CS 224: Advanced Algorithms Spring 2017

Lecture 19 — March 28, 2017

Prof. Jelani Nelson *Scribe: Nicholas Hasselmo*

1 Overview

In the last lecture we covered least squares regression.

In this lecture we cover the first of three algorithms for solving linear programs: the Simplex Algorithm, which was first described by George Dantzig in 1947 [1].

In following lectures we will cover the ellipsoid method, which runs in linear time but is very slow and not used much in practice, and Interior Point Methods, which are efficient and often used in practice.

2 Simplex

Simplex is more of a family of algorithms. There are examples that force each member of the family to take exponential time; however, the problems people solve in real life usually run more efficiently.

2.1 LP formulation

2.1.1 Before

A linear program is a quantity we want to maximize/minimize by changing x, and a set of linear constraints on x:

$$\begin{array}{rlll} \min(/\max) & c^\top x & & \\ s.t. & \langle a_i, x\rangle & \geq & b \\ & & \leq & b \\ & & = & b \end{array}$$

Any LP can be written

$$\begin{array}{rlll} \min & c^\top x & & \\ s.t. & Ax & \geq & b \\ & x & \geq & 0 \end{array}$$

Recall that we can use "slackness" to go between an inequality constraint and an equality constraint.

$$\begin{aligned} &\forall i,\ \langle a_i, x\rangle \geq b_i \\ &\implies \langle a_i, x\rangle - s_i = b_i \end{aligned}$$

$$\text{“slack”} \rightarrow s_i \geq 0$$

2.1.2 Simplex LP

The form we use for simplex is

$$\begin{array}{lrcl} \min & c^\top x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$

subject to the constraints

- $A \in \mathbb{R}^{m \times n}$
- $n \geq m$
- the rows of A are linearly independent

3 Simplex Algorithm

Simplex is a greedy algorithm that performs the following:

- Start at some vertex (it doesn't matter which).
- While there exists a neighboring vertex improving $c^\top v$, move to v.
- Halt.

An LP looks like

$$\min\ c^\top x,\ s.t.\ x \in P$$

Definition 1. *$x \in \mathbb{R}^n$ is **feasible** if $x \in P$*

Definition 2. *x is a **vertex** if*

$$(x + y \in P)\ and\ (x - y \in P) \implies y = 0$$

If we can find an x that lies in P, we can solve the linear program. If we have some guess r as the output of our LP, we can add the constraint $c^\top x \leq r$ and check if the new LP is satisfiable. If it is, we have an upper bound for the minimum, and we can continue making guesses with a binary search to solve the LP.

The binary search will terminate because the optimal x has finite precision (we assume it is a vertex computed by solving a linear system on A), and the search finds at least a bit of information at each step.

Quesitons:

1. How do we get a starting vertex?
2. Why is OPT always achieved at a vertex?
3. Why might we not get stuck at a local OPT?

3.1 Obtain a starting vertex

We can obtain a starting vertex using the following LP:

$$\begin{aligned} \min\ & t \\ \text{s.t. } & Ax = (1-t)b \\ & x \geq 0 \\ & 0 \leq t \leq 1 \end{aligned}$$

An easy starting vertex is $(\vec{x}, t)$ with $\begin{matrix} x = 0 \\ t = 1 \end{matrix}$, because this is clearly feasible as $Ax = 0 = (1-t)b$, and is clearly a vertex because $x - y, x + y \in P$ means y and $-y$ must be greater than 0. If the original LP is feasible, then the constraint can be met with $t = 0$, and we will eventually output $(\vec{x}^*, 0)$.

We use simplex on the above LP, starting with $x = 0, t = 1$.

3.2 OPT is always achieved at a vertex

Definition 3. *P is **feasible** if $\exists x \in P$*

Definition 4. *P **bounded** if $\inf c^\top x, x \in P$ is finite (else, **unbounded**)*

Claim 5. *If (c, P) bounded, then*

$$\forall x \in P,\ \exists \text{ vertex } x' \in P \text{ s.t. } c^\top x' \leq c^\top x$$

Proof.

- We are done if x is a vertex (set $x' = x$).
- Otherwise, x not vertex $\implies \exists y \neq 0$ s.t. $\begin{matrix} x + y \in P \\ x - y \in P \end{matrix}$

  Note that $Ay = 0$ since $\begin{matrix} A(x+y) = b \\ A(x-y) = b \end{matrix}$ and we know $Ax = b$ for $x \geq 0$.
- wlog $c^\top y \leq 0$ (if not, take $-y$, as either y or $-y$ will satisfy this)

- if $c^\top y = 0$, choose y s.t. $\exists j$ with $y_j < 0$:
  - Case 1: $\exists j$ s.t. $y_j < 0$
    - Look at $x(t) = x + ty$ for $t > 0$. Note $x_i = 0 \implies y_i = 0$.
    - Let $J = \{j : y_j < 0\}$
      - $|J| \geq 1$
      - $x_J > 0$ $(\forall j \in J, x_j > 0)$
    - pick

$$t^* = \min_{j \in J} \left| \frac{x_j}{y_j} \right|$$

    - Note $x(t^*) \in P$, and $t^* > 0$.
    - $|\mathsf{supp}(x(t^*))| < |\mathsf{supp}(x)|$
    - $|\mathsf{supp}(x)|$ starts at $\leq n$ and goes down by ≥ 1 on each step $\implies$ case 1 can happen only $\leq n$ times.

