Assignment 2
E0-230 Computational Methods of Optimization

Instructions

- This is an individual assignment, all the work must be your own.
- No copying or sharing of code is allowed. Any form of academic dishonesty will result in ZERO marks for this assignment.
- The deadline for submission is 18th October, 2024.
- Submissions will be accepted till 23rd October, 2024, with a penalty of 5 points per day.
- All the code must be written in Python, along with appropriate comments. The code needs to be submitted for any problem that requires any implementation or programming. Follow the naming convention suggested in the question and feel free to pass additional arguments if necessary with mention. These codes can be reused in subsequent assignments.
- The outputs of the coding problems and other analytical solutions should be reported in a single LaTeX-generated PDF file.
- Submit a single zip file in the `Last_Five_digits_of_sr_no_CMO_A2` which should contains one .py file and LaTeX-generated PDF report.

Oracle Access Instructions

- Based on your OS, download the corresponding .zip file and extract it to get the folder `CMO_A2`.
- You need to run your .py file in the extracted `CMO_A2` folder.
- Make sure to include the following lines of code:

  ```
  from CMO_A2 import f1, f2, f3
  import numpy as np
  ```

- All the oracle outputs are dependent on the last five digits of your serial number, which should be passed in the integer format.

1 Conjugate Gradient Descent (15 points)

In this question, we will explore different methods to solve a system of equations given by $\boldsymbol{Ax} = \boldsymbol{b}$.

1. **(2 points)** Suppose $\boldsymbol{A}$ is an $n \times n$ symmetric, PD matrix. Does this equation $\boldsymbol{Ax} = \boldsymbol{b}$ have a unique solution? State a convex quadratic minimization problem whose optimal solution is $\boldsymbol{x}^*$ such that $\boldsymbol{Ax}^* = \boldsymbol{b}$.

2. **(5 points)** Query the oracle for a PD matrix $\boldsymbol{A}$ and a vector $\boldsymbol{b}$. Implement conjugate gradient descent on the optimization problem in previous part. Report the optimum $\boldsymbol{x}^*$ and the number of iterations it took to reach the optimum.

3. **(3 points)** Suppose $\boldsymbol{A}$ is an $m \times n$ matrix with $m > n$. Does $\boldsymbol{Ax} = \boldsymbol{b}$ have a unique solution? One strategy to solve such equations is to minimize the error term $||\boldsymbol{Ax} - \boldsymbol{b}||$. Write down a quadratic minimization problem to minimize this error term. When does it have a unique solution?

4. **(5 points)** Query the oracle for an $m \times n$ matrix $\boldsymbol{A}$ with $m > n$ and a $m \times 1$ vector $\boldsymbol{b}$. Find an $\boldsymbol{x}^*$ that minimizes the error term $||\boldsymbol{Ax} - \boldsymbol{b}||$. Report the optimal $\boldsymbol{x}^*$ and the number of iterations it took to reach the optimum.

Oracle

Import the oracle `f1`. The arguments to be passed (strictly in the given order) to the function are:

1. `srno`: the last five digits of your serial number passed as an integer, and,

2. `subq`: a True or False value passed as a Boolean variable.

The oracle returns a matrix and a vector depending on the value of `srno` and `subq` as follows:

1. True: the oracle will return a symmetric, PD matrix $\boldsymbol{A}$ and an appropriate vector $\boldsymbol{b}$ for subquestion 2.

2. False: the oracle will return a $m \times n$ matrix $\boldsymbol{A}$ and an $m \times 1$ vector $\boldsymbol{b}$ for subquestion 4.

2 Newton's Method (10 points)

As discussed in class, the Newton-update is given by:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \boldsymbol{H}(\boldsymbol{x}^{(t)})^{-1} \nabla f(\boldsymbol{x}^{(t)}),$$

where, $\nabla f(\boldsymbol{x})$ and $\boldsymbol{H}(\boldsymbol{x})$ are the gradient and Hessian of a function $f : \mathbb{R}^d \to \mathbb{R}$ at the point $\boldsymbol{x}$, respectively. Write code to implement Newton's method to minimise an unconstrained objective function. This code-snippet will henceforth be referred to as `Newton` in the assignment.

