MASD 2022, Assignment 2

François Lauze, Stefan Sommer, Kasra Arnavaz

September 12, 2022

Guidelines for the assignment.

- This is a group assignment. Hand-in in groups of 2 or 3 latest 21.09.2022 at 21.59. One submission per group and remember to include the name of all group members.
- The assignment report must be uploaded in PDF format, we strongly recommand the use of LaTeX to create the PDF.
- Please pay careful attention to the plagiarism rules, see `https://absalon.ku.dk/courses/61325/pages/course-information`.

Exercise 1. (Concept of derivative)

1. In the sequel, $f : \mathbb{R} \to \mathbb{R}$ is a differentiable function, $t \mapsto f(t)$. Its derivative is denoted $\frac{df}{dt}$ (Leibnitz' notation) of $f'(t)$ (Euler and Lagrange's notation). Do the following expressions give $f'(t)$ at time t? Please justify your answers.
   (a) $\frac{f(t+h)-f(t)}{h}$, h small
   (b) $\lim_{h\to 0} \frac{f(t+4h)-f(t)}{h}$,
   (c) $\lim_{h\to 0} \frac{f(t+4h)-f(t)}{4h}$,
   (d) $\lim_{h\to 0} \frac{f(t+h)-f(t-h)}{2h}$,
2. Draw the shape of the derivative of the functions f shown in Figure 1. The important points are the signs of the derivatives, and when they are equal to 0 (one say that they "vanish"), no need for extreme accuracy for the locations of these vanishing points.

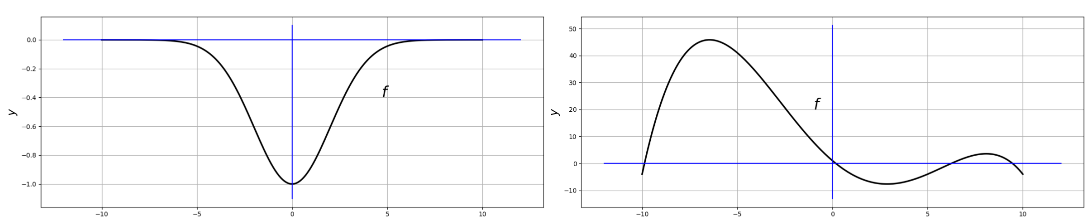

Figure 1: Draw the shapes of the derivative functions of these two functions

3. Assume that $f'(-1) = f'(1) = 0$ and that $f'(t) \neq 0$ when $t \notin \{-1, 1\}$. Draw the possible shapes of f' and of f. Explain why.
4. Assume that $f'(0) = 0$, $f''(0) > 0$. Draw the shape of f' and of f around 0. Do the same in the case $f'(0) = f''(0) = 0$. Here too, explain why.

Exercise 2 (Computing 1-D derivatives). In the following exercise, you have to compute derivatives of more or less complicated functions, using the different rules. For each computation, provide details of the rules used, e.g. chain rule, product rule etc. Don't go into to much details, just say: "from this form to that form, we used the x-rule".

1. $\frac{d}{dx}\sqrt{x^2+3}$,
2. $\frac{d}{dx}\sin(x)e^{-\cos(x)^2}$ (Remember that $\sin'(x) = \cos(x)$ and that $\cos'(x) = -\sin(x)$).
3. $\frac{d}{dx}\frac{\ln x}{x^3}$,
4. $\frac{d}{dx}\frac{\ln(1+e^{qx})}{q}$,
5. $\frac{d^2}{dx^2}\frac{\ln(1+e^{qx})}{q}$ (i.e., its second derivative)
6. $\frac{\partial}{\partial x}e^{(x^2+y)^3}$,
7. $\frac{\partial}{\partial x_i}\left(\mathbf{x}^T\mathbf{A}\mathbf{x}\right)$ where A is a square matrix and $\mathbf{x}$ a $n-$vector,

$$\mathbf{A} = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}.$$

   You may want to expand $\mathbf{x}^T\mathbf{A}\mathbf{x}$ when $n = 2$ or 3, and write with summation symbols for larger ns. *This is a very important example!*

8. $\frac{\partial}{\partial x_i}(\mathbf{A}\mathbf{x}-\mathbf{b})^T(\mathbf{A}\mathbf{x}-\mathbf{b})$, where this time A is a matrix in $\mathbb{R}^{m\times n}$, i.e. m lines and n columns, $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{x}$ a n-vector (*7. and 8. are very important examples!*).

Deliverable: The proof.

Exercise 3 (Gradients for data fitting). The file `A2-exercise3.ipynb` contains data on chocolate consumption (in kg per person per year) and Nobel prize laureates (per 100,000 inhabitants) for 20 countries. The data can be described as $((C_1, N_1), (C_2, N_2), \dots, (C_{20}, N_{20}))$ where C_i and N_i denote the chocolate consumption and Nobel prize laureate for country i. We assume a model for the data

$$N_i = aC_i + b, \quad , i = 1, 2, \dots, 20.$$

and want to discuss a and b values. To do so, we define a *badness* function:

$$badness : \mathbb{R}\times\mathbb{R} \to \mathbb{R}, \quad (a, b) \mapsto \frac{1}{20}\sum_{i=1} 20(N_i - aC_i - b)^2$$

which indicates how bad our (a, b)-model fits the data. The task is to find the pair of values (a, b) that minimise badness.

1. Implement *badness* as the function:
   `def badness(a, b, chocolate, nobel)` in the Python notebook.
2. Compute the gradient $\nabla_{(a,b)}badness$. Implement it as a function
   `def badness_gradient(a, b, chocolate, nobel)` in the Python notebook.
3. Gradient descent. Start with the values $a = 1$ and $b = 0$, compute the value of badness and its gradient and plot the model fit (using `plot_datafit`). Use the information in the gradient to find values of a and b that yield a badness of less than 50. Report the three values of a, b and badness. (Hint: If you move a tiny step into the negative direction of the gradient, i.e.

$$(a', b') = (a, b) - \lambda\nabla_{(a,b)}badness$$

   with a $\lambda > 0$ sufficiently small, you should be able to decrease *badness*. From there, recompute the gradient, move a step, and so on.
4. (Optional) what is the lowest badness you can achieve?

Deliverables: a) Code (properly commented) in the `.ipynb` and the `.pdf` files. b) formula; code – both uploaded in the ipython notebook and the report file. c) Values for a, b, and *badness*.