**Optimization Methods for Data Science (2023W)** 30 Oct 2023

# Assignment 1

Due to: 30 Nov 2023

You will have two assignments, this is the first one.

## Rules

1. **Theory.** If you write your solutions by hand, make sure they are readable. If I can't understand it, it won't be counted. You can send me a scan of your handwriting, but again, make sure it is of high quality, that the paper is not bent, etc.
2. **Practice.** This should be done in the form of a report: besides the actual code, you should explain in words what you're doing, what you're observing, etc. You can send me these reports in any form: as a Jupyter notebook, a link to Google Colab, a github repo, or you can even make a pdf report with all pictures/text and attached code.
3. In the future, if you disagree with my evaluation (or if I missed something), you have the right to appeal. But I will also reserve the right to ask you questions about how well you understand your solution.
4. In case of any questions, contact me by Moodle/email.

## Theory

1. **(1 point)** Show that the sum of two convex functions is a convex function.
2. **(2 points)** Compute the gradient and Hessian of $f(\mathbf{x}) = \log \sum_{i=1}^{n} e^{x_i}$.
3. **(1 point)** Let $f(x) = \frac{1}{1+e^{-x}}$. Prove the identity $f'(x) = f(x)(1-f(x))$.
4. **(2 point)** Which functions are differentiable (a) $f(\mathbf{x}) = \|\mathbf{x}\|$, (b) $f(\mathbf{x}) = \|\mathbf{x}\|_1$, (c) $f(\mathbf{x}) = \frac{\|\mathbf{x}\|^2}{x_1}$? Justify your answer (2 points only if all answers are correct).
5. **(2 points)** Let $f : \mathbb{R}^n \to \mathbb{R}$ be defined by $f(\mathbf{x}) = \sum_{i=1}^{m} \log(1 + e^{\mathbf{a}_i^\top \mathbf{x}})$, where $\mathbf{a}_i$ are given vectors. Compute $\nabla f(\mathbf{x})$.
6. **(4 points)** Compute the gradient $\nabla f$ (with derivations) of
   (a) $f : \mathbb{R}^n \to \mathbb{R}$: $f(\mathbf{x}) = \log(1 + \|\mathbf{x}\|^2)$;
   (b) $f : \mathbb{R}^n \to \mathbb{R}$: $f(\mathbf{x}) = \frac{1}{4}\|A\mathbf{x} - \mathbf{b}\|^4$;
   (c) $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$: $f(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\|\mathbf{x}\mathbf{y}^\top - A\|^2$.
7. **(2 points)** Prove that the following functions are convex
   (a) $f : \mathbb{R} \times \mathbb{R}_{++} \to \mathbb{R}$: $f(x, y) = \frac{x^2}{y}$;
   (b) $f : \mathbb{R}_{++} \times \mathbb{R}_{++} \to \mathbb{R}$: $f(x, y) = x \log x - x \log y$.
8. **(2 points)** Prove that the following functions are not convex
   (a) $f : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}$: $f(x, y) = xy$;
   (b) $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$: $f(x, y) = ye^x - xe^y$.
9. **(3 points)** Solve the following problem

$$\min_{x_1 > 0, x_2 > 0} \frac{1}{x_1 x_2} + x_1 + x_2.$$

10. **(3 points)** Consider the quadratic function $f(x) = \frac{1}{2}\langle A\mathbf{x}, \mathbf{x}\rangle + \langle \mathbf{b}, \mathbf{x}\rangle + c$, where $A$ is a symmetric $n \times n$ matrix. Prove that if the function $f$ is bounded from below on $\mathbb{R}^n$, then $A$ must be positive semidefinite.

11. **(3 points)** Show that the following functions are $L$-smooth: (a) $f(\mathbf{x}) = \frac{1}{2}\|A\mathbf{x} - b\|^2$; (b) $f(\mathbf{x}) = \sqrt{1 + \|\mathbf{x}\|^2}$; (c) $f(x) = x^3$ for $x \in [-3, 3]$.

12. **(2 points)** Let $f(x, y) = \sin(\pi x y^2)$. What is the linear approximation of $f$ at the point $(1, 1)$? Compare the values of $f$ and its linear approximation at this point.

13. **(2 points)** Consider $f$ from the previous example. What is the quadratic approximation of $f$ at this point? Compare the values of $f$ and its quadratic approximation at this point.

14. **(2 points)** Let $f(x, y) = \sin(x^2 + y) + y$. Find all critical points (stationary points) of $f$ (those points where the gradient vanishes: $\nabla f(x, y) = 0$).

15. **(2 points)** Show that for twice differentiable functions strict convexity is not equivalent to $\nabla^2 f(\mathbf{x}) \succ 0$.

16. **(2 points)** Prove that $1 - t \leq e^{-t}$, for all $t$, using only convexity arguments.

17. **(2 points)** Prove that if $f$ is a convex function, then $F(\mathbf{x}) = f(\mathbf{x}) + \frac{\mu}{2}\|\mathbf{x}\|^2$ is $\mu$-strongly convex.

