COMS 4995-8: Advanced Algorithms (Spring'21) Mar 16, 2021

Lecture 17: Linear Programming

Instructor: *Alex Andoni* Scribes: *Xingjian Zhao, Viren Bajaj*

1 Linear Programming

Let's state the problem of Linear Programming (LP) that was introduced last time. In the *general form* of LP, we are given

$$A \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^n \text{ and unknown } \mathbf{x} = [x_1, ..., x_n] \in \mathbb{R}^n$$

And we have to minimize the objective function

$$\min \mathbf{c}^\intercal \mathbf{x}$$

such that

$$A\mathbf{x} \geqslant \mathbf{b}$$

In this set up,

- $A\mathbf{x} \geqslant \mathbf{b} \Leftrightarrow (\ A_1\mathbf{x} \geqslant \mathbf{b}_1 \wedge ... \wedge A_m\mathbf{x} \geqslant \mathbf{b}_m)$
- n is the number of variables
- m is the number of constraints

Remarks:
Notice that this form can covers the case where we want to maximize the objective function and captures all linear constraints. This is because of following equivalence relations:

1. $\max_x[f(x)] = -\min_x[-f(x)]$
2. $A_i\mathbf{x} \leqslant \mathbf{b}_i \Leftrightarrow (-A_i)\mathbf{x} \geqslant -\mathbf{b}_i$
3. $A_i\mathbf{x} = \mathbf{b}_i \Leftrightarrow (A_i\mathbf{x} \leqslant \mathbf{b}_i \ \wedge \ A_i\mathbf{x} \geqslant \mathbf{b}_i)$

LP is a very powerful method since many problems can be reduced to it. For example, the 'max-flow' problem we saw in class previously.

1.1 Example: Max-Flow via LP

Lets briefly restate the problem in a manner that resembles our set up for LP.
The unknowns are the edge flow variables $f(u, v) \in \mathbb{R}$, $(u, v) \in E$

And we want to maximize the objective function:

$$\max \left[\left(\sum_{u:(s,u)\in E} f(s,u) \right) - \left(\sum_{u:(u,s)\in E} f(u,s) \right) \right]$$

Subject to the constraints:

$$f(u,v) \geqslant 0,\ \forall (u,v) \in E$$

$$f(u,v) \leqslant c_{u,v},\ \forall (u,v) \in E$$

$$\left(\sum_{v:(u,v)\in E} f(u,v) \right) - \left(\sum_{v:(v,u)\in E} f(v,u) \right) = 0,\ \forall v \neq s,t$$

Here the number of variables equals the number of edges which is m.

The number of constraints equals $m+m+(n-2) = 2m+n-2$, where n is the number of nodes and m is the number of edges, since the first two constraints are on each edge and the third constraint is on every node except for the source (s) and the sink (t).

Looking at the objective function we can deduce $\mathbf{c} \in \mathbf{R}^m$, $\mathbf{c}_{u,v} \in \{\pm 1, 0\}$. The right hand sides of the constraint equations make up $\mathbf{b}$. This puts the problem in the form of LP.

Our objective is to talk about the algorithm design for an LP solver. To achieve this goal, we must first talk about the structure of solutions to LP.

2 Structure of Solutions to LP

Definition 1. *Feasible set: set of all values of* $\mathbf{x}$ *that satisfies all the constraints.*

$$\begin{aligned} F &= \{\mathbf{x} : A\mathbf{x} \geqslant \mathbf{b}\} \\ &= \{\mathbf{x} : & A_1\mathbf{x} \geqslant \mathbf{b_1}, \\ & & A_2\mathbf{x} \geqslant \mathbf{b_2}, \\ & & . \\ & & . \\ & & A_m\mathbf{x} \geqslant \mathbf{b_m}\} \end{aligned}$$

Notice that each constraint $\{A_i\mathbf{x} = \mathbf{b_i}\}$ forms a hyper-plane that splits the space into two half spaces, with our solution lying in the half space $\{x_i : A_i\mathbf{x} \geqslant \mathbf{b_i}\}$. The intersection of all m half spaces forms a *polytope*, which is the feasible set F. An example of the polytope and F can be seen in Fig 1.
The nature of F determines the fate of the solution to LP, x^*. Based on the problem definition, one of the following three scenarios may occur:

1. $F = \varnothing \Rightarrow$ No solution exists because the problem has conflicting constraints. (e.g. adding $x_1 \leq 1$ to example in Fig 1)

