Lecture 1

LPs: Algebraic View*

1.1 Introduction to Linear Programming

Linear programs began to get a lot of attention in 1940's, when people were interested in minimizing costs of various systems while meeting different constraints. We care about them today because we can solve them efficiently and a very general class of problems can be expressed as LPs. A linear program has variables, linear constraints on those variables, and a linear objective function which we aim to maximize or minimize. This might look something like the following:

$$\begin{aligned} x_1 &\geq 0 \\ x_1 + x_2 &\leq 2 \\ x_1 - x_2 &\geq 1 \\ x_2 &\geq 2 \\ \min \quad 3x_1 &+ 2x_2 \end{aligned}$$

The "feasible region", the settings for x_1, x_2 that satisfy the above constraints, look like this:
trict

Here is a simple example of a linear program called the *Diet problem*. There are n foods and m nutrients you need to have enough of in your diet. You'd like to spend the least money possible while getting enough of each nutrient. So, let a_{ij} denote the amount of nutrient i in each unit of food j, b_i be the minimum amount of nutrient i you need in your diet, and c_j be the cost of one unit of food j, and x_j be the variable representing the amount of food j you are solving to buy. These constraints are written as:

$$\sum_j a_{ij} x_j \geq b_j$$

$$x_j \geq 0$$

*Lecturer: Anupam Gupta. Scribe: Jamie Morgenstern.

And the objective function to minimize is the cost:

$$\min \sum_j c_j x_j$$

As you might notice, we suggestively chose a_{ij} notation for the coefficients of the constraints: usually, we do write LPs with matrices. To rewrite this in matrix form, let A be an $m \times n$ matrix with $A_{ij} = a_{ij}$, B be a $m \times 1$ column matrix with $B_i = b_i$, x be a $n \times 1$ column vector of variables, and c be the $n \times 1$ column vector such that $c_i = c_i$. Then, we write the constraints as

$$Ax \geq b$$
$$x \geq 0$$

and the objective as

$$\min c^T x$$

We will also find it useful to write A in two other ways: as a concatenation of its rows or its columns:

$$A = \begin{pmatrix} | & | & \dots & | \\ A_1 & A_2 & \dots & A_n \\ | & | & \dots & | \end{pmatrix} = \begin{pmatrix} — & a_1 & — \\ — & a_2 & — \\ — & \dots & — \\ — & a_m & — \end{pmatrix}$$

There are several forms in which people write their LPs. The minimization of $c^T x$ can be recast as a maximization of $-c^T x$. Similarly, one can recast upper bound constraints of the form

$$a_i x \geq b_i$$

to the equivalent lower bound constraints

$$-a_i x \leq b_i.$$

It is also easy to translate equality constraints into pairs of inequality constraints:

$$a_i x = b_i \iff a_i x \leq b_i \text{ and } a_i x \geq b_i$$

One can starting from inequality constraints and get to equality constraints (along with nonnegativity inequalities) as follows:

$$a_i x \leq b_i$$

becomes

$$a_i x + s_i = b_i,$$
$$s_i \geq 0.$$

where we have introduced new variables s_i, the "slack" variables.

Finally, if we have unconstrained variables x_i (which are allowed to be positive or negative), and we would like only non-negative variables, we could introduce two non-negative variables $x_i^+ \geq 0$ and $x_i^- \geq 0$ for each such unconstrained x_i, and replace each x_i by $x_i^+ - x_i^-$.

This allows us to move between various forms of LPs: the two most common forms of LPs are the *general form*, which is

$$\begin{aligned} &\min c^T x \\ &Ax \geq b \end{aligned}$$

and the *equational* (or *standard*) form, which is

$$\begin{aligned} &\min c^T x \\ &Ax = b \\ &x \geq 0 \end{aligned}$$

To go from the general form to the equational form, we need two things. First, we can add slack variables for each constraint $a_i x \geq b_i$ to get equality constraints $a_i x - s_i = b_i$. Second, since the general form doesn't require x to be positive, we can use the idea of replacing each x_i with $x_i^+ - x_i^-$, where $x_i^+, x_i^- \geq 0$. Can you see why these two LPs have the same feasible solution set? Going in the other direction is easier: just replacing $a_i x = b_i$ by two inequalities $a_i x \geq b_i$ and $a_i x \leq b_i$ gives us an LP in general form.

Note that given m constraints and n variables, you may end up with $O(m+n)$ constraints and $O(m+n)$ variables in this conversion process. Note that if $m \gg n$ this may not always be a desireable conversion to make.

