LECTURE 3: LINEAR PROGRAMMING

1. Linear Programming

Recap: A mathematical programming (max/min) problem is **linear** if f and the constraints are linear.

We saw that any linear programming can be put in the following form:

$$\begin{aligned} \max z = & c^T \mathbf{x} \\ \text{subject to } & Ax \leq b \\ \text{and } & \mathbf{x} \geq 0 \end{aligned}$$

- c is the cost vector (think the price of each product)
- A is the constraint matrix
- b is the constraint vector

2. Standard Form

We can actually do better than that, and put a problem in **standard form**:

Rule 1: (Sign Rule)

Date: Thursday, September 15, 2022.

If a variable x_i has arbitrary sign, then we can write

$$x_i = (x_i)^+ - (x_i)^- \text{ , where } (x_i)^{\pm} \geq 0$$

For example, if $x_i = 3$ then $x_i = 3 - 0$ so $(x_i)^+ = 3$ and $(x_i)^- = 0$, but you can also write $x_i = 4 - 1$ so $(x_i)^+ = 4$ and $(x_i)^- = 1$ (so you can decompose x_i in multiple ways).

And if $x_i = -5$ then $x_i = 0 - 5$ so $(x_i)^+ = 0$ and $(x_i)^- = 5$ for example, but also $(x_i)^+ = 2$ and $(x_i)^- = 7$ would work.

Moral: It is actually ok to assume without loss of generality that all our variables are ≥ 0.

Rule 2: (Equality)

We can actually turn any inequality $A\mathbf{x} \leq b$ into an equality as follows:

Example 1: Suppose your constraint is given by

$$\begin{aligned} x_1 + 2x_2 &\leq 5 \\ 2x_1 + 3x_2 &\leq 8 \\ x_1 \geq 0, &x_2 \geq 0 \end{aligned}$$

Define new variables s_1 and s_2 called **slack variables** by

$$\begin{aligned} s_1 &= 5 - x_1 - 2x_2 \geq 0 \\ s_2 &= 8 - 2x_1 - 3x_2 \geq 0 \end{aligned}$$

Then by definition of s_1 and s_2, the constraints simply become

$$\begin{aligned} x_1 + 2x_2 + s_1 &= 5 \\ 2x_1 + 3x_2 + s_2 &= 8 \\ x_1, x_2, s_1, s_2 &\geq 0 \end{aligned}$$

So by redefining $\mathbf{x}$ and A to be

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \end{bmatrix} \quad A = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 2 & 3 & 0 & 1 \end{bmatrix}$$

The constraint now becomes $A\mathbf{x} = \mathbf{b}$

Example 2: (putting it all together)

Write the following problem in standard form:

$$\begin{aligned} \max z = & 2x_1 + 4x_2 \\ \text{subject to } & x_1 + x_2 \leq 3 \\ & 3x_1 + 2x_2 = 14 \\ & x_1 \geq 0 \end{aligned}$$

First of all, x_2 is of arbitrary sign, so write $x_2 = (x_2)^+ - (x_2)^-$.

Moreover, let $s_1 = 3 - x_1 - x_2$ then the problem becomes

$$\begin{aligned} \max z = & 2x_1 + 4(x_2)^+ - 4(x_2)^- \\ \text{subject to } & x_1 + (x_2)^+ - (x_2)^- + s_1 = 3 \\ & 3x_1 + 2(x_2)^+ - 2(x_2)^- = 14 \\ & x_1, (x_2)^-, (x_2)^+, s_1 \geq 0 \end{aligned}$$

Which you can write in matrix form as

$$\max z = c^T\mathbf{x}, \text{ subject to } A\mathbf{x} = b \text{ and } \mathbf{x} \geq 0$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ (x_2)^+ \\ (x_2)^- \\ s_1 \end{bmatrix}, \; c = \begin{bmatrix} 2 \\ 4 \\ -4 \\ 0 \end{bmatrix}, \; A = \begin{bmatrix} 1 & 1 & -1 & 1 \\ 3 & 2 & -2 & 0 \end{bmatrix}, \; b = \begin{bmatrix} 3 \\ 14 \end{bmatrix}$$

And if for example $\mathbf{x} = \begin{bmatrix} 8 \\ 1 \\ 6 \\ 0 \end{bmatrix}$ is a solution, then $x_1 = 8$ and $x_2 = (x_2)^+ - (x_2)^- = 1 - 6 = -5$ is a solution

And if $(x_1, x_2) = (4, -2)$ is a solution to the original problem, then for example $(x_2)^+ = 0$, $(x_2)^- = 2$ and $s_1 = 3 - x_1 - x_2 = 1 \geq 0$ and so $\mathbf{x} = \begin{bmatrix} 4 \\ 0 \\ 2 \\ 1 \end{bmatrix}$ is a solution to the new problem.

