MAS374 OPTIMIZATION THEORY
HW#9 (PROGRAMMING)

Let $\mathcal{X}$ be a nonempty closed convex subset of $\mathbb{R}^n$, and $f_0 : \mathbb{R}^n \to \mathbb{R}$ a differentiable convex function. Say we want to solve a convex constrained optimization problem

$$\begin{aligned} p^* = \min_{\boldsymbol{x}} \quad & f_0(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathcal{X}. \end{aligned} \tag{1}$$

By considering an equivalent unconstrained problem

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \quad f_0(\boldsymbol{x}) + I_{\mathcal{X}}(\boldsymbol{x})$$

where $I_{\mathcal{X}}$ is the indicator function of $\mathcal{X}$, we can use the proximal gradient method to solve (1). In particular, as the proximal map of $I_{\mathcal{X}}(\boldsymbol{x})$ is $[\boldsymbol{x}]_{\mathcal{X}}$, the Euclidean projection onto $\mathcal{X}$, we get the following *projected gradient method*.

Algorithm 1: Projected Gradient Method

Initialize: $\boldsymbol{x}_0 \in \operatorname{dom}(f_0 + I_{\mathcal{X}}) = \mathcal{X}$
Set $k = 0$ and accuracy $\epsilon > 0$
while accuracy not attained **do**
    Choose stepsize s_k
    Update: $\boldsymbol{x}_{k+1} = [\boldsymbol{x}_k - s_k \nabla f_0(\boldsymbol{x}_k)]_{\mathcal{X}}$
    $k \leftarrow k + 1$
return $\boldsymbol{x}_k$

Usually the most challenging part of actually implementing this algorithm into a computer program is building the projection map $[\,\cdot\,]_{\mathcal{X}}$. In most of the cases an explicit expression for $[\,\cdot\,]_{\mathcal{X}}$ is not known, and even if an explicit expression is known it is usually quite complicated.

In this problem, we will build a solver which solves a QP:

$$\begin{aligned} p^* = \min_{\boldsymbol{x} \in \mathbb{R}^n} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H} \boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}. \end{aligned} \tag{2}$$

From now on, let $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}\}$. For simplicity we assume that the feasible set $\mathcal{X}$ has a nonempty relative interior, and $\boldsymbol{H} \succ \boldsymbol{0}$. Still, in general, a closed form expression for the projection map $[\,\cdot\,]_{\mathcal{X}}$ is not known. Yet, by doing this assignment, you will see an approach to overcome this issue.

Problem 1. Let $\boldsymbol{x}$ be any point in $\mathbb{R}^n$. By definition, the Euclidean projection of $\boldsymbol{x}$ onto $\mathcal{X}$ is nothing but the optimal solution of the problem

$$\begin{aligned} \min_{\boldsymbol{y} \in \mathbb{R}^n} \quad & \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{y} \preceq \boldsymbol{b}. \end{aligned} \tag{3}$$

Denote the dual variable by $\boldsymbol{\lambda} \in \mathbb{R}^n$. Answer to the following question <u>in your report</u>.

(a) Derive the Lagrangian $\mathcal{L}(\boldsymbol{y}, \boldsymbol{\lambda})$ and the dual function $g(\boldsymbol{\lambda})$ of (3).

(b) Denote the optimal solution for (3) by $\boldsymbol{y}^*$, and the dual optimal solution by $\boldsymbol{\lambda}^*$. As mentioned above, we have $\boldsymbol{y}^* = [\boldsymbol{x}]_{\mathcal{X}}$. Find a closed form expression of $[\boldsymbol{x}]_{\mathcal{X}}$ using $\boldsymbol{x}$, $\boldsymbol{\lambda}^*$, $\boldsymbol{A}$, and/or $\boldsymbol{b}$. *Hint*: *Use the KKT conditions. You should explain why strong duality holds.*

Due Date: December 21, 2021.

Problem 2. The previous problem indicates that, in order to compute $[\boldsymbol{x}]_{\mathcal{X}}$ we should first determine the dual optimal point $\boldsymbol{\lambda}^*$. To this end, consider the dual problem of (3) in the minimization form:

$$\begin{aligned} \min_{\boldsymbol{\lambda}\in\mathbb{R}^n} \quad & -g(\boldsymbol{\lambda}) \\ \text{s.t.} \quad & \boldsymbol{\lambda} \succeq \mathbf{0}. \end{aligned} \tag{4}$$

Let $\boldsymbol{\Lambda}$ denote the dual feasible set, i.e. $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda} \in \mathbb{R}^n : \boldsymbol{\lambda} \succeq \mathbf{0}\}$.

(a) Write a Python/MATLAB function `dual_proj(l)` which takes a vector $\texttt{l} \in \mathbb{R}^n$ and returns its projection $[\texttt{l}]_{\boldsymbol{\Lambda}}$ onto $\boldsymbol{\Lambda}$.

(b) Write a Python/MATLAB function `dual_grad(l, x, A, b)` which takes a vector $\texttt{l} \in \mathbb{R}^n$ and returns $-\nabla_{\boldsymbol{\lambda}} g(\texttt{l})$, the gradient of the objective function in (4) at `l`.

(c) Write a Python/MATLAB function `solve_dual(x, A, b)` which uses the projected gradient method (Algorithm 1) to solve (4) within an accuracy of $\texttt{tol} = 2^{-40}$ and returns the computed optimal solution $\boldsymbol{\lambda}^*$. Recall that the ultimate goal of computing $\boldsymbol{\lambda}^*$ is to use it for obtaining $[\boldsymbol{x}]_{\mathcal{X}}$, rather than solving (4) itself. In this sense, for the stopping criterion, you should check how much $[\boldsymbol{x}]_{\mathcal{X}}$ is changed by using $\boldsymbol{\lambda}_{k+1}$ compared to using $\boldsymbol{\lambda}_k$, and terminate the iteration if the ℓ_2-norm of the change in $[\boldsymbol{x}]_{\mathcal{X}}$ is less than $\texttt{tol} = 2^{-40}$.

