ECE 506: Homework #1: Basic Optimization Math

To get help with the homework, please join me for the Saturday morning discussion sessions starting at 9am at https://unm.zoom.us/j/99977790315.

Problem #1. An Introduction to Linear Programming.
This problem is focused on manipulating the basic Linear Programming equation:

$$\min_x \; c^T x \quad \text{subject to:} \quad Ax = b \quad \text{and} \quad x \geq 0. \tag{1}$$

1(a) We begin with the simplest possible example! Consider the 1D problem:

$$\min_x \; c \cdot x \quad \text{subject to:} \quad ax = b \quad \text{and} \quad x \geq 0. \tag{2}$$

From this case, answer the following:

i. Give an example where there is no solution.

ii. Give an example with a simple solution.

iii. For your solution, did you minimize anything? Explain.

1(b) More generally, consider $Ax = b$ for many dimensions. Suppose that A is invertible. In this case, show that there is no minimization! To show this, compute the solution without minimizing $c^T x$.

1(c) The only case that is interesting is when we have many solutions to $Ax = b$. We then get to pick the one that minimizes $c^T x$. This can only happen when the number of equations is smaller than the number of unknowns. Here is an example:

$$\begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \end{bmatrix}.$$

Note that we have one equation in two unknowns. **We have more unknowns than we have equations!** It may be possible to set up a proper optimization problem!

To have a proper solution, we must also satisfy: $x_1, x_2 \geq 0$. These are called **feasible** solutions. They satisfy the constraints, and the optimal solution needs to satisfy them.

Plot all possible solutions of $Ax = b$ satisfying $x_1, x_2 \geq 0$ for this case.

1(d) For the case when $Ax = b$ described in 1(c), solve the proper optimization problem. For this case, solve:

$$\min_x \; [1,\, 1]\, x \quad \text{subject to:} \quad Ax = b \quad \text{and} \quad x \geq 0. \tag{3}$$

Is the solution at the endpoints? Explain.

Problem #2. Generalizing Linear Programming for inequalities. The goal of this problem is to expand our discussion in problem # 1 and connect it to software that solves linear programming problems.

2(a) Consider the following inequality in 1D:

$$l_b \leq x_1 \leq u_b \tag{4}$$

We break it into two inequalities:

$$l_b \leq x_1 \quad \text{and} \quad x_1 \leq u_b.$$

Based on the original formulation of problem #1, we only allow non-negative variables. Thus, here, we introduce non-negative variables to convert the inequalities:

$$y_1 = x_1 - l_b \quad \text{and} \quad y_2 = u_b - x_1.$$

i. Show that for **feasible** solutions, we have $y_1, y_2 \geq 0$.

ii. We also need to allow x_1 to be any real number! For this problem, the key is to view x_1 as two positive variables. The relationship is as follows:

$$x_1^+ = \max(0, x) = \text{ReLU}(x_1) \tag{5}$$
$$x_1^- = \max(0, -x) = \text{ReLU}(-x_1) \tag{6}$$

A. Give three examples for determining x^+ and x^- from x.

B. Show that both x^+ and x^- are non-negative.

C. Derive an expression for determining x from x^+ and x^-.

iii. Set a 4D $x = [y_1,\, y_2,\, x_1^+,\, x_1^-]^T$. Reformulate the problem in the standard form:

$$\min_x \; c^T x \quad \text{subject to:} \quad Ax = b \quad \text{and} \quad x \geq 0. \tag{7}$$

Here, assume that c is given to you and A is derived from $l_b \leq x_1 \leq u_b$.

iv. Show the general N-D form of the problem for constraints of the type:

$$\begin{aligned} l_1 &\leq x_1 \leq u_1 \\ l_2 &\leq x_2 \leq u_n \\ &\vdots \\ l_n &\leq x_n \leq u_n \end{aligned}$$

2(b) We can generalize $Ax = b$ to handle inequalities and arbitrary values. Let us start with the 1D case. Suppose that we want to formulate the problem:

$$\min_{x_1} \; c \cdot x_1 \quad \text{subject to:} \quad ax_1 \leq b. \tag{8}$$

We again set:

$$x_1^+ = \max(0, x) = \text{ReLU}(x) \tag{9}$$
$$x_1^- = \max(0, -x) = \text{ReLU}(-x) \tag{10}$$

We can consider any real x value based on the following process.

i. Rewrite $ax \leq b$ and cx in terms of x^+ and x^-.

ii. Set $x = [x_1^+, x_1^-]^T$. Reformulate the problem in the standard form:

$$\min_{x} \; c^T x \quad \text{subject to:} \quad Ax = b \quad \text{and} \quad x \geq 0. \tag{11}$$

Here, assume that c is given to you and A is derived from $ax_1 \leq b$.

iii. Reformulate the standard problem for the case where $Ax \leq b$ where x is n-dimensional.

