CS-GY 6763: Lecture 9
Dimension Dependent Optimization

NYU Tandon School of Engineering, Prof. Christopher Musco

ADMINISTRATIVE

- Project proposal due next Wednesday, 11/9.
- Problem set 3 will be released shortly.
- We are working grading pset 2 and midterms.

FIRST ORDER CONVEX OPTIMIZATION

First Order Optimization: Given a convex function f and a convex set $\mathcal{S}$,

Goal: Find $\hat{\mathbf{x}} \in \mathcal{S}$ such that $f(\hat{\mathbf{x}}) \leq \min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x}) + \epsilon$.

Assume we have:

- **Function oracle**: Evaluate $f(\mathbf{x})$ for any $\mathbf{x}$.
- **Gradient oracle**: Evaluate $\nabla f(\mathbf{x})$ for any $\mathbf{x}$.
- **Projection oracle**: Evaluate $P_{\mathcal{S}}(\mathbf{x})$ for any $\mathbf{x}$.

Gradient descent requires $O\left(\frac{R^2G^2}{\epsilon^2}\right)$ calls to each oracle to solve the problem.

We were only able to improve the ϵ dependence by making stronger assumptions on f (strong convexity, smoothness).

DIMENSION DEPENDENT BOUND

Alternatively, we can get much better bounds if we are willing to depend on the problem dimension. I.e. on d if $f(\mathbf{x})$ is a function mapping d-dimensional vectors to scalars.

We already know how to do this for a few special functions:

$$\boxed{\kappa} \cdot \log(1/\epsilon)$$

$$\min \quad f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 \qquad \text{where} \qquad \mathbf{A} \in \mathbb{R}^{n \times d}.$$

$$\underline{2A^T(Ax-b) = 0} \qquad \underline{A^TAx = A^Tb}$$

$$x = (A^TA)^{-1}A^Tb$$

$$\hookrightarrow d^3 \text{ time.}$$

$$O(nd^2)$$

DIMENSION DEPENDENT BOUND

Let $f(\mathbf{x})$ $-f(x^*)$ be bounded between $[-B, B]$ on $\mathcal{S}$.

Theorem (Dimension Dependent Convex Optimization)

There is an algorithm (the Center-of-Gravity Method) which finds $\hat{\mathbf{x}}$ satisfying $f(\hat{\mathbf{x}}) \leq \min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x}) + \epsilon$ using $O(d \log(B/\epsilon))$ calls to a function and gradient oracle for f.

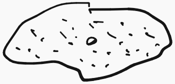
Caveat: Assumes we have some representation of $\mathcal{S}$, not just a projection oracle. We will discuss this more later.

Note: For an unconstrained problem with known starting radius R, can take $\mathcal{S}$ to be the ball of radius R around $\mathbf{x}^{(1)}$. If $\max_{\mathbf{x}} \|\nabla f(\mathbf{x})\|_2 = G$, we always have $B = O(RG)$.

$f(x^*) = 0$

CENTER OF GRAVITY METHOD

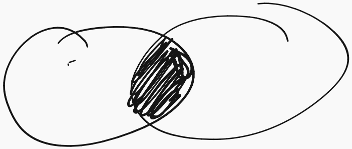
A few basic ingredients:



1. The center-of-gravity of a convex set $\mathcal{S}$ is defined as:

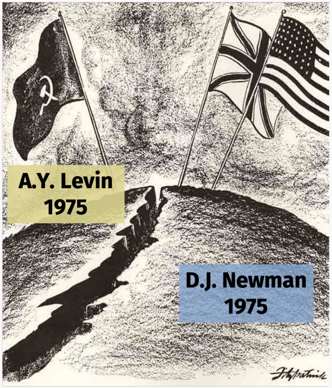
$$c = \frac{\int_{x\in\mathcal{S}} x\,dx}{\text{vol}(\mathcal{S})} = \frac{\int_{x\in\mathcal{S}} x\,dx}{\underline{\int_{x\in\mathcal{S}} 1\,dx}}$$

2. For two convex sets $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A}\cap\mathcal{B}$ is convex. Proof by picture:



CENTER OF GRAVITY METHOD

Natural "cutting plane" method. Developed simultaneous on opposite sides of iron curtain.

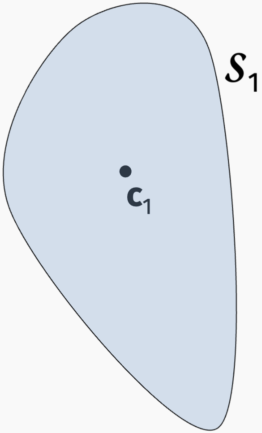


Not used in practice (we will discuss why) but the basic idea underlies many algorithms that are.