2. F is unbounded (e.g. remove $x_1 \leq 2$ from example in Fig 1)

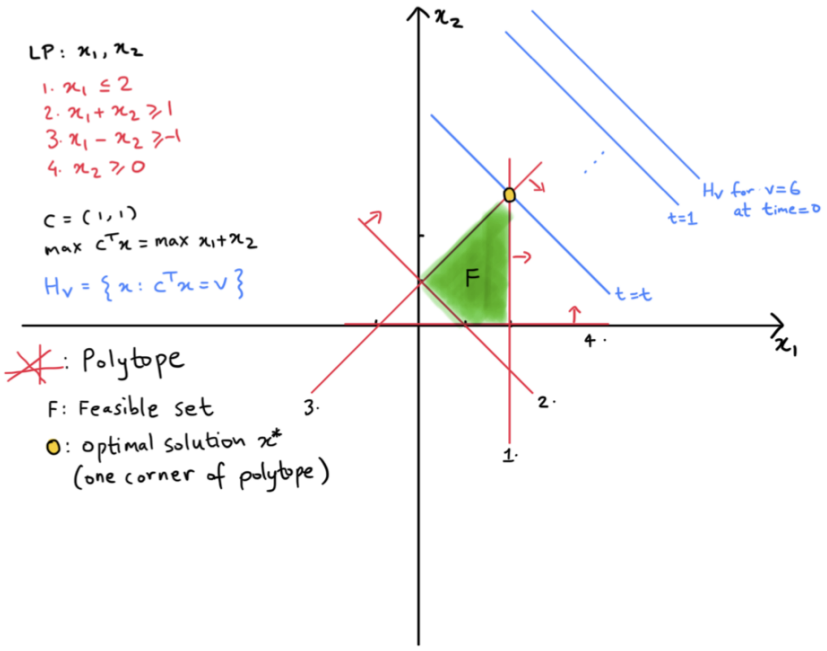

Figure 1: Geometric illustration of an LP Problem and Algorithm 1.

$$\Rightarrow \begin{cases} \text{The solution is infinite (e.g., if optimization was max } x_1 + x_2 \rightarrow x_1 = \infty, x_2 = 0), \text{ or} \\ \text{The solution is finite (e.g., if optimization was min } x_1 + x_2 \rightarrow x_1 = 0, x_2 = 1) \end{cases}$$

3. F is finite (bounded) $\Rightarrow$ The solution must be finite.

Now let's look into finding solutions to LP.

3 Finding Solutions to LP

In this section we are trying to answer the question: given an objective function $f(\mathbf{x}) = \mathbf{c}^{\mathrm{T}}\mathbf{x}$, how do we find the optimal value(s) for $\mathbf{x}$ that minimize (or maximize) the objective function?
Our first attempt to come up with a algorithm will be one that doesn't create a very realistic algorithm because it is difficult to implement. However, it will help us understand what solutions look like and conceptually lead us to more practical algorithms.

Algorithm 1

input $A \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^n$.

1: Guess $v = \min \mathbf{c}^{\mathrm{T}}\mathbf{x}$ such that $A\mathbf{x} \geqslant \mathbf{b}$. Notice, by construction, $v = \mathbf{c}^{\mathrm{T}}\mathbf{x}$.
2: Consider the hyperplane $H_v = \{\mathbf{x} : \mathbf{c}^{\mathrm{T}}\mathbf{x} = v\}$
3: Take v to be a very large value.
4: Decrease v continuously until $H_v \cap F \neq \phi$. In other words, move the hyperplane H_v towards the polytope F until they touch.
5: **return** any $\mathbf{x}^* \in H_v$, each of which is an optimal solution.

A geometric illustration of this algorithm can be found in Fig 1.

This algorithm is difficult to implement because we have to continuously decrease v and check whether H_v intersects F.

Notice that the first instance the hyperplane H_v intersects the feasible set F will be at a *corner* or vertex of the polytope F. We formally define a corner below.

Definition 2. *A corner/vertex of a polytope is defined by a linear system of n (the dimension of the space) equations of the form:*

$$\begin{aligned} A_{i_1}\mathbf{x} &= \mathbf{b_{i_1}} \\ A_{i_2}\mathbf{x} &= \mathbf{b_{i_2}} \\ &\cdot \\ &\cdot \\ A_{i_n}\mathbf{x} &= \mathbf{b_{i_n}} \end{aligned}$$

Definition 3. *For some $\mathbf{x} \in F$, if $A_i\mathbf{x} = \mathbf{b_i}$, the i^{th} constraint is tight.*

Thus the optimal solution x^* is in the corner of the polytope means that it is defined by the solution to a system of n tight constraints. Thus the number of optimal solutions can potentially be n, because we can take n different hyperplanes.

Based on this discussion, we can design an implementable but inefficient algorithm for solving LP as follows.

Algorithm 2

---

input $A \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^n$.

1: Try all possible corners/vertices.

- Try every $\binom{m}{n}$ sets $S = \{i_1, ...i_n\}$.
- Solve

$$\begin{aligned} A_{i_1}\mathbf{x} &= \mathbf{b_{i_1}} \\ A_{i_2}\mathbf{x} &= \mathbf{b_{i_2}} \\ &\cdot \\ &\cdot \\ A_{i_n}\mathbf{x} &= \mathbf{b_{i_n}} \end{aligned}$$

  Let the solution be $\mathbf{x}^S$.

