COMS 4995-8: Advanced Algorithms (Spring'21) Mar 18, 2021

Lecture 18: Linear Programming Continued

Instructor: *Alex Andoni* Scribes: Shengtan Mao, Yanhao Li

1 Systems of linear equations

Given a matrix $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, we want to find $x \in \mathbb{R}^n$ such that

$$Ax = b.$$

1.0.1 When $n \neq m$ or $\det A = 0$

Let S be the set of linearly independent columns from A. We write it as

$$S = \{S_1, S_2, \ldots, S_k\} \subset \mathbb{R}^m$$

Let $\bar{S} = \{S_1, \ldots, S_k, S_{k+1}, \ldots, S_m\}$ be the completion of S to a basis. Consider

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

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. Since $b \in \text{Span}(S)$, there exists a unique x' such that

$$\begin{bmatrix}\bar{S_1}, \ldots, \bar{S_k}\end{bmatrix} \begin{bmatrix} x' \end{bmatrix} = b.$$

This implies $y = 0$ because y would have contributed to the value of b if it was non-zero. Let $x = (x_1, \ldots, x_n)$ such that

$$x_i = \begin{cases} x_i', & \text{if } S_i \text{ is the } i^{th} \text{ column of } A \\ 0, & \text{otherwise.} \end{cases}$$

Then we have a unique solution to

$$Ax = b.$$

2 No solution to $Ax = b$

We want to be able to find a witness or certificate that there is no solution. The following claim, a variant of the Farkas' lemma, tells us that proving no solution is equivalent to solving another linear system of equations.

Claim 1. *$Ax = b$ has no solution if and only if there exists a $y \in \mathbb{R}^m$ such that $y^T A = 0$ and $y^T b \neq 0$.*

Proof. First we suppose $Ax = b$ has no solution. This implies

$$b \notin \{Ax : x \in \mathbb{R}^n\} = \text{Span}(\text{Col}(A)).$$

Let $\text{Proj}_A b$ be the projection of b on space Span(Col(A)). Let $y \triangleq b - \text{Proj}_A b$. Then we have $y \perp$ Span(Col(A)), so $y^T A = 0$. But we also have,

$$y^T b = y^T(y + \text{Proj}_A b) = ||y||^2 \neq 0.$$

Note that since y is non-zero, it can be rescaled so that $y^T b = 1$.

Now suppose there exists a $y \in \mathbb{R}^m$ such that $y^T A = 0$ and $y^T b \neq 0$. We proceed by contradiction and suppose there exists $x \in \mathbb{R}^n$ such that $Ax = b$. We then have

$$0 = 0x = (y^T A)x = y^T(Ax) = y^T b \neq 0.$$

Which is a contradiction. □

Note that we can find y by solving the following system of equations:

$$\begin{bmatrix} A^T \\ b^T \end{bmatrix} y = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \in R^{n+1}.$$

3 Back to linear programming

Let $c, x \in \mathbb{R}^n$, and $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$. Consider the two formulations of linear programming. The general form is

$$\text{minimize } c^T x \text{ such that } Ax \geq b,$$

and the standard form is

$$\text{maximize } c^T x \text{ such that } Ax = b \text{ where } x \geq 0.$$

Note these two forms are equivalent by considering $-\min(-c^T)x$.

We fix $x^+, x^- \in (R)$ where $x^+, x^- \geq 0$ and $x^+ - x^- = x$. $Ax \geq b$ if and only if $A(x^+ - x^-) \geq b$. For each constraint, we have

$$\begin{bmatrix} A_i & -A_i \end{bmatrix} \begin{bmatrix} x^+ \\ x^- \end{bmatrix} \geq b_i$$

where A_i is the i^{th} row of the matrix. We introduce new slack variables $\xi_i \geq 0$ so that

$$A_i(x^+ - x^-) \geq b_i \text{ if and only if } A_i(x^+ - x^-) - \xi_i = b_i.$$

We derived the form:

$$\text{maximize } c^T(x^+ - x^-) \text{ such that } A(x^+ - x^-) - \xi = b \text{ where } x^+, x^-, \xi \geq 0.$$

3.1 Basic definitions for linear programming

Definition 2. *An inequality is tight if the equality holds.*

Definition 3. *A real vector $x \in V$ is a convex combination of $y^1, ..., y^n \in V$ if there exists $\alpha_1, ..., \alpha_n \geq 0$ in $\mathbb{R}$ such that*

$$x = \sum_{i=1}^{n} \alpha_i y^i \ , \ \sum_{i=1}^{n+1} \alpha_i = 1.$$

Definition 4. *A solution $x \in F$, where $F = \{x : Ax = b, x \geq 0\}$, is a basic feasible solution if it is not a convex combination of other points in F;*

