Linear regression and data generation

In this exercise, you should write a Python program that can fit experimental data using linear regression. Furthermore, you are going to create a generator for randomized experimental data. You are not permitted to use any libraries that carry out the fitting for you, i.e., the algorithm described below must be fully implemented.

Requirements and basic functionality

Your program should be **split into two parts**: The first one should be the data generator, while the second one should be your implementation of the fitting algorithm. **Please split the two parts into separate files as described below.**

Part 1 - The data generator

Write your code for this part into the file `generator.py`. For this part, in addition to any built-in packages that come with Python, you are only allowed to use the package NumPy. The basic idea is that your program should generate data points (X_i, Y_i) with linear dependence $f(x) = a \cdot x + b$ of the Y_i on the X_i and random Gaussian noise. Your program should have the following **basic functionality**:

- There should be a main function that gets executed when the user runs the program from the command line.
- When the main function is run, the user should be asked for five parameters:
  - `n`: The number of data points (X_i, Y_i) to generate
  - `x_max`: The maximum value for X_i, such that $X_i \in [0, \texttt{x\_max}]$.
  - `a`: The slope of the linear dependence (the a in $f(x) = a \cdot x + b$)
  - `b`: The value of $f(0)$, where $f(x) = a \cdot x + b$ (the intersection with the y-axis)
  - `filename`: The name of the file to write the data to
- The program should use the given arguments to produce `n` data points (X_i, Y_i), where the X_i are **equally spaced** on $[0, \texttt{x\_max}]$ (*Hint:* Use NumPy for this). The data points should describe a linear dependence of the Y_i on the X_i with the given parameters `a` and `b` (see description above).
- The program should add **Gaussian noise** to the Y_i values, i.e., each Y_i should be displaced by a random distance from its original value. The random amount should be drawn from a normal distribution
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \tag{1}$$
with mean $\mu = 0$ and standard deviation $\sigma = 0.5$. (Alternatively, draw the Y_i directly from a normal distribution with $\sigma = 0.5$ and an appropriate value for μ)
- The resulting data points (X_i, Y_i) should be written to the file `filename` **in text format** using the appropriate NumPy function. Use a comma (,) as a delimiter.
- The user should be informed that the data was saved successfully.

If you are unsure of how the result should look like, have a look at Figure 1 in the section *Example output* below.

Part 2 - The fitting algorithm

The algorithm you are going to implement is the following: Let (X_i, Y_i) be n data points known from experiments, where $i = 1, 2, \ldots, n$. We suspect there to be a linear dependence of the Y_i on the X_i. Our goal is therefore to find a function

$$f(x) = a \cdot x + b \tag{2}$$

with parameters a, $b \in \mathbb{R}$ that fits the given data, such that the sum S of squared distances between $f(X_i)$ and Y_i is minimized, i.e.,

$$S = \sum_{i=1}^{n} [f(X_i) - Y_i]^2 \quad \text{minimal.} \tag{3}$$

To do this, we need to find the appropriate values for a and b that minimize S.

These values can be calculated using the equations

$$a = \frac{[n \cdot \sum_{i=1}^{n} X_i \cdot Y_i] - [\sum_{i=1}^{n} X_i] \cdot [\sum_{i=1}^{n} Y_i]}{[n \cdot \sum_{i=1}^{n} X_i^2] - [\sum_{i=1}^{n} X_i]^2}, \tag{4}$$

$$b = \frac{1}{n} \left[\sum_{i=1}^{n} Y_i - a \cdot \sum_{i=1}^{n} X_i \right]. \tag{5}$$

Your task: Write a Python program that implements this fitting algorithm. For this part, in addition to any built-in packages that come with Python, you are only allowed to use the package NumPy. **You are not allowed to use any fitting capabilities that come with NumPy** (you must implement the above algorithm manually).

Write the code for this part into the file `fit.py`. Your program should have the following **basic functionality**:

- Your program should have a main function that gets executed when the user runs the program from the command line.
- When the program is run, the user is asked to provide a `filename` as an input. You can expect the file to contain data points (X_i, Y_i), where each data point has its own line and the X_i and the Y_i are separated by commas (,). Have a look at the file `example_data.csv` given to you.
- The program should read the file `filename` and perform a linear regression using the algorithm described above. It should then print the determined values for the parameters a and b.

To test your program, you can use the file `example_data.csv`, which should, if fitted correctly, produce the parameters $a = 0.9999359614031943$ and $b = 5.048183002644452$. This file also shows you how the data is meat to be saved (two columns, the first one containing the X_i, the second one the Y_i)

Allowed/forbidden tools an material

You are permitted to use any online/offline resources you like, **except**

- Generative AIs like ChatGPT
- Communication with other people

Grading

Including the following in your solution will give you the specified number of points (partial points are of course possible, each of the two parts contributes equally to each of the following points, except if stated otherwise):

- The basic functionality described above is implemented [50/100 Pts.]
- Some part of your programs implements object orientated programming in a meaningful way [15/100 Pts.]
- The object oriented part of your programs uses getters and setters in a meaningful way [10/100 Pts.]
- Proper error handling (user input is checked and the program doesn't crash or end on illegal input/keyboard interrupt, the user is asked to re-enter illegal input instead of having to restart the program) [15/100 Pts.]
- The code is documented and well organized [10/100 Pts.]
- *(Bonus)* **For the data generator:** Alternatively to asking the user for the five parameters, use **docopt** to create a command line interface such that the user can provide the arguments before running the program [10 bonus Pts.]
- *(Bonus)* Plot the output of the data generator **and** the linear regression using a plotting library of your choice [10 bonus Pts.]