So we can really go back and forth between those two formulations.

3. Solving a linear programming problem

Let's now *finally* solve a linear programming problem!

Example 3:

Solve the following

$$\begin{aligned} \max z =& 3x_1 + 5x_2 \\ \text{subject to } & x_1 \leq 4 \\ & x_2 \leq 6 \\ & 3x_1 + 2x_2 \leq 18 \\ & x_1, x_2 \geq 0 \end{aligned}$$

STEP 1: Draw the feasible region

$x_1 = 4$ is a vertical line, $x_2 = 6$ is a horizontal line, and the last one is the diagonal line $x_2 = 9 - \frac{3}{2}x_1$

We then obtain the colored region as in the picture (on the next page)

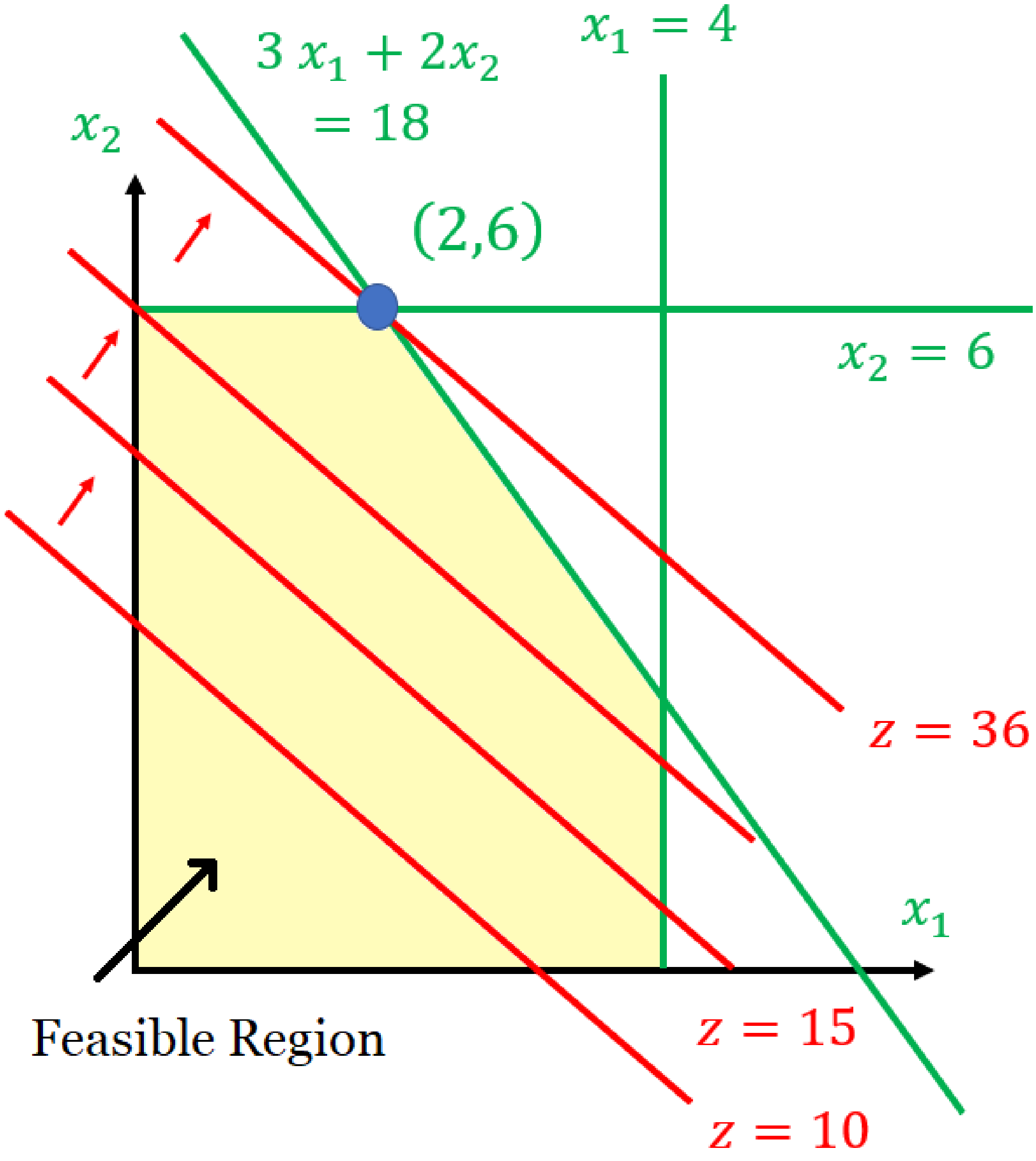
STEP 2: Motivation

Let's plot some level curves, that is curves of the form $z = k$ (for real numbers k)

For example, with $z = 10$, we get the line $3x_1 + 5x_2 = 10$, so $x_2 = -\frac{3}{5}x_1 + 2$. This line crosses the feasible region, so z is for sure at least 10.