Fact 5. *Basic feasible solutions are vertices of the polytope F.*

Claim 6. *If x^* is a linear programming solution to a system of n equations that is feasible and bounded, then there exists an optimal solution that is a basic feasible solution.*

Proof. Suppose x^* is not a basic feasible solution. Then there are less than n tight linearly independent constraints. Let C be the space defined by these $n-1$ tight linearly independent constraints. Then $\dim C \geq 1$. So there exists a direction $d \in \mathbb{R}^n, d \neq 0$ such that

$$x^* + \alpha d \in C, \forall \alpha \in \mathbb{R}.$$

Consider a small $\epsilon > 0$, Then $x^* \pm \epsilon d \in C$. Note that C take care of all the tight constraints on x^* All other constraints have the form $x_i > 0$. This implies there exists a small enough $\epsilon > 0$ such that $x^* \pm \epsilon d$ is still feasible. Now we consider

$$c^T(x^* \pm \epsilon d) = c^T x^* \pm \epsilon c^T d.$$

Since x^* is optimal, $c^T d = 0$. One of the directions decreases some coordinates of x^*, we want to push as much as allowed so that some coordinate of $x^* \pm \epsilon d$ becomes 0. This makes one more constraint tight. We iterate this process until we have n tight linearly independent constraints. We have found a basic feasible solution. □

As a summary, x is a vertex if and only if x it is basic feasible solution if and only if x has n tight linearly independent constraints.

3.2 Enumerate all vertices

This was covered in the lecture 17.

3.3 Simplex Algorithm:

Algorithm 1

Choose $x^0 \in F$, starting vertex, $t = 0$
while true **do**
  Take $N(x^t) = \textit{vertices}$ neighboring x^t(Vertices s.t. differ from x^t in only 1 tight constraint.)
  Choose $y \in N(x^t)$ s.t. $c^T y < c^T x^t$
  if y exists **then**
    Set $x^{t+1} = y$
    $t = t + 1$
  else
    return x^t
  end if
end while

$N(x)$ means the neighborhood of x. Since in high dimensional space we might provide more choices and more options to go to a better position, how can we choose y?

Pivoting rule:

To solve the problem of choosing y, firstly, we need to know how to find the starting point $x^0 \in F$. It's a common trick that to find a starting point for this, we will solve another linear programming which is easier to initialize. We have a general form that:

$$\begin{array}{ll} \text{minimize} & c^\top x \\ \text{subject to} & Ax \leq b \end{array}$$

For finding a feasible vertex point, we just care about the $Ax \leq b$ constraint.
To achieve this, we want to minimize t subject to the constraint that

$$Ax - t \cdot \mathbb{1} \leq b$$

Note that for optimal t that $t > 0$, the original LP is infeasible, which means $F = \emptyset$.
For optimal t that $t \leq 0$, x is a starting vertex.
Thus, we can initialize this by setting $x = 0$ and $t = -\min_i b_i$.

Remark 7. *Taking y^* which has the best improvement is not optimal. In other words, being greedy(go to the neighbor that improves the objective function best) is not necessarily the best strategy.*

Remark 8. *Simplex Algorithm takes exponential time for worst pivoting rules that we know.*

Remark 9. *In practice, Simplex Algorithm works well.*

Conjecture 10. $\{$*Hirsch Conjecture* $'57\}$*: For any starting vertex in polytope F and any other optimal vertex V, there exists a shortest path of length which is poly(n, m).*

The Hirsch Conjecture provides a really tight upper bound ($\leq m - n$) for the shortest path. However, it is disproved in $\{$Santos $'10\}$.

Smoothed Analysis $\{$Spielman-Teng $'00s\}$:
Consider the instance of LP:

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax \leq b \end{aligned}$$

Consider the instance$'$:

$$A' = A + \textit{gaussian noise}$$

$$b' = b + \textit{gaussian noise}$$

(c can remain as is.) Then Apply Simplex Algorithm (with a specific pivot rule) on

$$\begin{aligned} \text{minimize} \quad & c'^\top x \\ \text{subject to} \quad & A'x \leq b' \end{aligned}$$

which runs in polynomial time.

4 Acknowledgements

We referenced to the lecture notes, scribe notes from lecture 18 of the 2017 Advanced Algorithms course, scribe notes from lecture 12 to 14 from the 2020 Advanced Algorithms course.