MAD Assignment 3

Andreas V. W. Zacchi (nzl169)

December 8, 2024

Exercise 1

a)

After implementing PCA we get the percentage covered by the x-th components as (rounded to 2 decimals):

1	77.19%
2	92.77%
3	95.21%
4	96.38%
5	97.39%
6	98.24%
7	98.90%
8	99.10%
9	99.27%
10	99.40%

Table 1: Table showing the proportion of the variance explained by the first x-th components

b)

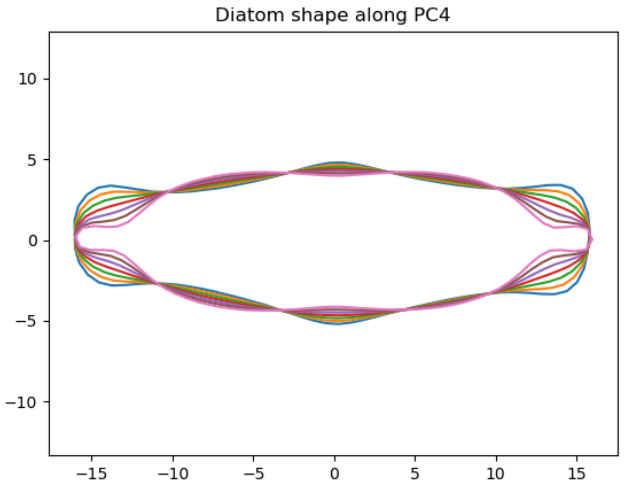

Figure 1: Image showing the fourth component of the PCA with the given multiplier

The mean is red and is smoothly curved, but when multiplying by either positive or negative values it becomes "jaggered" indicating that this component is in charge of how "jaggered" the curve is especially in the corners.

Exercise 2

X is given as a random variable with mean $E[X]$ and variance $E[(X-\mu)^2]$. Assuming g is a convex function Jensen inequality says that:

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

Using $g(Y) = Y^2$, which is convex, and setting $Y = (X-\mu)^2$ we get $g(Y) = ((X-\mu)^2)^2 = (X-\mu)^4$. We can now use Jensen inequality:

$$\begin{aligned} E[g(Y)] &\geq g(E[Y]) \Rightarrow \\ E[Y^2] &\geq E[Y]^2 \Rightarrow \\ E[(X-\mu)^4] &\geq E[(X-\mu)^2]^2 \end{aligned}$$

As $\sigma^2 = E[(X-\mu)^2]$ we can rewrite:

$$\begin{aligned} E[(X-\mu)^4] &\geq (\sigma^2)^2 \Rightarrow \\ E[(X-\mu)^4] &\geq \sigma^4 \end{aligned}$$

We have now shown the inequality holds.

Exercise 3

a)

In the lecture the confidence interval for μ of a Normal distributed sample with known variance (the estimator) is given by:

$$[\bar{X}_n - c\frac{\sigma}{\sqrt{n}}; \bar{X}_n + c\frac{\sigma}{\sqrt{n}}]$$

Inserting the two estimators we get the confidence interval as:

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

The critical value c is calculated by:

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

b)

We edit the sigma used in the calculations to fit the given estimator as seen below:

```
1     sig = np.sqrt(np.var(x, ddof=1))
```

We obtain that 3.6% is outside the confidence interval, and we are told a good fit should have less than $1 - \gamma = 1 - 0.99 = 0.01$ and since $0.036 > 0.01$ this confidence interval doesn't provide a good fit.

c)

As we have just changed the estimator $\hat{\sigma}$ the confidence interval remains the same:

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

However to calculate c we now need to use the inverse of the CDF for a student-t distribution:

$$c = F^{-1}(\frac{1+\gamma}{2})$$

Again we edit the code and calculate the c that is needed to calculate the two bounds correctly:

```
    c = scipy.stats.t.ppf((1 + gamma ) / 2, n-1)
    ac = xmean - c*sig/np.sqrt(n)
    bc = xmean + c*sig/np.sqrt(n)
```

After this we obtain that 0.97% lies outside the confidence interval, as $0.0097 < 0.01$ this means that we now have a good estimate for our confidence interval.

Exercise 4

a)

We chose the null hypothesis where the gene knockout has no effect. This is done as we would rather say that the gene knockout has no effect, even though it might have some, than to falsely say it has an effect (Type I error). Meaning we have:

$$H_0 : \mu = 0$$
$$H_A : \mu \neq 0$$

b)

We have the null and alternate hypothesis from the previous subtask, we are also given $\alpha = 0.05$. As we are checking the differences we first have to calculate those:

$$\begin{aligned} d &= [4.1 - 3.1, 4.8 - 4.3, 4.0 - 4.5, 4.5 - 3.0, 4.0 - 3.5] \\ &= [1, 0.5, -0.5, 1.5, 0.5] \end{aligned}$$

We make use of the two-sided t-test shown at the lecture:

$$T = \frac{\bar{X} - \mu_0}{S/\sqrt{n}}$$

Now we need to calculate the mean and standard deviation of the samples:

$$\bar{d} = \frac{1.0 + 0.5 - 0.5 + 1.5 + 0.5}{5} = 0.6$$

$$\begin{aligned} s &= \sqrt{\frac{\sum_{i=1}^{n}(d_i - \bar{d})^2}{n-1}} \\ &= \sqrt{\frac{(1.0-0.6)^2 + (0.5-0.6)^2 + (-0.5-0.6)^2 + (1.5-0.6)^2 + (0.5-0.6)^2}{4}} \\ &= \sqrt{\frac{2.2}{4}} = 0.74161984871 \end{aligned}$$

Now we can compute the test statistics as

$$\begin{aligned} t &= \frac{\bar{d} - \mu_0}{s/\sqrt{n}} \\ &= \frac{0.6 - 0}{0.74161984871/\sqrt{5}} \\ &= \frac{0.6}{0.74161984871/\sqrt{5}} \\ &= 1.80906806747 \end{aligned}$$

We find c_1 and c_2 by using `scipy.stats.t.ppf` which gives 2.776 and -2.776 respectively. As $1.809 < 2.776$ we don't reject the null hypothesis.

c)

Yes it's possible. The mean would remain the same while the standard deviation would become smaller when k increases. However the hypothesis test assumes independent and identically distributed random variables, and in the case of duplication this is not the case.

Appendix

confidenceinterv.py

```
#!/usr/bin/env python
# coding: utf-8
#
# # 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)) # Use the estimator given in the task
    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 2.576
    xmean = np.mean(x) # Sample mean
    a = xmean - fac2*sig/np.sqrt(n)
    b = xmean - fac1*sig/np.sqrt(n)

```

```
    c = scipy.stats.t.ppf((1 + gamma ) / 2, n-1) # computes the
    critical value for the given gamma with n-1 degrees of freedom

    ac = xmean - c*sig/np.sqrt(n) # TODO: adapt for c)
    bc = xmean + c*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/nexp) + "%")
print("c) Not matching in " + str(counter_c) + " (out of " + str(
    nexp) + ") experiments, " + str(100.0*counter_c/nexp) + "%")


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()
```

pca_StudentVersion.ipynb

For the Jupyter Notebook please see the file in the zip directory