Lecture “Advanced Data Analytics”

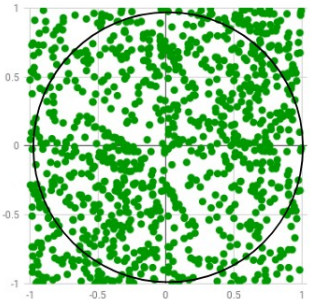
Problem Set 1

Simon Scheidegger

This problem set will give you the chance to practice the content thought in the first two lectures of this course.

Exercise 1

Monte Carlo estimation are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. One of the basic examples of getting started with the Monte Carlo algorithm is the estimation of Pi.
The idea is to simulate random (x, y) points in a 2-D plane with domain as a square of side 1 unit. Imagine a circle inside the same domain with same diameter and inscribed into the square. We then calculate the ratio of number points that lied inside the circle and total number of generated points. Refer to the image below:



We know that area of the square is 1 unit sq while that of circle is $\pi * (\frac{1}{2})^2 = \frac{\pi}{4}$
Now for a very large number of generated points,

$$\frac{\text{area of the circle}}{\text{area of the square}} = \frac{\text{no. of points generated inside the circle}}{\text{total no. of points generated or no. of points generated inside the square}}$$

that is,

$$\pi = 4 * \frac{\text{no. of points generated inside the circle}}{\text{no. of points generated inside the square}}$$

The beauty of this algorithm is that we don’t need any graphics or simulation to display the generated points. We simply generate random (x, y) pairs and then check if $x^2 + y^2 \leqslant 1$
If yes, we increment the number of points that appears inside the circle. In randomized and simulation algorithms like Monte Carlo, the more the number of iterations, the more accurate the result is.

Your task is to implement a code that computes Pi as described above. Moreover, your code should print out a table how the accuracy of the estimate of Pi improves with the number of samples.

Exercise 2

Make a program that simulates flipping a coin N times. Print out “tail” or “head” for each flip and let the program count the number of heads. (Hint: Use r = random.random() and define head as r <= 0.5, or draw an integer among {1, 2} with r = random.randint(1,2) and define head when r is 1.)

Exercise 3

Computing probabilities of rolling dice with Python.

1. You throw a die. What is the probability of getting a 6?

2. You throw a die four times in a row. What is the probability of getting 6 all the times?

3. Suppose you have thrown the die three times with 6 coming up all times. What is the probability of getting a 6 in the fourth throw?

4. Suppose you have thrown the die 100 times and experienced a 6 in every throw. What do you think about the probability of getting a 6 in the next throw?

First try to solve the questions from a theoretical or common sense point of view. Thereafter, make functions for simulating cases 1, 2, and 3.

Exercise 4

Compute probabilities of throwing two dices with Python.

Make a computer program for throwing two dice a large number of times. Record the sum of the eyes each time and count how many times each of the possibilities for the sum (2, 3, . . . , 12) appear. A dictionary with the sum as key and count as value is convenient here. Divide the counts by the total number of trials such that you get the frequency of each possible sum. Write out the frequencies and compare them with exact probabilities. (To find the exact probabilities, set up all the 6×6 possible outcomes of throwing two dice, and then count how many of them that has a sum s for s = 2, 3, . . . , 12).

Exercise 5

In a laboratory experiment waves are generated through the impact of a model slide into a wave tank. (The intention of the experiment is to model a future tsunami event in a fjord, generated by loose rocks that fall into the fjord.) At a certain location, the elevation of the surface, denoted by η , is measured at discrete points in time using an ultra-sound wave gauge. The result is a time series of vertical positions of the water surface elevations in meter: $\eta(t0)$, $\eta(t1)$, $\eta(t2)$, . . . , $\eta(tn)$. There are 300 observations per second, meaning that the time difference between two neighboring measurement values $\eta(ti)$ and $\eta(ti+1)$ is h = 1/300 second.

Write a Python program that accomplishes the following tasks:

1. Read h from the command line.

2. Read the η values in the file **dat/gauge.dat** into an array eta.

3. Plot eta versus the time values (use matplotlib)

4. Compute the velocity v of the surface by the formula

$$v_i \approx \frac{\eta_{i+1} - \eta_{i-1}}{2h}, \quad i = 1, \ldots, n-1.$$

Plot v versus time values in a separate plot.

