COS 330 - Lecture 13

Matt Weinberg

October 27th, 2025

These notes may contain errors, and any errors are the fault of the author. See Section 6 for the sources from which these notes were derived.

1 Linear Programming

Here is a motivating example to have something concrete in mind:

Example 1 (Smoothies)**.** Say you have a stock of 3 oranges, 10 bananas, and 6 strawberries. You can make and sell 3 kinds of smoothies:

	Smoothie Type 1 *Orange-Banana*	**Smoothie Type 2** *Strawberry-Banana*	**Smoothie Type 3** *Citro-Bananaberry*
Ingredients per Liter	**1** Orange **2** Bananas	**1** Banana **2** Strawberries	**1** Orange **1** Banana **1** Strawberry
Profit	**5** Dollars/Liter	**4** Dollars/Liter	**6** Dollars/Liter

How should you turn your fruits into smoothies in order to maximize your profit? You can make and sell fractions of liters of smoothies (e.g. you can sell $3/5$ liters of Orange-Banana smoothies to get 3 dollars), but you cannot make or sell negative smoothies (e.g. you can't pay to get the fruits back that you would've used for a smoothie).

Remark. The smoothies example is exactly the following problem: define the variables $x_i \in \mathbb{R}$ to be the number of liters of Smoothie Type i that you will make. Then your objective is to

$$\text{maximize } 5x_1 + 4x_2 + 6x_3 \ , \quad \text{your total profit.}$$

However, you have a few constraints. The variables x_1, x_2, x_3 must satisfy

$$\begin{aligned} 1x_1 + \qquad 1x_3 &\leq 3 \ , && \text{or you will run out of oranges;} \\ 2x_1 + 1x_2 + 1x_3 &\leq 10 \ , && \text{or you will run out of bananas;} \\ 2x_2 + 1x_3 &\leq 6 \ , && \text{or you will run out of strawberries} \\ x_1,\ x_2,\ x_3 &\geq 0 \ , && \text{since you cannot make negative liters of smoothie.} \end{aligned}$$

So our problem is maximizing a linear objective function subject to three linear constraints (plus that all variables are non-negative). In this class, we will not discuss how to go about solving this

(in case you are interested, look for "simplex method"). But let's say someone hands you a solution and claims that it is optimal. How would you go about trying to prove that you cannot do better?

Specifically, I will claim that the **optimal solution** is to make 3 liters of orange banana and 3 liters of strawberry banana ($x_1 = 3$, $x_2 = 3$, $x_3 = 0$). This gives you profit 27. Observe also that this uses 3 oranges, 9 bananas, and 6 strawberries, so it is feasible. But how do I know I cannot do better? I might be concerned that I can do better because I have a leftover banana, or because maybe I can make better use of my strawberries and oranges.

Here is a proof that I cannot do better:

- ***Any feasible solution*** satisfies both the inequalities $x_1 + x_3 \leq 3$ and $2x_2 + x_3 \leq 6$, the constraints on our orange usage and strawberry usage.
- Therefore, *any feasible solution* satisfies a positive linear combination of these constraints:

$$\begin{array}{rcl} 5 & \times & [\ x_1 + \quad\quad x_3 \leq 3\ ] \\ 2 & \times & [\ \quad 2x_2 + \ x_3 \leq 6\ ] \\ \hline & & 5x_1 + 4x_2 + 7x_3 \leq 27 \end{array}$$

- Finally, since *any feasible solution* satisfies $x_3 \geq 0$, we have

$$\text{Profits} \quad = \quad 5x_1 + 4x_2 + 6x_3 \quad \leq \quad 5x_1 + 4x_2 + 7x_3 \quad \leq \quad 27 \ .$$

This proves that our original solution is optimal! But how did we do this? How did we magically guess to add up five times the orange constraint plus twice the strawberries constraint? This is a special case of a more general concept called *linear programming duality*, which is the focus of the rest of this lecture.

2 LP Duality

Consider generally a linear program (LP) of the form:[1]

$$\begin{array}{rll} \max & \displaystyle\sum_i c_i x_i & \\ \text{s.t.} & \displaystyle\sum_i A_{ji} x_i \leq b_j & \forall j \\ & x_i \geq 0 & \forall i \ . \end{array}$$

We will call this the *primal LP*. We call $\vec{x}$ a *primal solution*, and our goal is to find a primal solution that maximizes our objective, subject to the feasibility constraints.