Can we do better? Yes! Let's try some bigger z, say $z = 15$. Geometrically, what this does is move the line to the right (the x_2 intercept increases). It still crosses the feasible region!

And in fact, if you think about it, we can increase z until we reach a corner point/vertex of the feasible region.



So in fact here the max is obtained at $(2, 6)$ and the max value is

$$z = 3x_1 + 5x_2 = 3(2) + 5(6) = 36$$

Answer: $(x_1, x_2) = (2, 6)$ and $z = 36$

Fact:

If the feasible region is bounded and non-empty, then an optimal solution can always be found at a corner point

Note: The optimal solution might not be unique, could be attained at two different corner points for example.

Note: For unbounded regions, either the optimal solution is at a corner point, or z can be arbitrarily small or large (think $\pm\infty$)

Summary: There are four possible scenarios in linear programming

Case 1: Unique solution

Case 2: Two or more solutions

Case 3: No solution. This happens when the feasible region is empty

Case 4: The LP problem has arbitrarily large (or small) solutions. This happens for unbounded feasible regions

4. Linear Programming and Cookies

As a fun application, let's use linear programming to$\cdots$ bake cookies!![1]

Suppose you want to bake the perfect (optimal) cookie with the following nutrition facts:

Goal: Figure out how much to use of each ingredient.

Decision Variables:

[1] Thank you Caroline Klivans for the idea

$$
\begin{aligned}
x_1 &= \text{Number of Servings of Chocolate Chips} \\
x_2 &= \text{Number of Servings of Flour} \\
x_3 &= \text{Number of Servings of Butter} \\
&\vdots \\
x_9 &= \text{Number of Servings of Vanilla}
\end{aligned}
$$

Constraints: To find the constraints, let's look at the nutritional data for the ingredients

	Grams	Calories	Fat (g)	Sat. Fat (g)	Cholesterol (mg)	Sodium (mg)	Carbs (g)	Sugar (g)	Protein (g)
1 Cup Choc Chips	160	840	48	40	0	0	108	96	10
1 Cup Flour	125	455	0	0	0	0	95	0	13
1 Stick Butter	113	810	91.7	58	242.9	12.4	0.1	0.1	1
1 Cup Sugar	200	773	0	0	0	0	200	200	0
1 Cup Brown Sugar	200	773	0	0	0	0	200	200	0
1 Egg	53	78	5	1.6	186	62	0.4	0.2	6.3
1 Tsp Baking Soda	5	0	0	0	0	1258.6	0	0	0
1 Tsp Salt	5	0	0	0	0	2325	0	0	0
1 Tsp Vanilla	4.2	10	0	0	0	0	0	0	0
1 Serving Size	28	140	7	4.5	25	150	18	12	2

Each ingredient has to be in decreasing order in terms of weights, and so we get the constraint

$$160x_1 \geq 125x_2 \geq 113x_3 \geq \cdots \geq 4.2x_9 \geq 0$$

Moreover, the total weight has to be 28 grams, the total calories has to be 140, which gives

$$
\begin{aligned}
160x_1 + 125x_2 + \cdots + 4.2x_9 &= 28 \\
840x_1 + 455x_2 + \cdots + 10x_9 &= 140 \\
&\vdots \\
10x_1 + \cdots + 0x_9 &= 2
\end{aligned}
$$

