Assignment 1

Numerical Optimization / Optimization for CS WS2023

Lea Bogensperger, lea.bogensperger@icg.tugraz.at
Alexander Falk, alexander.falk@icg.tugraz.at
Clemens Krenn, clemens.krenn@student.tugraz.at
Fabian Ofner, f.ofner@student.tugraz.at

October 24, 2023

Deadline: Nov 14th, 2023 at 23:59

Submission: Upload your report and your implementation to the TeachCenter. Please use the provided framework-file for your implementation. Make sure that the total size of your submission does not exceed 50MB. Include **all** of the following files in your submission:

- `report.pdf`: This file includes your notes for the individual tasks. Keep in mind that we must be able to follow your calculation process. Thus, it is not sufficient to only present the final results. You are allowed to submit hand written notes, however a compiled LaTeX document is preferred. In the first case, please ensure that your notes are well readable.
- `main.py`: This file includes your python code to solve the different tasks. Please only change the marked code sections. Also please follow the instructions defined in `main.py`.
- `figures.pdf`: This file is generated by running `main.py`. It includes a plot of all mandatory figures on separate pdf pages. Hence, you do not have to embed the plots in your report.

1 Characterization of Functions (8 P.)

For $\mathbf{x} = (x_1, x_2)^T \in \mathbb{R}^2$ consider the different functions $f : \mathbb{R}^2 \to \mathbb{R}$

a) $f(\mathbf{x}) = 2x_1^3 - 6x_2^2 + 3x_1^2 x_2$

b) $f(\mathbf{x}) = x_1^2 + x_1 \|\mathbf{x}\|_2^2 + \|\mathbf{x}\|_2^2$

c) $f(\mathbf{x}) = \ln\left(1 + \frac{1}{2}(x_1^2 + 3x_2^3)\right)$

d) $f(\mathbf{x}) = (x_1 - 2)^2 + x_1 x_2^2 - 2$

For each of the given functions do:

In Python:

1. Plot the contour sets of the above functions using a contour plot[1]. Ensure a reasonable numerical range to plot each of your functions.
2. Add markers to the stationary points (computed with Pen & Paper).

With Pen & Paper:

3. Does the function have a global minimizer/maximizer?
4. Compute the gradient and the Hessian.
5. Determine the set of stationary points.

[1] https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.contour.html

6. Characterize every stationary point whether it is a saddle point, (strict) local/global minimum or maximum.

2 Numerical Gradient Approximation (3 P.)

Validate your derived gradients of Section 1 by computing a numerical approximation using central differences

$$\nabla_{x_1} f(x_1, x_2) \approx \frac{f(x_1 + \epsilon, x_2) - f(x_1 - \epsilon, x_2)}{2\epsilon}$$
$$\nabla_{x_2} f(x_1, x_2) \approx \frac{f(x_1, x_2 + \epsilon) - f(x_1, x_2 - \epsilon)}{2\epsilon}.$$

In Python:

1. Write a short `python` script to check the gradients for all the functions in Section 1 numerically. Set ϵ to a suitable choice (state and explain this in your report) and choose 3 random points $(x_1, x_2)^T$.
2. For each point compare the result of your analytically computed gradient with the numerically approximation. Ensure that the numeric gradient approximates the analytic gradient sufficiently well using `numpy.allclose`. Report the results for each random point (state the random point that was generated) for all four functions and include a screenshot of your code output in the report.

3 Matrix Calculus (8 P.)

Given $\mathbf{x} \in \mathbb{R}^n$, consider the following functions:

a) $f(\mathbf{x}) = \frac{1}{2}\|AB\mathbf{x} - \mathbf{b}\|_2^2$ for $\mathbf{b} \in \mathbb{R}^m, A \in \mathbb{R}^{m \times o}, B \in \mathbb{R}^{o \times n}$

b) $f(\mathbf{x}) = \mathbf{1}^T\mathbf{x} - \frac{1}{2}(\mathbf{x} \odot \mathbf{t})^T K(\mathbf{x} \odot \mathbf{t})$ for $\mathbf{t} \in \mathbb{R}^n, K \in \mathbb{R}^{n \times n}, K = K^T, \mathbf{1} \in \mathbb{R}^n$

c) $f(\alpha) = \frac{1}{2}\|A(\mathbf{x} + \alpha\mathbf{y}) - \mathbf{b}\|_2^2$ for $\alpha \in \mathbb{R}, \mathbf{y} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$

The operator $\odot$ denotes the Hadamard multiplication and is defined element-wise for two vectors or matrices, e.g.

$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \odot \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1 b_1 \\ a_2 b_2 \end{pmatrix}.$$

For each of the given functions do the following:

With Pen&Paper:

1. Compute the gradient of f. Compute the derivative for each element using the summation notation as shown in the exercise. Convert the result back to multivariate notation. Show all your steps in the report.
2. Compute the Hessian of f. Proceed similarly to the gradient computation step. Show all your steps in the report.

In Python:

1. Check the correctness of your results for ∇f and $\nabla^2 f$ by utilizing the function `scipy.optimize.approx_fprime`[2] for $m = n = o = 2$ and 3 points in $\mathbb{R}^n$. For this, use the values provided in function `task3` in `main.py`:
2. For each point compare the result of your analytically computed gradient with the numerically approximation. Ensure that the numeric gradient approximates the analytic gradient sufficiently well using `numpy.allclose`. Report the results for each random point (state the random point that was generated) for all three functions and include a screenshot of your code output in the report.

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.approx_fprime.html

4 Student Task Selection Problem (6 P.)

Assume there are $I = 2$ students that need to complete an assignment sheet, which consists of $J = 15$ tasks. Each student has a total time budget which can be assigned to the individual tasks. To account for the different abilities of the students, they estimated for each student the required time for each task in hours as well as their time budget and inserted it into the following table:

	Task															
Student	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Time budget
1	0.5	0.25	0.25	0.25	1.0	1.0	0.5	0.5	1.0	0.5	1.5	2.5	1.0	2.5	3.5	9
2	0.75	1.0	0.75	0.5	0.5	0.25	0.25	0.25	2.0	1.25	1.0	4.0	2.5	3.0	2.0	6

Given the time budgets and the estimated task time of each student, they need to figure out the distribution of students to tasks that minimizes the time to complete the whole assignment sheet.

To do so, you have to complete the following steps.

With Pen & Paper:

1. Formulate the objective function of a linear program such that the total time when distributing the J tasks to the I students is minimized.
2. Formulate the corresponding constraints and state them in the report.
3. Convert the linear program into the standard linear program form. Give all dimensionalities of your vectors/matrices.
4. What is the time that each of the I students would need to solve the entire assignments on their own?

In Python:

5. Use `scipy`'s linear program solver[3] to solve the linear program.

With Pen & Paper:

6. Report the student to task assignment in your report and document each step.
7. What is the final time that is needed after optimally distributing the tasks?

[3] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html`