Lecture 24: Linear Programming Algorithms

Objectives of this lecture

- Understand why linear programs can be solved to high accuracy in polynomial time.
- Introduce cutting-plane methods, with the ellipsoid method as the main example.

1 Formal Setup for Linear Programming Analysis

In this lecture we focus on the dual formulation of a linear program:

$$\max\{b^\top y : A^\top y \le c\}.$$

Here $A^\top y \le c$ is interpreted coordinatewise. Our goal is not exact arithmetic; instead, we want an additive-ϵ approximation to the optimum in time polynomial in the input size and logarithmic in $1/\epsilon$.

Abstracting away the objective. Let

$$K := \{y : A^\top y \le c\}.$$

The key structural fact is that K is convex. Informally, convex means that for any $y_1, y_2 \in K$, the line segment connecting y_1 and y_2 lies inside K. Indeed, if $y_1, y_2 \in K$ and $\theta \in [0,1]$, then

$$A^\top(\theta y_1 + (1-\theta)y_2) = \theta A^\top y_1 + (1-\theta)A^\top y_2 \le \theta c + (1-\theta)c = c,$$

so $\theta y_1 + (1-\theta)y_2 \in K$.

Geometric parameters. For the remainder of the lecture, we will assume there are parameters $r, R > 0$ such that the convex set under consideration is either empty or contains a Euclidean ball of radius r, and is contained in the ball

$$B_R := \{x \in \mathbb{R}^d : \|x\|_2 \le R\}.$$

Baking the objective into the feasible set. To optimize $b^\top y$, we will binary search on a target value λ. For a guess λ, define

$$K_\lambda := \{y : A^\top y \le c,\ b^\top y \ge \lambda\}.$$

Then K_λ is again convex, by the same argument as above. Moreover,

1. if $K_\lambda \neq \emptyset$, then the optimum value is at least λ;
2. if $K_\lambda = \emptyset$, then the optimum value is less than λ.

Thus, if we can solve feasibility for sets of the form K_λ, then we can recover the optimum by binary search. Since all feasible points will lie in a ball of radius R, we also get the crude bound

$$|b^\top y| \le \|b\|_2 R,$$

so the binary search interval has length at most $2\|b\|_2 R$.

The weak feasibility problem. To make the geometry work cleanly, we will analyze the following promise problem. We are given a convex set $K \subseteq \mathbb{R}^d$ and parameters $r, R > 0$ such that

1. either $K = \emptyset$, or K contains a Euclidean ball of radius r;
2. $K \subseteq B_R := \{x \in \mathbb{R}^d : \|x\|_2 \le R\}$.

The task is to distinguish these two cases.

For linear programming, obtaining suitable values of r and R requires encoding arguments and a small amount of perturbation/rounding. We will not prove those facts in lecture. The important point for us is that one can choose r and R so that $\log(R/r)$ is polynomial in the bit complexity of the LP instance and logarithmic in $1/\epsilon$.

2 Separation Oracles

Above, we reduced linear programming to testing emptiness of a convex set that we only know implicitly. The correct abstraction for this type of access is a separation oracle.

Definition. A separation oracle for a convex set K is an algorithm that, given any point $x \in \mathbb{R}^d$, returns either

1. a certification that $x \in K$; or
2. a vector $d \neq 0$ such that
$$K \subseteq \{y \in \mathbb{R}^d : d^\top (y - x) \le 0\}.$$

So in the second case, the oracle gives a halfspace passing through x that still contains all of K. Geometrically, this is a certificate that $x \notin K$.

Why linear programs admit separation oracles. Consider the set

$$K_\lambda = \{y : A^\top y \le c,\ b^\top y \ge \lambda\}.$$

Given a point x, we can check the constraints directly.

1. If $A^\top x \le c$ and $b^\top x \ge \lambda$, then $x \in K_\lambda$.
2. Otherwise, if some coordinate j violates the inequality $a_j^\top x \le c_j$ (where a_j is the jth column of A), then for every $y \in K_\lambda$ we have

   $$a_j^\top (y - x) \le c_j - a_j^\top x < 0.$$

   Hence the halfspace

   $$\{y : a_j^\top (y - x) \le 0\}$$

   contains K_λ.
3. If all inequalities $A^\top x \le c$ hold but $b^\top x < \lambda$, then for every $y \in K_\lambda$ we have

   $$(-b)^\top (y - x) = b^\top x - b^\top y \le b^\top x - \lambda < 0,$$

   so the halfspace

   $$\{y : (-b)^\top (y - x) \le 0\}$$

   contains K_λ.

Therefore K_λ admits an efficient separation oracle: we simply look for a violated inequality and use its normal vector.

3 The Ellipsoid Method

The ellipsoid method is a particular cutting-plane method. The general template is:

1. Maintain a simple outer approximation E_t with $K \subseteq E_t$.
2. Query the center z_t of E_t.
3. If $z_t \in K$, stop.
4. Otherwise, use the separation oracle to obtain a halfspace through z_t that still contains K.
5. Replace E_t by a new simple set E_{t+1} that contains the cut set $E_t \cap H_t$.

The only question is how to choose the outer approximation family. In the ellipsoid method, each E_t is an ellipsoid.

Ellipsoids. For a positive definite matrix $Q \in \mathbb{R}^{d \times d}$, define

$$\mathcal{E}(z, Q) := \{x \in \mathbb{R}^d : (x - z)^\top Q^{-1} (x - z) \le 1\}.$$

This is an ellipsoid centered at z. The initial ellipsoid will be

$$E_0 = B_R = \mathcal{E}(0, R^2 I).$$

Central cuts. At iteration t, suppose we have an ellipsoid $E_t = \mathcal{E}(z_t, Q_t)$ containing K. We query the center z_t. If $z_t \notin K$, the separation oracle returns a vector d_t such that

$$K \subseteq \{y : d_t^\top (y - z_t) \le 0\}.$$

Because the boundary hyperplane passes through the center z_t, this is called a *central cut*.