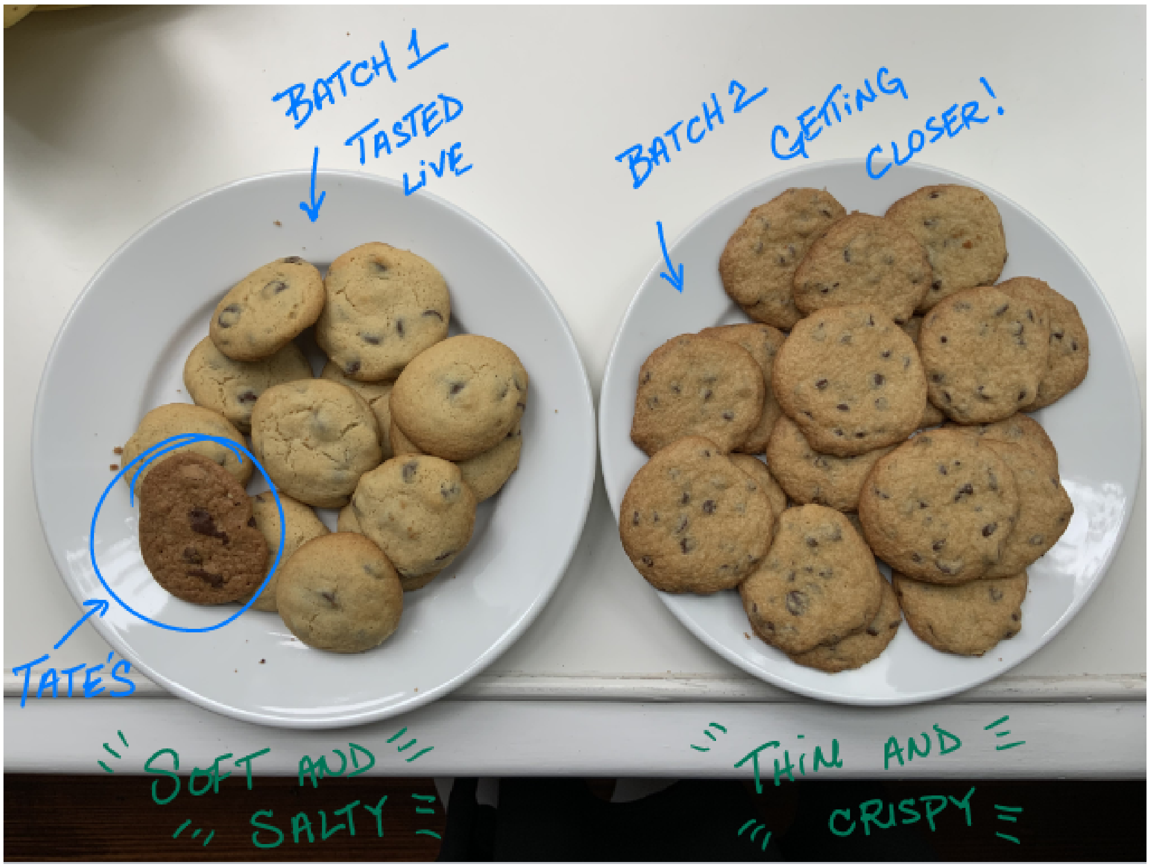
Objective Function: None actually, we just want one point in the feasible region

But here's the catch: If you solve this problem as it is, you will find that the feasible region is empty, so there would be no solution. To get around this, relax the constraint a little bit. For example, instead of requiring the second constraint to be *exactly* equal to 28, just require it to be between 26 and 32 (it'll still be a delicious cookie, I promise)

If you then implement the simplex method (see later) to solve this problem, then you get that the optimal proportions are:

Chocolate Chips	1.563 Cups
Flour	2 Cups
Butter	1.885 Sticks
Sugar	0.756 Cups
Brown Sugar	0.756 Cups
Eggs	1.502 Egg
Baking Soda	1.329 Tsp
Salt	1.328 Tsp
Vanilla	1.582 Tsp

And as a result, you get some wonderfully tasty cookies ☺



5. Appendix: Python Code

Here is the Python Code for the previous problem, in case you're interested

```
import numpy as np
import picos as pic

def recipe_lp(label,ingredients,eps=0.1):
```

```
n = len(ingredients[0])
m = len(ingredients)

model = pic.Problem()
x = model.add_variable('x',n)

model.add_list_of_constraints(
[pic.tools.sum([ingredients[i][j]*x[j] for j in range(n)]) <=
(label[i]+max(0.5,eps*label[i])) for i in range(m)]

model.add_list_of_constraints(
[pic.tools.sum([ingredients[i][j]*x[j] for j in range(n)]) >=
(label[i]-max(0.5,eps*label[i])) for i in range(m)]

model.add_list_of_constraints(
[pic.tools.sum([ingredients[0][i]*x[i] >=
ingredients[0][i+1]*x[i+1] for i in range (n-1)])

model.add_constraint(x >= 0)

model.solve(solver='cvxopt')

return x.value

label = [28., 140., 7., 4.5, 25., 150., 18., 12., 2.]

ingredients = [[160., 125., 113., 200., 200., 53., 5., 5., 4.2],
[840., 455., 810., 773., 773., 78., 0., 0., 10.],
[48., 0., 91.7, 0., 0., 5., 0., 0., 0.],
[40., 0., 58., 0., 0., 1.6, 0., 0., 0.],
[0., 0., 242.9, 0., 0., 186., 0., 0., 0.],
```

```
[0., 0., 12.4, 0., 0., 62., 1258.6, 2325., 0.],
[108., 95., 0.1, 200., 200., 0.4, 0., 0., 0.],
[96., 0., 0.1, 200., 200., 0.2, 0., 0., 0.,],
[10., 13., 1., 0., 0., 6.3, 0., 0., 0.]]
```