Use the initial point $\boldsymbol{\lambda}_0 = \mathbf{0} \in \boldsymbol{\Lambda}$. For the stepsize, show that $-g(\boldsymbol{\lambda})$ has a Lipschitz continuous gradient with some Lipschitz constant L, and use constant stepsize $s_k = 1/L$. Of course, L will depend on $\boldsymbol{A}$ and/or $\boldsymbol{b}$. <u>In your report</u>, explain how your code chooses L.

Problem 3. Now we get back to our original QP

$$\begin{aligned} p^* = \min_{\boldsymbol{x}\in\mathbb{R}^n} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}. \end{aligned} \tag{5}$$

Recall that we are denoting $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}\}$. Let us also write $f_0(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x}$.

(a) Write a Python/MATLAB function `primal_proj(x, A, b)` which takes a vector $\texttt{x} \in \mathbb{R}^n$ and returns its projection $[\texttt{x}]_{\mathcal{X}}$ onto $\mathcal{X}$. Use the results from Problems 1 and 2.

(b) Write two Python/MATLAB functions: `grad_f0(x, H, c)` which evaluates $\nabla_{\boldsymbol{x}} f_0(\texttt{x})$, and `f0(x, H, c)` which evaluates $f_0(\texttt{x})$.

(c) Now we have everything we need. Write your code which uses the projected gradient method (Algorithm 1) to solve (5) within an accuracy of $\texttt{eps} = 2^{-40}$. Use the ℓ_2-norm of the change on the iterates, $\|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}\|_2$, as the measure of accuracy.

Use the inital point $\boldsymbol{x}_0 = \mathbf{0}$. Here, note that $\mathbf{0}$ might not be in $\mathcal{X}$, but you will see that the algorithm still works fine. For the stepsize, show that $f_0(\boldsymbol{x})$ has a Lipschitz continuous gradient with some Lipschitz constant L, and use constant stepsize $s_k = 1/L$. Here, L will depend on $\boldsymbol{H}$ and/or $\boldsymbol{c}$. <u>In your report</u>, explain why taking $\boldsymbol{x}_0 = \mathbf{0}$ is okay, and how your code chooses L.

Your code should print out the computed optimal value and the computed optimal solution. (See page 3 for sample outputs.)

Caution:

- Fill out the provided code template appropriately, and submit it through KLMS.
- You can choose either using MATLAB or using Python 3 (with NumPy).
- You may define your own functions in the code if you need.
- Your report should contain answers and explanations to the things that are asked in Problems 1(a), 1(b), 2(c), and 3(c).
- 10 points each are assigned to the code and the report.

Examples

As a guidance, here are some examples of QP coefficients and the corresponding expected output of your code. These results may slightly differ from your results, due to rounding errors. A function which helps you to print out the results in the following format is implemented for you in the template.

$$\boldsymbol{H} = \begin{bmatrix} 6 & 4 \\ 4 & 14 \end{bmatrix}, \ \boldsymbol{c} = \begin{bmatrix} -1 \\ -19 \end{bmatrix}, \ \boldsymbol{A} = \begin{bmatrix} -3 & 2 \\ -2 & -1 \\ 1 & 0 \end{bmatrix}, \ \boldsymbol{b} = \begin{bmatrix} -2 \\ 0 \\ 4 \end{bmatrix}$$

```
optimal value p* =
   -4.002525252526707

optimal solution x* =
   1.1010101010099662
   0.6515151515152153
```

$$\boldsymbol{H} = \begin{bmatrix} 20 & -5 & 3 \\ -5 & 17 & 0 \\ 3 & 0 & 10 \end{bmatrix}, \ \boldsymbol{c} = \begin{bmatrix} 4 \\ 2 \\ 7 \end{bmatrix}, \ \boldsymbol{A} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix}, \ \boldsymbol{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{bmatrix}$$

```
optimal value p* =
   6.177018633524976

optimal solution x* =
   0.3726708074524678
   0.5279503105580037
   0.09937888198783879
```

$$\boldsymbol{H} = \begin{bmatrix} 34 & 4 & 15 & 10 & 2 \\ 4 & 35 & -17 & 3 & -8 \\ 15 & -17 & 36 & -12 & -4 \\ 10 & 3 & -12 & 37 & -11 \\ 2 & -8 & -4 & -11 & 38 \end{bmatrix}, \ \boldsymbol{c} = \begin{bmatrix} -1 \\ 17 \\ -2 \\ 9 \\ 0 \end{bmatrix}, \ \boldsymbol{A} = \begin{bmatrix} 10 & 18 & 12 & 2 & 0 \\ 18 & -1 & 1 & 19 & -2 \\ 6 & -7 & 18 & 8 & -6 \\ -13 & 19 & 11 & -10 & -2 \\ 3 & 5 & -1 & -4 & -13 \\ -4 & 0 & 11 & 19 & -8 \end{bmatrix}, \ \boldsymbol{b} = \begin{bmatrix} 16 \\ 7 \\ 10 \\ 4 \\ 17 \\ 18 \end{bmatrix}$$

```
optimal value p* =
   -23.750664014073177

optimal solution x* =
   1.6420579101585024
   -1.843202697415582
   -2.0992582072077504
   -1.4873078388514014
   -1.0185653995430897
```