CENTER OF GRAVITY METHOD

Natural "cutting plane" method.

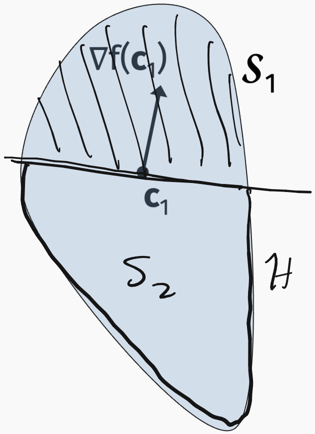
- $\underline{\mathcal{S}_1} = \underline{\mathcal{S}}$
- For $t = 1, \ldots, T$:
    - $\underline{\mathbf{c}_t}$ = center of gravity of $\underline{\mathcal{S}_t}$.
    - Compute $\nabla f(\mathbf{c}_t)$.
    - $\mathcal{H} = \{\mathbf{x} \,|\, \langle \nabla f(\mathbf{c}_t), \mathbf{x} - \mathbf{c}_t \rangle \leq 0\}$.
    - $\mathcal{S}_{t+1} = \mathcal{S}_t \cap H$
- Return $\hat{\mathbf{x}} = \arg\min_t f(\mathbf{c}_t)$



CENTER OF GRAVITY METHOD

Natural “cutting plane” method.

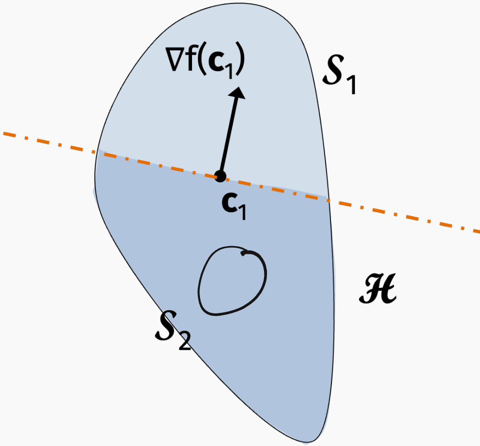
- $\mathcal{S}_1 = \mathcal{S}$
- For $t = 1, \ldots, T$:
  - $\mathbf{c}_t$ = center of gravity of $\mathcal{S}_t$.
  - Compute $\nabla f(\mathbf{c}_t)$.
  - $\underline{\mathcal{H}} = \{\mathbf{x} \,|\, \langle \nabla f(\mathbf{c}_t), \mathbf{x} - \mathbf{c}_t \rangle \leq 0\}$.
  - $\underline{\mathcal{S}_{t+1}} = \underline{\mathcal{S}_t} \cap \mathcal{H}$
- Return $\hat{\mathbf{x}} = \underline{\arg\min_t f(\mathbf{c}_t)}$



CENTER OF GRAVITY METHOD

Natural "cutting plane" method.

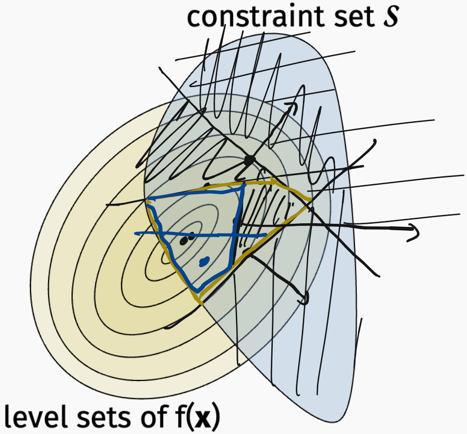
- $\mathcal{S}_1 = \mathcal{S}$
- For $t = 1, \ldots, T$:
    - $\mathbf{c}_t$ = center of gravity of $\mathcal{S}_t$.
    - Compute $\nabla f(\mathbf{c}_t)$.
    - $\mathcal{H} = \{\mathbf{x} \mid \langle \nabla f(\mathbf{c}_t), \mathbf{x} - \mathbf{c}_t \rangle \leq 0\}$.
    - $\mathcal{S}_{t+1} = \mathcal{S}_t \cap H$
- Return $\hat{\mathbf{x}} = \underline{\arg\min_t f(\mathbf{c}_t)}$



CENTER OF GRAVITY METHOD

Intuitively, why does it make sense to search in $\mathcal{S}_t \cap \mathcal{H}$ where:

$$\mathcal{H} = \{\mathbf{x} \big| \langle \nabla f(\mathbf{c}_t), \mathbf{x} - \mathbf{c}_t \rangle \leq 0\}?$$



