Assignment 1

Numerical Optimization / Optimization for CS WS2022

Lea Bogensperger, lea.bogensperger@icg.tugraz.at
Nives Križanec, nives.krizanec@student.tugraz.at
Jakob Ederer, ederer@student.tugraz.at

October 25, 2022

Deadline: Nov 15th, 2022 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 (6 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) $f(\mathbf{x}) = \ln\left(1 + \frac{1}{2}(x_1^2 + 3x_2^3)\right)$

b) $f(\mathbf{x}) = (x_1 - 2x_2)^4 + 64x_1x_2$

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

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. Compute the gradient and the Hessian.
4. Determine the set of stationary points.
5. Characterize every stationary point whether it is a saddle point, (strict) local/global minimum or maximum.

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

2 Matrix Calculus (6 P.)

Given $\mathbf{x} \in \mathbb{R}^n$

a) $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{c} \odot (\mathbf{x} - \mathbf{k})\|^2$ for $\mathbf{x}, \mathbf{k}, \mathbf{c} \in \mathbb{R}^n$

b) $f(\mathbf{x}) = \sum_{i=1}^{n} h\left((\mathbf{A}\mathbf{x})_i\right)$ with $h(t) = \frac{1}{2}t^2 + 2t$ for $t \in \mathbb{R}$, $\mathbf{A} \in \mathbb{R}^{n \times n}$, subscript i denoting the i-th element

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

The operator $\odot$ denotes the Hadamard-Product which is the element-wise product of two vectors or matrices, e.g.

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \odot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{pmatrix} .$$

For each of the given functions:

With Pen & Paper:

1. Compute the gradient of the respective function. If needed in the multivariate case, compute the derivative for each element using the summation formulation as shown in the exercise lecture and convert your result back to multivariate notation.
2. Compute the Hessian of the respective function. Proceed similarly to the gradient computation step. Show all your steps in the report.

3 Numerical Gradient Approximation (4 P.)

To see whether the computed gradient is correct one can easily verify this by computing a numerical approximation of the gradient using central differences

$$\nabla f(\mathbf{x}) \approx \frac{1}{2\epsilon} \begin{pmatrix} f((x_1 + \epsilon, x_2)^T) - f((x_1 - \epsilon, x_2)^T) \\ f((x_1, x_2 + \epsilon)^T) - f((x_1, x_2 - \epsilon)^T) \end{pmatrix}$$

In Python:

1. Write a function to check the gradients for all the functions in Task 1 numerically as shown above. To do so, set $\epsilon = 1 \times 10^{-6}$ and choose a random point (x_1, x_2). For this point compare the result of your analytically computed gradient with the numerically approximation. Make a table where you report all analytic and approximated gradient values for each function. Further, plot and compare your results in bar plot. Use the provided functions and variables to compute the gradient, function value and gradient approximation.
2. In a general setting for $\mathbf{x} \in \mathbb{R}^n$ choose $n = 5$ and again perform the verification of the gradients but for your results from Task 2. This time use scipy's `approx_fprime()` function[2]. Create a bar plot for each function $f(\mathbf{x})$ comparing your analytical result vs. the numerical approximation. Use the provided functions and variables to compute the gradient, function value and gradient approximation.

4 Vectors, Norms and Matrices (4 P.)

With Pen & Paper:

1. Show that $\|\cdot\|_{\frac{1}{2}}$ for $\mathbf{x} \in \mathbb{R}^n$ is not a norm.
2. Show that for any $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$ the following inequality holds
$$\|\mathbf{x} - \mathbf{z}\| \leq \|\mathbf{x} - \mathbf{y}\| + \|\mathbf{y} - \mathbf{z}\|.$$
3. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ be two orthogonal vectors, prove that
$$\|\mathbf{x} + \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 .$$

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

4. Let $\mathbf{Q} \in \mathbb{R}^{n \times n}$ be a positive definite matrix. Show that the following is a norm:

$$\|\mathbf{x}\|_Q = \sqrt{\mathbf{x}^T \mathbf{Q} \mathbf{x}}.$$

5 Automatic Differentiation (5 P.)

Consider a very simple feedforward neural network with one hidden layer (see Fig. 5) that takes an input vector $\mathbf{x} = (x_1, x_2)^T \in \mathbb{R}^2$ and outputs $y \in \mathbb{R}$ with weights $\mathbf{W}^{(0)} \in \mathbb{R}^{2\times 2}$ and $\mathbf{W}^{(1)} \in \mathbb{R}^{1\times 2}$. The computational chain is given by

$$\mathbf{z} = \mathbf{W}^{(0)}\mathbf{x} \tag{1}$$
$$\mathbf{a} = \sigma(\mathbf{z}) \tag{2}$$
$$y = \mathbf{W}^{(1)}\mathbf{a} \tag{3}$$

The activation function is given by $\sigma(t) = \frac{1}{1+\exp(-t)}$ for $t \in \mathbb{R}$ which is applied elementwise to a vector. Note that $\mathbf{a} = (a_1, a_2)^T \in \mathbb{R}^2$ and $\mathbf{z} = (z_1, z_2)^T \in \mathbb{R}^2$.

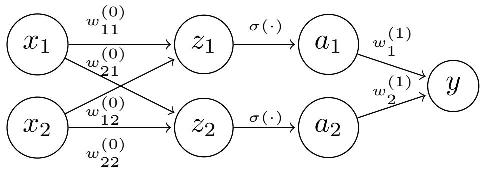

Figure 1: Simple feedforward network with one hidden layer.

We are interested in the gradient $\nabla_{\mathbf{x}} y = \left(\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}\right)^T$. Therefore it is convenient to make use of automatic differentiation which exploits the fact that every function can be decomposed into a sequence of elementary arithmetic operations and functions. We will consider the forward and the reverse mode of automatic differentiation, both relying on the chain rule of decomposing the gradient but they differ in the directions from which they traverse the chain rule.

With Pen & Paper:

1. Compute the derivative of $\sigma(t)$ for $t \in \mathbb{R}$, which you will need in the subsequent tasks.
2. Compute all partial derivatives $\frac{\partial y}{\partial x_i}$ using automatic differentitaion in **forward mode**. Do this manually, showing how *each* partial derivative can be evaluated in parallel to one forward sweep.
3. Compute the gradient $\nabla_{\mathbf{x}} y$ using automatic differentiation in **backward mode**. Note that you first have to a forward sweep before evaluating all partial derivatives.
4. Consider a general function $f : \mathbb{R}^n \to \mathbb{R}^m$ and the two cases
   - $m \ll n$
   - $m \gg n$

   Reason for both cases whether you would use the forward or the backward mode to compute the gradient.

In Python:

6. Set $\mathbf{x} = (1.0, 0.5)^T$ and weight matrices $\mathbf{W}^{(0)} = \begin{pmatrix} 1.0, 0.2 \\ 0.5, -1. \end{pmatrix}$, $\mathbf{W}^{(1)} = \begin{pmatrix} 1.0, 0.5 \end{pmatrix}$. Evaluate the gradient using automatic differentiation in both forward and backward mode.
7. Check the gradient for the given $\mathbf{x}$ numerically using central difference approximation from Task 3 with $\epsilon = 1 \times 10^{-6}$ and report your results.