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

Andre Milzarek · Fall Semester 2024/25

MDS 6106 — Optimization and Modeling

Exercise Sheet 4

Assignment A4.1 (Implementing the Gradient Method): (approx. *70* points)
Implement the gradient descent method (Lecture L-11) that was presented in the lecture as a function `gradient_method` in `MATLAB` or `Python`.

The following input functions and parameters should be considered:

- `obj`, `grad` – function handles that calculate and return the objective function $f(\boldsymbol{x})$ and the gradient $\nabla f(\boldsymbol{x})$ at an input vector $\boldsymbol{x} \in \mathbb{R}^n$. 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 `gradient_method(obj,grad,...)`).
- $\boldsymbol{x}^0$ – the initial point.
- `tol` – a tolerance parameter. The method should stop whenever the current iterate $\boldsymbol{x}^k$ satisfies the criterion $\|\nabla f(\boldsymbol{x}^k)\| \leq$ `tol`.

We want to analyze the performance of the gradient method for different step size strategies. In particular, we want to test and compare backtracking, exact line search, and diminishing step sizes. The following parameters will be relevant for these strategies:

- $\sigma, \gamma \in (0, 1)$ – parameters for backtracking and the Armijo condition.
- `alpha` – a function that returns a pre-defined, diminishing step size $\alpha_k =$ `alpha(k)` satisfying $\alpha_k \to 0$ and $\sum \alpha_k = \infty$.
- You can use the golden section method from Assignment A3.1 to determine the exact step size α_k. The parameters for the golden section method are: `maxit` (maximum number of iterations), `tol` (stopping tolerance), $[0, \mathtt{a}]$ (the interval of the step size).

You can organize the latter parameters in an appropriate `options` class or structure. It is also possible to implement separate algorithms for the different step size mechanisms. The method(s) should return the final iterate $\boldsymbol{x}^k$ that satisfies the stopping criterion.

Test your implementation on the following problem:

$$\min_{\boldsymbol{x} \in \mathbb{R}^2} \; f(\boldsymbol{x}) := \frac{1}{2}x_1^4 - x_1^3 - x_1^2 + x_1^2 x_2^2 + \frac{1}{2}x_2^4 - x_2^2. \tag{1}$$

This problem has the stationary points $\boldsymbol{x}_1^* = [0,\, 0]^\top$, $\boldsymbol{x}_2^* = [2,\, 0]^\top$, $\boldsymbol{x}_3^* = [-\frac{1}{2},\, 0]^\top$, $\boldsymbol{x}_4^* = [0,\, 1]^\top$, and $\boldsymbol{x}_5^* = [0,\, -1]^\top$. It can be shown that $\boldsymbol{x}_1^*$ is a local maximum, $\boldsymbol{x}_2^*$ is a global minimum, and the points $\boldsymbol{x}_3^*$, $\boldsymbol{x}_4^*$, and $\boldsymbol{x}_5^*$ are saddle points of the problem $\min_{\boldsymbol{x}} f(\boldsymbol{x})$.

Let us define the set of initial points

$$\mathcal{X}^0 := \left\{ \begin{bmatrix} -\frac{1}{2} \\ 1 \end{bmatrix}, \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix}, \begin{bmatrix} -\frac{1}{4} \\ -\frac{1}{2} \end{bmatrix}, \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \end{bmatrix}, \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix} \right\}$$

Run the methods:

– Gradient descent method with backtracking and $(\sigma, \gamma) = (0.5, 0.1)$,

– Gradient method with diminishing step sizes $\alpha_k = \frac{1}{\sqrt{k+2}}$,

– Gradient method with exact line search and `maxit` $= 100$, `tol` $= 10^{-6}$, `a` $= 1$,

for every initial point in $\mathcal{X}^0$ using the stopping tolerance `tol` $= 10^{-5}$.

– Report the behavior and performance of the different methods. In particular, for each of the step size strategies, compare the number of required iterations and the points to which each algorithm converged.

– For each algorithm/step size strategy 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 each figure.

Assignment A4.2 (Inertial Gradient Method): (approx. *30* points)
Implement the inertial gradient method/gradient method with momentum (Lecture L-17) that was presented in the lecture as a function `inertial_gradient_method` in `MATLAB` or `Python`.

The following input functions and parameters should be considered:

- `obj`, `grad` – function handles that calculate and return the objective function $f(\boldsymbol{x})$ and the gradient $\nabla f(\boldsymbol{x})$ at an input vector $\boldsymbol{x} \in \mathbb{R}^n$. You can treat these handles as functions or fields of a class or structure `f` or use them directly as input.
- $\boldsymbol{x}^0$ – the initial point.
- `tol` – a tolerance parameter. The method should stop whenever the current iterate $\boldsymbol{x}^k$ satisfies the criterion $\|\nabla f(\boldsymbol{x}^k)\| \leq$ `tol`.
- $\beta \in (0, 1)$, $\ell > 0$ – the momentum parameter and an estimate of the Lipschitz constant of the gradient mapping ∇f.

A pseudo code of the inertial gradient method is shown below.

Algorithm 1: The Gradient Method with Momentum

1 Initialization: Select $\boldsymbol{x}^0 = \boldsymbol{x}^{-1} \in \mathbb{R}^n$ and $\beta \in (0, 1)$, $\ell > 0$ and set $\alpha = 1.99(1 - \beta)/\ell$.
for $k = 0, 1, \ldots$ **do**
2 If $\|\nabla f(\boldsymbol{x}^k)\| \leq$ `tol`, then STOP and $\boldsymbol{x}^k$ is the output.
3 Compute the update $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha \nabla f(\boldsymbol{x}^k) + \beta(\boldsymbol{x}^k - \boldsymbol{x}^{k-1})$.
while $f(\boldsymbol{x}^{k+1}) - f(\boldsymbol{x}^k) \geq \nabla f(\boldsymbol{x}^k)^\top (\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) + \frac{\ell}{2}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2$ **do**
4 Set $\ell = 2\ell$ and $\alpha = 1.99(1 - \beta)/\ell$.
5 Recompute $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha \nabla f(\boldsymbol{x}^k) + \beta(\boldsymbol{x}^k - \boldsymbol{x}^{k-1})$.

The procedure in step 4 and 5 estimates the Lipschitz constant, i.e., ℓ is increased if the initial estimate is too small. Afterwards the step size α and the inertial gradient step are recomputed.

It is possible to introduce additional safeguards like $\alpha = \max\{\alpha_{\min}, \min\{\alpha, \alpha_{\max}\}\}$ or to reduce the Lipschitz estimate ℓ (and increase $\alpha = 1.99(1-\beta)/\ell$) from time to time to avoid stagnation due to too large choices of ℓ. (You don't need to necessarily consider such potential adjustments).

Test your implementation on the optimization problem (1) discussed in the previous assignment:

– Repeat the tasks from Assignment A4.1 and run the inertial gradient method with parameters $\ell = \beta$ and $\beta \in \{0.3, 0.5, 0.7, 0.9\}$ to solve (1) with different initial points selected from the set $\mathcal{X}^0$.

  You can use $\texttt{tol} = 10^{-5}$. As before, for each choice of β 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 f in the background of your figure.

– Report the behavior and performance of the inertial gradient method and compare it to the convergence of the gradient method tested in A4.1. In particular, discuss the number of required iterations (on average) and the effect of β.

Sheet 4 is due on **Nov, 22nd**. Submit your solutions before **Nov, 22nd**, **11:00 pm**.