15.081J/6.251J Introduction to Mathematical Programming

Lecture 18: The Ellipsoid method

1 Outline

Slide 1

- Efficient algorithms and computational complexity
- The key geometric result behind the ellipsoid method
- The ellipsoid method for the feasibility problem
- The ellipsoid method for optimization

2 Efficient algorithms

Slide 2

- The LO problem

$$\begin{array}{ll} \min & \boldsymbol{c'x} \\ \text{s.t.} & \boldsymbol{Ax} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{array}$$

- A LO instance

$$\begin{array}{lrcrcl} \min & 2x & + & 3y & & \\ \text{s.t.} & x & + & y & \leq & 1 \\ & x & , & y & \geq & 0 \end{array}$$

- A problem is a collection of instances

2.1 Size

Slide 3

- The **size** of an instance is the number of bits used to describe the instance, according to a prespecified format
- A number $r \leq U$

$$r = a_k 2^k + a_{k-1} 2^{k-1} + \cdots + a_1 2^1 + a_0$$

  is represented by $(a_0, a_1, \ldots, a_k)$ with $k \leq \lfloor \log_2 U \rfloor$
- Size of r is $\lfloor \log_2 U \rfloor + 2$
- Instance of LO: $(\boldsymbol{c}, \boldsymbol{A}, \boldsymbol{b})$
- Size is

$$(mn + m + n)\big(\lfloor \log_2 U \rfloor + 2\big)$$

2.2 Running Time

Slide 4

Let A be an algorithm which solves the optimization problem Π.
If there exists a constant $\alpha > 0$ such that A terminates its computation after at most $\alpha\, f(I)$ elementary steps for each instance I, then A runs in $\mathrm{O}(f)$ time.

Elementary operations are

- variable assignments
- random access to variables
- conditional jumps
- comparison of numbers
- arithmetic operations
- ...

Slide 5

A "brute force" algorithm for solving the min-cost flow problem:

Consider all spanning trees and pick the best tree solution among the feasible ones.

Suppose we had a computer to check 10^{15} trees in a second. It would need more than 10^9 years to find the best tree for a 25-node min-cost flow problem.
It would need 10^{59} years for a 50-node instance.

That's not efficient! SLIDE 6

Ideally, we would like to call an algorithm "efficient" when it is sufficiently fast to be usable in practice, but this is a rather vague and slippery notion.

The following notion has gained wide acceptance:

> An algorithm is considered efficient if the number of steps it performs for any input is bounded by a polynomial function of the input size.

Polynomials are, e.g., n, n^3, or $10^6 n^8$.

2.3 The Tyranny of Exponential Growth

SLIDE 7

	$100\,n \log n$	$10\,n^2$	$n^{3.5}$	2^n	$n!$	n^{n-2}
10^9/sec	$1.19 \cdot 10^9$	600,000	3,868	41	15	13
10^{10}/sec	$1.08 \cdot 10^{10}$	1,897,370	7,468	45	16	13

Maximum input sizes solvable within one hour.

2.4 Punch line

SLIDE 8

The equation

$$\text{efficient} \quad = \quad \text{polynomial}$$

has been accepted as the best available way of tying the empirical notion of a "practical algorithm" to a precisely formalized mathematical concept.

2.5 Definition

SLIDE 9

An algorithm runs in *polynomial time* if its running time is $O(|I|^k)$, where $|I|$ is the input size, and all numbers in intermediate computations can be stored with $O(|I|^k)$ bits.

3 The Ellipsoid method

SLIDE 10

- $\boldsymbol{D}$ is an $n \times n$ positive definite symmetric matrix
- A set E of vectors in $\Re^n$ of the form

$$E = E(\boldsymbol{z}, \boldsymbol{D}) = \left\{ \boldsymbol{x} \in \Re^n \mid (\boldsymbol{x} - \boldsymbol{z})' \boldsymbol{D}^{-1} (\boldsymbol{x} - \boldsymbol{z}) \leq 1 \right\}$$