Now, import the oracle `f2`, which encodes a function $f : \mathbb{R}^5 \to \mathbb{R}$. The arguments to be passed (strictly in the gien order) to the function are:

1. $\boldsymbol{x} \in \mathbb{R}^5$: a 1×5 numpy array,

2. `srno`: the last five digits of your serial number passed as an integer, and,

3. `order` $\in \{0, 1, 2\}$.

The oracle returns the function-value at $\boldsymbol{x}$ if the `order` is 0, $\nabla f(\boldsymbol{x})$ if the `order` is 1, and $\boldsymbol{H}(\boldsymbol{x})^{-1}\nabla f(\boldsymbol{x})$ if the `order` is 2. Also, let $\boldsymbol{x}_0 = [0,0,0,0,0]^\top$.

1. **(2 points)** Run gradient descent (implemented in Assignment 1) for 100 iterations with five choices of step-sizes with $\boldsymbol{x}_0$ as the initial point. Plot the function value at each iteration for each choice of step-size. Report $\boldsymbol{x}$ at the final iteration.

2. **(2 points)** Execute `Newton` for 100 iterations starting from $\boldsymbol{x}_0$, and plot the function values at each iteration. Compare the results with those from gradient descent. Report $\boldsymbol{x}$ at the final iteration.

3. **(3 points)** Execute `Newton` for 100 iterations starting from five different initial points, and plot the function values at each iteration. What do you observe? There are two important observations that can be made.

4. **(3 points)** Based on the above observation, guess the nature of the function in the oracle, and prove that the observation holds for such functions.

3 Newton's Method continued (15 points)

For this problem, use the oracle `f3`, which takes the same arguments as `f2` and returns values in a similar format. Also, for all of the following subparts, start with $\boldsymbol{x}^{(0)} = [1,1,1,1,1]^\top$ as the initial point.

1. **(2 points)** Run gradient descent with a step-size of 0.1 for a 100 iterations and plot the function-values. Report the best function value obtained over all the iterations.

2. **(2 points)** Did you observe oscillations in the above plot? Try to guess possible reasons for this phenomenon, and suggest ways to overcome them.

3. **(3 points)** Execute `Newton` for a 100 iterations, and report the function-values for only the first 10 iterations. Did the algorithm converge? If not, why?

4. **(8 points)** Now, let's play a game! Note that the cost of calculating the Hessian is often close to $\mathcal{O}(d^2)$ for a function in $\mathbb{R}^d$. And, to make matters worse, Newton's method necessitates the inversion of the Hessian, which is in $\mathcal{O}(d^3)$. Since $d = 5$ for this problem, the cost of each Newton-update is about $25\times$ that of a gradient-update! To circumvent the high cost of Newton's method, and to avoid issues with its convergence, a common practice is to run gradient-descent for K iterations, followed by the use of Newton's method to take advantage of the quadratic convergence once the iterates are in the vicinity of a (local) minimum.

   Replicate the aforesaid strategy by running a total of 100 iterations, of which $K \leq 100$ iterations are gradient-based, and the rest are Newton-based. Let each gradient-update have a unit-cost, and consequently, the cost of each Newton-update is 25 units. Experiment with different values of step-sizes and K, and track the total cost of the optimisation procedure. Report the least observed cost, the corresponding function-value and $\boldsymbol{x}$ at the last iteration for that value of K. Also, plot the function-values at each iteration, denoting each gradient-step with a blue dot (●), and each Newton-step with a red-coloured cross (×). One may even interleave the gradient and Newton updates, if that leads to lower costs.

4 Quasi-Newton Methods (10 points)

In this question, we will explore Quasi-Newton methods. Any approximation of the Hessian in the secant approximation can be considered a Quasi-Newton method.

1. **(5 points)** There may be cases where you are severely constrained by memory and you can only store one extra variable apart from two values of $\boldsymbol{x}$ and two values of $\nabla f(\boldsymbol{x})$. A good idea here is to approximate the Hessian by a scalar multiple of the identity matrix. Derive an update expression for this approach. You may find two solutions.

2. **(5 points)** Use the oracle from question 2. Compare your quasi-Newton solutions from the previous part with gradient descent (try a few different step sizes) and the quasi-Newton rank 1 update as discussed in class. Plot the values of f for 100 iterations for these methods. Report $\boldsymbol{x}^*$ obtained by each method.