Problem #3. Visualizing and solving 2D Linear Programming Problems Using 2D Plots and Neural Networks

3(a) Consider a modified version of the inequalities:

$$Ax + b \geq 0. \tag{12}$$

This equation does not appear to be in the right form for using the Linear Programming software. We need to transform the equation following the example of problem # 2 so that $A'x \leq b'$. Reformulate ((**??**)) by defining A' and b' so as to satisfy $A'x \leq b'$.

Hint: Note that multiplying both sides of an inequality $a \leq b$ by -1 will reverse the order to: $-a \geq -b$.

3(b) Define y as follows:

$$y = Ax + b. \tag{13}$$

Now, answer the following:

i. Compute an expression for $c^T y$ in terms of x.

ii. Derive an expression for c' so that the following problems are equivalent:

$$\min_y \; c^T y \quad \text{is the same as} \quad \min_x \; c'^T x. \tag{14}$$

3(c) The equations described in 3(b) can be simulated with a neural network structure that uses:

Implement $Ax + b$ using a linear layer and `bias=b`.
Select `ReLU` activation function for the linear layer.
Implement the second layer using $c^T y$.

Visualize the interface given by this Linear Program formulation through the interactive interface given at Neural Network playground. Of-course, in this case, our Neural Network does not perform the minimization :-(. Attach a screenshot that shows the plots.

3(d) In this problem, we want to consider optimizing the output in a two-layer Neural Network. The goal here is to demonstrate the use of Linear Programs in practice.

Building on our discussion for 3(c), consider the following system:

$$\min_y \; [1 \quad 2]^T y \quad \text{subject to:} \begin{bmatrix} -1 & +1 \\ 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

where:

$$y = \begin{bmatrix} -1 & +1 \\ 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

For this problem, answer the following:

i. Compute the solution region for the inequalities. Sketch your answer.

ii. List the corners of the line intersections from i.

iii. Calculate $[1 \quad 2]^T y$ at the corner points.

iv. What is the minimum value of y at the corner points?

Note: If there is a solution to LP, it will always be at one of the corner points.

3(e) Confirm your answer in 3(d) using `linprog` from linprog documentation.

Problem #4. Basics of the use of the l_1 norm.
4(a) Consider the following constraint on the l_1 norm:

$$\sum_{i=1}^{i=n} |x_i| \leq 1. \tag{15}$$

Draw the feasible region that satisfies the constraint for $n = 2$.

4(b) Restate (**??**) in standard form using: $c_i(x)$, $i = 1, 2, \ldots, 4$.

4(c) Consider the general case where

$$\sum_{i=1}^{i=n} |x_i| \leq M \tag{16}$$

where $M < n$ and i is an integer.

1. List the corner points.
2. For each corner point, show the count of the non-zero entries.

Hint: Corner points satisfy $x_i = 0$ for some indices. For example, in 2D, we have that $x_1 = 0, x_2 = 1$ is a corner point with one non-zero entry.

4(d) Restate (**??**) using $c_i(x)$ for $n = 3$. How many $c_i(x)$ constraints do you need for arbitrary n? Why do you think that the use of the l_1-norm requires the use of approximations of the constraints?

The l_1 norm can be used to solve tough combinatorial optimization problems such as minimizing the number of connections in Neural Networks, compressive sensing, minimizing the number of factors used in Statistical models, etc.

Problem #5. Unconstrained optimization.
Consider the following ideal, convex function:

$$f(x_1, x_2) = (x_1 - 2)^2 + 10 \cdot (x_2 - 3)^2 \tag{17}$$

1. Plot the contours of the function and its gradient.
2. Compute the point where $\nabla f(x) = 0$.

3. Solve the unconstrained optimization problem:

$$\min_x \quad f(x_1, x_2). \tag{18}$$

4. Verify $\nabla f(x) = 0$ at the optimal point.

Problem #6. Constrained optimization.
Consider the following ideal, convex optimization problem:

$$\min_x \quad f(x_1, x_2) \tag{19}$$

subject to:

$$\sum_{i=1}^{i=n} |x_i| \leq 1.$$

6(a) Let $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 4)^2$.

1. Plot the contours of the function and its constraints.
2. Solve the unconstrained optimization problem:

$$\min_x \quad f(x_1, x_2). \tag{20}$$

3. Solve the constraint optimization problem by finding the point in the line constraint that is closest to the unconstrained optimal point.

6(b) Let $f(x_1, x_2) = (x_1 - 5)^2 + x_2^2$.

1. Plot the contours of the function and its constraints.
2. Solve the unconstrained optimization problem:

$$\min_x \quad f(x_1, x_2). \tag{21}$$

3. Solve the constraint optimization problem by finding the point that is closest to the unconstrained optimal point.

6(c) Let $f(x_1, x_2) = (x_1 - 0.5)^2 + x_2^2$.

