

*Note:* Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

## 1 LP Basics

**Linear Program.** A *linear program* is an optimization problem that seeks the best value of a linear objective over linear constraints. Let  $x \in \mathbb{R}^n$  be the variables,  $A \in \mathbb{R}^{m \times n}$ ,  $b \in \mathbb{R}^m$ , and  $c \in \mathbb{R}^n$ . One standard canonical form is

$$\begin{aligned} & \text{maximize } \langle c, x \rangle \\ & \text{subject to } Ax \leq b \\ & \quad x \geq 0. \end{aligned}$$

Here,  $\langle c, x \rangle = \sum_{i=1}^n c_i x_i$  is the inner product of  $c$  and  $x$ , and inequalities between vectors are entry-wise.

**Example.** Consider the LP

$$\begin{aligned} & \text{maximize } 3x_1 + 2x_2 \\ & \text{subject to } x_1 + x_2 \leq 4 \\ & \quad 2x_1 + x_2 \leq 5 \\ & \quad x_1, x_2 \geq 0. \end{aligned}$$

This same LP can be written in matrix form as

$$\begin{aligned} & \text{maximize } \left\langle \begin{pmatrix} 3 \\ 2 \end{pmatrix}, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right\rangle \\ & \text{subject to } \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 4 \\ 5 \end{pmatrix} \\ & \quad \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \end{aligned}$$

So in this example,

$$A = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 4 \\ 5 \end{pmatrix}, \quad c = \begin{pmatrix} 3 \\ 2 \end{pmatrix}.$$

Let's check that many LPs can still be converted into this canonical form:

- (i) What if the objective is minimization instead of maximization?
- (ii) What if one of the constraints is  $a^\top x \geq d$ ?

(iii) What if one of the constraints is  $a^\top x = d$ ?

(iv) What if a variable is constrained by  $x \geq -2$ ?

(v) What if one of the variables is unrestricted in sign, so it can be positive or negative?

### Solution:

(i) Negate the objective. Minimizing  $\langle c, x \rangle$  is the same as maximizing  $\langle -c, x \rangle$ .

(ii) Negate both sides to get  $-a^\top x \leq -d$ .

(iii) Replace the equality by the two inequalities  $a^\top x \leq d$  and  $a^\top x \geq d$ , then rewrite the second as  $-a^\top x \leq -d$ .

(iv) Write that variable as the difference of two nonnegative variables:  $x = x^+ - x^-$  with  $x^+, x^- \geq 0$ .

(v) Shift the variable so that it becomes nonnegative: let  $x = y - 2$  where  $y \geq 0$ .

### Simplex:

1. Start at an arbitrary vertex.
2. Denote the current vertex as  $x$ .
3. Look at all neighboring vertices  $y$  to  $x$ .
4. Find the neighbor  $y^*$  with the best objective value (if the LP is minimizing,  $y^*$  should have the smallest objective value, etc.).
5. If  $y^*$  has a better value than  $x$ , then we move to (i.e. visit)  $y^*$  and repeat steps 2 to 5. Otherwise, we have found an optimal solution of the LP, and simply return  $x$ .

$\hookrightarrow$  Runtime: worst case exponential time for all commonly used pivot rules, average case polynomial time.

**LP Solver Runtime:** Any LP can be solved in (worst case) polynomial time using the Ellipsoid or Interior Point Method (IPM). *Note: you do not need to know how the Ellipsoid method or IPM work, but you should know that they can be used to solve an LP in polynomial time.*

## 2 Baker

You are a baker who sells batches of brownies and cookies (unfortunately no brookies... for now). Each brownie batch takes 4 kilograms of chocolate and 2 eggs to make; each cookie batch takes 1 kilogram of chocolate and 3 eggs to make. You have 80 kilograms of chocolate and 90 eggs. You make a profit of 60 dollars per brownie batch you sell and 30 dollars per cookie batch you sell, and want to figure out how many batches of brownies and cookies to produce to maximize your profits.