is called an **ellipsoid** with center $\boldsymbol{z} \in \Re^n$

3.1 The algorithm intuitively

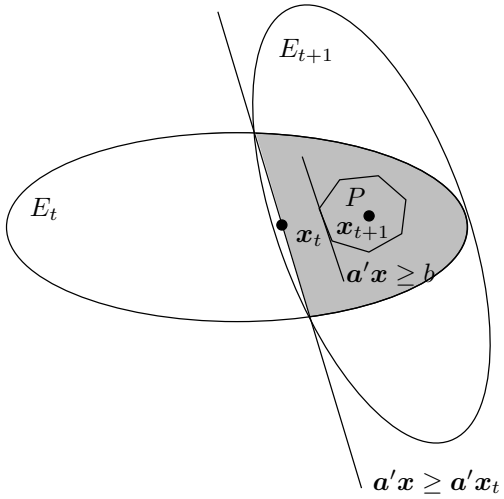
SLIDE 11

- Problem: Decide whether a given polyhedron

$$P = \left\{ \boldsymbol{x} \in \Re^n \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} \right\}$$

is nonempty

SLIDE 12



- Key property: We can find a new ellipsoid E_{t+1} that covers the half-ellipsoid and whose volume is only a fraction of the volume of the previous ellipsoid E_t

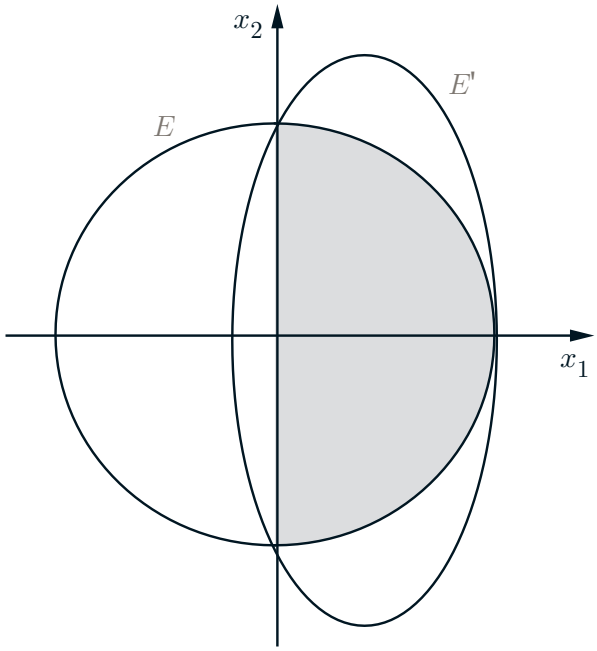
3.2 Key Theorem

SLIDE 13

- $E = E(\boldsymbol{z}, \boldsymbol{D})$ be an ellipsoid in $\Re^n$; $\boldsymbol{a}$ nonzero n-vector.
- $H = \{\boldsymbol{x} \in \Re^n \mid \boldsymbol{a}'\boldsymbol{x} \geq \boldsymbol{a}'\boldsymbol{z}\}$

$$\begin{aligned} \overline{\boldsymbol{z}} &= \boldsymbol{z} + \frac{1}{n+1} \frac{\boldsymbol{D}\boldsymbol{a}}{\sqrt{\boldsymbol{a}'\boldsymbol{D}\boldsymbol{a}}}, \\ \overline{\boldsymbol{D}} &= \frac{n^2}{n^2-1} \left(\boldsymbol{D} - \frac{2}{n+1} \frac{\boldsymbol{D}\boldsymbol{a}\boldsymbol{a}'\boldsymbol{D}}{\boldsymbol{a}'\boldsymbol{D}\boldsymbol{a}} \right). \end{aligned}$$

- The matrix $\overline{\boldsymbol{D}}$ is symmetric and positive definite and thus $E' = E(\overline{\boldsymbol{z}}, \overline{\boldsymbol{D}})$ is an ellipsoid
- $E \cap H \subset E'$
- $\mathrm{Vol}(E') < e^{-1/(2(n+1))}\ \mathrm{Vol}(E)$