On the other hand, we also need to think about how to prove that a solution is optimal once we find it. That is, we need to think about searching for good upper bounds on how good a primal can possibly be. This is called the *dual* problem. How can we derive an upper bound on how good a primal can possibly be? We will follow the approach suggested by the smoothies example.

[1] If you're used to matrices and find the expression $\sum_i A_{ji} x_i$ odd, note that it's the same as $\sum_i A_{ij} x_j$, the one you might have expected. We index rows by j to emphasize they correspond to constraints.

x_1	x_2	x_3	x_4	x_5	
A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	b_1
A_{21}	A_{22}	A_{23}	A_{24}	A_{25}	b_2
A_{31}	A_{32}	A_{33}	A_{34}	A_{35}	b_3
A_{41}	A_{42}	A_{43}	A_{44}	A_{45}	b_4
A_{51}	A_{52}	A_{53}	A_{54}	A_{55}	b_5
c_1	c_2	c_3	c_4	c_5	

Table 1: A visualization of the form of an LP. The i's range across the columns and index the variables $x_1, \ldots, x_5$, and the j's range across the rows and index the inequalities. The objective function and constraints read by multiplying the variable x_i with the coefficients in its corresponding column, then summing across rows.

Consider the following: if we have a weight $w_j \geq 0$ for each inequality j, and take a linear combination of the feasibility constraints, we may directly conclude that any feasible $\vec{x}$ must satisfy

$$\sum_j w_j \left(\sum_i A_{ji} x_i \right) \quad \leq \quad \sum_j w_j b_j \ . \tag{1}$$

In our example, for instance, by summing the orange and strawberry constraints we can see that we must have $x_1 + 2x_2 + 2x_3 \leq 9$ for *any* feasible combination of smoothies. This is certainly cool, but does not appear particularly useful, as it tells us nothing about how much profit we can possibly make. We further saw, however, that if we chose $w_1 = 5$, $w_2 = 0$, $w_3 = 2$, then we got something interesting.

More generally, if we happen to have chosen our w_j's so that $\sum_j w_j A_{ji} \geq c_i$ for all i and $w_j \geq 0$ for all j, we are in business! This is because we could then conclude that

$$\sum_i c_i x_i \quad \leq \quad \sum_i \left(\sum_j w_j A_{ji} \right) x_i \quad = \quad \sum_j w_j \left(\sum_i A_{ji} x_i \right) \quad \leq \quad \sum_j w_j b_j \ ,$$

where the first inequality follows from $\sum_j w_j A_{ji} \geq c_i$ and $x_i \geq 0$, the equality simply exchanges order of summation, and the last inequality follows from (1) (because $w_j \geq 0$).

This means that *any* set of weights w_j which satisfy $\sum_j w_j A_{ji} \geq c_i$ for all i and $w_j \geq 0$ for all j yields a valid inequality of the form $\sum_i c_i x_i \leq \sum_j w_j b_j$. This exactly reads off as an upper bound of $\sum_j w_j b_j$ on how good any feasible solution can possibly be!

So now, we can think of the following "dual" approach: search over all weights w_j to find the ones that minimize $\sum_j w_j b_j$, subject to the constraints $\sum_j w_j A_{ji} \geq c_i$ for all i and $w_j \geq 0$ for all j. The constraints matter because if they aren't satisfied, then $\sum_j w_j b_j$ doesn't actually guarantee an upper bound on the primal.

Notice that this itself is another linear program:

$$\begin{aligned} \min \quad & \sum_j w_j b_j \\ \text{s.t.} \quad & \sum_j w_j A_{ji} \geq c_i \quad \forall i \\ & w_j \geq 0 \quad \forall j \ . \end{aligned}$$

This is called the *dual LP*.

	x_1	x_2	x_3	x_4	x_5	
w_1	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	b_1
w_2	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}	b_2
w_3	A_{31}	A_{32}	A_{33}	A_{34}	A_{35}	b_3
w_4	A_{41}	A_{42}	A_{43}	A_{44}	A_{45}	b_4
w_5	A_{51}	A_{52}	A_{53}	A_{54}	A_{55}	b_5
	c_1	c_2	c_3	c_4	c_5	

Table 2: The visualization in Table 1 with the weight variables from the dual LP included. Again, the i's range across the columns and index the variables $x_1, \ldots, x_5$ of the primal. However, notice also that the i's index the inequalities of the dual. Similarly, while the j's range across the rows and index the inequalities of the primal, they also index the variables $w_1, \ldots, w_5$ of the dual (this makes sense, because our initial interpretation of the w_j were as weights being multiplied to each inequality of the primal). The objective function and constraints of the dual read by multiplying the weight variable w_j with the coefficients in its corresponding row, then summing across columns. The primal and dual LPs are thus closely connected.

