Math 482: Linear Programming[1] Mikhail Lavrov

Lecture 1: What Is a Linear Program?

January 22, 2020 *University of Illinois at Urbana-Champaign*

1 Useful information about the class

See the course webpage (in the footnote at the bottom of this page) for information about when things happen, how grading works, and so forth.

- The first homework assignment is due Friday, January 31st in class. The first exam will be in the evening on Wednesday, February 19th. You can register for the 4-credit version of the class within the first eight weeks of the semester.
- Homework grades and solutions will be posted on Moodle.
- Notes for each lecture will be posted on the course webpage. (But consider taking your own notes, as well.)
- Office hours are Monday, Wednesday, and Friday after class (11:00am to 11:50am).
- There is an official textbook for this class, and several other sources you can access through the university library; none of them are required, but all of them might be helpful.

  Watch out: each of these textbooks uses its own notation for just about everything. Feel free to ask me about what a textbook is trying to say.

2 Linear programs and optimization

"Linear programming" suggests telling a computer to do linear stuff. But that's not what the word "programming" means in the title of this class. In this context, "programming" means "optimization": finding a point in a set that maximizes the value of a function.

Here is an example of a linear program:

$$\begin{aligned} \underset{x,y\in\mathbb{R}}{\text{maximize}} \quad & x - y \\ \text{subject to} \quad & y \leq 3, \\ & y \geq 2x - 5, \\ & x + y \geq 1. \end{aligned}$$

2.1 The feasible region

The linear program has a set of *constraints*: $y \leq 3$, $y \geq 2x - 5$, and $x + y \geq 1$. In general, we will allow our constraints to be linear inequalities or linear equations.

[1] This document comes from the Math 482 course webpage: https://faculty.math.illinois.edu/~mlavrov/courses/482-spring-2020.html

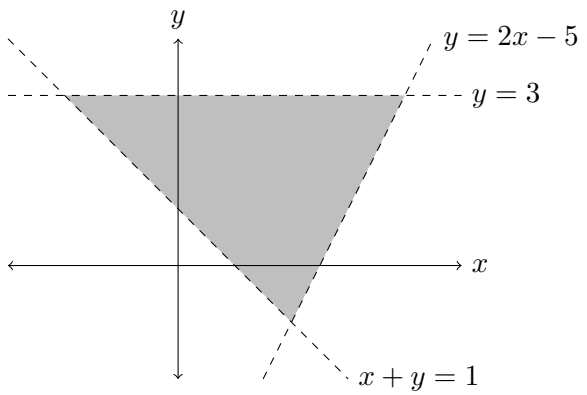
After a bit of rearrangement, we could write down these four inequalities in matrix form:

$$\begin{bmatrix} 0 & 1 \\ 2 & -1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} 3 \\ 5 \\ -1 \end{bmatrix}.$$

Here, if $\mathbf{a}, \mathbf{b}$ are two vectors in $\mathbb{R}^n$, $\mathbf{a} \leq \mathbf{b}$ means that $a_i \leq b_i$ for every $i = 1, 2, \ldots, n$.

In general, if a linear program has variables $\mathbf{x} \in \mathbb{R}^n$ (that is, variables $x_1, x_2, \ldots, x_n \in \mathbb{R}$), then we can write down the constraints as a matrix inequality $A\mathbf{x} \leq \mathbf{b}$, where A is an $m \times n$ matrix, and $\mathbf{b}$ is an $m \times 1$ vector.

The set $\{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} \leq \mathbf{b}\}$ of all $\mathbf{x}$ which satisfy the constraints is called the *feasible region*. A point $\mathbf{x}$ in the feasible region is called a *feasible solution*. (This is weird terminology, since we haven't solved the problem at this point!) In our example, the feasible region is the following region in the plane:



2.2 The objective function

The linear program also has an *objective function*: we want to maximize $x - y$. In general, we might be maximizing or minimizing an arbitrary linear function. In terms of variables $\mathbf{x} \in \mathbb{R}^n$, we can write the objective function as $\mathbf{c}^\mathsf{T}\mathbf{x}$ for some constant vector $\mathbf{c} \in \mathbb{R}^n$.

As a result, any linear program can be written as

$$\begin{aligned} \underset{\mathbf{x} \in \mathbb{R}^n}{\text{maximize}} \quad & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b}. \end{aligned}$$

What about minimizing? Well, minimizing $\mathbf{c}^\mathsf{T}\mathbf{x}$ would be the same as maximizing its negative $(-\mathbf{c})^\mathsf{T}\mathbf{x}$. We will encounter both kinds of linear programs in class, but we don't lose any generality by focusing on one kind whenever it's convenient.

Whether we're minimizing or maximizing, a point $\mathbf{x} \in \mathbb{R}^n$ with the best value of the objective function is called an *optimal solution*. In our example, the point $(x, y) = (2, -1)$ is the unique optimal solution.

