The Chinese University of Hong Kong, Shenzhen
SDS · School of Data Science

Andre Milzarek · Fall Semester 2021/22

MDS 6106 – Introduction to Optimization

Exercise Sheet 4

Assignment A4.1 (Globalized Newton's Method): (approx. *50* points)
Implement the globalized Newton method (Lecture L-08, slide 07) with backtracking that was presented in the lecture as a function `newton_glob` in `MATLAB` or `Python`.

The following input functions and parameters should be considered:

- `obj`, `grad`, `hess` – function handles that calculate and return the objective function $f(x)$, the gradient $\nabla f(x)$, and the Hessian $\nabla^2 f(x)$ at an input vector $x \in \mathbb{R}^n$. You can treat these handles as functions or fields of a class or structure `f` or use them directly as input. (For example, your function can have the form `newton_glob(obj,grad,hess,...)`).
- x^0 – the initial point.
- `tol` – a tolerance parameter. The method should stop whenever the current iterate x^k satisfies the criterion $\|\nabla f(x^k)\| \leq$ `tol`.
- $\beta_1, \beta_2 > 0$ – parameters for the Newton condition.
- $s > 0$, $\sigma, \gamma \in (0,1)$ – parameters for backtracking and the Armijo condition.

You can again organize the latter parameters in an appropriate `options` class or structure. You can use the backslash operator `A\b` in `MATLAB` or `numpy.linalg.solve(A,b)` to solve the linear system of equations $Ax = b$. If the computed Newton step $d^k = -\nabla^2 f(x^k)^{-1}\nabla f(x^k)$ is a descent direction and satisfies

$$-\nabla f(x^k)^\top d^k \geq \beta_1 \min\{1, \|d^k\|^{\beta_2}\}\|d^k\|^2,$$

we accept it as next direction. Otherwise, the gradient direction $d^k = -\nabla f(x^k)$ is chosen. The method should return the final iterate x^k that satisfies the stopping criterion.

a) Test your implementation on the following problem:

$$\min_{x\in\mathbb{R}^2} \; f(x) = f_1(x)^2 + f_2(x)^2,$$

where $f_1 : \mathbb{R}^2 \to \mathbb{R}$ and $f_2 : \mathbb{R}^2 \to \mathbb{R}$ are given by:

$$f_1(x) := -1 + x_1 + ((5 - x_2)x_2 - 2)x_2,$$
$$f_2(x) := -1 + x_1 + ((x_2 + 1)x_2 - 10)x_2.$$

(This problem was discussed in Assignment A3.3).

- Generate a plot of the solution paths of Newton's method for a variety of initial points similar to Assignment A3.3b. Let us again define the set $\mathcal{X}^0 := \{x \in \mathbb{R}^2 : x_1 \in \{-10, 0\}, x_2 \in [-5, 5]\} \cup \{x \in \mathbb{R}^2 : x_1 \in [-10, 0], x_2 \in \{-5, 5\}\}$. Run the globalized Newton method with parameters $(s, \sigma, \gamma) = (1, 0.5, 0.1)$ and $(\beta_1, \beta_2) = (10^{-6}, 0.1)$ to solve the problem $\min_x f(x)$ with p different initial points selected from the set $\mathcal{X}^0$.

The initial points should uniformly cover the different parts of the set $\mathcal{X}^0$ and you can use the tolerance $\texttt{tol} = 10^{-8}$ and $p \in [10, 25] \cap \mathbb{N}$. Create a single figure that contains all of the solution paths generated for the different initial points. The initial points and limit points should be clearly visible. Add a contour plot of the function f in the background of your figure.

Report the behavior and performance of the Newton method and compare it to the convergence of the gradient method tested in A3.3. In particular, discuss the number of required iterations (on average). Which type of convergence can typically be observed?

b) Test your approach on the Rosenbrock function $f : \mathbb{R}^2 \to \mathbb{R}$

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

with initial point $x^0 = (-1.2, 1)^\top$ and parameter $(s, \sigma, \gamma) = (1, 0.5, 10^{-4})$ and $(\beta_1, \beta_2) = (10^{-6}, 0.1)$. ($\gamma$ is smaller here). Besides the globalized Newton method also run the gradient method with backtracking ($(s, \sigma, \gamma) = (1, 0.5, 10^{-4})$) on this problem and compare the performance of the two approaches for different tolerances $\texttt{tol} \in \{10^{-1}, 10^{-3}, 10^{-5}\}$.