Formally, a *feasible solution* is some $x \in \mathbb{R}^n$ such that x satisfies all constraints. We say that x is *optimal* if it maximizes the objective function subject to all constraints.

It is possible that LP's have either bounded feasible sets or unbounded feasible sets. In the case that the feasible set is unbounded, it may also be the case that the optimal value is also unbounded.

1.2 Fourier–Motzkin elimination

The first way we'll describe to solve LP's is known as the *Fourier–Motzkin elimination* algorithm. Consider an LP in general form:

$$\begin{aligned} &\min c^T x \\ &Ax \geq b \end{aligned}$$

Let us rewrite it using one additional variable in this slightly modified, but equivalent form:

$$\begin{aligned} &\min x_{n+1} \\ &Ax \geq b \\ &c^T x \leq x_{n+1} \end{aligned}$$

Now we will eliminate variables in the following way. For variable x_1, arrange the constraints we have into three groups: those where x_1 has positive coefficients (let the indices for these constraints belong to set $P \subseteq [m]$), those where it has negative coefficients (let $N \subseteq [m]$ be the set of these indices), and those which don't involve x_1 at all (let $Z = [m] \setminus (P \cup N)$ be the indices for such constraints). Consider any constraints $a_i x \geq b_i$ for $i \in P$: we can divide out by the coefficient a_{i1} of x_1 for each one, leaving you with constraints of the form:

$$\begin{aligned} & x_1 + \frac{a_{i2}}{a_{i1}} x_2 + \cdots + \frac{a_{in}}{a_{i1}} \geq \frac{b_i}{a_{i1}} x_n \\ \iff & x_1 \geq \frac{b_i}{a_{i1}} - \left(\sum_{j=2}^{n} \frac{a_{ij}}{a_{i1}} x_j \right) \end{aligned}$$

Note that such constraints give us lower bounds for x_1. Now we do a similar operation for the constraints $a_i x \geq b$ for $i \in N$: remember that $a_{i1} < 0$ for these constraints, so we need to take care that dividing by a negative number changes the inequality direction, leaving you with constraints of the form:

$$\begin{aligned} & a_{i1} x_1 + a_{i2} x_2 + \cdots + a_{in} x_n \geq b_i \\ \iff & x_1 + \frac{a_{i2}}{a_{i1}} x_2 + \cdots + \frac{a_{in}}{a_{i1}} x_n \leq \frac{b_i}{a_{i1}} \\ \iff & x_1 \leq \frac{b_i}{a_{i1}} - \left(\sum_{j=2}^{n} \frac{a_{ij}}{a_{i1}} x_j \right) \end{aligned}$$

Now we create new constraints as follows: for each $i \in P$ and $i' \in N$, we get $blah_i \leq x_i$ and $x_{i'} \leq blah_{i'}$, so we get the new constraint $blah_i \leq blah_{i'}$. More formally, for each such pair $i \in P, i' \in N$, we get the constraint:

$$\frac{b_i}{a_{i1}} - \left(\sum_{j=2}^{n} \frac{a_{ij}}{a_{i1}} x_j \right) \leq \frac{b_{i'}}{a_{i'1}} - \left(\sum_{j=2}^{n} \frac{a_{i'j}}{a_{i'1}} x_j \right)$$

(All the constraints in Z just get copied over to this new LP.) It is easy to check the following lemma:

Lemma 1.1. *Given LP1 on k variables, suppose eliminating x_1 gives the new linear program LP2. Show that (a) if $(x_1 x_2 \cdots x_k)$ was feasible for LP1 then $(x_2 x_3 \cdots x_k)$ is feasible for LP2, and if $(x_2 \cdots x_k)$ is feasible for LP2 then there exists some value $x_i' \in \mathbb{R}$ such that $(x_1' x_2 x_3 \cdots x_k)$ is feasible for LP1.*

Note that we took the $|P| + |N|$ constraints, and replaced them with $|P| \cdot |N|$ constraints. Hence from the m constraints, we now have at most $m^2/4$ constraints, but one fewer variable. Continuing this process for each of the variables $x_2, x_3, \ldots, x_n$, we get at most m^{2^n} constraints. And when we have a single variable x_{n+1} remaining, each constraint is of the form $x_{n+1} \leq \textit{something}$, or $x_{n+1} \geq \textit{something else}$. These can be combined to get values ℓ, h, such that $x_{n+1} \in [\ell, h]$. (If $\ell > h$ then the LP is infeasible, and if there is no lower bound

then the LP is unbounded.) Now since the LP sought to minimize x_{n+1}, we get that the optimal value of the LP is $x_{n+1} = \ell$. Moreover, it is easy to proceed backwards through this process to get settings for the eliminated variables $x_1, x_2, \ldots, x_n$ such that $\sum_{j=1}^{n} c_j x_j = \ell$. (Exercise!)