5. Compute the acceleration a of the surface by the formula

$$a_i \approx \frac{\eta_{i+1} - 2\eta_i + \eta_{i-1}}{h^2}, \quad i = 1, \ldots, n-1.$$

Plot a versus the time values in a separate plot.

Exercise 6

Differentiate noisy signals in Python.

The purpose of this exercise is to look into numerical differentiation of time series signals that contain measurement errors. This insight might be helpful when analyzing the noise in real data from a laboratory experiment in Exercise 5.

1. Compute a signal

$$\bar{\eta}_i = A \sin\left(\frac{2\pi}{T} t_i\right), \quad t_i = i\frac{T}{40}, \; i = 0, \ldots, 200.$$

Display η_bar versus time t_i in a plot. Choose A = 1 and T = 2π . Store the η_bar values in an array etabar.

2. Compute a signal with random noise E_i ,

$$\eta_i = \bar{\eta}_i + E_i,$$

E_i is drawn from the normal distribution with mean zero and standard deviation σ = 0.04A. Plot this η_i signal as circles in the same plot as η_bar. Store the E_i in an array E for later use.

3. Compute the first derivative of η_bar by the formula

$$\frac{\bar{\eta}_{i+1} - \bar{\eta}_{i-1}}{2h}, \quad i = 1, \ldots, n-1,$$

and store the values in an array detabar. Display the graph.

4. Compute the first derivative of the error term by the formula

$$\frac{E_{i+1} - E_{i-1}}{2h}, \quad i = 1, \ldots, n-1,$$

and store the values in an array dE. Calculate the mean and the standard deviation of dE.

5. Plot detabar and detabar + dE. Use the result of the standard deviation calculations to explain the qualitative features of the graphs.

6. The second derivative of a time signal η_i can be computed by

$$\frac{\eta_{i+1} - 2\eta_i + \eta_{i-1}}{h^2}, \quad i = 1, \ldots, n-1.$$

Use this formula on the etabar data and save the result in d2etabar. Also apply the formula to the E data and save the result in d2E. Plot d2etabar and d2etabar + d2E. Compute the standard deviation of d2E and compare with the standard deviation of dE and E. Discuss the plot in light of these standard deviations.

Exercise 7

Consider the following list of household appliances.

Category	Prize	kWh/Jahr	Energy Efficiency	Weight	Brand	Sales
Freezer	699	210	A	99	Mühle	27.96
Washing machine	399	302	B	50	Boss	99.75
Freezer	779	475	C	75	Nerd	33.558
Washing machine	989	705	E	125	Prolet	3.996
tableware dryer	199	790	D	42	Boss	2.786
tableware dryer	479	810	D	37	Mühle	21.076

a) List for all of the seven features whether is is a nominal, ordinal, or a numerical feature.

b) Encode the nominal and ordinal features such that one can use them for linear regression.

Exercise 8 (solve after lecture week 3!)

Go to the **dat/Boston.dat** and look at the data set of housing prices from Boston.

a) Read the description of the features, and explain whether they are nominal, ordinal, or numerical.

b) Use the numerical features to fit linear models (you can choose the features to fit – based on generating scatter plots and correlations of the different features). Use scikit-learn.
Report the Regression parameters, R^2, and the MSE. Do not only use linear regression, but also higher-order polynomials.

c) Implement the linear regression in Python yourself. To do so, use the formula

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

derived in the lecture. Compare the resulting Parameters, R^2, and the MSE computed here with the onces computed in b).

Exercise 9

Solve the optimization problem stated below with a Python optimization tool-kit of your choice.
To do so, solve the problem once by providing the i) analytical Jacobian and Hessian,
and once by providing ii) a first-order finite-difference implementation of the Hessian and Jacobian to the optimizer.
Compare the results of i) and ii) – what do you observe?

$\min x_1 x_4 (x_1 + x_2 + x_3) + x_3$	Objective
$\text{s.t.} \quad x_1 x_2 x_3 x_4 \geq 25$	Inequality constraint
$x_1^2 + x_2^2 + x_3^2 + x_4^2 = 40$	Equality constraint
$1 \leq x_1, x_2, x_3, x_4 \leq 5$	Bounds
$x_0 = (1, 5, 5, 1)$	Initial guess