Does the Newton method always utilize the Newton direction? Which type of convergence can be observed? Does the method always use full step sizes $\alpha_k = 1$? In contrast, what is the typical behavior of the gradient method?

Assignment A4.2 (Accelerated Gradient Method): (approx. *50* points)

In this exercise, we consider the ℓ_1-optimization problem

$$\min_x \; f(x) = \frac{1}{2}\|Ax - b\|^2 + \mu\|x\|_1, \tag{1}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $\mu > 0$ are given. In order to handle the nonsmoothness of the ℓ_1-norm, we substitute $\varphi(x) = \|x\|_1$ in (1) with one of the following smooth options:

$$\varphi_1(x) = \|x\|_2^2 \quad \text{and} \quad \varphi_2(x) = \sum_{i=1}^{n} \varphi_{\texttt{hub}}(x_i),$$

where

$$\varphi_{\texttt{hub}}(t) := \begin{cases} \frac{1}{2\delta}t^2 & \text{if } |t| \le \delta, \\ |t| - \frac{1}{2}\delta & \text{if } |t| > \delta. \end{cases}$$

The data A and b is generated as follows: choose $n, m, s \in \mathbb{N}$ with $s \le m \le n$ and create a mask $\texttt{mask} = \texttt{randperm(n,s)}$. (The vector `mask` contains s different integers from 1 to n). We then construct a sparse signal $x^* \in \mathbb{R}^n$ via

$$x^* = \texttt{zeros(n,1)} \quad \text{and} \quad x^*(\texttt{mask}) = \texttt{randn(s,1)}.$$

Hence, x^* is an n-dimensional vector with s-nonzero (randomly selected) entries that follow a normal distribution. We choose $A = \texttt{randn(m,n)}$ and generate the partial measurements b via $b = Ax^* + 0.01 \cdot \texttt{randn(m,1)}$. The goal is to reconstruct the original sparse signal x^* from the much smaller measurements b via solving the problem (1) and its smooth variants.

a) Implement the accelerated gradient method (Lecture L-08, slide 42) for problem (1) using φ_1 and φ_2. The extrapolation parameter and step size should be chosen as follows:

$$\alpha_k = \frac{1}{L}, \quad \beta_k = \frac{t_{k-1} - 1}{t_k}, \quad t_k = \frac{1}{2}(1 + \sqrt{1 + 4t_{k-1}^2}), \quad t_{-1} = t_0 = 1.$$

Here, L denotes the Lipschitz constants of the gradient mappings $\nabla f_1(x) = A^\top(Ax - b) + \mu\nabla\varphi_1(x)$ and $\nabla f_2(x) = A^\top(Ax - b) + \mu\nabla\varphi_2(x)$, respectively. It can be shown that the two constants are given by

$$L_1 = 2\mu + \|A^\top A\| \ \text{(for } \nabla f_1) \quad \text{and} \quad L_2 = \mu\delta^{-1} + \|A^\top A\| \ \text{(for } \nabla f_2).$$

b) Test your implementations for $m = 300$, $n = 3000$, and $s = 30$. Report and compare the performance of the accelerated gradient method using the different sparse models. For f_1 and f_2 you can choose $\mu \in [0.1, 10]$ and $\delta \in [10^{-5}, 10^{-3}]$. You can select $x^0 = 0$ as initial point and $\texttt{tol} = 10^{-4}$. (Performance can be measured by comparing $\|\nabla f(x^k)\|$ or $|f(x^k) - f^*|/\max\{1, |f^*|\}$ where f^* is an approximation of the optimal objective function value). Compare the performance of the accelerated gradient method with the standard gradient method (using, e.g., backtracking).

Plot the reconstructed solutions and compare them with the true solution x^*. Check whether your solutions are truly sparse – which of the models performed best in these tasks?

c) Discuss whether the globalized Newton method can be applied to minimize the functions $f_1(x) = \frac{1}{2}\|Ax - b\|^2 + \mu\varphi_1(x)$ and $f_2(x) = \frac{1}{2}\|Ax - b\|^2 + \mu\varphi_2(x)$. How would you realize an efficient implementation of the Newton method?

Hint: You don't need to implement Newton's method here.

Sheet 4 is due on **Dec, 1st**. Submit your solutions before **Dec, 1st**, **11:00 pm**.