2.3 Misbehaving linear programs

In the previous example, our linear program had a unique optimal solution, but this is not always guaranteed to be true.

The optimal solution might not be unique.

Suppose that we have the same constraints, but we try to maximize y instead. By going as far in the y direction as possible, we run into the line $y = 3$. Any feasible point on this line is an optimal solution, so there are infinitely many.

(In higher dimensions, this can happen for less trivial reasons than "we're trying to optimize a linear function, and one of the constraints puts an upper bound on that exact function.")

The optimal solution might not exist, because the objective function can be arbitrarily large.

Suppose that we forget the constraint $y \geq 2x - 5$, and just try to solve

$$\begin{aligned} \underset{x,y \in \mathbb{R}}{\text{maximize}} \quad & x - y \\ \text{subject to} \quad & y \leq 3, \\ & x + y \geq 1. \end{aligned}$$

Here, the point $(100, 0)$ satisfies both conditions, and has $x - y = 100$. But $(1000, 0)$ is even better: it has $x - y = 1000$. There is no optimal solution, because we can keep going in the positive x direction for as long as we like, and make $x - y$ as large as we want.

We call a linear program like this *unbounded*.

The optimal solution might not exist, because there are no feasible solutions.

Suppose we change the constraint $y \leq 3$ in our original program to be $y \leq -3$:

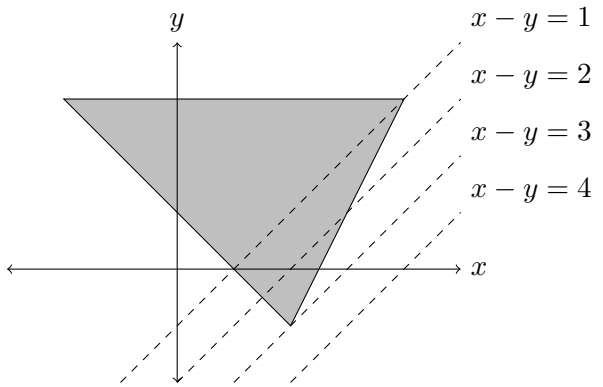
$$\begin{aligned} \underset{x,y \in \mathbb{R}}{\text{maximize}} \quad & x - y \\ \text{subject to} \quad & y \leq -3, \\ & y \geq 2x - 5, \\ & x + y \geq 1. \end{aligned}$$

Here, there are no solutions at all! In the original feasible region, the lowest point is the corner $(2, -1)$ where $y = -1$. So asking for $y \leq -3$ eliminates every single point. There is no optimal solution, because it's impossible to satisfy all the constraints.

We call a linear program like this *inconsistent*.

3 The naive approach to solving linear programs

Let's go back to the original linear program. Here are some possible values of the objective function $x - y$, and the places where they occur:



Pick a small value of $x - y$ (such as 1) and the feasible points with that value of $x - y$ are a line segment. Pick a large value of $x - y$ (such as 4) and there are no feasible points with that value of $x - y$. But when $x - y = 3$, just before the value becomes impossible, the segment shrinks to a single point: a corner of the feasible region.

(Formally, a corner is called a *vertex* or *extreme point*, depending on who you ask.)

By combining any two of the equations for the edges of our feasible region, we can solve for the coordinates of a vertex. We have $x+y = 1$ and $y = 3$ when $(x, y) = (-2, 3)$. We have $y = 2x-5$ and $y = 3$ when $(x, y) = (4, 3)$. Finally, we have $x + y = 1$ and $y = 2x - 5$ when $(x, y) = (2, -1)$.

(With more inequalities in the linear program, an extra step would be required: we'd have to also check that each of these solutions also satisfies the other inequalities we didn't use to solve it.)

Therefore we know ahead of time that one of the points

$$(-2, 3) \qquad (4, 3) \qquad (2, -1)$$

is going to be an optimal solution, and we can solve the linear program just by comparing their values of $x - y$.

This is the "naive" approach to solving linear programs. It's quick to explain, and for small examples, especially ones you can draw in the plane, it may be the easiest thing to do.

Imagine, however, that you have a linear program with 50 variables and 100 inequalities. (This is a "tiny" linear program: my computer solves one of these in approximately 0.03 seconds.) With the naive approach, there are $\binom{100}{50} = 100\,891\,344\,545\,564\,193\,334\,812\,497\,256$ combinations of 50 of the equations bounding the region. Each of these combinations (in general) intersects at a single point, so we need to compare that many points to find the best one.

So our goal in this class is going to be to try to do less work than this. Ahead of us is the simplex method, which starts at one vertex of a linear program, and moves from vertex to vertex until it finds the best one: hopefully long before it visits all the vertices.