Assignment 2

E0 230: Computational Methods of Optimization

October 2023

No copying allowed. Thorough plagiarism check will be run. You are given four questions, each of which has subpart(s) that require(s) python programming.

SUBMISSION INSTRUCTIONS

1. You should submit **five individual files** (NOT a zip file) with the following naming convention.

   ▷ `LastFiveDigitsOfSRNumber.pdf` → Answers to all the problems.

   ▷ `LastFiveDigitsOfSRNumber_1.py` → Code for the first problem.

   ▷ `LastFiveDigitsOfSRNumber_2.py` → Code for the second problem.

   ▷ `LastFiveDigitsOfSRNumber_3.py` → Code for the third problem.

   ▷ `LastFiveDigitsOfSRNumber_4.py` → Code for the fourth problem.

   For example, if the last five digits of your SR Number is 20000, then you should submit five *individual* files `20000.pdf`, `20000_1.py`, `20000_2.py`, `20000_3.py`, `20000_4.py`. Every file should run without any errors with the command `python3 ⟨filename⟩.py`.

2. **Any deviation from the above rule will incur serious penalty!**

3. All the code must be written in Python 3.10 or higher (no MATLAB allowed). All code written in Jupyter notebooks must be converted to `.py` format before submitting.

4. For the coding questions, you are asked to report some values, e.g., the number of iterations. These values should be reported in the `.pdf` file that you submit.

5. At the top of the `.pdf` file that you submit, write your name and SR Number.

ORACLE ACCESS INSTRUCTIONS

1. If you are a Mac user, download the zip file `MAC_CMO_A2.zip`. Otherwise, download the zip file `NONMAC_CMO_A2.zip`. Extract it to get the folder `CMO_A2`.

2. You need to run your `.py` files by placing them in the `CMO_A2` folder.

3. All the oracles are provided by the library file `CMO_A2.py`. Moreover, all vectors are represented as numpy arrays. Hence, make sure to include the following lines in your code.

```python
import numpy as np
import CMO_A2
```

4. Instructions on how to use the appropriate oracle for a problem are given below that problem.

5. All the oracle outputs are dependent on the last five digits of your SR Number, which should be passed to the oracle in `int` format.

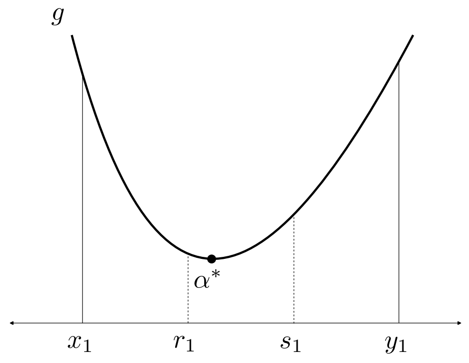
6. Although the second problem requires you to code, it does not need any oracle access.

1 Trisection Method

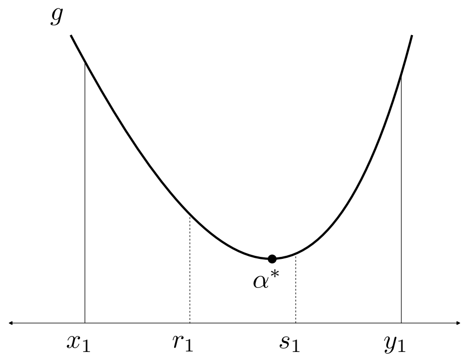
(5 marks)

Recall from the tutorial that one way to perform exact line search is the bisection method. However, it requires us to compute the gradient, which can be expensive. An alternative to find the minimum of a given function $g(\alpha)$ is the *Trisection Method*.

For our purposes, assume that g is strictly convex with the minimum at α^*. Say, we are given two points x_1, y_1 such that $x_1 \leq \alpha^* \leq y_1$. Suppose we partition the interval $[x_1, y_1]$ into three *equal* intervals $[x_1, r_1), [r_1, s_1]$, and $(s_1, y_1]$. If $g(r_1) \leq g(s_1)$, then we can be sure that the minimizer α^* lies in the interval $[x_1, s_1]$. Otherwise, we can be sure that α^* lies in $[r_1, y_1]$. See the below figures for some intuition.



