MAA 307 – Convex optimization and optimal control

Homework Due date: Wednesday 29 of November - 8:35am

Part I: submit either a pdf by email or hand-in a written document at the beginning of the class of November 29.
Part II: submit a Python notebook (or equivalent) including answers and code by email with subject "MAA 307 - Homework Part 2 - Name1 Name2" with the lastname of the group members.

1 Convergence of the subgradient method

In this exercice $\|.\|$ denotes the euclidean norm in $\mathbb{R}^n$.

In the class we have stated and proven the convergence of gradient descent (GD) for a convex, differentiable and L-smooth function. More precisely we have shown that if $x^\star$ is a minimum and $f(x^\star) = f^\star$, and the step-size $\alpha \in (0, 1/L]$, the sequence x_k produced by the gradient descent algorithm satisfies

$$f(x_k) - f^\star \leq \frac{\|x_0 - x^\star\|^2}{2\alpha k}.$$

You will investigate in this exercice the convergence of the subgradient method where at each iteration the update reads:

$$x_{k+1} = x_k - \alpha_k g_k \ , \ k = 0, 1, \dots$$

where g_k is an element from the subdifferential of f at x_k and $\alpha_k > 0$ a step-size at iteration k.

We use the same notations as above for $x^\star$ and $f^\star$ and assume that the objective function f is:

- convex with $\mathrm{dom} f = \mathbb{R}^n$
- Lipschitz-continuous with continuity parameter $G > 0$.

We want first to prove that for all x, for all $g \in \partial f(x)$

$$\|g\| \leq G. \tag{1}$$

.

(a) Let $x \in \mathbb{R}^n$ and $g \in \partial f(x)$. If $g = 0$ deduce (1). Assume that $g \neq 0$, show that there exists z such that

$$z - x = \mathrm{argmax}_{v, \|v\|=1} \langle v, g \rangle.$$

Verify that $\langle v, g \rangle \leq \langle z - x, g \rangle$ for all v with $\|v\| = 1$ and $\|z - x\| = 1$.

(b) Deduce that $\|g\| \leq G$.

(c) Using properties of the subgradient show that

$$\|x_{k+1} - x^\star\|^2 - \|x_k - x^\star\|^2 \leq \alpha_k^2 \|g_k\|^2 - 2\alpha_k (f(x_k) - f(x^\star))$$

and then use a telescoping sum to show that

$$\|x_N - x^\star\|^2 - \|x_0 - x^\star\|^2 \leq \sum_{k=0}^{N-1} \alpha_k^2 \|g_k\|^2 - 2 \sum_{k=0}^{N-1} \alpha_k (f(x_k) - f(x^\star))$$

The subgradient method is not a strict descent method, so we need to keep track of the best iterate x_k^{best} such that $f(x_k^{\mathrm{best}}) = \min_{i=0,\dots,k} f(x_i)$.

(d) Use the result in the previous part to prove that

$$f(x_{N-1}^{\text{best}}) - f(x^\star) \leq \frac{R^2 + G^2 \sum_{k=0}^{N-1} \alpha_k^2}{2 \sum_{k=0}^{N-1} \alpha_k}$$

where $R = \|x_0 - x^\star\|$.

(d) Deduce that by using a fixed step-size $\alpha_k = \alpha$ then the subgradient method satisfies

$$\lim_{N \to \infty} f(x_N^{\text{best}}) \leq f^\star + G^2 \alpha / 2.$$

(e) We call diminishing step-size, some step-size chosen according to $\sum_{k=0}^{+\infty} \alpha_k^2 < +\infty$ and $\sum_{k=0}^{+\infty} \alpha_k = +\infty$. Prove that with diminishing step-size, then the subgradient method converges to the optimum, namely

$$\lim_{N \to \infty} f(x_N^{\text{best}}) = f^\star.$$

2 Testing Gradient Descent, BFGS and Coordinate Descent

This part contains theoretical questions and numerical questions. I recommend you to prepare a notebook with the answer to the questions (also the theoretical ones), the code and graphs. Please read the remark in the end and the Python helper section before to start implementing.

We consider the following class of objective functions defined in $\mathbb{R}^n$

$$f_{\text{elli,cond}}(x) = \frac{1}{2} \sum_{i=1}^{n} (\text{cond})^{\frac{i-1}{n-1}} x_i^2 \tag{2}$$

for cond > 0 a real value that will be taken in $\{1, 10, 10^2, \ldots, 10^6\}$. We also consider $R \in \mathbb{R}^{n \times n}$ a (random) rotation matrix non equal to the identity and define

$$f_{\text{rot-elli,cond}}(x) = f_{\text{elli,cond}}(Rx) \ . \tag{3}$$

Last we consider the Rosenbrock function

$$f_{\text{rosen}}(x) = \sum_{i=1}^{n-1} \left[100 \cdot (x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right] \ . \tag{4}$$

2.1 Introduction

(a) Compute the gradients and Hessian matrix of $f_{\text{elli,cond}}(x)$ and $f_{\text{rot-elli,cond}}(x)$. What are the condition numbers of the Hessian matrices of $f_{\text{elli,cond}}(x)$ and $f_{\text{rot-elli,cond}}(x)$?