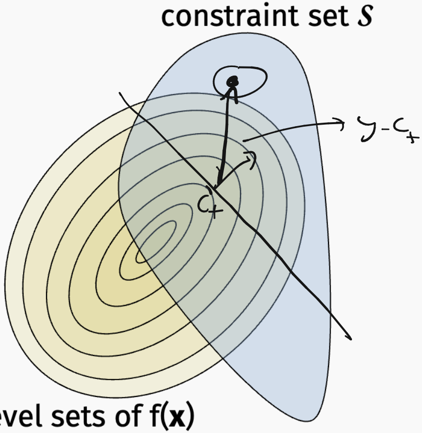
CENTER OF GRAVITY METHOD

Intuitively, why does it make sense to search in $\mathcal{S}_t \cap \mathcal{H}$ where:

$$\mathcal{H} = \{\mathbf{x} \big| \langle \nabla f(\mathbf{c}_t), \mathbf{x} - \mathbf{c}_t \rangle \leq 0\}?$$

By convexity, if $\mathbf{y} \notin \{\mathcal{S}_t \cap \mathcal{H}\}$ then:

$$\begin{aligned} \underline{f(\mathbf{y})} &\geq \underline{f(\mathbf{c}_t)} + \langle \nabla f(\mathbf{c}_t), \mathbf{y} - \mathbf{c}_t \rangle \\ &> \underline{\underline{f(\mathbf{c}_t)}} \end{aligned}$$



CONVERGENCE THEOREM

Theorem (Center-of-Gravity Convergence)

Let f be a convex function with values in $[-B, B]$. Let $\hat{\mathbf{x}}$ be the output of the center-of-gravity method run for T iterations. Then:

$$\underline{f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)} \leq 2B\left(1 - \frac{1}{e}\right)^{T/d} \leq 2Be^{-T/3d}.$$

If we set $\underline{T = 3d\log(2B/\epsilon)}$, then $f(\hat{\mathbf{x}}) - f(\mathbf{x}^*) \leq \epsilon$.

$T = O(d \log(B/\varepsilon))$

$\log(B/\varepsilon)$

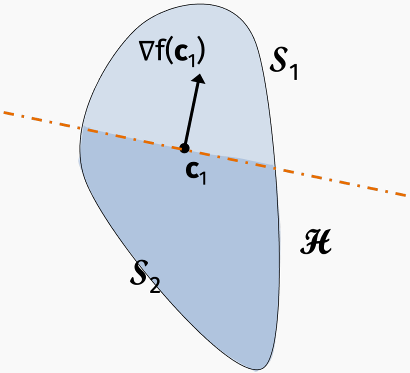
$\frac{\varepsilon}{B}$

$\underline{.63}$

ε

KEY GEOMETRIC TOOL

Want to argue that, at every step of the algorithm, we "cut off" a large portion of the convex set we are searching over:

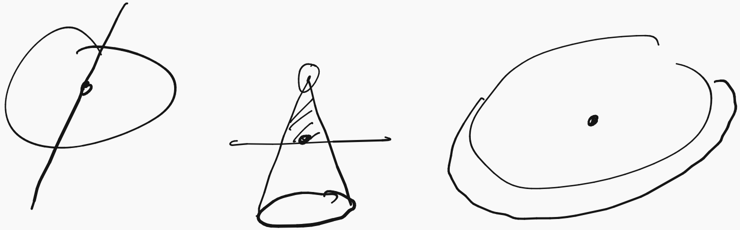


KEY GEOMETRIC TOOL

Theorem (Grünbaum's Theorem)

For any convex set $\mathcal{S}$ with center-of-gravity $\mathbf{c}$, *and any halfspace* $\mathcal{Z} = \{\mathbf{x} \,|\, \langle \mathbf{a}, \mathbf{x} - \mathbf{c}\rangle \leq 0\}$ *then:*

$$\frac{\mathrm{vol}(\mathcal{S} \cap \mathcal{Z})}{\mathrm{vol}(\mathcal{S})} \geq \frac{1}{e} \approx \boxed{.368}$$



KEY GEOMETRIC TOOL

Want to argue that, at every step of the algorithm, we “cut off” a large portion of the convex set we are searching over.

