MAD Assignment 3

Ask Jensen

13. december 2021

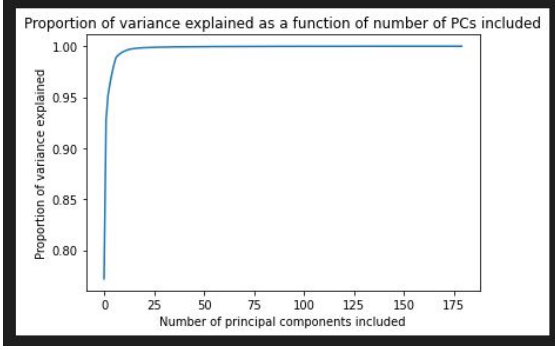
Indhold

1 Problem 1

1.1 (a)

```
import numpy.matlib

def pca(data):
    # Creating "clone" of matrix
    data_cent = np.full_like(data,0)

    # Iterate the matrix subtracting the mean diatiom
    #  from each row
    for i in range(780):
        data_cent[:,i] = diatoms[:,i] - mean_diatom

    # Create the covariance matrix
    cov_matrix = np.cov(data_cent)

    # Calculate the eigenvecotrs and eigenvalues
    PCevals, PCevecs = np.linalg.eigh(cov_matrix)

    # linalg.eigh returns the vectors and values
    # in the wrong order.
    # Np.flip will reverse the order so it is correct and
    # corrosponding to the exercise requirements
    PCevals = np.flip(PCevals)
    PCevecs = np.flip(PCevecs, axis=1)
    return PCevals, PCevecs, data_cent

PCevals, PCevecs, data_cent = pca(diatoms)
```

```
Proportion of variance explained by the first 1 principal components: 0.7718721493017527
Proportion of variance explained by the first 2 principal components: 0.9276996293043024
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
```

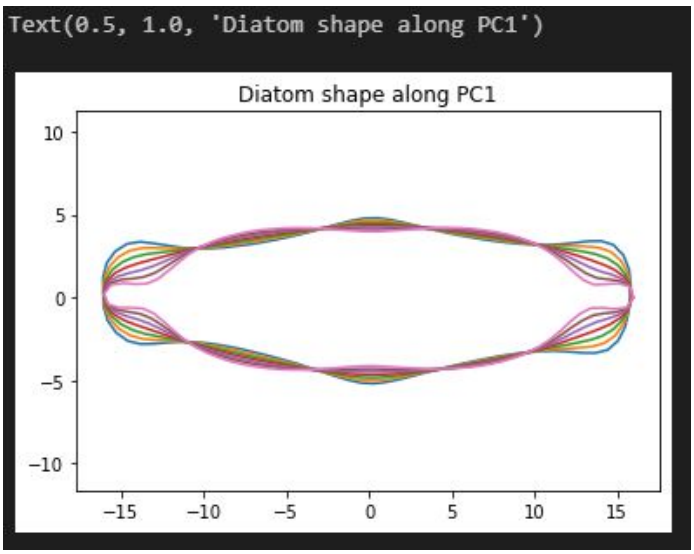


Figur 1: Values for the first 10 proportions of variance, and the corrosponding graph

The proportion and the figure shows the context between the principal components and the ammount of variance each iteration captures.

1.2 (b)

```
    # gets the fourth eigenvector
    e4 = PCevecs[:, 3]
    # gets the fourth eigenvalue
    lambda4 = PCevals[3]
    # In case the naming std is confusing --
    # the eigenvalues have a statistical interpretation
    # print(std4)
    std4 = np.sqrt(lambda4)

    # Makes matrix filled with zeros
    diatoms_along_pc = np.zeros((7, 180))

    # Iterates the length of the matrix
    # For each row, add the mean diatom with added
    # values
    for i in range(7):
        diatoms_along_pc[i] = mean_diatom + ( e4 * std4 * (i-3))

    # Plotting each diatom
    for i in range(7):
        plot_diatom(diatoms_along_pc[i])

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



Figur 2: Plotted diatom

2 Problem 2

assesses the given claim $E[(X-\mu)^4] \geq \sigma^4$

X has the mean μ and the variance σ^4 which can be rewritten as $\sigma^4 = (Var(X))^2$
$E[(X-\mu)^4]$ kan be rewritten as $E(g(x))$ where $g(x) = (x-\mu)^4$
It is possible to Jensen's inequality if $g''(x) \geq 0$
$g(x) = (x-\mu)^4$
$g(x) = 4(x-\mu)^3$
$g(x) = 12(x-\mu)^2$
g(x) is convex, since the second derivative of the function is quadratic. Hence it will always be greater than zero. Which means that it its possible to make use of Jensen's inequality.
$E(cX-\mu)^4 \geq (E(X-\mu))^4$
$(E(X-\mu))^4$
$((E(X-\mu))^2)^2 = (Var(X))^2$

thus the claim is true, and shown by Jensen's inequality

$$E[(X-\mu)^4] \geq \sigma^4$$

3 Problem 3

3.1 (a)