- (a) Formulate this problem as a linear programming problem; in other words, write a linear program (in canonical form) whose solution gives you the answer to this problem. Draw the feasible region, and find the solution via inspection. Briefly explain how you found your solution.

- (b) Suppose instead that the profit per brownie batch is  $P$  dollars and the profit per cookie batch remains at 30 dollars. For each vertex you listed in the previous part, give the range of  $P$  values for which that vertex is the optimal solution.

**Solution:**(a)  $x$  = number of brownie batches $y$  = number of cookie batchesMaximize:  $60x + 30y$ 

Linear Constraints:

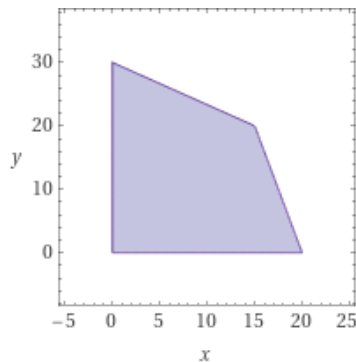
$$4x + y \leq 80$$

$$2x + 3y \leq 90$$

$$x \geq 0$$

$$y \geq 0$$

The feasible region:



The vertices are  $(x = 0, y = 0)$ ,  $(x = 20, y = 0)$ ,  $(x = 15, y = 20)$ ,  $(x = 0, y = 30)$ , and the objective is maximized at  $(x = 15, y = 20)$ , where  $60x + 30y = 1500$ . In other words, we can bake 15 batches of brownies and 20 batches of cookies to yield a maximum profit of 1500 dollars.

*Note: We treat this as an LP; the optimum happens to be integral.*

- (b) There are lots of ways to solve this part. The most straightforward is to write and solve a system of inequalities, checking when the objective of one vertex is at least as large as the objective of the other vertices. For example, for  $(x = 15, y = 20)$  the system of inequalities would be  $P \cdot 15 + 30 \cdot 20 \geq P \cdot 20$  and  $P \cdot 15 + 30 \cdot 20 \geq 30 \cdot 30$ . Doing this for each vertex gives the following solution:

$$(x = 0, y = 30) : P \leq 20$$

$$(x = 15, y = 20) : 20 \leq P \leq 120$$

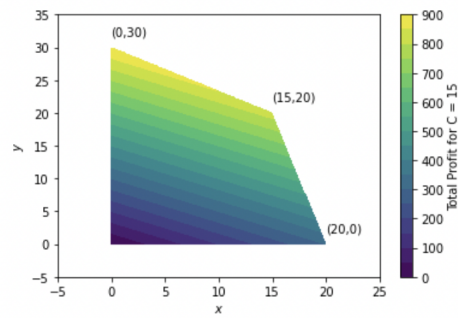
$$(x = 20, y = 0) : 120 \leq P$$

$$(x = 0, y = 0) : \text{never optimal}$$

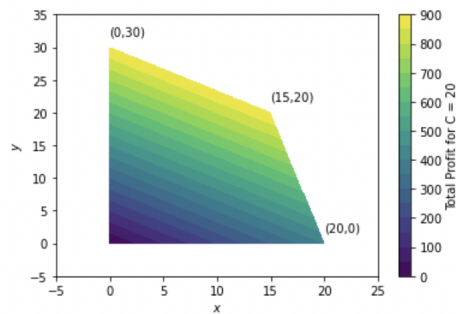
One should note that there is a nice geometric interpretation for this solution: Looking at the graph of the feasible region, as  $P$  increases, the vector  $(P, 30)$  starts pointing closer to the  $x$ -axis. The objective says to find the point furthest in the direction of this vector, so the optimal solution also moves closer to the  $x$ -axis as  $P$  increases. When  $P = 20$  or  $P = 120$ , the vector  $(P, 30)$  is perpendicular to the boundary line of one of the constraints, and there are multiple optimal solutions all lying on that constraint, which are all equally far in the direction  $(P, 30)$ .