Theorem (Grünbaum’s Theorem)

For any convex set $\mathcal{S}$ with center-of-gravity $\mathbf{c}$, *and any halfspace* $\mathcal{Z} = \{\mathbf{x} \big| \langle \mathbf{a}, \mathbf{x} - \mathbf{c}\rangle \leq 0\}$ *then:*

$$\frac{\mathrm{vol}(\mathcal{S} \cap \mathcal{Z})}{\mathrm{vol}(\mathcal{S})} \geq \frac{1}{e} \approx .368$$

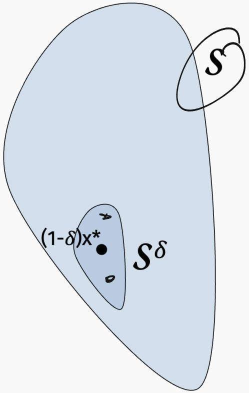
Let $\mathcal{Z}$ be the compliment of $\mathcal{H}$ from the algorithm. Then we cut off at least a $1/e$ fraction of the convex body on every iteration.

Corollary: After t steps, $\mathrm{vol}(\mathcal{S}_t) \leq \Big(\underbrace{1 - \tfrac{1}{e}}_{.63}\Big)^t \mathrm{vol}(\mathcal{S})$.

CONVERGENCE PROOF

Let δ be a small parameter to be chosen later.

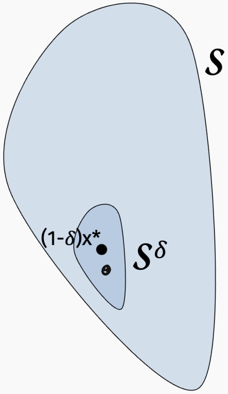
Let $\underline{\mathcal{S}^\delta} = \{\underline{(1-\delta)\mathbf{x}^* + \underline{\delta \mathbf{x}}} \mid \text{for } \mathbf{x} \in \mathcal{S}\}$.



Claim: Every point $\mathbf{y}$ in $\mathcal{S}^\delta$ has good function value.

CONVERGENCE PROOF

$y = (1-\delta)x^* + \delta x$

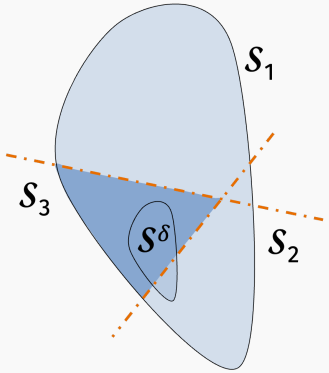


For any $\mathbf{y} \in \mathcal{S}^\delta$:

$$
\begin{aligned}
\boxed{f(\mathbf{y})} &= \underline{f\left((1-\delta)\mathbf{x}^* + \delta\mathbf{x}\right)} \\
&\le (1-\delta)f(\mathbf{x}^*) + \delta f(\mathbf{x}) \\
= \quad &\cancel{\le} f(\mathbf{x}^*) - \delta f(\mathbf{x}^*) + \delta f(\mathbf{x}) \\
&\le f(\mathbf{x}^*) + \cancel{2}B\delta.
\end{aligned}
$$

$= f(x^*) + \delta\left(\underline{f(x)} - \underline{f(x^*)}\right)$

CONVERGENCE PROOF



We also have: $\underline{\text{vol}(\mathcal{S}^\delta)} = \underline{\delta^d}\,\underline{\text{vol}(\mathcal{S})}$.

Set $\underline{\delta} = \underline{\left(1 - \frac{1}{e}\right)}^{T/d}$. After T steps, $\underline{\text{vol}(\mathcal{S}_t)} \leq \underline{\text{vol}(\mathcal{S}^\delta)}$.

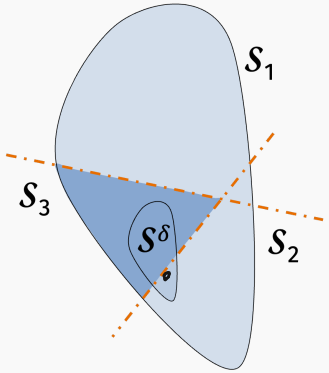
Either $\underline{S_T}$ exactly equals $\underline{S^\delta}$, in which case our current centroid gives error $\leq 2B\delta$.

Or we must have "chopped off" at least one point $\mathbf{y}$ in $\mathcal{S}^\delta$ by the time we reach step T.

$$\text{Vol}(S_t) \leq \left(1-\frac{1}{e}\right)^T \text{Vol}(S) = \delta^d\, \text{Vol}(S) = \text{Vol}(S^\delta)$$

$$\hookrightarrow = \delta^d$$

CONVERGENCE PROOF



Claim: If we "chopped off" at least one point $\mathbf{y}$ in $\mathcal{S}^\delta$ by the time we reach step T then for some centroid $\mathbf{c}_1, \ldots, \mathbf{c}_t$, $f(\mathbf{c}_t) < \cancel{2}B\delta$.