1. Plot the contours of the function and its constraints.
2. Solve the unconstrained optimization problem:

$$\min_x \quad f(x_1, x_2). \tag{22}$$

3. Based on your contour-plot show that the optimal point cannot be on the boundary.

6(d) Vertify that the KKT conditions are satisfied at the optimal points for 6(a), 6(b), and 6(c).

KKT Summary: In constrained-optimization, the KKT conditions are always satisfied at the optimal point. Note that this is not enough to determine if a point is optimal. If the KKT conditions are satisfied, we cannot infer that we have an optimal point. To understand the KKT conditions, we need to apply them. Here is an intuitive summary of how to apply and verify the KKT conditions:

```
IF (the solution is at an interior point)
   THEN The gradient of f should be zero at that point.

IF (the solution is on a boundary point) AND (LICQ holds there)
  THEN
     The Lagrangian condition is satisfied at that point.
```

To understand the KKT conditions, note that they only tell you when you can reject a candidate optimal point. To see how to do this, let us review the elements of a contra-positive proof. Suppose that we have:

```
IF A THEN B
```

Then the statement is equivalent to:

```
IF (NOT B) THEN (NOT A)
```

In other words, since A implies B, if B does not hold, it must be that A does not hold either. If we apply this statement to the first statement, we have that if the gradient of f is not zero in the interior, we can conclude that the optimal solution is not in the interior. If we apply this statement to the second statement, we then have that if the Lagrangian condition is not satisfied, we then have that either the solution is not on the boundary or the LICQ does not hold.

If the KKT conditions are satisfied, then we MAY be at an optimal point. There are no guarantees that we will be optimal. Back in our example, if A implies B, we cannot say that B implies A.

Next, we explain each condition carefully. For the first condition, if x^* is inside the feasible region, we require that $\nabla f(x^*) = 0$. Again, this is not enough to guarantee that we are at an optimal point. However, if this condition is violated, we are clearly not at an optimal point in the interior.

If the solution is not inside the feasible region, then it could be on the boundary of the feasible region. In this case, we need to first check the LICQ condition before we look at the full KKT conditions.

For the LICQ condition, consider the vectors evaluated at the candidate optimal point. Here, we are only concerned with the constraints that are **active**. We use the term **active** to describe the constraints that satisfy $c_i(x^*) = 0$. Thus, if a solution is on a line, the $c_i(x)$ that gives the equation of the line will satisfy $c_i(x^*) = 0$. For a corner, we will have the $c_i(x^*) = c_j(x^*) = 0$ where c_i and c_j represent each intersecting line. The LICQ requires that $\nabla c_i(x^*)$ from the active constraints remain **linearly independent**. To prove LICQ, you need to show that

$$\sum_{\textbf{Active } i} a_i \nabla c_i(x^*) = 0 \tag{23}$$

is only satisfied if $a_i = 0$ for all i. Intuitively, we say that the **active** constraint gradients are independent from each other. In other words, the constraints pull you into different directions inside the boundary.

To check the Lagrangian condition, we form the Lagrangian:

$$\mathcal{L}(x, \lambda) = f - \sum_{\textbf{Active } i} \lambda_i c_i(x). \tag{24}$$

The Lagrangian condition requires that we can solve

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = 0 \tag{25}$$

at the optimal point x^* for some $\lambda_i^* \geq 0$.

It is important to understand what (**??**) is saying. Basically, it is saying that going inside the feasible region will only increase the function. Note that ∇f is the direction where the function is increasing. Furthermore, the solution implies that

$$\nabla f(x^*) - \sum_{\textbf{Active } i} \lambda_i^* \nabla c_i(x^*) = 0. \tag{26}$$

In turn, this implies that:

$$-\nabla f(x^*) = - \sum_{\textbf{Active } i} \lambda_i^* \nabla c_i(x^*). \tag{27}$$

Thus, the descent direction of $-\nabla f(x^*)$ will move us outside the feasible region.

Contour and gradient plots in Python

For information on how to plot contours, the following links can help a lot:

- A simple unofficial tutorial on how to generate contours in Python is given at `https://www.python-course.eu/matplotlib_contour_plot.php`.
- The official tutorial of how to plot 2D contours from regularly-spaced points.
- The official tutorial of how to plot 2D contours from irregularly-spaced points (needed later).

- An official advanced tutorial for using advanced options for contour plots can be found at advanced contour tutorial.
- An official simple demo of how to use `quiver` to plot gradient fields.
- In Matplotlib, you can plot the contours followed by the gradient and they will appear together. In other words, this is equivalent to having `hold on` as the default behavior.

Advanced Demo for Optimization Methods in Python
You can find a very nice demonstration of how to produce convergence videos of several unconstrained optimization algorithms.