> **Note:** Kevin asked what happens if, say, N was empty, and x_1 only had lower bound constraints (constraints in P). In that case there are no constraints in $P \times N$, and hence we would end up throwing away all the constraints in P and N. Indeed, this makes sense, since whatever the setting of variables $x_2, \ldots, x_n$, having np upper bound constraints on x_1 means we could set x_1 as large as needed to satisfy constraints involving x_1.

1.2.1 Gaussian Elimination

This is a good time to just mention Gaussian elimination (converting a matrix to an upper triangular matrix; this can be used to solve systems of linear equations $Ax = b$). If we just had a collection of equality constraints, the elimination could proceed by taking the first constraint $\sum_j a_{1j} x_j = b_1$, rewriting this as $x_1 = a_{11}^{-1}(b_1 - \sum_{j=2}^{n} a_{1j} x_j)$, and substituting this into the other constraints. This is pretty much what we do using Gaussian elimination.

Gaussian elimination can be done in strongly polynomial time, meaning that

- The number of operations done is polynomial in n and m, and
- The size of the numbers in the intermediate stages of the algorithm are $poly(n, m, \log |a_{ij}|)$ (i.e., the size of the input). Hence we can ensure the matrix entries don't grow too large during the algorithm.

It remains an interesting (and open) problem whether all of linear programming can be done in strongly polynomial time.

> **Note:** Remember that the *size* of numbers measures the *number of bits* used to represent the numbers. Hence, if the entries of the matrix are $a_{ij} \in \{0, 1\}$, then 2^n has size polynomial in the size of the input, but 2^{2^n} does not.
>
> Formally, let us define the size: for an integer k, define $\text{size}(k) = 1 + \lceil \log_2(|k| + 1) \rceil$; for a rational p/q (with p, q coprime, $q > 0$), define $\text{size}(p/q) = \text{size}(p) + \text{size}(q)$; for a matrix $R = (r_{ij})$ of rationals, define $\text{size}(M) = \sum_{i,j} \text{size}(r_{ij})$

1.3 Equational Form Solving

We've seen that Fourier–Motzkin gives us a solution in at most m^{2^n} time. Now let's consider a faster method for solving linear programs. For this section, assume our LP is in the equational form

$$\min\{c^T x \mid Ax = b, x \geq 0\}$$

Let us make two assumptions (without loss of generality). Firstly, we assume that $Ax = b$ has a solution (otherwise our LP is infeasible). And we can use Gaussian elimination to check if $Ax = b$ has a solution or not. Secondly, we assume that the rows of A are linearly independent (otherwise there is some constraint which is superfluous and we can throw it out).

With these assumptions, note that $rank(A) = m$, the number of constraints. Given a subset $B \subseteq [n]$, we define A_B to be the concatenation of the B columns of A. Similarly, we define x_B to be the column vector consisting of the variables $\{x_i \mid i \in B\}$. Suppose we had some subset B with $|B| = m$ such that A_B's columns were linearly independent. Then, A_B would have full rank, and thus be invertible, so

$$A_B x_B = b$$

would have a unique solution

$$x_B = A_B^{-1} b.$$

We can extend this x_B to all n variables (by setting $x_i = 0$ for all indices $i \notin B$): this vector x we call a *basic solution*. Note that a basic solution satisfies $Ax = b$, but it may not satisfy the non-negativity constraints. We say the basic solution is feasible (called a *basic feasible solution* or *BFS*) if $x_B \geq 0$, i.e., it also satisfies the non-negativity constraints.

So suppose we knew that the optimal value was achieved at a BFS, we could just try all $\binom{n}{m}$ subsets of columns B with $|B| = m$ which are linearly independent, and take the optimal one. However, we don't yet know this: we really have two questions at this point. Firstly, *what if there exists a solution to the LP, but no BFS exists*? And secondly, *what if the optimum is attained only at non-BFS points*? It turns out neither of these is a problem.