Proof:

$$\cancel{2}B\delta \geq f(\mathbf{y}) \geq f(\mathbf{c}_t) + \langle \nabla f(\mathbf{c}_t), \mathbf{y} - \mathbf{c}_t \rangle$$
$$> f(\mathbf{c}_t).$$

Algorithm returns $\arg\min_{\mathbf{c}_i} f(\mathbf{c}_i)$.

CONVERGENCE THEOREM

Theorem (Center-of-Gravity Convergence)

Let f be a convex function with values in $[-B, B]$. Let $\hat{\mathbf{x}}$ be the output of the center-of-gravity method run for T iterations. Then:

$$f(\hat{\mathbf{x}}) - f(\mathbf{x}^*) \leq 2B\left(1 - \frac{1}{e}\right)^{T/d} \leq 2Be^{-T/3d}.$$

If we set $T = O\left(\underline{d \log}(B/\epsilon)\right)$, then $f(\hat{\mathbf{x}}) - f(\mathbf{x}^*) \leq \epsilon$.

In terms of (<u>gradient-oracle</u> complexity, this is essentially optimal. So why isn't the algorithm used?

CENTROID COMPUTATION

In general computing the centroid is hard. #P-hard even when when $\mathcal{S}$ is an intersection of half-spaces (a polytope).

Even if the problem isn't hard for your starting convex body $\mathcal{S}$, it likely will become hard for $\mathcal{S} \cap \mathcal{H}_1 \cap \mathcal{H}_2 \cap \mathcal{H}_3 \ldots$.

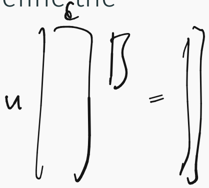
So while the oracle complexity of dimension-dependent optimization was settled, in the 70s a number of basic questions in terms of computational complexity.

LINEAR PROGRAMMING

Linear programs (LPs) are one of the most basic convex constrained, convex optimization problems:

Let $\mathbf{c} \in \mathbb{R}^d, \mathbf{b} \in \mathbb{R}^n, \mathbf{A} \in \mathbb{R}^{n\times d}$ be fixed vectors that define the problem, and let $\mathbf{x}$ be our variable parameter.

$$\min \underline{f(\mathbf{x})} = \underline{\mathbf{c}^T\mathbf{x}}$$
$$\text{subject to } \underline{\mathbf{A}\mathbf{x} \geq \mathbf{b}}.$$



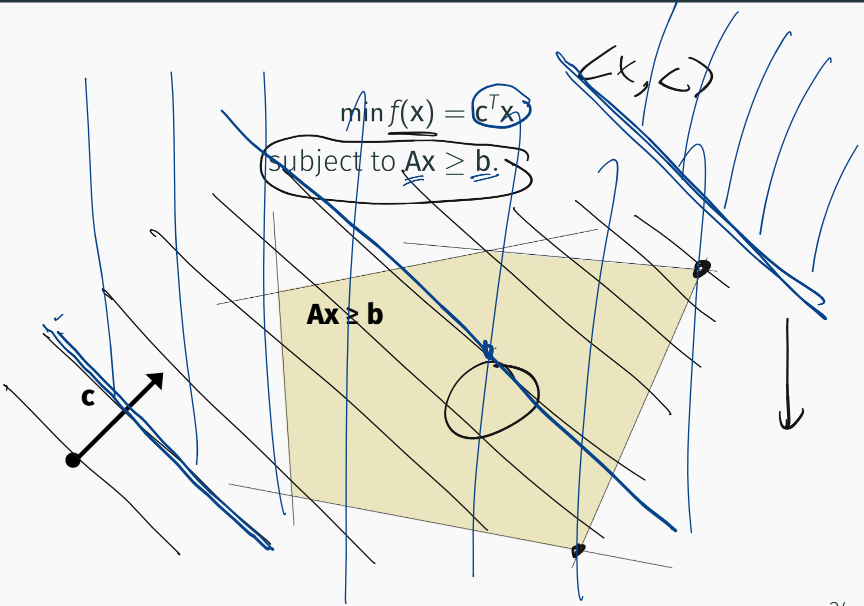
Think about $\mathbf{A}\mathbf{x} \geq \mathbf{b}$ as a union of half-space constraints:

$$\{\mathbf{x} : \mathbf{a}_1^T\mathbf{x} \geq b_1\}$$
$$\{\mathbf{x} : \mathbf{a}_2^T\mathbf{x} \geq b_2\}$$
$$\vdots$$
$$\{\mathbf{x} : \mathbf{a}_n^T\mathbf{x} \geq b_n\}$$