Check Your Understanding.

- Why did we insist that $\sum_j w_j \cdot A_{ij} \geq c_i$ for all i? Think of this as a "usefulness constraint". If this fails for some i, then we don't necessarily learn anything about how good the primal can be.
- Why did we insist that $w_j \geq 0$ for all j? Think of this as a "validity" constraint". If this fails for some j, then we just multiplied an inequality by a negative number, which results in an invalid inequality.

Exercise (Dual of the dual)**.** Verify that the dual of the dual LP is the primal LP itself.

Note that we have already proved that *every* feasible solution of the dual provides an upper bound on how good any primal solution can possibly be (also known as **Weak LP Duality**). Magically, this turns out to be a *complete* approach to proving bounds! If the optimum of the primal LP is some finite value x, then there exists a solution to the dual LP that has value x as well.

In other words, if the primal LP has a finite optimal value x, there exists a proof using linear combinations of the constraints (such as the proof we gave for the smoothies example) proving that x is optimal.

Theorem 2 (Strong LP Duality)**.** *Let LP1 be any maximization (resp., minimization) LP and let LP2 be its dual, which is a minimization (resp., maximization) LP. Then if the optimum of LP1 is finite, then the optimum of LP2 is also finite, and they are equal.*

You do not need to know the proof of Strong LP Duality for the course, but if you are interested, see Anupam Gupta's scribed lecture notes here.

3 Cool, what now?

We'll now use LP duality to prove something interesting. Recall the following two concepts:

Definition 3 (Matching)**.** *Given a graph $G = (V, E)$, a* matching *is a subset of edges $M \subseteq E$ such that no node $v \in V$ appears in multiple edges $e \in M$. That is, for all $v \in V$, $|\{e \in M,\ v \in e\}| \leq 1$.*

Definition 4 (Vertex Cover)**.** *Given a graph $G = (V, E)$, a* vertex cover *is a subset of vertices $C \subseteq V$ such that all edges $e \in E$ have at least one endpoint in C. That is, for all $e \in E$, $|e \cap C| \geq 1$.*

We will now go through the following exercise:

1. Write an Integer Program for the max-weight matching.
2. Relax it to a Linear Program.
3. Take the Dual Linear Program.
4. Add Integrality constraints, and observe that this is exactly the minimum Vertex Cover.
5. Stare at a related chain of inequalities, and see what we can conclude from it.

First, we claim that the following Integer Program exactly captures the max-weight matching in G:

$$\text{Maximize } \sum_{e \in E} x_e, \text{ such that:} \forall v \in V,\ \sum_{e,\ v \in e} x_e \leq 1$$
$$\forall e \in E,\ x_e \text{ is a non-negative integer.}$$

Here's a quick side-question – why is it OK to only constrain x_e to be an integer instead of further constraining $x_e \in \{0, 1\}$? Why won't we accidentally wind up with $x_e = 2$?[2]

We'll refer to the value of the optimal solution to this Integer Program as $MM(G)$. The LP relaxation is as follows:

$$\text{Maximize } \sum_{e \in E} x_e, \text{ such that:} \forall v \in V,\ \sum_{e,\ v \in e} x_e \leq 1$$
$$\forall e \in E,\ x_e \geq 0$$

We'll refer to the value of the optimal solution to this Linear Program as $LP(G)$. Its Dual is as follows. In taking the dual, recall:

1. We make a variable for each constraint, and there is a constraint for every $v \in V$. So let's make a variable y_v for the constraint corresponding to v.
2. We want the RHS of our mega-constraint to be as small as possible, so we want to minimize $\sum_{v \in V} y_v$.

[2] Answer: because the inequality above implies $x_e \leq 1$ for all $e \in E$.

3. Our mega-constraint is useful if and only if the coefficient of each x_e in our mega-constraint exceeds 1 (its coefficient in the the objective function of MLP). This is tricky to think through – if A_{ve} denotes the coefficient of x_e in the equation for node v, the coefficient of x_e in our mega-constraint is $\sum_v y_v \cdot A_{ve}$. What is this?

   - Well, $A_{ve} = 0$ whenever $v \notin e$, and $A_{ve} = 1$ whenever $v \in e$. So this simplifies to: $\sum_v y_v \cdot A_{ve} = y_v + y_u$ when $e = (u, v)$.
   - Therefore, our mega-constraint is useful if and only if $y_u + y_v \geq 1$ for all $e = (u, v) \in E$.

