Convex Optimization Homework #2 ,

Due: Thursday June 3, 2021, 11am.

r09942082 楊雅婷

1. (100%) In this problem, we use the Newton's method to solve an unconstrained optimization problem. We wish to minimize

$$f(x) = c^T x - \sum_{i=1}^{m} \log(-a_i^T x + b_i)$$

where $A \in \mathbf{R}^{m\times n}$ with a_i being the transponse of the ith row of A, $b \in \mathbf{R}^m$, $c \in \mathbf{R}^n$.

We choose $m = 3, n = 2$, and

$$A = \begin{bmatrix} 1 & 3 \\ 1 & -3 \\ -1 & 0 \end{bmatrix}, b = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}, \quad c = \begin{bmatrix} 1 \\ 2 \end{bmatrix}. \tag{1}$$

(a) Find **dom** f. Show that **dom** f is a convex set.

(b) Derive $\nabla f(x)$ and $\nabla^2 f(x)$ for any $x \in$ **dom** f.

(c) Write a matlab m-file as a function which takes inputs A, b, c, and x, and evaluates the objective function at the point x, as well as the gradient and the Hessian.
Hint: the function can have a header that looks like the following.

```
function [f, g, H] = my_objective(x, A, b, c)
```

(d) From this step on, write another m-file as the main function for your program to solve the unconstrained problem. Set A and b as in Eq. (1) and set the initial point $x^{(0)} = 0$.

(e) Use the Newton step's formula: $\Delta x_{\text{nt}} = -(\nabla^2 f(x))^{-1}\nabla f(x)$ and calculate the first Newton step $\Delta x_{\text{nt}}^{(0)}$.

(f) Calculate the Newton decrement $\lambda(x) = (-\nabla f(x)^T \Delta x_{\text{nt}})^{1/2}$.

(g) Perform backtracking line search along search direction using $\beta = 0.7$ starting from $t = 1$ until $x^{(0)} + t\Delta x_{\text{nt}}^{(0)} \in$ **dom** f. (Note: $t^+ := \beta t$)

(h) Continue the backtracking line search until $f(x + t\Delta x_{\text{nt}}^{(0)}) \le f(x) - \alpha t\lambda(x)^2$, where $\alpha = 0.1$.

(i) Perform the update $x^{(1)} = x^{(0)} + t\Delta x_{\text{nt}}^{(0)}$. Determine whether $\lambda^2/2 \le \epsilon$ where $\epsilon = 10^{-10}$.

(j) Repeat (e)-(i) for the next iteration. Wrap up these steps in a main loop. For the kth iteration, record the following items: (1) $f(x^{(k)})$, (2) $\lambda(x^{(k)})$, (3) $t^{(k)}$. Plot $\lambda(x^{(k)})^2/2$ versus k.

(k) Use the `cvx` toolbox to solve the problem. Compare the results (i.e., optimal value and optimal point) with those you obtained through Newton's method. Do they coincide with each other?

(l) Set $A = \begin{bmatrix} 1 & 3 \\ 2 & -3 \\ -1 & 0 \end{bmatrix}, b = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}, \quad c = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $x^{(0)} = 0$. Repeat (e)-(k).

Homework submission guidelines:

- Submit your answer online as a set of three files: two m-files and a document file (in *.pdf or *.docx) that contains all answers (including plots) in this problem set.
- If you choose to do the homework in Python, then submit *.py files instead of m-files.
- Submit your files online at the NTU Cool website. No paper shall be handed in.
- Late submissions will be treated under the principle elaborated as follows.
  (1) Homework received by 11am, June 3 (t_1) will be counted fully.
  (2) Homework received after 2pm, June 3 (t_2) will not be counted.
  (3) Homework received between t_1 and t_2 will be counted with a discount rate

$$\frac{t_2 - t}{t_2 - t_1}$$

  where t is the received time. Note that $t_2 - t_1$ is three hours.
- Plagiarism is strongly prohibited. While discussions among classmates are allowed (and encouraged), you shall not ask anyone else to share his/her codes with you, nor should you attempt to share with anyone your codes. You should write every line of the code by yourself. If any part of your submission is found to be copied from someone else's submission, then both of your homework submissions will be counted zero.

r09942082 楊雅婷

1. $f(x) = c^T x - \sum_{i=1}^{m} \log(-a_i^T x + b_i)$ $\quad f(x) = x_1 + 2x_2 - \log(-x_1 - 3x_2 + 0.1) - \log(-x_1 + 3x_2 + 0.1) - \log(x_1 + 0.1)$

$A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ $\quad a_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, a_2 = \begin{bmatrix} 1 \\ -3 \end{bmatrix}, a_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$

$\Rightarrow m = 3,\ n = 2,\quad A = \begin{bmatrix} 1 & 3 \\ 1 & -3 \\ -1 & 0 \end{bmatrix},\ b = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \end{bmatrix},\ c = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

(a) Find dom f, and show dom f is a convex set

① $\text{dom}\ \log(\cdot) = \mathbb{R}_+^n$, $-a_i^T x + b_i \geq 0$, $i = 1, 2, \dots, n$

$\Rightarrow \text{dom}\, f = \{x \mid x \in \mathbb{R}^n, -Ax + b \succeq 0\}$ $\quad \text{dom}\, f = \{x \mid x \in \mathbb{R}^2, -x_1 - 3x_2 + 0.1 \geq 0, -x_1 + 3x_2 + 0.1 \geq 0, x_1 + 0.1 \geq 0\}$