LINEAR PROGRAMMING

$$\min \underline{f(\mathbf{x})} = \mathbf{c}^T\mathbf{x}$$

$$\text{subject to } \mathbf{A}\mathbf{x} \geq \mathbf{b}.$$



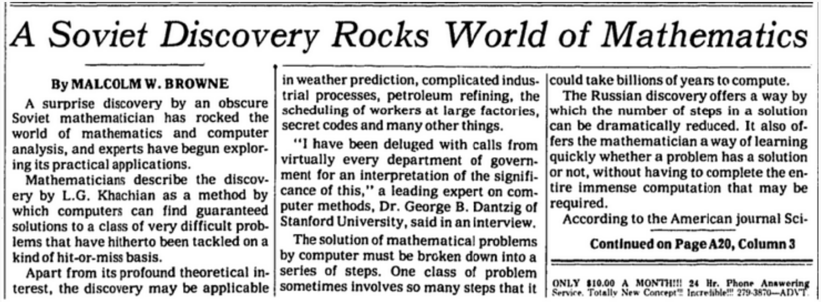
LINEAR PROGRAMMING APPLICATIONS

- Classic optimization applications: industrial resource optimization problems were killer app in the 70s.
- Robust regression: $\min_x \|Ax - b\|_1$.
- *L*1 constrained regression: $\min_x \|x\|_1$ subject to $Ax = b$. Lots of applications in sparse recovery/compressed sensing.
- Solve $\min_x \|Ax - b\|_\infty$.
- Polynomial time algorithms for Markov Decision Processes.
- Many combinatorial optimization problems can be solved via LP relaxations.

Theorem (Khachiyan, 1979)

Assume $n = d$. The ellipsoid method solves any linear program with L-bit integer valued constraints in $O(n^4 L)$ time. I.e. linear programming is in (weakly) polynomial time!

Using a relatively simple center-of-gravity like method!



A Soviet Discovery Rocks World of Mathematics

By MALCOLM W. BROWNE

A surprise discovery by an obscure Soviet mathematician has rocked the world of mathematics and computer analysis, and experts have begun exploring its practical applications.

Mathematicians describe the discovery by L.G. Khachian as a method by which computers can find guaranteed solutions to a class of very difficult problems that have hitherto been tackled on a kind of hit-or-miss basis.

Apart from its profound theoretical interest, the discovery may be applicable in weather prediction, complicated industrial processes, petroleum refining, the scheduling of workers at large factories, secret codes and many other things.

"I have been deluged with calls from virtually every department of government for an interpretation of the significance of this," a leading expert on computer methods, Dr. George B. Dantzig of Stanford University, said in an interview.

The solution of mathematical problems by computer must be broken down into a series of steps. One class of problem sometimes involves so many steps that it could take billions of years to compute.

The Russian discovery offers a way by which the number of steps in a solution can be dramatically reduced. It also offers the mathematician a way of learning quickly whether a problem has a solution or not, without having to complete the entire immense computation that may be required.

According to the American journal Sci-

Continued on Page A20, Column 3

ONLY $10.00 A MONTH!!! 24 Hr. Phone Answering Service. Totally New Concept!!! Incredible!!! 279-3870—ADVT.

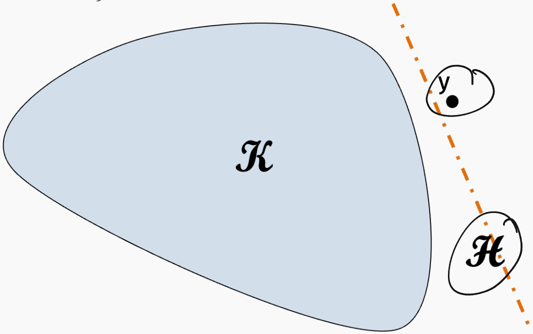
Front page of New York Times, November 9, 1979.

PROBLEM SIMPLIFICATION

Simplifying the problem: Given a convex set $\mathcal{K}$ via access to separation oracle $S_{\mathcal{K}}$ for the set, determine if $\mathcal{K}$ is empty, or otherwise return any point $\mathbf{x} \in \mathcal{K}$.

$$S_k(\mathbf{y}) = \begin{cases} \emptyset & \text{if } \underline{\mathbf{y} \in \mathcal{K}}. \\ \text{seperating hyperplane } (\mathbf{a}, c) & \text{if } \mathbf{y} \notin \mathcal{K}. \end{cases}$$

Let $\mathcal{H} = \{\mathbf{x} : \mathbf{a}^T\mathbf{x} = c\}$.