4. And that's it! So we get the Dual LP as:

$$\text{Minimize } \sum_{v \in V} y_v, \text{ such that:} \forall e \in E,\ y_u + y_v \geq 1$$
$$\forall v \in V,\ y_v \geq 0$$

We'll call this $DLP(G)$. Finally, let's insist that each y_v is integral and see what happens.

$$\text{Minimize } \sum_{v \in V} y_v, \text{ such that:} \forall e \in E,\ y_u + y_v \geq 1$$
$$\forall v \in V,\ y_v \text{ is a non-negative integer}$$

We'll call this $VC(G)$. Now, we claim that $VC(G)$ is exactly the minimum Vertex Cover of G! To see this, observe that any $\{0, 1\}$-solution to the above IP is indeed a Vertex Cover (because at least one node from each edge is in the cover), and that every Vertex Cover is a $\{0, 1\}$-solution to the above IP. Why is it OK that we didn't constrain $y_v \in \{0, 1\}$? Why won't the optimal solution have $y_v = 2$?[3]

Now, we claim the following interesting chain of inequalities:

$$MM(G) \leq LP(G) \leq DLP(G) \leq VC(G).$$

$MM(G) \leq LP(G)$ simply because $MM(G)$ only has integral solutions, whereas $LP(G)$ can have fractional solutions. $LP(G) \leq DLP(G)$ due to Weak LP Duality. And $DLP(G) \leq VC(G)$ again simply because $VC(G)$ is restricted to integral solutions whereas $DLP(G)$ can use fractional solutions.

So all together, this establieshes that for all graphs G, the maximum matching is at most the minimum vertex cover. Cool, but there's a simpler way to prove this.[4] What's cooler about this proof is the following:

- Is it possible that $MM(G) = LP(G)$ for all graphs? No.[5] But, it turns out that $MM(G) = LP(G)$ for all bipartite graphs G![6]
- Is it possible that $LP(G) = DLP(G)$ for all graphs? Yes, by Strong LP Duality!

[3] Answer: in this case, it is *feasible* in the IP to set $y_v = 2$. But, any *optimal* solution will never do this, because if a solution is feasible with $y_v = 2$, it is certainly feasible to update $y_v = 1$, which strictly improves the objective.

[4] Every matching of size C witnesses a set of C edges such that no node can cover more than one edge from this set. Therefore, at least C distinct nodes must be used to cover them.

[5] Example: consider a triangle. Then the max-matching is 1, but $LP(G) = 3/2$ by setting each $x_e = 1/2$.

[6] This is a consequence of the Birkhoff-Von Neumann Theorem.

- Is it possible that $DLP(G) = VC(G)$ or all graphs? No.[7] But, it turns out that $DLP(G) = VC(G)$ for all bipartite graphs G![8]

- So, after doing a little bit of graph theory, we can conclude that $MM(G) = VC(G)$ for all bipartite graphs G! Importantly, $LP(G) = DLP(G)$ *for all graphs* G *by strong LP duality*, and *whether or not* $MM(G) = VC(G)$ *depends entirely on whether* $MM(G) = LP(G)$ *and* $DLP(G) = VC(G)$.

4 Minor extensions

Note: This was *not* covered in lecture, but it is a good "Check your understanding" exercise to see if you can read this on your own.

Just to get some more comfort with duals, let us consider the following minor extension, which works even when a linear program is not exactly of the form we described earlier. It is also the form we will use in next lecture. Specifically, we consider a linear program of the form:

$$\begin{aligned} \max \quad & \sum_i c_i x_i \\ \text{s.t.} \quad & \sum_i A_{ji} x_i \leq b_j \quad \forall j \leq m \\ & \sum_i A_{ji} x_i = b_j \quad \forall j > m \\ & x_i \geq 0 \quad \forall i \leq n \ . \end{aligned}$$

The difference is that some constraints are now equalities instead of inequalities, and some variables are allowed to be negative.

Observe that we can write any LP of this form in the same format as the original; we just need to write two inequalities for each equality constraint ($\sum_i A_{ji} x_i \leq b_j$ and $\sum_i -A_{ji} x_i \leq -b_j$), and for all unconstrained x_i, replace it by two variables $x_i^+, x_i^- \geq 0$ to represent the positive and negative components of x_i. Then we can replace x_i by $x_i^+ - x_i^-$.

Alternatively, we can just start from scratch and go through the same reasoning. What changes from our previous approach?