In this case, since $g(r_1) \leq g(s_1)$, the minimizer α^* lies in the interval $[x_1, s_1]$.



In this case, since $g(r_1) > g(s_1)$, the minimizer α^* lies in the interval $[r_1, y_1]$.

This way, we can reduce our search interval from $[x_1, y_1]$ to a strictly smaller interval $[x_2, y_2]$ where

- $[x_2, y_2] = [x_1, s_1]$, if $g(r_1) \leq g(s_1)$.
- $[x_2, y_2] = [r_1, y_1]$, otherwise.

We recurse until we reach a search interval $[x_t, y_t]$ of length at most ε. Once this stopping condition is reached, we return our estimate of α^* as $(x_t + y_t)/2$.

Now, answer the following questions.

1. Compute r_1, s_1 in terms of x_1, y_1.

$$r_1 = x_1 + \frac{y_1 - x_1}{3},$$
$$r_1 = x_1 + \frac{2(y_1 - x_1)}{3}.$$

Notes: Carries 1 mark.

2. In how many iterations does the process end? Prove it and report your answer in terms of x_1, y_1, ε.

In each iteration, the search interval's size drops to a factor of $2/3$. The search interval's size, therefore, at the end of the k^{th} iteration is given by

$$\left(\frac{2}{3}\right)^k \left(\frac{y_1 - x_1}{3}\right).$$

The process ends when we reach a search interval of size at most ε. Hence, if we let k to be the total number of iterations, both the following inequalities must be satisfied.

$$\left(\frac{2}{3}\right)^k \left(\frac{y_1 - x_1}{3}\right) \leq \varepsilon \quad \text{and} \quad \left(\frac{2}{3}\right)^{k-1} \left(\frac{y_1 - x_1}{3}\right) > \varepsilon. \tag{1.1}$$

Combining with the fact that k must be an integer gives us

$$k = \left\lceil \frac{\log((y_1 - x_1)/3)}{\log(1/\varepsilon)} \right\rceil.$$

Notes: Carries 1 mark. Students who have directly arrived at the above equality for k from the left inequality of (1.1) without proper explanation have lost marks.

3. You are given oracle access to a function $g(\alpha)$. (See below.) Minimize it using the trisection method. Choose the initial interval as $[x_1, y_1] = [-1, 1]$ and the tolerance $\varepsilon = 10^{-4}$. Report the number of iterations, the estimate of the minimizer α^*, and the estimate of the minimum value $g(\alpha^*)$ (both up to 9 decimal places).

Irrespective of your SR Number, the process takes 25 iterations.

Notes: Number of iterations - 1 mark, Minimizer - 1 mark, Minimum - 1 mark.

```
# INSTRUCTIONS TO ACCESS THE ORACLE g(alpha)
import numpy as np
import CMO_A2
func_val_at_alpha = CMO_A2.oracle1(LastFiveDigitsOfSRNumber, alpha)
# Input: last five digits of your SR Number -> integer format, alpha -> float format.
# Returns the value of the function g at the point alpha
# For example, val = CMO_A2.oracle1(20000,0.1)
```

2 Damped Newton's Method using Backtracking

(5 marks)

In the tutorial, we have seen that the Newton's update $\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\nabla^2 f(\mathbf{x}_k)^{-1}\right) \nabla f(\mathbf{x}_k)$ may not converge. One of the ways to fix this is to select an appropriate step size. Hence, the new update step would be

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \left(\nabla^2 f(\mathbf{x}_k)^{-1}\right) \nabla f(\mathbf{x}_k) \tag{2.1}$$

for some good choice of α_k. The resulting algorithm is called *Damped Newton's Method.*

In this question, we will select the step size α_k using a simple but popular method called *backtracking.* Let $\beta, c \in (0,1)$ be some constants and let $\mathbf{p}_k$ be a descent direction. The backtracking algorithm starts off with an initial step size of $\alpha_k = 1$. If the sufficient decrease condition is met, i.e., if

$$f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) < f(\mathbf{x}_k) + c\alpha_k \nabla f(\mathbf{x}_k)^\top \mathbf{p}_k, \tag{2.2}$$

then we freeze α_k. Otherwise, we keep updating the step size as $\alpha_k \leftarrow \beta\alpha_k$ until the sufficient decrease condition, given by eq. (2.2), is satisfied.