SEPARATION ORACLE

Example: How would you implement a seperation oracle for a polytope $\underline{\{\mathbf{x} : \mathbf{A}\mathbf{x} \geq \mathbf{b}\}}$.

$a_1^T x \geq b_1$

$\boxed{a_2^T x \geq b_2}$

$\vdots$

$a_n^T x \geq b_n$

a_2

$$\begin{tikzcd} a, \arrow[d] & c \arrow[d] \\ \underline{a_2^T x} < & b_2 \end{tikzcd}$$

but for all x' in S,

$\underline{a_2^T} x \geq b_2$

FROM MEMBERSHIP TO OPTIMIZATION

Original problem:

$Ax \geq b$

$$\min_{\mathbf{x}} f(\mathbf{x}) \text{ subject to } \mathbf{x} \in \underline{\mathcal{S}}$$

How can we reduce to determining if a convex set $\mathcal{K}$ is empty or not?

$x^T \vec{c} \leq c$

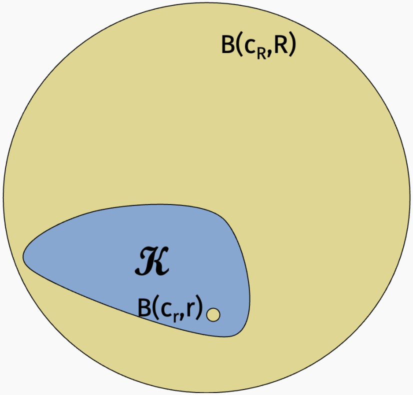
Binary search! For a convex function $f(\mathbf{x})$, $\{\underline{\mathbf{x}} : \underline{f(\mathbf{x})} \leq \bigcirc{c}\}$ is convex, and you can get a seperation oracle via the gradient.

- Start with upper bound and lower bounds u and l on optimal solution (can be obtained for many problems).
- Check if the convex set $\mathcal{S} \cap \{\mathbf{x} : f(\mathbf{x}) \leq (u+l)/2\}$ contains a point.
- Update $u = (u+l)/2$ if it does, $l = (u+l)/2$ if not.
- Continue until $|u - l| \leq \epsilon$.

ELLIPSOID METHOD SKETCH

Goal of ellipsoid algorithm: Solve "Is $\mathcal{K}$ empty or not?" given a seperation oracle for $\mathcal{K}$ under the assumptions that:

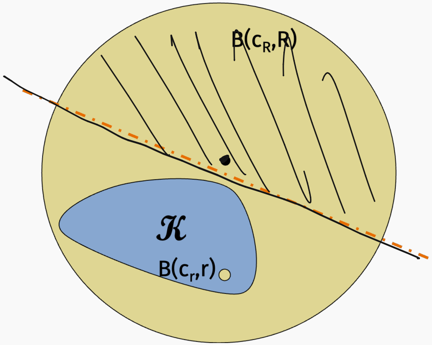
1. $\mathcal{K} \subset B(\mathbf{c}_R, R)$.
2. If non-empty, $\mathcal{K}$ contains $B(\mathbf{c}_r, r)$ for some $r < R$.



ELLIPSOID METHOD SKETCH

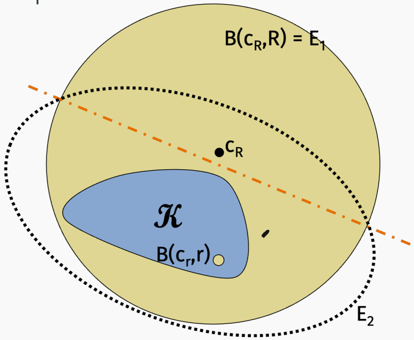
Iterative method similar to center-of-gravity:

1. Check if center $\mathbf{c}_R$ of $B(\mathbf{c}_R, R)$ is in $\mathcal{K}$.
2. If it is, we are done.
3. If not, cut search space in half, using seperating hyperplane.