Below are some graphics to help build up your intuition about this problem.

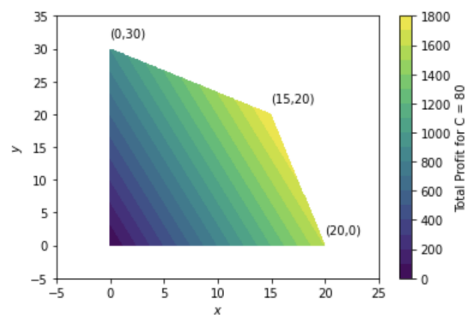
$P = 15$  :  $(x = 0, y = 30)$  optimal



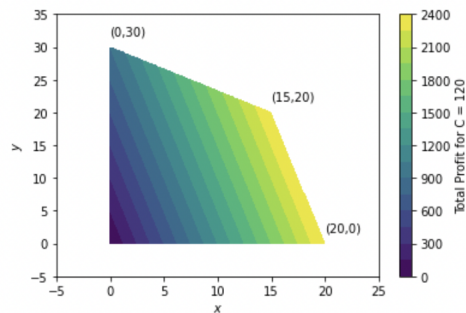
$P = 20$  :  $(x = 0, y = 30)$  and  $(x = 15, y = 20)$  optimal



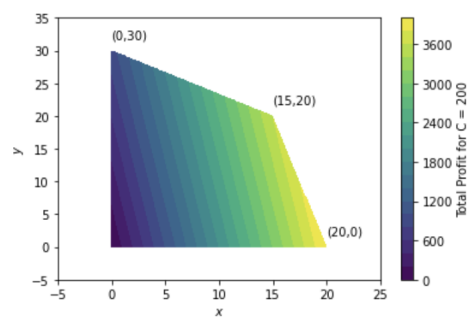
$P = 80$  :  $(x = 15, y = 20)$  optimal



$P = 120$ :  $(x = 15, y = 20)$  and  $(x = 20, y = 0)$  optimal



$P = 200$  :  $(x = 20, y = 0)$  optimal



### 3 LP Meets Linear Regression

One of the most important problems in the field of *statistics* is the *linear regression problem*. Roughly speaking, this problem involves fitting a straight line to statistical data represented by points  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$  on a graph. Denoting the line by  $y = a + bx$ , the objective is to choose the constants  $a$  and  $b$  to provide the “best” fit according to some criterion. The criterion usually used is the *method of least squares*, but there are other interesting criteria where linear programming can be used to solve for the optimal values of  $a$  and  $b$ .

Suppose instead we wish to minimize the sum of the absolute deviations of the data from the line:

$$\min \sum_{i=1}^n |y_i - (a + bx_i)|$$

Write a linear program with variables  $a, b$  to solve this problem.

*Hint: Create new variables  $z_i$  and new constraints to help represent  $|y_i - (a + bx_i)|$  in a linear program.*

#### Solution:

Note that the smallest value of  $z$  that satisfies  $z \geq x, z \geq -x$  is  $z = |x|$ . Now, consider the following linear programming problem:

$$\begin{aligned} & \min \sum_{i=1}^n z_i \\ & \text{subject to } \begin{cases} y_i - (a + bx_i) \leq z_i & \text{for } 1 \leq i \leq n \\ (a + bx_i) - y_i \leq z_i & \text{for } 1 \leq i \leq n \end{cases} \end{aligned}$$

Since  $\sum_i z_i$  is minimized,  $z_i$  will be set to the  $\max(y_i - (a + bx_i), (a + bx_i) - y_i)$ . Note that  $\max(y_i - (a + bx_i), (a + bx_i) - y_i)$  is, in fact,  $|y_i - (a + bx_i)|$ .

If for some solution we have that  $z_i > |y_i - (a + bx_i)|$ , then by setting  $z_i = |y_i - (a + bx_i)|$  we will get a solution with a smaller value of the objective function, therefore the initial solution was not optimal. Hence, the constraints require that the optimal solution will set  $z_i = |y_i - (a + bx_i)|$ , so the new problem is indeed equivalent to the original problem. However, now it is a linear programming problem.