2: Consider the hyperplane $H_v = \{\mathbf{x} : \mathbf{c}^{\mathrm{T}}\mathbf{x} = v\}$

3: Check if $\mathbf{x}^S$ is feasible by checking if $A\mathbf{x}^S \geqslant \mathbf{b}$ is satisfied.

4: **return** $\min \mathbf{c}^{\mathrm{T}}\mathbf{x^S}$ for an $\mathbf{x}^S$ that is feasible.

Runtime:

$$\begin{aligned}\text{The runtime of this algorithm is} = &\text{ number of corners of the polytope} * \\ &\text{(time to solve the system of n equations+} \\ &\text{time to check feasibility of } x^S + \\ &\text{computing the optimal value } c^\top x^S) \\ = &\binom{m}{n}([\text{time to solve } A_s x = b] + m \cdot n + n)\end{aligned}$$

Notice that $\binom{m}{n} \approx m^n$, which is very slow. Also note that it is not immediately clear that the time taken to write down a solution to the linear program can be done efficiently. For instance, if a solution involved the number pi, or other transcendental numbers, it might take some time to write down the solution. First, let's look at the time taken to solve a system of linear equations.

3.1 Time to solve a system of linear equations

Given $Ax \in b, A \in \mathbb{R}^{m \mathbf{x} n}, b \in \mathbb{R}^m$ and unknown $x \in \mathbb{R}^n$, we want to solve $Ax = b$.

3.1.1 Special case: A is a square matrix ($n = m$)

First, we define a well known theorem from linear algebra.

Theorem 4. *The following are equivalent:*

1. *A is invertible ($A^{-1}exists$)*
2. *$det(A) \neq 0$*
3. *Columns of A are linearly independent*
4. *Rows of A are linearly independent*
5. *$Ax = b$ has a unique solution. This also means, $x = A^{-1}b$.*

Some helpful relations:

1. $det(A) = \sum_{\pi:[n]\to[n]} sign(\pi) \prod_{i=1}^{n} A_{i,\pi(i)}$, where π is a permutation.
2. $x_i = \frac{det(A_i)}{det(A)}$, where A_i is A with the ith column replaced with the vector b.

Corollary 5. *Suppose A_{ij}, b_j are all integer, described by B bits, then the solution x_i is rational and needs only $O(nlogn + Bn)$ bits to describe.*

Proof. Using helpful relation 1, we can see that to describe x_i we need log(max value of $det(A_i)$) + log(max value of $det(A)$) bits.From helpful relation 2, we can see that $|det(A)| \leq n!(2^B)^n$. This is because there are $n!$ many permutations, the max value of each A_{ij} of length B bits is 2^B, which is multiplied n times. Since $n!$ can be upper bounded by n^n, we can say $|det(A)| \leq n^n 2^{Bn}$. This implies the number of bits needed to describe $x_i = O(log(n^n 2^{Bn})) = O(nlogn + Bn)$. □

This runtime bound on a solution of a system linear equations also translates to the runtime bound for solutions for linear programs.

Notice that since it takes at least time n to write down x_i, it will take time at least n^2 to write down any solution to a system of linear equations, and therefore any LP. This time is non-trivial.

3.1.2 General case: A is rectangular ($n \neq m$ or $det(A) = 0$)

Before we tackle this case we look at some more definitions.

Definition 6. *col(A) = set of columns of* $A \subset \mathbb{R}^m$

Definition 7. $span(col(A)) = \{y \in \mathbb{R}^m : \exists x \in \mathbb{R}^n | y = \sum_{i=1}^{n} x_i A_i = Ax\}$

Note that $span(col(A))$ is the space of all possible vectors that equal Ax. This is important because we want to solve the equation $Ax = b$. Therefore

- if $b \notin span(col(A))$ then there is no solution, and
- if $b \in span(col(A))$ then one or more solutions exist.

Now, given a solution exists, how do we find it?
Consider the following sets.

Definition 8. S *= set of linearly independent vectors from col(A)*

Definition 9. $\bar{S}$ *= completion of S to a basis*

Thus, $span(S) = span(col(A)) \subseteq \mathbb{R}^m$ and $|\bar{S}| = m$.
We can index entries of $\bar{S}$ and write $S = \{\bar{S_1}, \ldots, \bar{S_m}\}$. Now, if we solve

$$\left[\bar{S_1}, \ldots, \bar{S_m}\right] \begin{bmatrix} x' \\ y \end{bmatrix} = b$$

where x' are the entries of the vector x that will multiply with the columns of A, and y are the entries that will multiply with the columns that form the completion, i.e. $\bar{S} \backslash S$.
Since $\bar{S}$ is a basis of $\mathbb{R}^m$, there exists a unique solution to the equation above $\begin{bmatrix} x' \\ y \end{bmatrix}$

Then our solution to the system of equations becomes:

$$x_i^* = \begin{cases} x_i', & \text{if if col } i \text{ of } A \text{ is in } S \\ 0, & \text{otherwise} \end{cases}$$

The remaining portion of this topic will be covered in the next lecture.