A1

MASD 2021
Department of Computer Science
University of Copenhagen

Mikkel/pwz854 | partners = mfh144 & qfh987

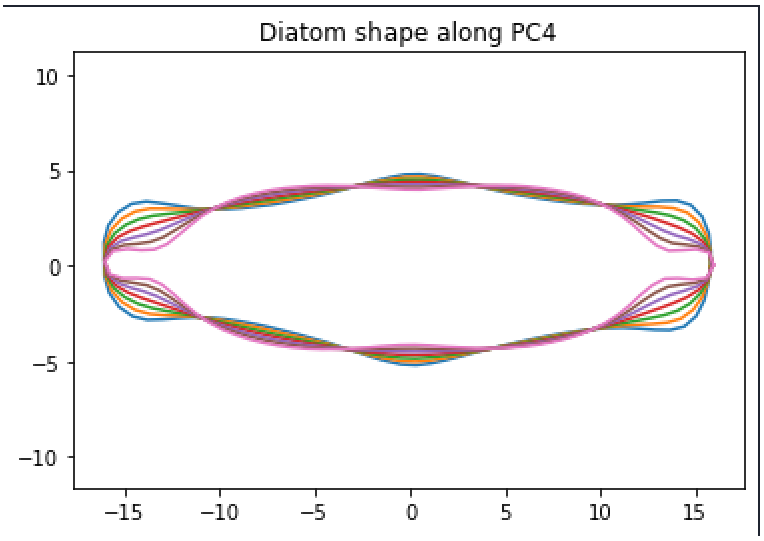
13/12/2021

Contents

Exercise 1

a) The proportion of variance explained by each components can be seen in the following

```
first 1 principal components: 0.7718721493017527
first 2 principal components: 0.9276996293043025
first 3 principal components: 0.9521198453942007
first 4 principal components: 0.9637878603999529
first 5 principal components: 0.9739084497954094
first 6 principal components: 0.98236065164916
first 7 principal components: 0.9889975933245944
first 8 principal components: 0.9910287023941854
first 9 principal components: 0.9926692113360289
first 10 principal components: 0.9939926229665051
```



b) Above can be seen the result of projecting the mean diatom onto the fourth component. The shape of the mean can actually be seen in the plot above, being the red line in between the other lines. This is achieved when we scale the std4 by 0, since we actually end up having just having the mean actually. All the other lines are representations of scaling the std4 to x degree while projecting the mean onto this fourth component, which could be seen as an axis. So they are the result of projecting the mean onto this 4 fourth component axis.

Exercise 2

a) Assuming that X is a random variable with mean μ and variance σ^2 we will show that

$$E\left[(X-\mu)^4\right] \geq \sigma^4 \tag{1}$$

we notice that

$$E\left[(X-\mu)^4\right] \geq \sigma^4 \Longleftrightarrow E\left[((X-\mu)^2)^2\right] \geq (\sigma^2)^2$$

which, by using that $\sigma^2 = VAR(X) = E\left[(X-\mu)^2\right]$, can then be written as

$$E\left[((X-\mu)^2)^2\right] \geq (E\left[(X-\mu)^2\right])^2$$

Now we see that on both sides of the equality one can find an expression to the second power, meaning that if we define $g(x)$ as

$$g(x) = (x)^2$$

then we can write the expression as

$$E\left[g((X-\mu)^2)\right] \geq g(E\left[(X-\mu)^2\right]) \tag{2}$$

Then computing $g''(x) = 2$ we see that $g''() \geq 0$ for all x. Then Jensen's Inequality states

$$E[g(X)] \geq g(E[x])$$

And thus equation 2 must hold. Now again using $\sigma^2 = VAR(X) = E\left[(X-\mu)^2\right]$ we go back to

$$E\left[g((X-\mu)^2)\right] \geq g(\sigma^2)$$

$$E\left[((X-\mu)^2)^2\right] \geq (\sigma^2)^2$$

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

and therefor Jensen's Inequality must also hold for equation 1

Exercise 3

a) Constructing a γ-confidence interval for μ we will start by defining our problem as

$$P\left(-c \leq \sqrt{n}\frac{\hat{\mu}-\mu}{\sigma} \leq c\right) = P\left(-c \leq \frac{\hat{\mu}-\mu}{\sigma/\sqrt{n}} \leq c\right) = \gamma$$

The middle expression is achieved using the rule $a : \frac{b}{c} = a \cdot \frac{c}{b}$
Now because it is the normal distribution we can formulate it using the CDF as $\Phi(c)$. By looking at the lower and upper bound of the expression we see that it can be formulated as

$$P\left(-c \leq \frac{\hat{\mu}-\mu}{\sigma/\sqrt{n}} \leq c\right) = \Phi(c) - \Phi(-c) = \gamma$$

Now by using that our normal distribution is symmetric we can write $\Phi(-c)$ as $1 - \Phi(c)$ Putting this into our expression we get

$$\Phi(c) - (1 - \Phi(c)) = \gamma \Longrightarrow \Phi(c) - 1 + \Phi(c) = \gamma$$