## 4 Egg Drop Revisited

Recall the Egg Drop problem from Discussion 7:

*You are given  $m$  identical eggs and an  $n$ -story building. You need to figure out the highest floor  $b \in \{0, 1, 2, \dots, n\}$  that you can drop an egg from without breaking it. Each egg will never break when dropped from floor  $b$  or lower, and always breaks if dropped from floor  $b + 1$  or higher ( $b = 0$  means the egg always breaks). Once an egg breaks, you cannot use it any more. However, if an egg does not break, you can reuse it.*

*Let  $f(n, m)$  be the minimum number of egg drops needed to find  $b$  (regardless of the value of  $b$ ).*

Instead of solving for  $f(n, m)$  directly, we define a new subproblem  $M(x, m)$  to be the maximum number of floors for which we can always find  $b$  in at most  $x$  drops using  $m$  eggs.

For example,  $M(2, 2) = 3$  because a 3-story building is the tallest building for which we can always find  $b$  in at most 2 drops using 2 eggs.

- (a) Give a recurrence relation for  $M(x, m)$  that can be computed in constant time given smaller subproblems, along with its base cases.

*Hint: What is the highest floor from which we can drop the first egg and still be guaranteed to solve the problem with the remaining  $x - 1$  drops and  $m - 1$  eggs if the egg breaks?*

- (b) Give an algorithm to compute  $M(x, m)$  given  $x$  and  $m$ , and analyze its runtime.

- (c) Modify your algorithm from part (b) to compute  $f(n, m)$  given  $n$  and  $m$ .

*Hint: If we can find  $b$  when there are more than  $n$  floors, we can also find  $b$  when there are  $n$  floors.*

- (d) Show that the runtime of your algorithm from part (c) is  $O(nm)$ . Compare this to the runtime you found in last week's homework.

- (e) Show that your algorithms from parts (b) and (c) can be implemented using only  $O(m)$  space.

### Solution:

(a)

$$M(x, m) = M(x - 1, m - 1) + M(x - 1, m) + 1,$$

with base cases  $M(0, m) = 0$  for all  $m \geq 0$  and  $M(x, 0) = 0$  for all  $x \geq 0$ .

(Drop the first egg from floor  $M(x - 1, m - 1) + 1$ . If it breaks, the  $M(x - 1, m - 1)$  floors below remain. If it doesn't, the  $M(x - 1, m)$  floors above remain.)

- (b) Fill in a table row by row: for  $i = 1, 2, \dots, x$ , compute  $M(i, j)$  for all  $1 \leq j \leq m$  using the recurrence and the values from row  $i - 1$ . Return  $M(x, m)$ .

There are  $xm$  subproblems, each taking constant time, so the runtime is  $\Theta(xm)$ .

- (c) Run the same row-by-row computation for  $x = 1, 2, 3, \dots$  without fixing  $x$  in advance. Stop at the first  $x$  for which  $M(x, m) \geq n$ , and return that  $x$ . (If  $n = 0$ , return 0.)

- (d) With one egg, dropping from floors  $1, 2, \dots$  in order gives  $M(x, 1) = x$ . So  $M(x, m) \geq x$  for all  $m \geq 1$ , and the algorithm stops at some  $x \leq n$ . Each row takes  $O(m)$  time, so the total runtime is  $O(nm)$ .

This bound is loose, but it is still much better than the  $O(n^2m)$  runtime of the direct DP on  $f(n, m)$ .

- (e) Row  $x$  depends only on row  $x - 1$ , so we keep just two arrays of length  $m + 1$ : the previous row and the current row. That is  $O(m)$  space.

(We can even use a single array by updating  $j$  from  $m$  down to 1:  $M[j] \leftarrow M[j - 1] + M[j] + 1$ , since  $M[j - 1]$  still holds its value from row  $x - 1$  at that point.)