3.3 Illustration

Slide 14

3.4 Assumptions

Slide 15

- A polyhedron P is **full-dimensional** if it has positive volume
- The polyhedron P is bounded: there exists a ball $E_0 = E(\boldsymbol{x}_0, r^2\boldsymbol{I})$, with volume V, that contains P
- Either P is empty, or P has positive volume, i.e., $\text{Vol}(P) > v$ for some $v > 0$
- E_0, v, V, are a priori known
- We can make our calculations in infinite precision; square roots can be computed exactly in unit time

3.5 Input-Output

Slide 16

Input:

- A matrix $\boldsymbol{A}$ and a vector $\boldsymbol{b}$ that define the polyhedron $P = \{\boldsymbol{x} \in \Re^n \mid \boldsymbol{a}_i'\boldsymbol{x} \geq b_i,\ i = 1, \ldots, m\}$
- A number v, such that either P is empty or $\text{Vol}(P) > v$

- A ball $E_0 = E(\boldsymbol{x}_0, r^2\boldsymbol{I})$ with volume at most V, such that $P \subset E_0$

Output: A feasible point $\boldsymbol{x}^* \in P$ if P is nonempty, or a statement that P is empty

3.6 The algorithm

SLIDE 17

1. (Initialization)
Let $t^* = \left\lceil 2(n+1)\log(V/v) \right\rceil$; $E_0 = E(\boldsymbol{x}_0, r^2\boldsymbol{I})$; $\boldsymbol{D}_0 = r^2\boldsymbol{I}$; $t = 0$.

2. (Main iteration)

- If $t = t^*$ stop; P is empty.
- If $\boldsymbol{x}_t \in P$ stop; P is nonempty.
- If $\boldsymbol{x}_t \notin P$ find a violated constraint, that is, find an i such that $\boldsymbol{a}_i'\boldsymbol{x}_t < b_i$.
- Let $H_t = \{\boldsymbol{x} \in \Re^n \mid \boldsymbol{a}_i'\boldsymbol{x} \geq \boldsymbol{a}_i'\boldsymbol{x}_t\}$. Find an ellipsoid E_{t+1} containing $E_t \cap H_t$: $E_{t+1} = E(\boldsymbol{x}_{t+1}, \boldsymbol{D}_{t+1})$ with

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t + \frac{1}{n+1}\frac{\boldsymbol{D}_t\boldsymbol{a}_i}{\sqrt{\boldsymbol{a}_i'\boldsymbol{D}_t\boldsymbol{a}_i}}, \\ \boldsymbol{D}_{t+1} &= \frac{n^2}{n^2-1}\left(\boldsymbol{D}_t - \frac{2}{n+1}\frac{\boldsymbol{D}_t\boldsymbol{a}_i\boldsymbol{a}_i'\boldsymbol{D}_t}{\boldsymbol{a}_i'\boldsymbol{D}_t\boldsymbol{a}_i}\right). \end{aligned}$$

- $t := t + 1$.

3.7 Correctness

SLIDE 18

Theorem: Let P be a bounded polyhedron that is either empty or full-dimensional and for which the prior information $\boldsymbol{x}_0$, r, v, V is available. Then, the ellipsoid method decides correctly whether P is nonempty or not, i.e., if $\boldsymbol{x}_{t^*-1} \notin P$, then P is empty

3.8 Proof

SLIDE 19

- If $\boldsymbol{x}_t \in P$ for $t < t^*$, then the algorithm correctly decides that P is nonempty
- Suppose $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{t^*-1} \notin P$. We will show that P is empty.
- We prove by induction on k that $P \subset E_k$ for $k = 0, 1, \ldots, t^*$. Note that $P \subset E_0$, by the assumptions of the algorithm, and this starts the induction.