Now, consider minimizing the function $f(x) = \sqrt{1 + 2x^2}$ with the starting point denoted by x_0.

1. Show that $f(x)$ is convex.

We obtain

$$f''(x) = \frac{2}{(1+2x^2)^{3/2}},$$

which is always positive. Hence f is convex.

Notes: Carries 1 mark.

2. Say, we choose $x_0 = 1$. Show that Newton's method diverges.

Newton's update is given by $x_k = x_{k-1} - f'(x_{k-1})/f''(x_{k-1})$. Plugging in the expressions for f' and f'' gives us

$$x_k = -2x_{k-1}^3.$$

This is a recursive formula for x_k and it can be shown using induction that

$$x_k = \frac{1}{\sqrt{2}}(-1)^{(3^k-1)/2}\left(\sqrt{2}x_0\right)^{3^k}.$$

Whether $|x_k|$ goes to infinity or not with increasing k solely depends on the value of x_0. If $\sqrt{2}\,|x_0| < 1$, then x_k goes to zero, which is also the minimizer of f. If $\sqrt{2}\,|x_0| = 1$, the iterate alternates between ± 1. If $\sqrt{2}\,|x_0| > 1$, then x_k goes to infinity as k goes to infinity, i.e., Newton's method diverges.

Notes: Carries 1 mark. The only way to show divergence of a method is to prove that the function value eventually goes unbounded with increasing iterations. Many students have based their reasoning on the argument that $f(x_{k+1}) > f(x_k)$ for all k. This is neither a necessary condition nor a sufficient condition for divergence. In fact, a method can very well converge (albeit not to the optimum) while satisfying $f(x_{k+1}) > f(x_k)$ for all k.

3. For what values of x_0 does Newton's method converge?

The solution to the above part gives the answer as $x_0 \in \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$.

Notes: Carries 1 mark.

4. Now, apply Damped Newton's method from $x_0 = 1$ where, in each iteration, the step size is found using backtracking with parameters $\beta = 0.9$ and $c = 0.1$. Take the stopping condition as $|f'(x)| \leq 10^{-4}$. Report the final point (up to 9 decimal places) and the number of iterations, i.e., the final value of k.

The number of iterations required is 6. The final point is 0.000000812.

Notes: Number of iterations - 1 mark, Final point - 1 mark.

3 Nesterov's Accelerated Gradient Descent

(5 marks)

Consider minimizing a differentiable function $f : \mathbb{R}^5 \to \mathbb{R}$. In the gradient descent update rule $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)$, we only use the current iterate $\mathbf{x}_k$ to compute the next iterate $\mathbf{x}_{k+1}$. A smarter way would be to use the previous two iterates $\mathbf{x}_k, \mathbf{x}_{k-1}$. Consider the following update method which uses one more constant parameter θ:

$$\mathbf{x}_{k+1} = \mathbf{y}_k - \alpha \nabla f(\mathbf{y}_k) \quad \text{where} \quad \mathbf{y}_k = \mathbf{x}_k + \theta(\mathbf{x}_k - \mathbf{x}_{k-1}).$$

When $k = 0$, the vector $\mathbf{x}_{k-1}$ is taken to be $\mathbf{0}$, i.e., $\mathbf{y}_0 = (1+\theta)\mathbf{x}_0$. Note that if we take $\theta = 0$, then the above update corresponds to gradient descent.

