In-class Exercise 4.1
Least squares fitting methods
AMATH 301
University of Washington
Jakob Kotas

1. Given the dataset:

x	1.05	1.1	1.35	1.6	1.95	2.15	2.4	2.5	2.65	2.8
y	1.68	1.71	1.55	0.74	0.31	−0.22	−0.92	−1.37	−1.32	−1.74

   (a) Plot the data points as blue dots on xy-axes.

   (b) Solve the system of linear equations:

$$\begin{bmatrix} \sum_{k=1}^n x_k^2 & \sum_{k=1}^n x_k \\ \sum_{k=1}^n x_k & n \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^n x_k y_k \\ \sum_{k=1}^n y_k \end{bmatrix}$$

   and plot the best-fit line:

$$y = Ax + B.$$

   (c) Use `np.polyfit` to find the coefficients of the 1st order polynomial (linear) fit. These should be the same values as part (b).

   (d) Make a surface plot in 3D of the mean squared error:

$$E_2(A, B) = \sum_{k=1}^{n} (Ax_k + B - y_k)^2$$

   and show that the minimum (A, B) is the same value found in parts (b) and (c).
   Visualization of #1d

   (e) Find the system of linear equations that would need to be solved to find the best-fit parabola:

$$y = Ax^2 + Bx + C.$$

The sum of the square errors is:

$$E_2 = \sum_{k=1}^{n} (Ax_k^2 + Bx_k + C - y_k)^2$$

$$\frac{\partial E_2}{\partial A} = \sum_{k=1}^{n} 2(Ax_k^2 + Bx_k + C - y_k)(2x_k^2) = 0$$

$$\frac{\partial E_2}{\partial B} = \sum_{k=1}^{n} 2(Ax_k^2 + Bx_k + C - y_k)(x_k) = 0$$

$$\frac{\partial E_2}{\partial C} = \sum_{k=1}^{n} 2(Ax_k^2 + Bx_k + C - y_k) = 0$$

So the linear system is:

$$\begin{bmatrix} \sum_{k=1}^n x_k^4 & \sum_{k=1}^n x_k^3 & \sum_{k=1}^n x_k^2 \\ \sum_{k=1}^n x_k^3 & \sum_{k=1}^n x_k^2 & \sum_{k=1}^n x_k \\ \sum_{k=1}^n x_k^2 & \sum_{k=1}^n x_k & n \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^n x_k^2 y_k \\ \sum_{k=1}^n x_k y_k \\ \sum_{k=1}^n y_k \end{bmatrix}$$

(f) Solve the system from part (e) and show that it matches `np.polyfit` for a 2nd order polynomial (parabolic) fit.

2. Given the dataset:

x	-2	-1	0	1	2
y	0.53	1.11	1.89	7.68	18.0

We wish to fit an exponential function:

$$y = Ce^{Ax}$$

to the dataset, where A and C are unknown constants.

(a) As on p. 89 of the textbook, the sum of squared errors is:

$$E_2(A, C) = \sum_{k=1}^{n} (Ce^{Ax_k} - y_k)^2$$

so the system of nonlinear equations to be solved is:

$$\frac{\partial E_2}{\partial A} = \sum_{k=1}^{n} 2(Ce^{Ax_k} - y_k)(Cx_k e^{Ax_k}) = 0$$

$$\frac{\partial E_2}{\partial C} = \sum_{k=1}^{n} 2(Ce^{Ax_k} - y_k)e^{Ax_k} = 0$$

After importing: `from scipy.optimize import fsolve`, use `fsolve` to find the minimum. Use a starting guess of $(A, C) = (1, 3)$. Plot the dataset as blue dots and the exponential fit function as a red curve.

Visualization of #2a

(b) Repeat (a) with a starting guess of $(A, C) = (0, 0)$ or $(A, C) = (1, 1)$. We can see a sensitive dependence on initial guess.

(c) A more robust approach is to let $z = \ln(y)$ and find the best-fit line of x vs z, then recover $y = e^z$. Then, we only need to solve a linear system of equations (much easier!) Implement this in Python and plot the fit as a green curve. Are A and C the same as part (a)? Should they be?

$$z = \ln(y) = \ln(Ce^{Ax}) = \ln(C) + Ax$$

3. Other alternatives for defining the "best fit" involve the errors below:

Maximum error (worst-case):

$$E_\infty = \max_{k \in 1,2,\cdots,n} |f(x_k) - y_k|$$

Average error:

$$R_1 = \frac{1}{n} \sum_{k=1}^{n} |f(x_k) - y_k|$$

(we note that minimizing R_1 is equivalent to minimizing E_1:)

$$E_1 = \sum_{k=1}^{n} |f(x_k) - y_k|$$

Root-mean-square error:

$$R_2 = \left(\frac{1}{n} \sum_{k=1}^{n} |f(x_k) - y_k|^2 \right)^{1/2}$$

(we note that minimizing R_2 is equivalent to minimizing E_2:)

$$E_2 = \sum_{k=1}^{n} (f(x_k) - y_k)^2$$

Given the dataset:

x	1	2	3
y	0	0	1

(a) Plot the data set as blue dots.

(b) Find the best-fit line in the least-squares (root-mean-square error) sense by minimizing E_2. You can use `np.polyfit`. Plot the line in red.

(c) Find the best-fit line in the maximum error sense by minimizing E_∞. You can use `scipy.optimize.fmin`. Plot the line in green.

(d) Find the best-fit line in the average error sense by minimizing E_1. You can use `scipy.optimize.fmin`. Plot the line in yellow.

4. This problem uses a data set from this source.

   Suppose you are a social science researcher who has queried 500 random recent college graduates about their income and happiness level. Their income is in ten thousands of dollars and their happiness is on a scale of 1 to 10. Is there a correlation between income and happiness?

   Load the data set `incomehappiness.csv` using `np.loadtxt`. Perform a linear regression. How many happiness points (higher or lower) does the average person feel when their income is increased by $10,000?