CS240-Week-1: Linear programs and Quadratic programs

Instructions

- This lab intends to make you comfortable with the use of solvers for optimization problems, particularly, the use of `PuLP` and `cvxopt` libraries. Additionally, it also reinforces the understanding of the classic simplex method covered in class.
- This lab will be **graded**.
- Please read the problem statement and the submission guidelines carefully.
- All code fragments need to be written within the *%% Student Code Start* and *%% Student Code End* blocks in the given Python files. Do not change any other part of the code.
- **Do not** use any *print* statements in the final submission since the submission will be evaluated automatically.
- For any doubts or questions, please contact either the TA assigned to your lab group or one of the 2 TAs involved in making the lab.
- The submission will be evaluated automatically, so stick to the naming conventions and follow the directory structure strictly.
- The deadline for this lab is **15th January, 5 PM**.
- The submissions will be checked for plagiarism, and any form of cheating will be appropriately penalized.

The directory structure should be as follows (nothing more nothing less). **Since your submission will go through an automated grading tool, if these structures are not properly maintained, it may not be graded properly (even if the program is correct)**.

```
cs240_rollno_lab1 /
     |- q1 /
         |- q1_pulp.py
         |- q1_simplex.py
     |- q2 /
         |- q2.py
```

After creating your directory, package it into a tarball: **cs240_rollno_lab1.tar.gz**

Command to generate tarball:
`tar -czvf cs240_rollno_lab1.tar.gz cs240_rollno_lab1/`

1 Question 1: Linear programs [60 + 20 = 80 marks]

Implement solvers to solve linear programs of the form:

$$\begin{aligned} &\max\ c^\top x \\ &\text{subject to:} \\ &Ax \leq b \\ &x \geq 0 \end{aligned}$$

- First, implement simplex method covered in class from scratch to solve the LP. Refer simplex-tableau for a recap of the algorithm.
- Second, explore and make use pulp python package utilities to solve the LP. `PuLP` is a python package that provides utilities (classes, objects, and functions) to solve optimization problems, particularly LPs. Refer pulp-examples, pulp-classes for examples and functionalities.

1.1 Code

You are given two python files **q1_pulp.py**, **q1_simplex.py** and test LPs (folders: **test_case_1**, **test_case_2**).

- **Test LPs**: The two test LPs can be used to test your solver's outputs. Each folder contains input files ***A.csv, b.csv,*** *and* ***c.csv*** which describe an LP in the standard form above. These inputs will be used by your solvers. The ***solution.txt*** contains the expected outputs for your verification purposes.
  Note: Your code will be tested on a different set of test LPs which are not provided here. Any hard coding will result in zero marks. Your implementation must be able to solve any LP in the given standard form.
- **q1_pulp.py**:
  1. This python file takes a command line parameter - - testDirectory which indicates the directory where the input files for your test LP resides.
  2. To run the python file the following command can be used. Make sure you're in the same directory as the python file, otherwise change the path appropriately.

```
>python q1_pulp.py --testDirectory test_case_1
```

Figure 1: Command q1_pulp.py

  3. It reads the A matrix and b, c vector that describes the test LP (from path provided by the user) as a numpy array and solves the LP by calling pulp_solver() function.
  4. **Your task is to complete this function pulp_solver() appropriately.**
     - **Input: (A_matrix: numpy.array, c: numpy.array, b: numpy.array)**
     - **Return: The function must return the solution x^* (as a numpy array), and optimal value (as a float). See code for details, do not use any additional print statements.**
  5. All code fragments need to be written within the *%% Student Code Start* and *%% Student Code End* blocks in the given Python files. Do not change any other part of the code.
- **q1_simplex.py**:
  1. The instructions to run this file are similar as follows.

```
>python q1_simplex.py --testDirectory test_case_1
```

Figure 2: Command q1_simplex.py

2. It reads the A matrix and b, c vector that describes the test LP (from path provided by the user) as a numpy array and solves the LP by calling simplex_solver() function.
3. **Your task is to complete this function simplex_solver() appropriately.**
   - **Input: (A_matrix: numpy.array, c: numpy.array, b: numpy.array)**
   - Write your program such that it solves the LP with simplex method using the tableaus as intermediate steps. This question is asking you to return some specific values of the tableaus as described below.
   - The column corresponding to the non-basic variable that turns into a basic variable in an iteration is called the **pivot column**. Similarly, the row corresponding to the basic variable that turns into a non-basic variable in an iteration is called the **pivot row**. The unnormalized value corresponding to the pivot row and column is called the pivot value.
   - **Return: The function must return the pivot values (as a list) encountered in each iteration of the simplex method (in that order). See code for details, do not use any additional print statements.**
4. All code fragments need to be written within the *%% Student Code Start* and *%% Student Code End* blocks in the given Python files. Do not change any other part of the code.