The key geometric claim. The key claim behind the ellipsoid method is the following: if an ellipsoid is intersected with a halfspace through its center, then the surviving set is contained in another ellipsoid whose volume is smaller by a definite factor. Repeating this claim is what forces progress.

Reducing to a canonical picture. To see why the claim is true, start with an ellipsoid

$$E = \mathcal{E}(z, Q)$$

and a halfspace through its center

$$H = \{x : g^\top (x - z) \le 0\}.$$

Apply the linear change of variables

$$u = Q^{-1/2}(x - z).$$

This sends E to the unit ball

$$B := \{u \in \mathbb{R}^d : \|u\|_2 \le 1\},$$

and it sends H to another halfspace

$$\{u : h^\top u \le 0\}, \qquad h := Q^{1/2} g.$$

Now rotate coordinates so that h points in the e_1 direction. Rotations preserve both the unit ball and volume, so it is enough to analyze the very special set

$$B \cap \{u_1 \le 0\}.$$

Writing down the new ellipsoid explicitly. Consider the ellipsoid

$$E^\star := \left\{ u \in \mathbb{R}^d : \frac{(u_1 + \frac{1}{d+1})^2}{\left(\frac{d}{d+1}\right)^2} + \frac{d^2 - 1}{d^2} \sum_{i=2}^{d} u_i^2 \le 1 \right\}.$$

We claim that

$$B \cap \{u_1 \le 0\} \subseteq E^\star.$$

Indeed, if $u \in B$ and $u_1 \le 0$, then

$$\sum_{i=2}^{d} u_i^2 \le 1 - u_1^2.$$

Therefore the left-hand side in the definition of $E^\star$ is at most

$$\frac{(u_1+\frac{1}{d+1})^2}{\left(\frac{d}{d+1}\right)^2}+\frac{d^2-1}{d^2}(1-u_1^2).$$

A direct expansion shows that this equals

$$1+\frac{2(d+1)}{d^2}u_1(u_1+1).$$

Since $u_1 \in [-1,0]$, we have $u_1(u_1+1) \le 0$, so the above quantity is at most 1. This proves the containment.

Why the volume goes down. The semi-axis lengths of $E^\star$ are

$$\frac{d}{d+1}, \quad \frac{d}{\sqrt{d^2-1}}, \quad \dots, \quad \frac{d}{\sqrt{d^2-1}}.$$

Hence

$$\frac{\mathrm{vol}(E^\star)}{\mathrm{vol}(B)}=\frac{d}{d+1}\left(\frac{d}{\sqrt{d^2-1}}\right)^{d-1}.$$

This quantity is strictly less than 1, and in fact it is at most $e^{-c/d}$ for some absolute constant $c>0$. Thus the containing ellipsoid has volume smaller by a definite multiplicative factor.

Undoing the rotation and the linear map gives an ellipsoid E_{t+1} such that

$$E_t \cap H_t \subseteq E_{t+1} \qquad \text{and} \qquad \mathrm{vol}(E_{t+1}) \le e^{-c/d}\,\mathrm{vol}(E_t).$$

Algorithm. The ellipsoid method for weak feasibility is now straightforward.

1. Start with $E_0 = B_R$.
2. At iteration t, let z_t be the center of E_t.
3. Query the separation oracle on z_t.
4. If the oracle says $z_t \in K$, output z_t and stop.
5. Otherwise, use the key geometric claim above to replace $E_t \cap H_t$ by a new ellipsoid E_{t+1} of smaller volume, and continue.

Why it terminates quickly. Assume $K \neq \emptyset$ and in fact contains a ball of radius r. Since every ellipsoid E_t contains K, it must also contain that ball, so

$$\mathrm{vol}(E_t) \ge \mathrm{vol}(B_r).$$

On the other hand, after T iterations,

$$\mathrm{vol}(E_T) \le e^{-cT/d}\,\mathrm{vol}(B_R).$$

Using $\mathrm{vol}(B_R) = R^d \mathrm{vol}(B_1)$ and $\mathrm{vol}(B_r) = r^d \mathrm{vol}(B_1)$, once

$$e^{-cT/d} R^d < r^d,$$

we get a contradiction. Rearranging, it suffices to take

$$T = O\left(d^2 \log \frac{R}{r}\right).$$

Therefore:

1. if K contains a ball of radius r, the algorithm must find a point of K within $O(d^2 \log(R/r))$ oracle calls;
2. if the algorithm runs for that many iterations without finding a feasible point, we can conclude that K is empty.

Consequence for linear programming. For linear programming, each oracle call is efficient, because it just checks the inequalities defining K_λ. Combining the ellipsoid method with binary search on λ gives an additive-ϵ approximation to the LP value.

More explicitly, the number of binary-search steps is

$$O\left(\log \frac{\|b\|_2 R}{\epsilon}\right),$$

so the total number of separation-oracle calls is

$$O\left(d^2 \log \frac{R}{r} \cdot \log \frac{\|b\|_2 R}{\epsilon}\right).$$

Since the oracle itself is polynomial-time and $\log(R/r)$ can be bounded by a polynomial in the input size and $\log(1/\epsilon)$, this yields a polynomial-time algorithm for approximately solving LPs.

Historically, Khachiyan's ellipsoid method was the first polynomial-time algorithm for linear programming. In practice, modern solvers are usually based on interior-point methods or simplex variants, but the ellipsoid method remains a foundational theoretical result.