$$(c^\top x(t^*) \leq c^\top x \text{ because } c^\top y \leq 0)$$

  - Case 2: $y \geq 0 \implies c^\top y < 0$

$$\implies x + ty \text{ feasible for all } t \geq 0$$

We can make $c^\top y$ arbitrarily small as $t \to \infty$:

$$c^\top x(t) = c^\top x + t \cdot \underbrace{c^\top y}_{<0}$$

$$\implies \text{case 2 impossible if } (c, P) \text{ bounded}$$

□

3.3 Simplex doesn't get stuck at local **OPT**

- Given vertex $x \in \mathbb{R}^n$, define $B_x = \{j \in [n] : x_j \neq 0\}$ (B_x is the "basis" corresponding to vertex x)
- A is a matrix with columns $A^1, \ldots, A^n$,

$$Ax = \sum_{i=1}^{n} x_i A^i = A_{B_x} x_B$$

Claim 6. $\overset{(i)}{\boxed{\textit{x is a vertex}}}$ *iff* $\overset{(ii)}{\boxed{\textit{columns of } A_{B_x} \textit{ are linearly independent}}}$.

(A_{B_x} is the $mx|B_x|$ submatrix of A obtained by taking cols in B_x)

Proof. – $(\bar{i}) \implies (\overline{ii})$

$$\begin{aligned} &\text{x not a vertex} \\ &\implies \exists y \neq 0,\ x + y, x - y \in P \\ &\implies Ay = 0 \end{aligned}$$

$$\begin{aligned} x_i = 0 \implies y_i &= 0 \text{ (so } B_y \subseteq B_x) \\ Ay &= A_{B_y} y \\ &= A_{B_x} y = 0 \end{aligned}$$

✓

- $(\overline{ii}) \implies (\bar{i})$
  A_{B_x} has linearly dependent columns
  $\implies \exists y \in \mathbb{R}^{|B_x|}$ with $A_{B_x} y = 0$
  * extend y to $\mathbb{R}^n$ by putting 0's in $[n] \setminus B_x$
  * $Ay = 0 \implies \forall t \geq 0, A(x+ty) = b = A(x - t'y)$
  * set t small enough so that $x + ty \geq 0, t > 0$
    $\implies x$ not vertex.

✓

□

3.4 Rewriting the LP

- Vertices come from subset of $\leq m$ linearly independent columns.
- For vertex $v \in P$, will have associated "basis" $B \subseteq [n]$ indexing linearly independent columns

$$A_B v_B = b$$

- Simplex pads all bases to have size exactly m, full rank (A_B is invertible $m \times m$ matrix)

Input LP

$$\begin{array}{lll} \min & c^\top x & \\ s.t. & Ax & \geq b \\ & x & \geq 0 \end{array}$$

Currently at vertex v with basis B ($N = [n] \setminus B$)

Rewrite

$$\begin{array}{ll} \min & c_B^\top x_B + c_N^\top x_N \\ \text{s.t.} & \boxed{A_B x_B + A_N x_N = b} \implies * \\ & x_B, x_N \geq 0 \end{array}$$

$$* \ \boxed{x_B = A_B^{-1} b - A_B^{-1} A_N x_N}$$

Rewrite LP again

$$\boxed{c_B^\top A_B^{-1} b - c_B^\top A_B^{-1} A_N x_N + c_N^\top X_N}$$
$$\downarrow$$
$$\min\ c_B^\top A_B^{-1} b + \underbrace{(c_N - A_N^\top (A_B^{-1})^\top c_B)^\top}_{\text{“}\tilde{c}_N\text{” (“reduced cost vector”)}} x_N$$
$$s.t.\ A_B x_B + A_N x_N = b$$
$$x_B, x_N \geq 0$$

3.5 Algorithm:

1. We have vertex v and its basis B
2. Rewrite LP as above
3. If $\tilde{c}_N \geq 0$, halt and return x since we are at OPT. ($x = A_B^{-1} b$ is optimal)
4. If $\exists j \in [n]$ such that $\tilde{c}_j < 0$ then increment x_j while maintaining feasibility.

While incrementing x_N it I am implicitly changing x_B. I kick out all people in the basis who became 0, and add j.

One worry: Since we padded B to be of size m there may be $i \in B$ s.t. $x_i = 0$

- Thus we might not be able to increment x_j for $j \in N$ by any nonzero amount without violating the non-negativity contraint $x_B \geq 0$. (taking j with $(\tilde{c}_N)_j < 0$)

- In degenerate cases, add j to B then choose which $k \in B$ to remove.

- Need to do carefully to avoid ∞-loops.

On each iteration of the simplex method we must choose which elements to pivot on. We can choose a “pivoting rule” such as “Bland’s Rule”, developed by Robert Bland in 1977 [2], which provably does not have infinite loops.

3.6 Runtime

Each iteration is poly(m, n) time.

$$\#\text{iters} \leq \#\text{vertices} \leq \binom{n}{m} \geq \left(\frac{n}{m}\right)^m$$
$$\#\text{vertices} \leq \binom{n - \frac{m}{2}}{\frac{m}{2}}$$

by the “Upper bound theorem”, which was proved by Peter McMullen in 1970 [3].

For every way of choosing j, there are examples that run in exponential time.

4 Next Time

We will use Simplex to prove strong duality.

Finally, we will cover the ellipsoid algorithm and begin covering Interior Point Methods.

References

[1] George B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. T.C. Koopmans (ed.): Activity Analysis of Production and Allocation, New York-London 1951 (Wiley & Chapman-Hall), pp. 339-347, 1947.

[2] Robert Bland. New Finite Pivoting Rules for the Simplex Method. *Mathematics of Operations Research*, 2(2):103–107, 1977.

[3] Peter McMullen. The maximum numbers of faces of a convex polytope. *Mathematika*, 17:179–184, 1971.