Optimization Methods for Data Science (2023W) 17 December 2023

Assignment 2

Due to: 18 January 2024

RULES

1. **Theory.** If you write your solutions by hand, make sure they are readable. This time I will be stricter about poor quality photos of your solutions, and we may not have enough time to appeal. You can use LaTeX (or any alternative) and get a bonus +5 points. I provide the LaTeX template `template.tex`, alternatively you can write your solutions in Overleaf (just don't send me the link) or Jupyter notebooks.
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. Whenever I say implement algorithm, it means that you cannot use an optimization library. But you can always reuse code from the class.
4. 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.
5. In case of any questions, contact me by Moodle/email.

THEORY

1. **(6 points)** Consider the problem

$$\min_x f(x) := \frac{1}{2m}\sum_{i=1}^m (x-u_i)^2,$$

where $x, u_i \in \mathbb{R}$ and $u_1, \dots, u_m$ are given.

   (a) What is the solution of this problem?

   (b) What is the Lipschitz constant of ∇f?

   (c) Is this problem nonconvex/convex/strongly convex?

   (d) Apply GD to this problem (on paper). What is the reasonable stepsize? Compute analytically a few iterates.

   (e) We want to apply incremental gradient descent to this problem. Let $f_i(x) = \frac{1}{2}(x-u_i)^2$, that is $f(x) = \frac{1}{m}\sum_{i=1}^m f_i(x)$. Incremental means that in the first iteration you only use ∇f_1, in the second only ∇f_2 and so on until the cycle is over and then you start the cycle again. It is basically the same as SGD, but without randomness. Run this method starting from $x_0 = 0$ and with steps $\alpha_k = \frac{1}{k+1}$. What can you say about its iterates?

2. **(1 point)** Assume that $\nabla^2 f(x) \succ 0$ for all x. Prove that minimimization of $f_k(x)$

$$f_k(x) := f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2}\langle \nabla^2 f(x_k)(x-x_k), x - x_k\rangle$$

leads to the Newton method applied to $\min_x f(x)$.

3. **(3 points)** Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable and μ-strongly convex function. Prove that for its minimzer x^* it holds that

$$\|\nabla f(x)\| \geqslant \mu \|x - x^*\| \qquad \forall x.$$

4. **(2 points)** Apply (on paper) Newton's method for minimizing $f(x) = \sqrt{1+x^2}$. How does the method behave?

5. **(2 points)** Solve a system of equations by Newton's method (you can do it numerically, but no need to show code).

$$\begin{cases} 3x^2y + y^2 &= 1 \\ x^4 + xy^3 &= 1. \end{cases}$$

6. **(5 points)** Let p be a polynomial with real coefficients whose roots are all real $u_n \geqslant \ldots \geqslant u_1$. Prove that if we initialize $x_0 \geqslant u_n$, then the Newton method applied to $p(x) = 0$ converges to u_n.

7. **(3 points)** Let $g : \mathbb{R} \to \mathbb{R}$ be an increasing function and $f : \mathbb{R}^n \to \mathbb{R}$ be convex. Suppose that both functions are twice differentiable. Consider two optimization problems: $\min_x f(x)$ and $\min_x g(f(x))$. Show that these two problems are equivalent. How does the Newton method look like when applied to both problems?

8. **(3 points)** Solve the problem

$$\min_x \|x\|^2 \quad \text{subject to} \quad x_1 + \cdots + x_n = 1,$$

using Lagrange multipliers.

9. **(3 points)** Let $A \in \mathbb{R}^{m\times n}$ have full rank with $m \leqslant n$ and $u \in \mathbb{R}^n$. Solve the problem

$$\min_x \|x - u\|^2 \quad \text{subject to } Ax = 0,$$

using Lagrange multipliers.

PRACTICE

10. **(5 points)** You are given N points in $\mathbb{R}^2$: $(x_1, y_1), \ldots, (x_N, y_N)$. You know that this data should follow approximately a circular shape, but you don't know the center $c = (x, y)$ of the circle and its radius r. Naturally, you want to find them from the minimization problem $\min_{x,y,r} f(x, y, r)$, where f is defined as

$$f(x, y, r) = \sum_{i=1}^{N} \left((x - x_i)^2 + (y - y_i)^2 - r^2\right)^2.$$

Solve this problem by any optimization method you would like to implement by yourself. Make the correspondent plot of how well the final solution represents the shape of data. For implementation, consider three cases $N = 20, 100, 1000$. Use the template from `homework-2_code.ipynb` to generate all the data.