In this question, we will compare gradient descent and accelerated gradient descent. You are given access to a oracle that, on input $\mathbf{x}$, returns a tuple of the form `(float, numpy.array)`. The first entry of this tuple is the value $f(\mathbf{x})$ and the second entry is the gradient $\nabla f(\mathbf{x})$ represented as a numpy array. (See below.)

1. Perform gradient descent from the starting point $\mathbf{x}_0 = \mathbf{0}$ with a constant step size of $\alpha = 0.00004$. Use the stopping condition as $\|\nabla f(\mathbf{x})\| \leq 10^{-4}$. Report the number of iterations, the final point, and the function value at the final point (both up to 9 decimal places).

   Notes: Number of iterations - 1 mark, Final point and value - 1 mark.

2. Perform accelerated gradient descent from the starting point $\mathbf{x}_0 = \mathbf{0}$ with parameters $\alpha = 0.00004$ and $\theta = 0.142$. Use the stopping condition as $\|\nabla f(\mathbf{x})\| \leq 10^{-4}$. Report the number of iterations, the final point, and the function value at the final point (both up to 9 decimal places).

   Notes: Number of iterations - 2 marks, Final point and value - 1 mark.

```
# INSTRUCTIONS TO ACCESS THE ORACLE f(x) AND gradient(f(x))
import numpy as np
import CMO_A2
(val_at_x,grad_at_x) = CMO_A2.oracle3(LastFiveDigitsOfSRNumber, x)
# Input: last five digits of your SR Number -> integer format, x -> np.array format.
# Returns a tuple of two entries
    # first entry: the value f(x) in float format
    # second entry: the gradient of f at x in np.array format.
# For example, (val,grad) = CMO_A2.oracle3(20000,[0.1,0.2,-3,4,0.5])
```

4 Conjugate Directions Method

(5 marks)

For this question, you are required to query the given oracle to get a positive definite matrix $\mathbf{Q}_{5\times5}$. (See the instructions below.) Let $\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_4$ be the standard basis vectors where $\mathbf{e}_i$ has its $(i+1)^{\text{th}}$ entry as one and the rest of the entries as zeros. The following iterative method can be used to find $\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_4$ that are $\mathbf{Q}$-conjugate.

$$\mathbf{u}_0 = \mathbf{e}_0, \tag{4.1}$$

$$\text{for } i \text{ from 1 to 4: } \mathbf{u}_i = \mathbf{e}_i - \sum_{j=0}^{i-1} \left(\frac{\mathbf{e}_i^\top \mathbf{Q}\mathbf{u}_j}{\mathbf{u}_j^\top \mathbf{Q}\mathbf{u}_j} \right) \mathbf{u}_j. \tag{4.2}$$

Answer the following questions.

1. Use the procedure in eq. (4.1) and eq. (4.2) to find the $\mathbf{Q}$-conjugate vectors $\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_4$. Report the vectors. Keep your precision of every coordinate up to 9 decimal places.

   Notes: Carries 3 marks.

2. Let $\mathbf{b} = [s_1, s_2, s_3, s_4, s_5]^\top$ where $s_1 s_2 s_3 s_4 s_5$ is the suffix of length 5 of your SR Number. Use the conjugate directions method to minimize $\frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{b}^\top \mathbf{x}$ using the $\mathbf{u}_i$-s as the conjugate directions. Use $\mathbf{x}_0 = \mathbf{0}$ as your starting point.

   (a) Report the final minimum $\mathbf{x}^*$ and $f(\mathbf{x}^*)$ (both up to 9 decimal places).

   Notes: Carries 1 mark.

   (b) How many iterations does your method take?

   Irrespective of your SR Number, the method takes exactly 5 iterations.

   Notes: Carries 1 mark.

```
# INSTRUCTIONS TO OBTAIN THE MATRIX Q
import numpy as np
import CMO_A2
Q = CMO_A2.oracle4(LastFiveDigitsOfSRNumber)
# Input: last five digits of your SR Number -> integer format.
# Returns Q in np.array format
# For example, Q = CMO_A2.oracle4(20000)
```