{\frac{std}{\sqrt{N}}} = \frac{0.6 - 0}{\frac{0.742}{\sqrt{5}}} = 1.81$$

Siden ikke ligger uden for c_1 og c_2 kan vi ikke afvise nulhypotesen.

c)

Han kan ændre resultat en smule, da standard deviation vil ændre sig når, N bliver større.

Appendix

pca_StudentVersion.py

```
import numpy as np

diatoms = np.loadtxt('diatoms.txt', delimiter=',').T
diatoms_classes = np.loadtxt('diatoms_classes.txt', delimiter=',')
print('Shape of diatoms:', diatoms.shape)
print('Shape of diatoms_classes:', diatoms_classes.shape)
#print('Classes:', diatoms_classes)

d,N = diatoms.shape
print('Dimension:', d)
print('Sample size:', N)


# Here's a function that will plot a given diatom. Let's try it on the first diatom in the datas

import matplotlib.pyplot as plt

def plot_diatom(diatom):
    xs = np.zeros(91)
    ys = np.zeros(91)
    for i in range(90):
        xs[i] = diatom[2*i]
        ys[i] = diatom[2*i+1]

    # Loop around to first landmark point to get a connected shape
    xs[90] = xs[0]
    ys[90] = ys[0]

    plt.plot(xs, ys)
    plt.axis('equal')

plot_diatom(diatoms[:,0])


# Let's next compute the mean diatom and plot it.

mean_diatom = np.mean(diatoms, 1)
plot_diatom(mean_diatom)


# ### Task1: Implementing PCA
#
```

```
# To implement PCA, please check the algorithm explaination from the lecture.
# Hits:
#
# 1) Noramilize data subtracting the mean shape. No need to use Procrustes Analysis or other mor
#
# 2) Compute covariance matrix (check np.cov)
#
# 3) Compute eigenvectors and values (check np.linalg.eigh)

import numpy.matlib

def pca(data):
    data_cent = np.zeros((data.shape[0],data.shape[1]))
    for i in range(data.shape[0]):
        for j in range(data.shape[1]):
            data_cent[i][j] = data[i][j] - mean_diatom[i]
    data_cent = np.cov(data_cent)
    PCevals, PCevecs = np.linalg.eigh(data_cent)
    PCevals = np.flip(PCevals, 0)
    PCevecs = np.flip(PCevecs, 1)
    return PCevals, PCevecs, data_cent

PCevals, PCevecs, data_cent = pca(diatoms)
# PCevals is a vector of eigenvalues in decreasing order. To verify, uncomment:
# PCevecs is a matrix whose columns are the eigenvectors listed in the order of decreasing eigen


# ***Recall:***
# * The eigenvalues represent the variance of the data projected to the corresponding eigenvecto
# * Thus, the 2D linear subspace with highest projected variance is spanned by the eigenvectors
# * We extract these eigenvectors and plot the data projected onto the corresponding space.

# ### Compute variance of the first 10 components
#
# How many components you need to cover 90%, 95% and 99% of variantion. Submit the resulting num

variance_explained_per_component = PCevals/np.sum(PCevals)
cumulative_variance_explained = np.cumsum(variance_explained_per_component)

plt.plot(cumulative_variance_explained)
plt.xlabel('Number of principal components included')
plt.ylabel('Proportion of variance explained')
plt.title('Proportion of variance explained as a function of number of PCs included')

# Let's print out the proportion of variance explained by the first 10 PCs
```

```
for i in range(10):
    print('Proportion of variance explained by the first '+str(i+1)+' principal components:', cu


# ### Task2: Plot varianace accosiated with the first component
#
# Please fill the gaps in the code to plot mean diatom shape with added FOURTH eigenvector mulit
#
# Submit the resulting plot for grading.


e4 = PCevecs[:, 3] # gets the second eigenvector
lambda4 = PCevals[3] # gets the second eigenvalue
std4 = np.sqrt(lambda4) # In case the naming std is confusing -- the eigenvalues have a statisti
temp = [- 3, - 2, - 1, 0, 1, 2, 3]

diatoms_along_pc = np.zeros((7, 180))
for i in range(7):
    diatoms_along_pc[i] = mean_diatom + e4 * std4 * temp[i]
plt.figure()
for i in range(7):
    plot_diatom(diatoms_along_pc[i])

plt.title('Diatom shape along PC1')
plt.show()
```