Fact 1.2. *Every linearly independent set B with $|B| = m$ gives exactly one basic solution and at most one BFS.*

Theorem 1.3. *For any LP in equational form, one of the following holds*

1. *The LP is infeasible*
2. *The LP has unbounded optimum*
3. *The LP has a BFS which is optimal*

Proof. Suppose our LP is feasible, and has a bounded optimum. Additionally, assume we have some x^* which is feasible (i.e., $Ax^* = b, x^* \geq 0$). Now we will show that there exists a BFS x with value $c^T x \leq c^T x^*$. Hence, for any feasible solution, there is a BFS with no higher value, and so there must exist an optimal BFS.

OK, so given x^*, pick a feasible solution $\tilde{x}$ among those that have $c^T x \leq c^T x^*$, and where $\tilde{x}$ has the fewest number of nonzero coordinates. Let

$$P = \{i \mid \tilde{x}_i > 0\}$$

be the set of corrdinates that are *strictly* positive. (Note that since all the other coordinates in $\tilde{x}$ are zero, we get $\sum_{j \in P} A_j \tilde{x}_j = \sum_j A_j \tilde{x}_j = b$, or $A_P \tilde{x}_P = b$.)

There are two cases. Case I is when the columns corresponding to the indices in P are linearly independent. I.e., the columns of A_P are linearly independent. Since A has full rank (and so contains m linearly independent columns), if needed we can add some more columns

from $[n] \setminus P$ to P, to get a set B with $|B| = m$ such that A_B's columns are also linearly independent, and so A_B is invertible. Now consider

$$A_B x_B = b.$$

There is a unique solution x_B to this equation (but we don't know if that satisfies $x_B \geq 0$.) No worries: we already know that $\tilde{x}_B$ is a solution to this equation, so it must be the unique solution. And since $\tilde{x}$ is feasible, it is the BFS corresponding to B.

In case II, suppose the columns of A_p are not linearly independent, so there exists some (not all zero) coefficients w_i such that

$$\sum_{j \in P} w_j A_j = 0 \qquad \text{or, equivalently} \qquad A_p w_P = 0.$$

By setting $w_j = 0$ for all $j \notin P$, we get a vector w which is itself non-zero, but where $Aw = 0$. Hence, if we consider the vector $y = \tilde{x} - \epsilon\, w$, note that

$$Ay = A(\tilde{x} - \epsilon w) = b - \epsilon\, 0 = b.$$

Moreover, since w is non-zero only in the coordinates in P, and x is strictly positive in those coordinates, then for small enough ϵ, we know that $y = \tilde{x} - \epsilon w \geq 0$. So y is also a feasible solution for small enough epsilon.

Suppose, fortuitously, $c^T w = 0$. Then $c^T y = c^T(\tilde{x} - \epsilon w) = c^T \tilde{x}$. We can assume that w has some positive entry, else we can negate all entries of w. So as we increase ϵ, we are decreasing some of the (positive) entries in y, without changing the objective function. And at some point we will make some entry zero, contradicting that $\tilde{x}$ had the fewest non-zeroes among all x such that $c^T x \leq c^T x^*$.

Now suppose $c^T w > 0$ (if $c^T w < 0$, we can just negate all entries of w to reduce to this case). Again, if there existed one positive w_j, we could do the same argument as above, and get a contradiction. But maybe $c^T w > 0$ and all of the entries of w are negative. (Now flipping the signs does not help.) But this is now easy: note that now $y = \tilde{x} - \epsilon w$ is non-negative and hence feasible for *all* $\epsilon \geq 0$. Moreover, the objective function value $c^T y = c^T \tilde{x} - \epsilon(c^T w)$ goes to $-\infty$ as $\epsilon \to \infty$. This contradicts the fact that the LP has a bounded optimum, and we are done! □

Note: Ankit asked a question about how Theorem 1.3 helped solve an equational form LP in time $\binom{n}{m}$. Specifically, his question was this: Theorem 1.3 says that if the LP is feasible and bounded, then the optimum is achieved at a BFS (and we could try all of them). But what if the LP was unbounded or infeasible? How could we detect those cases in the same amount of time? Here's one way.

To start, Fact 2.1 says any equational form LP that is feasible has a BFS. So if all the basic solutions are infeasible (i.e., there is no BFS), we can safely answer "Infeasible".

So now it suffices to differentiate between the bounded/unbounded subcases. So consider the BFS that achieves the lowest objective function value among all the BFSs (assuming the LP is a minimization LP). We know the optimal value is either this value (call it δ), or it is $-\infty$. Consider the LP obtained by adding in the new constraint $c^\top x = \delta - 1$. This is another equational form LP with $m + 1$ constraints, and we can use the above argument to decide its feasibility. If this new LP is feasible, the original LP had optimum value $latex - \infty$, else the optimum of the original LP was $latex\delta$.