```
    for i in range(nexp):
    # simulates n realizations from a Gaussian
    # with mean mu and var sigma^2
    x = np.random.normal(mu,sigma,n)
    # TODO: adapt for b)
    sig = np.sqrt(np.var(x, ddof=1))
    # computes the 0.5% quantile of a Gaussian, roughly -2.576
    fac1 = scipy.stats.norm.ppf((1-gamma)/2, 0, 1)
    # computes the 99.5% quantile of a Gaussian, roughly 2.576
    fac2 = scipy.stats.norm.ppf((1-gamma)/2 + gamma, 0, 1)

    # computes the 0.5 quantile using the t-test
    fac3 = scipy.stats.t.ppf((1+gamma)/2, n-1)
    # computes the 99.5 quantile using the t-test
    fac4 = scipy.stats.t.ppf((1+gamma)/2-gamma, n-1)
    xmean = np.mean(x) # Sample mean
    a = xmean - fac2*sig/np.sqrt(n)
    b = xmean - fac1*sig/np.sqrt(n)
    ac = xmean - fac3*sig/np.sqrt(n) # TODO: adapt for c)
    bc = xmean - fac4*sig/np.sqrt(n) # TODO: adapt for c)
```

```
99.0%-confidence interval:
b) Not matching in 360 (out of 10000) experiments, 3.6%
c) Not matching in 97 (out of 10000) experiments, 0.97%
```

Figur 3: Result from 10000 experiments

4 Problem 4

4.1 (a)

In this exercise I'm asked to choose the null hypothesis.
My Null hypothesis is:, $H_0 : \mu_0 = 0$ My alternative hypothesis is, $H_A : \mu \neq \mu_0$
Which means, that I assume that there is no difference in flowering time, since the value $X_3 - Y_3 = -0.5$ shows that the scientists claim does not hold for all of the samples. With the specified alternative hypothesis, I would have to perform af two-sided t-test
$\mu_0 = 0$ since the assumption is, that there is no difference between the two types of flowers

4.2 (b)

Performing the corrosponding t-test (Assuming that I have to perform the corrosponding test using my claim from (a)).
Following the "six steps"from the lecture. I will be starting from step 3 since both step one and two are defined in question (a).
The dataset is $X_i - Y_i$, which gives me

1	0.5	-0.5	1.5	0.5

Calculating the observed mean:
$\frac{1+0.5-0.5+1.5+0.5}{5} = 0.6$

Calculating the standard deviation for my sample
$S = \sqrt{Var(Z)}$

$Var(Z) = \sum_{i=1}^{5} \frac{(x_i - \overline{x})^2}{n-1}$

$Var(Z) = \frac{(1-0.6)^2+(0.5-0.6)^2+(-0.5-0.6)^2+(1.5-0.6)^2+(0.5-0.6)^2}{4} = \frac{2.2}{4} = 0.55$

$S = \sqrt{0.55} = 0.7416$

$t = \frac{\overline{x}-\mu_0}{S\sqrt{n}} = \frac{0.6-0}{0.7416\sqrt{5}} = \frac{0.6}{0.3317} \approx 1.81$

I've calculated c_1 and c_2 using the following code I made.

```python
from scipy.stats import t

# Mean of my sample
mean = 0.6

# Samples (5 flowers)
n_samples = 5

# I need to divide alpha since I'm making use of
# the two-side test
alpha_val = 0.05 / 2
std_deviation = 0.7416

# Performing the t.ppf, in order to find c1 and c2
c1 = t.ppf(alpha_val, n_samples-1,
           loc= mean, scale = std_deviation)

c2 = t.ppf((1 - alpha_val), n_samples-1,
           loc= mean, scale = std_deviation)

print("c1: lower_cutoff", c1)
print("c2: upper_cutoff", c2)
```

which gives me:

$$c_1 \approx -1.4590 \qquad \text{and} \qquad c_2 \approx= 2.6590$$

Since $c_1 < t < c_2$ meaning that t is in the "acceptance"region, I can accept that the flowering time with the two types of flowers appears to be indifferent.

4.3 (c)

No, it is not that simple. If the scientist were to multiply the whole dataset with a number k it would change the calculations of t. The standard deviation will cause problems since the degrees of freedom will not be the same

$$Var^* = \sum_{i=1}^{k \cdot n} \frac{(x_i - \mu)^2}{kn-1} \neq k \cdot \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{n-1}$$