SLIDE 20

- Suppose $P \subset E_k$ for some $k < t^*$. Since $\boldsymbol{x}_k \notin P$, there exists a violated inequality: $\boldsymbol{a}_{i(k)}'\boldsymbol{x} \geq \boldsymbol{b}_{i(k)}$ be a violated inequality, i.e., $\boldsymbol{a}_{i(k)}'\boldsymbol{x}_k < \boldsymbol{b}_{i(k)}$, where $\boldsymbol{x}_k$ is the center of the ellipsoid E_k

- For any $\boldsymbol{x} \in P$, we have

$$\boldsymbol{a}'_{i(k)}\boldsymbol{x} \geq \boldsymbol{b}_{i(k)} > \boldsymbol{a}'_{i(k)}\boldsymbol{x}_k$$

- Hence, $P \subset H_k = \left\{\boldsymbol{x} \in \Re^n \mid \boldsymbol{a}'_{i(k)}\boldsymbol{x} \geq \boldsymbol{a}'_{i(k)}\boldsymbol{x}_k\right\}$
- Therefore, $P \subset E_k \cap H_k$

SLIDE 21

By key geometric property, $E_k \cap H_k \subset E_{k+1}$; hence $P \subset E_{k+1}$ and the induction is complete

$$\frac{\text{Vol}(E_{t+1})}{\text{Vol}(E_t)} < e^{-1/(2(n+1))}$$
$$\frac{\text{Vol}(E_{t^*})}{\text{Vol}(E_0)} < e^{-t^*/(2(n+1))}$$
$$\text{Vol}(E_{t^*}) < Ve^{-\lceil 2(n+1)\log\frac{V}{v}\rceil/(2(n+1))} \leq Ve^{-\log\frac{V}{v}} = v$$

If the ellipsoid method has not terminated after t^* iterations, then $\text{Vol}(P) \leq \text{Vol}(E_{t^*}) \leq v$. This implies that P is empty

3.9 Binary Search

SLIDE 22

- $P = \left\{x \in \Re \mid x \geq 0, x \geq 1, x \leq 2, x \leq 3\right\}$
- $E_0 = [0, 5]$, centered at $x_0 = 2.5$
- Since $x_0 \notin P$, the algorithm chooses the violated inequality $x \leq 2$ and constructs E_1 that contains the interval $E_0 \cap \{x \mid x \leq 2.5\} = [0, 2.5]$
- The ellipsoid E_1 is the interval $[0, 2.5]$ itself
- Its center $x_1 = 1.25$ belongs to P
- This is binary search

3.10 Boundedness of P

SLIDE 23

Let $\boldsymbol{A}$ be an $m \times n$ integer matrix and let $\boldsymbol{b}$ a vector in $\Re^n$. Let U be the largest absolute value of the entries in $\boldsymbol{A}$ and $\boldsymbol{b}$.
Every extreme point of the polyhedron $P = \{\boldsymbol{x} \in \Re^n \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ satisfies

$$-(nU)^n \leq x_j \leq (nU)^n, \quad j = 1, \ldots, n$$

SLIDE 24

- All extreme points of P are contained in

$$P_B = \left\{\boldsymbol{x} \in P \;\middle|\; |x_j| \leq (nU)^n,\ j = 1, \ldots, n\right\}$$

- Since $P_B \subseteq E\big(\boldsymbol{0}, n(nU)^{2n}\boldsymbol{I}\big)$, we can start the ellipsoid method with $E_0 = E\big(\boldsymbol{0}, n(nU)^{2n}\boldsymbol{I}\big)$
-

$$Vol(E_0) \leq V = \left(2n(nU)^n\right)^n = (2n)^n(nU)^{n^2}$$

3.11 Full-dimensionality

SLIDE 25

Let $P = \{\boldsymbol{x} \in \Re^n \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\}$. We assume that $\boldsymbol{A}$ and $\boldsymbol{b}$ have integer entries, which are bounded in absolute value by U. Let

$$\epsilon = \frac{1}{2(n+1)}\big((n+1)U\big)^{-(n+1)}.$$

Let

$$P_\epsilon = \big\{\boldsymbol{x} \in \Re^n \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} - \epsilon\boldsymbol{e}\big\},$$

where $\boldsymbol{e} = (1, 1, \ldots, 1)$.

(a) If P is empty, then P_ϵ is empty.

(b) If P is nonempty, then P_ϵ is full-dimensional.