Example runs

This in an example run of the **data generator** (how it COULD look like)

```
Number of data points (n): 100
Maximum x-value (x_max): 10
Slope (a): 1
Intersection with y-axis (b): 5
Filename to write results to: example_data.csv

Saved results to example_data.csv
```

In Figure 1 you can see an example result of the generator.

This is an example run of the **fitting program** (how it COULD look like):

```
Filename to read data from: example_data.csv

Results:
a = 0.9999359614031943
b = 5.048183002644452
```

In Figure 2 you can see an example of the result of the fit.

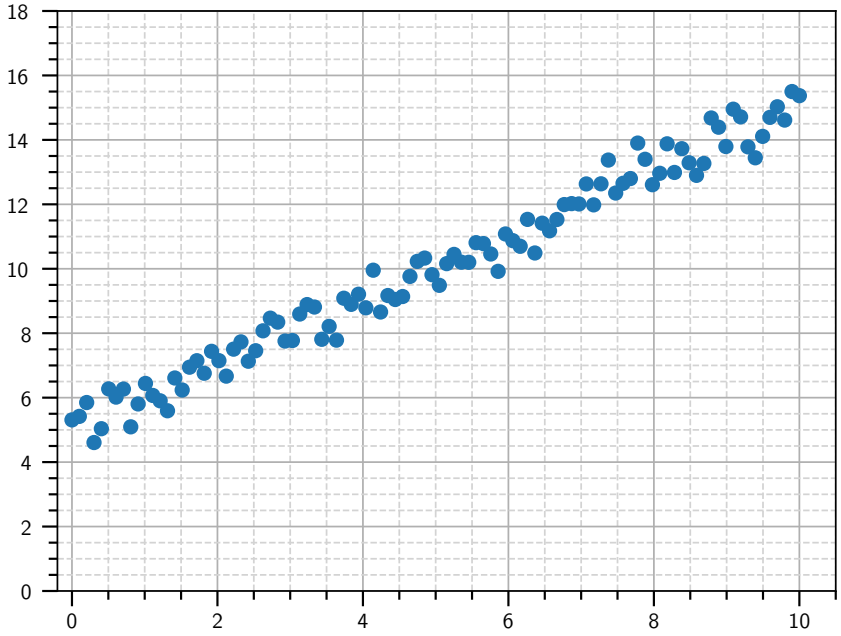

Figure 1: Sample output of the data generator ($a = 1$, $b = 5$, $n = 100$, `x_max` $= 10$). You can easily see the linear dependence here, where the slope is 1 and the intersection with the y-axis is 5.

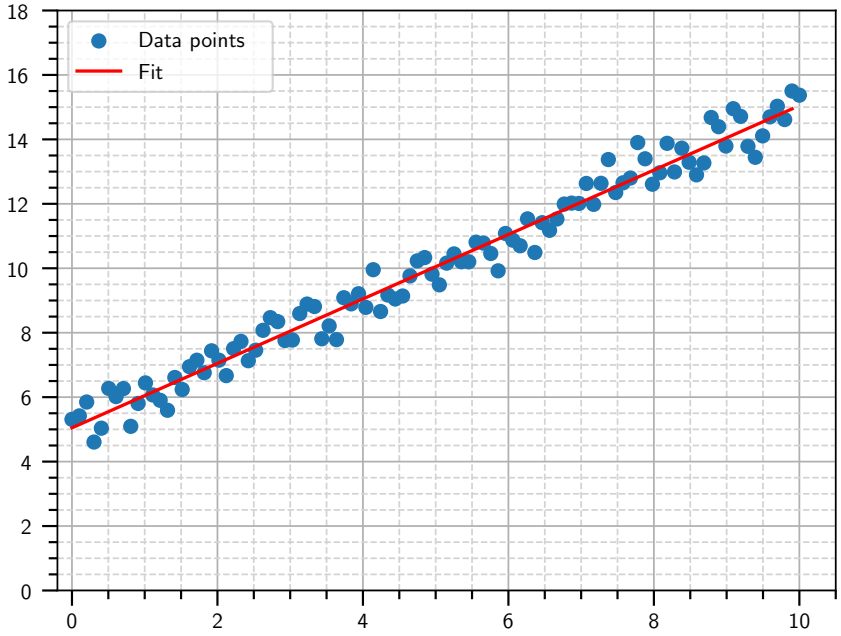

Figure 2: Sample output of the fit ($a = 0.9999359614031943$, $b = 5.048183002644452$)