MASD 2022, Assignment 3

François Lauze, Stefan Sommer, Kasra Arnavaz

September 20, 2022

Guidelines for the assignment.

- This is a group assignment. Hand-in in groups of 2 or 3 latest 28.09.2022 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`.

1 Identifying critical points

Exercise 1. (Extremal and critical points.) For each of the following functions, find all critical points, classify them as minimum, maximum, inflection or saddle point. When multiple variables are present, we use x_1 and x_2 for scalar variables instead of x and y. You can rewrite some of the equations as matrix/vector ones and are allowed to use Python library functions for some of the calculations (solving linear systems of equations or computing eigenvalues), though all calculations should easily be performed on paper.

a) $f(x) = x^3 - 2x^2 + 11x + 7$.

b) $f(x) = x^n \ln x$, $n \in \mathbb{N}\backslash\{0\}$

c) $f(x_1, x_2) = 3x_1^2 + 2x_1x_2 + 2x_2^2 - 6$.

d) $f(x_1, x_2) = 3x_1^2 + 4x_1x_2 + x_2^2$.

2 The Netflix problem

Table 1 contains movie ratings ranging from 1 (bad) to 10 (very ggod) for 10 different movies and 6 movie lovers. The "-" symbol represents unrated movies, because the viewer has not seen them yet. The *Netflix* (or HBO, Prime etc.) problem is to predict the missing movie ratings to recommend movies that the viewers are likely to enjoy.

	Love Actually	Pride and Prejudice	Titanic	Lala Land	Bridget Jones' Diary	Scream	Halloween	It	Sharknado 3	Pride, Prejudice, and Zombies
Sophia	7	8	9	-	-	1	4	2	3	9
Anton	-	-	10	9	10	2	3	-	-	5
Fabio	10	9	-	8	-	-	-	2	1	3
Magda	1	-	2	-	-	9	8	9	-	-
Marietta	-	1	1	-	2	-	9	-	7	-
Carl	2	1	-	-	1	10	-	9	-	8

Table 1: A set of movie ratings with missing values corresponding to unseen movies.

In this exercise, we shall provide a solution to this problem using *matrix factorisation* (this is a relatively common problem in statistics). The movie rating table is represented by a matrix M, where the symbol "-" is replaced by 0, as in Equation (2.1), and seek two low -rank matgrices A and B such that $A \cdot B \approx M$in those entries that have data for M. We shall assume that A is a 6×2 matrix, while B is a 2×10 matrix, as follows.

$$M = \begin{pmatrix} 7 & 8 & 9 & 0 & 0 & 1 & 4 & 2 & 3 & 9 \\ 0 & 0 & 10 & 9 & 10 & 2 & 3 & 0 & 0 & 5 \\ 10 & 9 & 0 & 8 & 0 & 0 & 0 & 2 & 1 & 3 \\ 1 & 0 & 2 & 0 & 0 & 9 & 8 & 9 & 0 & 0 \\ 0 & 1 & 1 & 0 & 2 & 0 & 9 & 0 & 7 & 0 \\ 2 & 1 & 0 & 0 & 1 & 10 & 0 & 9 & 0 & 8 \end{pmatrix}, \quad A = \begin{pmatrix} a11 & a_{12} \\ a21 & a_{22} \\ \vdots & \vdots \\ a61 & a_{62} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{110} \\ b_{21} & b_{22} & \dots & b_{210} \end{pmatrix} \tag{2.1}$$

Matrices A an B are obtained by solving an optimisation problem

$$(A^*, B^*) = \text{argmin}_{A\in\mathbb{R}^{6\times 2}, B\in\mathbb{R}^{2\times 10}} E(A,B) = \|I \odot (M - A\cdot B)\|^2,$$

where $\odot$ denotes matrix element-wise multiplication[1] and $A \cdot B$ is matrix multiplication[2], and the matrix norm $\|\cdot\|$ is the *Frobenius norm*,

$$\|L\| = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} l_{ij}^2}, \quad L = \begin{pmatrix} l_{11} & l_{12} & \dots & l_{1m} \\ l_{21} & l_{22} & \dots & l_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nm} \end{pmatrix}$$

that is, by concatening all the entries of L into a long vector and computing its Euclidean norm. The matrix I is a *binary indicator matrix*, of size 6×10 i.e. the same size as M and such that

$$\begin{cases} I_{ij} = 1 & \text{if } M_{ij} \neq 0 \\ I_{ij} = 0 & \text{if } M_{ij} = 0 \end{cases}$$

it records the position of the known vs. unknow entries.

Exercise 2. (Netflix – I)

a) Explain in your own words what this model does, and how (if at all) we can interpret the two matrices A and B. It is completely fine to include ideas that refers to the results in Exercise 4.

b) Show that, in coordinates, the error function $E(A,B)$ can be written as

$$E(A,B) = \|I \odot (M - AB)\|^2 = \sum_{i=1}^{6}\sum_{j=1}^{10} I_{ij}\left(M_{ij} - (a_{i1}b_{1j} + a_{i2}b_{2j})\right)^2$$

Deliverables. a) A short explanation. b) The derivation.

Exercise 3. (Netflix – II)

a) For any $k \in \{1,\dots,6\}$, $l \in \{1,\dots,10\}$, $m \in \{1,2\}$ (specifying the indicies of A and B), prove step by step that the following partial derivatives of E are correct:

$$\frac{\partial E}{\partial a_{km}} = 2\sum_{j=1}^{10} I_{kj}\left(-M_{kj}b_{mj} + a_{k1}b_{1j}b_{mj} + a_{k2}b_{2j}b_{mj}\right)$$

$$\frac{\partial E}{\partial b_{ml}} = 2\sum_{i=1}^{6} I_{il}\left(-M_{il}a_{im} + a_{i1}a_{im}b_{1l} + a_{i2}a_{im}b_{2l}\right)$$

Deliverables. The derivations.

Exercise 4. (Netflix – III)

a) Using these partial derivatives, fill in the Jupyter notebook template `A3template.ipynb`, Exercise 4, to implement a gradient descent algorithm[3] that minimizes E with respect to A and B. Provide a concise description of your implementation. Apply your implementation to the matrix M , which you find in the supplied file `netflix_matrix.txt`. Please include both your code and your final matrices A and B, as well as the matrix M' obtained by rounding all entries of AB to their nearest integer. How does M' compare with the original matrix M ? Can you interpret the result?

b) What strengths and weaknesses do you see with this approach to the original Netflix problem? Do you have ideas for how it could be improved?

Deliverables. a) Your code in the form of a filled-out Jupyter notebook template, a short implementation description, the matrices, and a few lines discussion in and interpretation. b) A few lines of discussion.

[1] with `numpy`, one can use `np.multiply`, equivalent to `A * B` for arrays `A` and `B` with the same shape.

[2] With `numpy`, it corresponds to `A @ B` for 2D arrays `A` and `B` of compatible shapes.

[3] You could concatenate the elements of A and B in a very long vector $(a_{11},\dots,a_{62},b_{11},\dots,b_{210})$ and compute the gradient with respect to this variable. But, as the gradients are defined element-wise, this is actually equivalent to *simultaneously* minimizing w.r.t the separate matrices A and B, using the same learning rate for both and updating both in the same iteration. Note that the norm of the gradient is just the norm of the vector of all partial derivatives from c) concatenated. This can be implemented as `np.linalg.norm(A)**2 + np.linalg.norm(B)**2` or `(A**2).sum() + (B**2).sum()`.