SLIDE 26

Let $P = \big\{\boldsymbol{x} \in \Re^n \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\big\}$ be a full-dimensional bounded polyhedron, where the entries of $\boldsymbol{A}$ and $\boldsymbol{b}$ are integer and have absolute value bounded by U. Then,

$$\mathrm{Vol}(P) > v = n^{-n}(nU)^{-n^2(n+1)}$$

3.12 Complexity

SLIDE 27

- $P = \{\boldsymbol{x} \in \Re^n \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\}$, where $\boldsymbol{A}$, $\boldsymbol{b}$ have integer entries with magnitude bounded by some U and has full rank. If P is bounded and either empty or full-dimensional, the ellipsoid method decides if P is empty in $O\big(n\log(V/v)\big)$ iterations
- $v = n^{-n}(nU)^{-n^2(n+1)}, \qquad V = (2n)^n(nU)^{n^2}$
- Number of iterations $O\big(n^4\log(nU)\big)$

SLIDE 28

- If P is arbitrary, we first form P_B, then perturb P_B to form $P_{B,\epsilon}$ and apply the ellipsoid method to $P_{B,\epsilon}$
- Number of iterations is $O\big(n^6\log(nU)\big)$.
- It has been shown that only $O(n^3\log U)$ binary digits of precision are needed, and the numbers computed during the algorithm have polynomially bounded size
- The linear programming feasibility problem with integer data can be solved in polynomial time

4 The ellipsoid method for optimization

SLIDE 29

$$\begin{array}{lll} \min & \boldsymbol{c}'\boldsymbol{x} & \\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, & \end{array} \qquad\qquad \begin{array}{lll} \max & \boldsymbol{b}'\boldsymbol{\pi} & \\ \text{s.t.} & \boldsymbol{A}'\boldsymbol{\pi} & = \boldsymbol{c} \\ & \boldsymbol{\pi} & \geq \boldsymbol{0}. \end{array}$$

By strong duality, both problems have optimal solutions if and only if the following system of linear inequalities is feasible:

$$\boldsymbol{b}'\boldsymbol{p} = \boldsymbol{c}'\boldsymbol{x}, \qquad \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \qquad \boldsymbol{A}'\boldsymbol{p} = \boldsymbol{c}, \qquad \boldsymbol{p} \geq \boldsymbol{0}.$$

LO with integer data can be solved in polynomial time.

4.1 Sliding objective

SLIDE 30

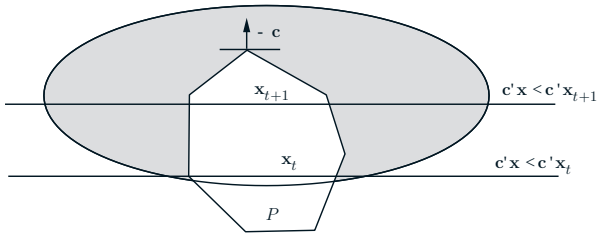
- We first run the ellipsoid method to find a feasible solution $\boldsymbol{x}_0 \in P = \{\boldsymbol{x} \in \Re^n \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\}$.
- We apply the ellipsoid method to decide whether the set

$$P \cap \{\boldsymbol{x} \in \Re^n \mid \boldsymbol{c}'\boldsymbol{x} < \boldsymbol{c}'\boldsymbol{x}_0\}$$

  is empty.
- If it is empty, then $\boldsymbol{x}_0$ is optimal. If it is nonempty, we find a new solution $\boldsymbol{x}_1$ in P with objective function value strictly smaller than $\boldsymbol{c}'\boldsymbol{x}_0$.

SLIDE 31

- More generally, every time a better feasible solution $\boldsymbol{x}_t$ is found, we take $P \cap \{\boldsymbol{x} \in \Re^n \mid \boldsymbol{c}'\boldsymbol{x} < \boldsymbol{c}'\boldsymbol{x}_t\}$ as the new set of inequalities and reapply the ellipsoid method.



4.2 Performance in practice

SLIDE 32

- Very slow convergence, close to the worst case
- Contrast with simplex method
- The ellipsoid method is a tool for classifying the complexity of linear programming problems

6.251J / 15.081J Introduction to Mathematical Programming
Fall 2009