(b) Show that $f_{\text{elli,cond}}(x)$ and $f_{\text{rot-elli,cond}}(x)$ are μ-strongly convex and L-smooth and identify the largest (respectively smallest) μ (respectively L) depending on the parameter cond.

(c) Display the sublevel sets of the Rosenbrock function in dimension 2. Is the function convex, quasi-convex?

You will now consider three optimization algorithms. Gradient descent with fixed step-size that you will implement yourself, the quasi-Newton BFGS algorithm (you can use the implementation in scipy.optimize, see below) and the coordinate-descent algorithm defined as

Coordinate descent (CD):

Given $f : \mathbb{R}^n \to \mathbb{R}$.
Initialize $x_0 \in \mathbb{R}^n$
For $k = 1, 2, \ldots,$
Define x_{k+1} from x_k by iteratively optimizing f along each coordinate, namely, for each coordinate $i = 1, \ldots, n$

$$x_{k+1}^i = \arg\min_{y \in \mathbb{R}} f(x_{k+1}^1, \ldots, x_{k+1}^{i-1}, y, x_k^{i+1}, \ldots, x_k^n)$$

We choose the convention to call k the iteration index. Remark that at each iteration, the algorithm performs n 1-dimensional optimization procedures.

2.2 Invariance of coordinate descent

(a) How many iterations are needed to minimize (2)? Does it depend on the parameter cond? Provide a small theoretical proof.

(b) We now want to study quantitatively the impact of rotating the search space on the performance of CD. Implement CD to minimize numerically (2) and (3). To perform the 1D-search in the coordinate descent, one can use the `minimize_scalar` function from scipy optimize implementing the Brent method.

```
from scipy.optimize import minimize_scalar
```

(c) Display the number of iterations needed by CD to reach a function value which is smaller than 10^{-8} as a function of the parameter cond $\in \{1, 10, \ldots, 10^5\}$ when minimizing (2) and (3) for $n = 10$. Display on the same graph the results for (2) and (3). Describe the results that appear on the graph.

(d) Is it surprising that CD needs more iterations on (3)? Do the results suggest that CD is rotational invariant?

(e) Find a theoretical explanation to your conclusion on the (non) rotational invariance of CD.

2.3 Influence of cond on GD and BFGS.

(a) Reproduce the graph shown in class 9 (slide 1 entitled "Sensibility of gradient descent wrt condition number"), showing the sensibility of gradient descent wrt condition number (realize that there is a difference of 1/2 between the function used and the function (2)). Comment what you observe on the graph.

(b) Create a similar graph for GD and BFGS optimizing (3). Discuss the difference observed.

2.4 Convergence graphs

(a) Display some convergence graphs (i.e. function value on the y-axis and number of iterations on the x-axis) of GD (with a well chosen step-size) and CD on the function (3) for cond $= 10^3$. Explain how you chose the step-size for GD and detail the implementation of the gradient. You can consider $n = 10$ and vary the initial vector x_0.

(b) Describe the results you obtain. Which type of convergence do you observe? Is it surprising? Comment on the impact of the initial vector.

(c) [**Bonus question: graded with extra points**] Investigate the convergence of GD, CD, and BFGS on the Rosenbrock function (4) in dimension 10. Explain in details how you proceed, which tests you performed, which graphs you decide to display and comment on the difference observed, on the impact of the initial vector, of the step-size choice for GD. Beware that it is not obvious to make fair comparisons among algorithms as you cannot always compare two iterations of an algorithm if one

algorithm uses the gradient and the other not. One way to solve this issue is to compare the number of calls to the objective function and assume a cost of n function evaluations for one gradient call (this comes from the cost of evaluating the gradient numerically via finite differences where the cost is $n+1$).

Remarks:

- When presenting your results on graphs, choose the right scale (log or not log) such that the graph is informative. In general: Log-scales are your friends. Add grids to your plots `plt.grid(True)`
- Start by running quick experiments (take possibly a smaller dimension, use small condition numbers).

Python helper

To implement a rotation sampled randomly, one can use:

```
>>> import cma, numpy as np
>>> R = cma.transformations.Rotation()
>>> x = np.array((1,2,3))
>>> y = R(x)
>>> list(np.round(R(R(x), inverse=1), 9))
```

(see `https://cma-es.github.io/apidocs-pycma/cma.fitness_transformations.Rotated.html`).

For the Rosenbrock function, you can use the implementation from scipy optimize:

```
from scipy.optimize import rosen as f
from scipy.optimize import rosen_der as fp
```

For the BFGS implementation, you can use the one from scipy optimize:

```
from scipy.optimize import minimize
result = minimize(f, x0, method='BFGS', jac=fp, options={'maxiter': iterations, 'disp': True},
callback=callbackf)
```

In the call of the minimize function, the last argument is to call the following function

```
# Callback function to collect the trajectory
def callbackf(x):
    trajectory.append(x.copy())
```

that is called at every iteration of the algorithm. This way you can record the trajectory (x-value and then deduce the f-value you need for plotting).