Prof. Dr. Volker Roth
volker.roth@unibas.ch
6^{th} floor

Sebastian Keller
sebastianmathias.keller@unibas.ch
1^{st} floor

Departement Mathematik und Informatik
Spiegelgasse 1
4051 Basel

University of Basel

Solutions for exercises marked with a '$\star$' will be made available online, usually in the following week. Only if needed will these exercises be discussed during the tutorial. All other exercises are to be prepared at home and presented by the participants during the tutorial session.

Exercise 5

The setting for the general linear model can be formulated as follows: Two quantities x and y are related by some function $y = f(x)$. The function itself is unknown, but assumed to be a *linear combination of several base functions*. The task is to find the appropriate coefficients based on a set of observations of x and y.

5.1 Geometric Approach*

Consider the model $y = b_1 x + b_2$. The geometric approach aims to minimize the sum of squared residuals $\sum_i r_i^2 = \sum_i [y_i - (b_1 x_i + b_2)]^2$. Computing the partial derivatives and setting them to zero, we get the following normal equations:

$$\underbrace{\begin{pmatrix} \sum_i x_i^2 & \sum_i x_i \\ \sum_i x_i & m \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}}_{\boldsymbol{b}} = \underbrace{\begin{pmatrix} \sum_i x_i\, y_i \\ \sum_i y_i \end{pmatrix}}_{\boldsymbol{c}}$$

Write a Python script to solve the following problem:

1. Compute matrix A and vector $\boldsymbol{c}$ for the $m = 20$ samples supplied in the file `data.dat`.
2. Using a SVD, solve for coefficients b_1 and b_2.
3. Plot the data points from `data.dat` and overlay the fitted model.
4. The true polynomial (with added noise) used to generate the data points in `data.dat` was $y(x) = -2x$. Add this to your plot as well.

5.2 Algebraic Approach

Assume we have samples (x_i, y_i) where $y_i = f(x_i) = b_1\, x_i^2 + b_2\, x_i + b_3$. Using this, we can construct the following system:

$$\underbrace{\begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_m^2 & x_m & 1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}}_{\boldsymbol{b}} = \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}}_{\boldsymbol{y}}$$

Matrix A is called the polynomial expansion of x and depends on the degree of the polynom (here: 2), resulting in 3 columns in A and 3 coefficients b_1, b_2 and b_3. The algebraic approach is solved by singular value decomposition.

(turn the page, please)

Write a Python script to solve the following problem:

1. Set up matrix A as shown above. $m = 20$ samples are supplied in `data2.dat`.
2. Decompose A using singular value decomposition.
3. Determine the condition number.
4. Compute the coefficients b_1, b_2 and b_3 using the singular value decomposition, i.e. by applying the pseudo inverse.
5. Calculate the residual (= deviation of the fitted function from the samples).
6. Zero the smallest singular value and re-compute condition number and the coefficients.
7. Calculate the condition number for the zeroed system.
8. Calculate the new residual.
9. Perform some meaningful plotting that will explain the differences (or advantages) of zeroing. (The true polynomial (with added noise) used to generate the data points in `data2.dat` was $y(x) = x^2 - x$.)