☞ Bonus (+2) for interesting experiments/observations.

11. **(5 points)** We want to reconstruct a function f from its samples $f(s_1), \ldots, f(s_M)$. We will approximate our function f by a trigonometric polynomial p:

$$p(t) = a_0 + \sum_{k=1}^{N} (a_k \cos(kt) + b_k \sin(kt)).$$

Specifically, we find a_k and b_k by making $\sum_{j=1}^{M}(p(s_j)-f(s_j))^2$ as small as possible. In other words, we want to minimize φ given by

$$\varphi(a,b) = \sum_{j=1}^{M}(p(s_j)-f(s_j))^2 = \sum_{j=1}^{M}\left(a_0 + \sum_{k=1}^{N}\left(a_k\cos(ks_j)+b_k\sin(ks_j)\right)-f(s_j)\right)^2.$$

By introducing

$$A = \begin{pmatrix} 1 & \cos s_1 & \dots & \cos(Ns_1) & \sin s_1 & \dots & \sin(Ns_1) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cos s_M & \dots & \cos(Ns_M) & \sin s_M & \dots & \sin(Ns_M) \end{pmatrix}, \quad x = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \\ b_1 \\ \vdots \\ b_N \end{pmatrix}, \quad y = \begin{pmatrix} f(s_1) \\ \vdots \\ f(s_M) \end{pmatrix},$$

we can equivalently rewrite the problem $\min_{a,b}\varphi(a,b)$ as $\min_x \varphi(x)$, or as

$$\min_x \|Ax-y\|^2.$$

(a) Solve this problem by any optimization method (with your implementation). Plot the residuals.

(b) Plot the original function and its reconstructed version.

(c) Bonus (+2) for interesting experiments/observations. If you notice something interesting, write down your observation. If this is the right observation and you find a plausible explanation for it, you'll get extra 2 points.

The details below are also listed in the `homework-2_code.ipynb` and the correspondent html file.

- $M = 2N+1$, $s_k = \frac{2\pi k}{M}$ for $k = 0,\dots M-1$;
- $f(t) = \sin(\sqrt{2}t)+\sin(t^2)$. We pretend that we don't know it, but only know $f(s_1),\dots,f(s_M)$.
- Make two sets of experiments: with $N = 10$ and with $N = 100$.

12. **(6 points)** Given $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, we want to solve $Ax = b$. Let a_i be the i-th row of A. Generate any reasonable data, $m \geqslant 100$, $n \geqslant 100$, but make sure that the linear system has a solution.

Kaczmarz's method:

$$x_{k+1} = P_{\text{row } i}x_k = x_k - \frac{\langle a_i, x_k\rangle - b_i}{\|a_i\|^2}a_i,$$

where $i = k \mod (m+1)$.

Randomized Kaczmarz's method:

$$\text{Sample } i \in \{1,\dots,m\} \text{ with probabilities } \frac{\|a_i\|^2}{\|A\|_F^2}$$
$$x_{k+1} = P_{\text{row } i}x_k = x_k - \frac{\langle a_i, x_k\rangle - b_i}{\|a_i\|^2}a_i,$$

(a) Implement Kaczmarz method and randomized Kaczmarz method

(b) Find two instances of A and b (of any dimensions, random or deterministic), on which each method will show better performance. We are interested in the objective value $\|Ax_k - b\|^2$ with respect to the number of iterations (or epochs).

(c) Implement gradient descent and compare it with randomized Kaczmarz method on a data of your choice. Be careful with the comparison: one iteration of Kaczmarz method is roughly m times cheaper than the one of GD.

13. **(5 points)** Use the same data as in Exercise 10 with $N = 100$. Implement Gauss-Newton method from Tutorial-4 and apply it to solve

$$\min_{x,y,r} \sum_{i=1}^{N} f_i(x,y,r)^2 := \left((x-x_i)^2 + (y-y_i)^2 - r^2\right)^2.$$

The Gauss-Newton method is given by

$$x_{k+1} = x_k - \alpha_k d_k,$$

where d_k is the Gauss-Newton direction (see Tutorial-4) and $\alpha_k > 0$ is a stepsize. First try $\alpha_k = 1$, if it doesn't work, make it smaller.

Make two plots: energy $f(x_k, y_k, r_k)$ vs. the number of iterations and $\|\nabla f(x_k, y_k, r_k)\|$ vs. the number of iterations.

☞ As a bonus (+1), compare it with SciPy version `scipy.optimize.least_squares` for the same problem.