Then add 1 to the expression so it becomes

$$2\Phi(c) = \gamma + 1$$

Divide it by 2

$$\Phi(c) = \frac{\gamma+1}{2}$$

Now taking the inverse of the CDF will give us an expression for c

$$c = \Phi^{-1}\left(\frac{1+\gamma}{2}\right)$$

Now having an expression for c that can be used in our expression

$$P\left(-c \leq \frac{\hat{\mu}-\mu}{\sigma/\sqrt{n}} \leq c\right) = \gamma$$

we can start figuring out what the γ-confidence interval is. Doing this we start by multiplying all sides of the inequality with the denominator $\sigma/\sqrt{n}$ so we have

$$P\left(-c\frac{\sigma}{\sqrt{n}} \leq \hat{\mu} - \mu \leq c\frac{\sigma}{\sqrt{n}}\right) = \gamma$$

Then subtract $\hat{\mu}$ on all sides of the inequality

$$P\left(-c\frac{\sigma}{\sqrt{n}} - \hat{\mu} \leq -\mu \leq c\frac{\sigma}{\sqrt{n}} - \hat{\mu}\right) = \gamma$$

Lastly multiply all sides of the inequality with -1

$$P\left(c\frac{\sigma}{\sqrt{n}}+\hat{\mu}\geq\mu\geq\hat{\mu}-c\frac{\sigma}{\sqrt{n}}\right)=\gamma$$

Now from this we know that the γ-confidence interval for μ is

$$\left[\hat{\mu}-c\frac{\sigma}{\sqrt{n}};\hat{\mu}+c\frac{\sigma}{\sqrt{n}}\right]$$

b) The correct parameter lies outside of the confidence interval 3.6% of the time as can be seen below in the raw output:

```
Not matching in 360 (out of 10000) experiments, 3.6%
```

c) The correct parameter lies outside of the confidence interval 0.97% of the time as can be seen below in the raw output:

```
Not matching in 97 (out of 10000) experiments, 0.97%
```

To find the confidence interval we will repeat the same steps as in task a) however, this time we will not spend time finding an expression for c, since as seen in the lecture, they are gonna be similar.
So having the expression

$$P\left(-c\leq\frac{\hat{\mu}-\mu}{\sigma/\sqrt{n}}\leq c\right)=\gamma$$

we repeat the exact steps taken in exercise a), but we keep in mind that we now have S instead of σ

$$P\left(-c\leq\frac{\hat{\mu}-\mu}{S/\sqrt{n}}\leq c\right)=\gamma$$

$$\implies P\left(-c\frac{S}{\sqrt{n}}\leq\hat{\mu}-\mu\leq c\frac{S}{\sqrt{n}}\right)=\gamma$$

$$\implies P\left(-c\frac{S}{\sqrt{n}}-\hat{\mu}\leq-\mu\leq c\frac{S}{\sqrt{n}}-\hat{\mu}\right)=\gamma$$

$$\implies P\left(c\frac{S}{\sqrt{n}}+\hat{\mu}\geq\mu\geq\hat{\mu}-c\frac{S}{\sqrt{n}}\right)=\gamma$$

Now from this we know that the γ-confidence interval for μ with unknown variance is

$$\left[\hat{\mu}-c\frac{S}{\sqrt{n}};\hat{\mu}+c\frac{S}{\sqrt{n}}\right]$$

Exercise 4

a) Choose the null hypothesis $\mu_0 = 0.5$. This choice is based off of the that fact that our distribution is defined as $X_i - Y_i$ and looking at the table in the assignment, we notice that for 3 of the plants the difference is 0.5 and the two others it is 1.0 and 1.5.

b) Using our null hypothesis from task a) we start by formulating an alternative hypothesis which is gonna be $\mu_A > \mu_0$. Hence we have to do a right-sided t-test and from the assignment we are given the level $\alpha = 0.05$.
Our test statistics is then t-distributed with d.f. $n - 1 = 4$

$$T = \frac{\hat{\mu} - \mu_0}{S/\sqrt{n}}$$

Now we start by computing the sample mean

$$\hat{\mu} = \frac{1}{5}(4.1 - 3.1 + 4.8 - 4.3 + 4.0 - 4.5 + 4.5 - 3.0 + 4.0 - 3.5) = 0.6$$

The next step is then to compute the standard deviation S

$$S = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(X_j - \hat{\mu})^2}$$

$$S = \sqrt{\frac{1}{4}((1 - 0.6)^2 + (0.5 - 0.6)^2 + (0.5 - 0.6)^2 + (1.5 - 0.6)^2 + (0.5 - 0.6)^2)}$$

$$S = 0.5$$

From this we can input our values into T

$$T = \frac{0.6 - 0.5}{0.5/\sqrt{5}} = 0.4472135954$$

The next step is to calculate c

c) Yes the scientist can change the result by illegally duplicating the experiment results, since this will increase the value n and then decrease the weight of the other tests. This will lead to a different sample mean, standard deviation and denominator for T.