② If $x \in \text{dom}\, f$ and $y \in \text{dom}\, f$, $-Ax + b \succeq 0$, $-Ay + b \succeq 0$

For $0 \leq \theta \leq 1$, $-A[\theta x + (1-\theta) y] + b = -\theta Ax + \theta b - (1-\theta) Ay + (1-\theta) b = \theta(-Ax + b) + (1-\theta)(-Ay + b) \succeq 0$

$\Rightarrow$ dom f is a convex set

(b) $\nabla f(x)$ and $\nabla^2 f(x)$

① $\nabla f(x) = c - \sum_{i=1}^{m} \frac{1}{-a_i^T x + b_i}(-a_i) = c + \sum_{i=1}^{m} \frac{a_i}{-a_i^T x + b_i} \Rightarrow \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} \frac{1}{-x_1 - 3x_2 + 0.1} \\ \frac{3}{-x_1 - 3x_2 + 0.1} \end{bmatrix} + \begin{bmatrix} \frac{1}{-x_1 + 3x_2 + 0.1} \\ \frac{-3}{-x_1 + 3x_2 + 0.1} \end{bmatrix} + \begin{bmatrix} \frac{-1}{x_1 + 0.1} \\ 0 \end{bmatrix}$

② $\nabla^2 f(x) = \sum_{i=1}^{m} \frac{1}{-(-a_i^T x + b_i)^2} a_i (-a_i^T) = \sum_{i=1}^{m} \frac{a_i a_i^T}{(-a_i^T x + b_i)^2}$

(c) my_objective_(x, A, b, c) in hw2_objective.py

(d) solver() in hw2_main.py

(e) $\Delta x_{nt}^{(0)} = \begin{bmatrix} -0.036667 \\ -0.001111 \end{bmatrix}$

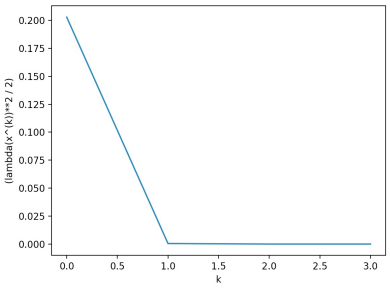
(f) $\lambda(x) = 0.06368324$

[(e) (f) (g) (h) (i) (j) are in newton_method(x, A, b, c) in hw2_main.py]

(i) After performing the update $x^{(1)} = x^{(0)} + t\Delta x_{nt}^{(0)}$

Determine whether $\lambda^2/2 \leq \varepsilon$ where $\varepsilon = 10\text{–}10$ "No" in this iter

(j) and (k)

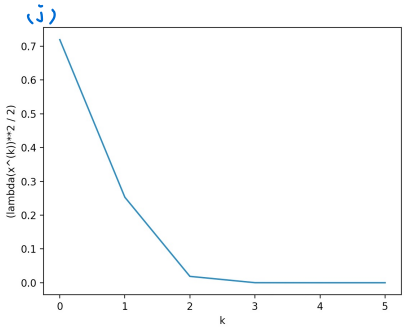



```
For original A, b, c:
A:
 [[ 1  3]
 [ 1 -3]
 [-1  0]]
b:
 [[0.1]
 [0.1]
 [0.1]]
c:
 [[1]
 [2]]
Getting result from the Newton's method...
First Newton step:  [[-0.03666667 -0.00111111]]   First Newton decrement:  [[0.63683244]]
***Optimal point of x =  [[-0.03631456 -0.00206039]]
***Optimal value =  6.701002496254256
Obj value in each iter:
 [6.907755278982137, 6.701470483515969, 6.701002571803488, 6.701002496254256]
Decrement in each iter:
 [0.6368324391514267, 0.030686579364916217, 0.0003887110433103496, 6.562049993772393e-08]
t in each iter:
 [1.0, 1.0, 1.0, 1.0]
Getting result from cvx toolbox...
***Optimal point of x =  [-0.03631456 -0.00206038]
***Optimal value =  6.701002456197811
```

It seems that the Newton's method coincide with cvx toolbox.

(d).

(j)




```
For A, b, c in 1.(l):
A:
 [[ 1  3]
 [ 2 -3]
 [-1  0]]
b:
 [[0.2]
 [0.3]
 [0.4]]
c:
 [[1]
 [1]]
Getting result from the Newton's method...
First Newton step:  [[-0.13265393 -0.01506016]]   First Newton decrement:  [[1.1995871]]
***Optimal point of x =  [[-0.23772562 -0.07652448]]
***Optimal value =  2.514093678290154
Obj value in each iter:
 [3.7297014486341915, 2.7984780333725174, 2.5327325253978135, 2.5141806919755845, 2.5140936827970943, 2.514093678290154]
Decrement in each iter:
 [1.1995870957471517, 0.7113525045741452, 0.19388436279863105, 0.01321841767055216, 9.494026722271074e-05, 4.659407762315283e-09]
t in each iter:
 [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Getting result from cvx toolbox...
***Optimal point of x =  [-0.23772563 -0.07652448]
***Optimal value =  2.5140935930954975
```

(e) & (f)

(k) It seems that the Newton's method coincide with cvx toolbox.