## PRACTICE

18. **(4-6 points)** Let $f : \mathbb{R}^n \to \mathbb{R}$ be given by $f(\mathbf{x}) = 100 \sum_{i=2}^{n} (x_i - x_{i-1}^2)^2 + (1 - x_1)^2$. Prove that $\mathbf{x}^* = (1, \ldots, 1)$ is its global minimum. Choose any $n > 1$ you want and run gradient descent (any variant you want) with different starting point $\mathbf{x}^0$. Explain briefly, possibly with supporting evidence from the plots, what you're observing. For example, does the method converge to the global minimum? Does convergence depend on the starting point? You may add other questions if you find them interesting.

19. **(2 points)** Plot level sets of

$$f(x_1, x_2) = \sqrt{x_1^2 + x_2^2} + \sin\left(4 \arctan \frac{x_2}{x_1}\right)$$

on $[-3, 3] \times [-3, 3]$. Is this function differentiable/convex? Bonus (+1) for finding the simplest explanation (one sentence).

20. **(6 points)** Consider $f : \mathbb{R}^n \to \mathbb{R}$ given by

$$f(\mathbf{x}) = \frac{1}{m} \log\left(\sum_{i=1}^{m} e^{\mathbf{a}_i^\top \mathbf{x}}\right) + \frac{\mu}{2}\|\mathbf{x}\|^2.$$

We choose the following datum

```
import numpy as np
m = 100
n = 50
mu = 1.0
A = np.random.randn(m, n)
x0 = np.zeros(n)
```

(a) Prove that this function is strongly convex.

(b) Compare GD, accelerated GD and accelerated GD for the strongly convex functions on this problem. For a stepsize $\alpha$ you can choose whatever you prefer
   - compute $L$ analytically and set $\alpha = \frac{1}{L}$;
   - trial and error. In this case ensure that taking a much larger step won't work.
   - linesearch

(c) Plot function values wrt the number of iterations.

(d) Do the same, but this time set $\mu = 0.01$. What changes do you observe?

21. **(8 points)** Given dataset $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_m, b_m)$, with $\mathbf{a}_i \in \mathbb{R}^n$, $b_i \in \{-1, 1\}$, we want to solve

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(x) := \frac{1}{m}\sum_{i=1}^{m} \log(1 + e^{-b_i \mathbf{a}_i^\top \mathbf{x}}).$$

Here we only care about minimization problem, not about classification accuracy.

1) **Data.** You can choose either synthetic not interesting data or take some existing ML dataset:
   (a) Synthetic non-interesting data.
   $m = 100$, $n = 90$, $A \in \mathbb{R}^{m\times n}$ with Gaussian iid entries and set $\mathbf{a}_i$ to be the $i$-th row of $A$. Set $b$ as signs of a vector in $\mathbb{R}^m$ also with Gaussian entries. In Python it is done by

```
import numpy as np
m = 100
n = 90
A = np.random.randn(m, n)
b = np.sign(np.random.randn(m))
```

   (b) Real datasets. You can use any ML dataset you find, for example check UCI datasets. Alternatively, some of them you can find directly in Python, perhaps with some extra library.

2) **GD.** Run Gradient Descent on this problem.
   - For the problem above and the dataset you have chosen, implement gradient descent method with a fixed stepsize $\alpha$. For this you need to choose right $\alpha$. Either compute the Lipschitz constant $L$ of $\nabla f$ or tune $\alpha$ by trial and error.
   - Compare both methods in terms of $\|\nabla f(\mathbf{x}_k)\|$ or $f(\mathbf{x}_k) - f_*$ vs. the number of iterations $k$. Make plot, write down some observations/conclusions.

3) **SGD.** Run SGD on this problem.
   For the same problem implement SGD with different regimes for stepsizes:
   (i) $\alpha_k = \alpha$ for all $k$;
   (ii) $\alpha_k = \frac{c_1}{c_2+k}$;
   (iii) $\alpha_k = \frac{c_1}{c_2+\sqrt{k}}$;
   (iv) $\alpha_k = \frac{c_1}{c_2+\sqrt{\lceil \frac{k}{m} \rceil}}$, where $\lceil a \rceil$ denotes the integer part of $a$.
   (v) $\alpha_k = \frac{c_1}{\sqrt{10^{-6}+\sum_{i=0}^{k}\|g_i\|^2}}$, where $g_i$ are stochastic gradients from previous iterations $0, \ldots, k$
   Instead of some stepsizes as above, you can choose your own. Your first goal is to find good constants $c_1, c_2, \alpha$ for every method. They don't need to be the best, just good enough. Then compare all 5+ methods. One iteration of SGD is quite cheap, so be ready to run the method for many iterations. To compare algorithms, compute $f(\mathbf{x}_k) - f_*$ or $\|\nabla f(\mathbf{x}_k)\|$ every $m$ iterations (every epoch), where for $f_*$ you can use the value you obtained from the running GD in the previous task.

4) Compare GD and the best variant of SGD. Make the comparison fair, since one iteration of SGD is much cheaper than the one of GD.