Appendix

pca_StudentVersion.ipynb

```
import numpy.matlib

def pca(data):
    mean = data.mean(1)
    norm = (data.T - mean).T #Data transformed to have zero mean

    C = np.cov(norm) #covariace matrix

    PCevals, PCevecs = np.linalg.eig(C)

    #https://stackoverflow.com/questions/8092920/sort-eigenvalues-and-associated-eigenvectors-after-using-numpy
    idx = PCevals.argsort()[::-1]
    PCevals = PCevals[idx]
    PCevecs = PCevecs[:,idx]

    return PCevals, PCevecs, norm

PCevals, PCevecs, norm = pca(diatoms) #Data_cent unaware of what was desired with this
print(norm.shape)
print(PCevecs.shape)
#PCevals is a vector of eigenvalues in decreasing order. To verify, uncomment:
#print(PCevals)
#print(PCevecs)
#PCevecs is a matrix whose columns are the eigenvectors listed in the order of decreasing eigenvectors

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

diatoms_along_pc = np.zeros((7, 180))
for i in range(7):
    diatoms_along_pc[i,:] = mean_diatom + (e4 * ((i-3) * std4))


for i in range(7):
    plot_diatom(diatoms_along_pc[i])

plt.title('Diatom shape along PC4')
```

confidenceinterv.py

```
#!/usr/bin/env python
# coding: utf-8
#
```

```python
# # Confidence Intervals
#
# Jonas Peters, 20.11.2018
# Modified by Kim Steenstrup Pedersen, 23.11.2020

# You will have to modify the code in the places marked with TODO
# Notice that the code is constructed such that for small number of experiments
# (nexp) the code also makes a plot of the confidence interval from each experiment


import scipy.stats
import matplotlib.pyplot as plt
import numpy as np

# Fix the random generator seed
np.random.seed(111)


# Ground truth values
mu = 3.7
sigma = 2

# Number of samples in each experiment
n = 9

# Confidence level
gamma = 0.99 # 99 %

# Number of experiments to carry out
nexp = 10000 # TODO: Change this when you have developed your code


counter = 0
counter_c = 0
for i in range(nexp):
    x = np.random.normal(mu,sigma,n) # simulates n realizations from a Gaussian with mean mu and var sigma^2
    sig = np.sqrt(np.var(x, ddof = 1)) #sigma # TODO: adapt for b)
    fac1 = scipy.stats.norm.ppf((1-gamma)/2, 0, 1) # computes the 0.5% quantile of a Gaussian, roughly -2.576
    fac2 = scipy.stats.norm.ppf((1-gamma)/2 + gamma, 0, 1) # computes the 99.5% quantile of a Gaussian, roughly
    xmean = np.mean(x) # Sample mean
    a = xmean - fac2*sig/np.sqrt(n)
    b = xmean - fac1*sig/np.sqrt(n)
    fac1t = scipy.stats.t.ppf((1-gamma)/2, n-1)
    fac2t = scipy.stats.t.ppf((1-gamma)/2 + gamma, n-1)
    ac = xmean - fac2t*sig/np.sqrt(n) # TODO: adapt for c)
    bc = xmean - fac1t*sig/np.sqrt(n) # TODO: adapt for c)

    # b) plotting and counting code
    if (a <= mu) & (mu <= b):
        if nexp < 1000:
            plt.figure(1)
            plt.plot((a, b), (i, i), 'k-')
    else:
        counter = counter + 1
        if nexp < 1000:
            plt.figure(1)
            plt.plot((a, b), (i, i), 'r-')

    # c) plotting and counting code
    if (ac <= mu) & (mu <= bc):
        if nexp < 1000:
            plt.figure(2)
            plt.plot((ac, bc), (i, i), 'k-')
    else:
        counter_c = counter_c + 1
        if nexp < 1000:
            plt.figure(2)
            plt.plot((ac, bc), (i, i), 'r-')


# Number of times the correct mu and confidence interval is not matching
print(str(100.0 * gamma) + "%-confidence interval:")
print("b) Not matching in " + str(counter) + " (out of " + str(nexp) + ") experiments, " + str(100.0*counter/ne
print("c) Not matching in " + str(counter_c) + " (out of " + str(nexp) + ") experiments, " + str(100.0*counter_


if nexp < 1000:
    plt.figure(1)
    plt.plot((mu, mu), (0, nexp), 'b-')
    plt.xlabel('$\hat{\mu}$')
    plt.ylabel('Number of experiments')
    plt.title('b) The correct $\mu$ value in blue')

    plt.figure(2)
    plt.plot((mu, mu), (0, nexp), 'b-')
    plt.xlabel('$\hat{\mu}$')
    plt.ylabel('Number of experiments')
    plt.title('c) The correct $\mu$ value in blue')
    plt.show()
```