2 Question 2: Quadratic programs [20 Marks]

- Consider a given matrix X (2D points) and a target vector y. On this given data, the goal of a least square fit is to find a vector w such that

$$\sum_{i=1}^{n}(x_{i1} \cdot w_1 + x_{i2} \cdot w_2 - y_i)^2$$

is minimized, where n is the number of given data points. Note that it can be written as an unconstrained optimization problem of the form

$$\min \ \frac{1}{2}(Xw - y)^\top (Xw - y) \tag{1}$$

- The standard form of a quadratic program (QP) is given by the following

$$\begin{aligned} &\min \frac{1}{2}x^\top Px + q^\top x \\ &\text{subject to:} \\ &Gx \leq h \\ &Ax = b \end{aligned} \tag{2}$$

where P, G, and A are matrices and q, b, h are vectors. The vector x denotes the vector of variables in the QP.

- Convert the given least square fit problem (Equation (1)) to a quadratic programming problem (Equation (2)) and solve it using `cvxopt`. Examples and documentation can be found at cvxopt, quadratic programming.

2.1 Code

You are given **q2.py** and a test dataset (folder: **QP_test_case**).

- **Test case:** The test dataset can be used to test your output. It contains two files ***X.csv*** and ***Y.csv*** where X is the input matrix and Y is the target vector. The ***solution.txt*** contains the expected output w^* for your verification purposes.
  Note: Your code will be tested on a different set of test datasets which are not provided here. Any hard coding will result in zero marks.

- **q2.py:**

1. This python file takes a command line parameter `--testDirectory` which indicates the directory where the input files for your test QP resides.

```
$ python q2.py --testDirectory QP_test_case
```

Figure 3: Command q2.py

2. To run the python file the following command can be used. Make sure you're in the same directory as the python file, otherwise change the path appropriately.
3. It reads the X and Y matrices and will solve the QP by calling **QP_solver()** function.
4. **Your task is to complete this function QP_solver() appropriately.**
   - **Input: (X: numpy.array, Y: numpy.array)**
   - Convert the least squares problem into a quadratic program and use appropriate functions from cvxopt to solve the QP.
   - **Return: The function must return the solution of the QP (i.e., the solution vector w^*) as a numpy array. See code for details, do not use any additional print statements.**
5. All code fragments need to be written within the *%% Student Code Start* and *%% Student Code End* blocks in the given Python files. Do not change any other part of the code.
6. Once your QP has been solved a plot will automatically be generated of the least square fit curve and the original data points.

Note: The data used for Figure 4 is different from the test case provided.

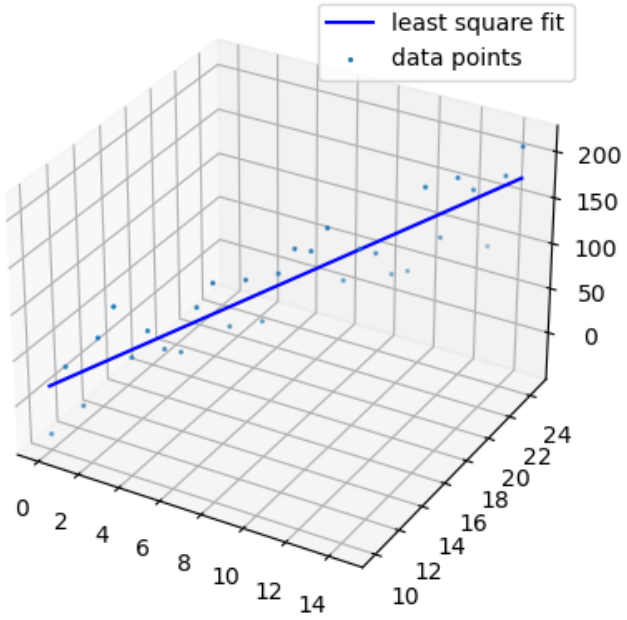

Figure 4: Comparison of original data and least square fit