- First, we previously insisted that the weight w_j for constraint j satisfy $w_j \geq 0$, because if we multiply an inequality by a negative number, we have to flip the sign. **But for an equality, we can take negative multipliers just fine.**

- Second, we previously declared a linear combination to be a valid upper bound on our objective as long as $\sum_j w_j A_{ji} \geq c_i$ for all i. This is true as long as $x_i \geq 0$. **But if x_i is allowed to be negative, we need to get the coefficient exactly right, and require $\sum_j w_j A_{ji} = c_i$.**

[7] Example: consider a triangle. Then the min vertex cover is 2, but $DLP(G) = 3/2$ by setting each $y_v = 1/2$.

[8] We omit a proof of this, but here is a randomized rounding scheme: pick z uniformly at random from $[0, 1]$. For all left-hand nodes with $y_v \geq z$, add them to the cover. For all right-hand nodes with $y_v \geq 1 - z$, add them to the cover. Then: (a) this is definitely a vertex cover (because for all $e = (u, v)$, we had $y_u + y_v \geq 1$, so either $y_u \geq z$ or $y_v \geq 1 - z$, and (b) the expected size is exactly the same as $DLP(G)$, because each node v is in the cover with probability exactly y_v. Therefore, there must be an integral vertex cover that's at least as good as $DLP(G)$ (and in fact, *all* integral covers resulting from this rounding scheme must be exactly as good as $DLP(G)$, because the expected value is $DLP(G)$ but no vertex cover can be strictly better, so they must not be strictly worse either).

One can indeed derive these by using our "standard form dual" as a black box, with the modifications in the above paragraph, but it is illustrative to see both ways. In any case, the dual for LPs of the above form is as follows:

$$\begin{aligned} \min \quad & \sum_j w_j b_j \\ \text{s.t.} \quad & \sum_j w_j A_{ji} \geq c_i \quad \forall i \leq n \\ & \sum_j w_j A_{ji} = c_i \quad \forall i > n \\ & w_j \geq 0 \quad \forall j \leq m \ . \end{aligned}$$

Check Your Understanding.

- Why is it OK to have an inequality for $i \leq n$, but we need equality for $i > n$? This is because $x_i \geq 0$ for $i \leq n$, and not necessarily for $i > n$. When we know that $x_i \geq 0$, we know that $c_i' x_i \geq c_i x_i$ whenever $c_i' \geq c_i$. But if we don't know that $x_i \geq 0$, then it could be that $c_i' \geq c_i$ but $c_i' x_i < c_i x_i$, and then we don't learn anything useful about the primal. Still, if $c_i' = c_i$, than $c_i' x_i = c_i x_i$ no matter what x_i is.
- Why did we need $w_j \geq 0$ only for $j \leq m$, but not $j > m$? Remember that all variables w_j multiply an inequality, and inequalities can only be multiplied by non-negative numbers and remain valid. However, all constraints $j > m$ were *equalities*. Equalities remain valid even after you multiply them by a negative number, so we don't need to constrain $w_j \geq 0$, any w_j is fine.

Key Takeaway. In general, I *strongly, strongly* do not recommend trying to memorize a long list of rules to figure out how to take the dual. Instead, I recommend trying to remember the following key principles, and then applying them each time you want to take a dual:

- You make a variable w_j for every constraint j.
- You put *validity constraints* on the w_j which guarantee that any feasible solution must satisfy

$$\sum_j w_j \left(\sum_i A_{ji} x_i \right) \quad \leq \quad \sum_j w_j b_j \ .$$

  If constraint j is an inequality, then we must have $w_j \geq 0$, or the inequality may not continue to hold. If constraint j is an equality, then w_j can be anything.
- You put *usefulness constraints* on the w_j which guarantee that the new constraint derived above actually says something useful about the objective function. That is,

$$\sum_i c_i x_i \quad \leq \quad \sum_i \left(\sum_j w_j A_{ji} \right) x_i \quad \leq \quad \sum_j w_j b_j \ .$$

  If $x_i \geq 0$ is guaranteed, you just need $c_i \leq \sum_j w_j A_{ji}$. Otherwise, you need $c_i = \sum_j w_j A_{ji}$.
- You minimize the RHS above to get the best upper bound.

5 Sources

These notes were derived from the following sources. These sources also serve as great further reading, if you would like to read more.

- [Goe15] Sections 1 and 4.

6 Acknowledgements

Thank you to Freddy Qiu '23 for helping to overhaul these notes.

References

[Goe15] Michel Goemans. Lecture notes in linear programming. https://math.mit.edu/~goemans/18310S15/lpnotes310.pdf, March 2015. 9