ELLIPSOID METHOD SKETCH

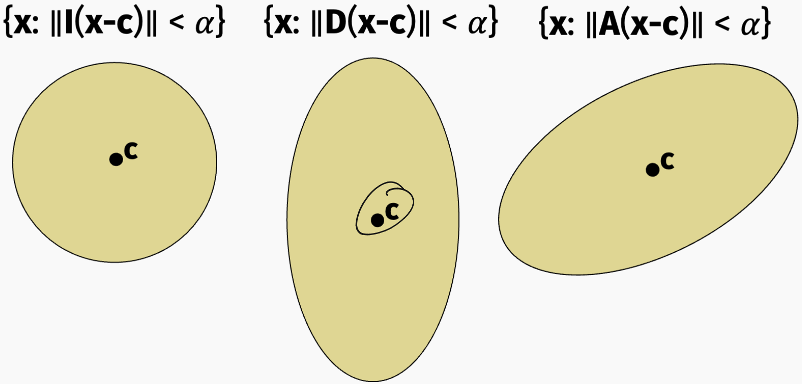
Key insight: Before moving on, approximate new search region by something that we can easily compute the centroid of. Specifically an ellipse!!



Produce a sequence of ellipses that always contain $\mathcal{K}$ and decrease in volume: $B(\mathbf{c}_R, R) = E_1, E_2, \ldots$. Once we get to an ellipse with volume $\leq B(\mathbf{c}_r, r)$, we know that $\mathcal{K}$ must be empty.

ELLIPSE

An ellipse is a convex set of the form: $\{\mathbf{x} : \|\mathbf{A}(\mathbf{x} - \mathbf{c})\|_2^2 \leq \alpha\}$ for some constant c and matrix $\mathbf{A}$. The center-of-mass is $\mathbf{c}$.



Often re-parameterized to say that the ellipse is all $\mathbf{x}$ with $\{\mathbf{x} : (\mathbf{x} - \mathbf{c})^T\mathbf{Q}^{-1}(\mathbf{x} - \mathbf{c}) \leq 1\}$

ELLIPSOID UPDATE

There is a closed form solution for the equation of the smallest ellipse containing a given half-ellipse. I.e. let E_i have parameters $\mathsf{Q}_i, \mathsf{c}_i$ and consider the half-ellipse:

$$\mathsf{E}_i \cap \{\mathsf{x} : \mathsf{a}_i^T \mathsf{x} \leq \mathsf{a}_i^T \mathsf{c}_i\}.$$

Then E_{i+1} is the ellipse with parameters:

$$\mathsf{Q}_{i+1} = \frac{d^2}{d^2-1}\left(\mathsf{Q}_i - \frac{2}{d+1}\mathsf{h}\mathsf{h}^T\right) \qquad \mathsf{c}_{i+1} = \mathsf{c}_i - \frac{1}{n+1}\mathsf{h},$$

where $\mathsf{h} = \sqrt{\mathsf{a}_i^T \mathsf{Q}_i \mathsf{a}_i} \cdot \mathsf{a}_i$.

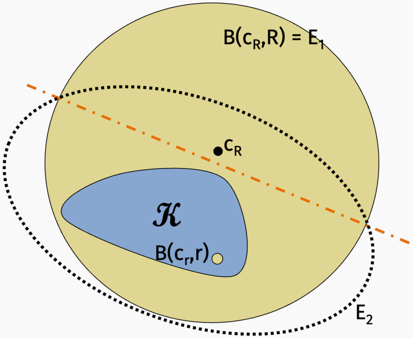
GEOMETRIC OBSERVATION

Claim: $\text{vol}(E_{i+1}) \leq \underline{(1 - \frac{1}{2d})}\text{vol}(E_i)$.   .63

Proof: Via reduction to the "isotropic case". I will post a proof on the course website if you are interested.

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

$\approx \frac{1}{e}$



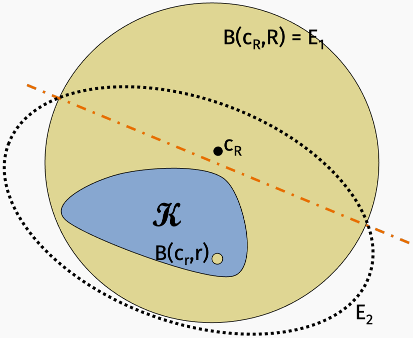
$\underline{O(d^2)}$

Not as good as the $(1 - \frac{1}{e})$ constant-factor volume reduction we got from center-of-gravity, but still very good!

GEOMETRIC OBSERVATION

Claim: $\mathrm{vol}(E_{i+1}) \leq (1 - \frac{1}{2d})\,\mathrm{vol}(E_i)$

d^{7}



After $O(d)$ iterations, we reduce the volume by a constant.

In total require $O(\underline{d^2 \log(R/r)})$ iterations to solve the problem.

ELLIPSOID FOR LPs

Theorem (Khachiyan, 1979)

Assume $n = d$. The ellipsoid method solves any linear program with L-bit integer valued constraints in $O(n^4L)$ time. I.e. linear programming is in (weakly) polynomial time!

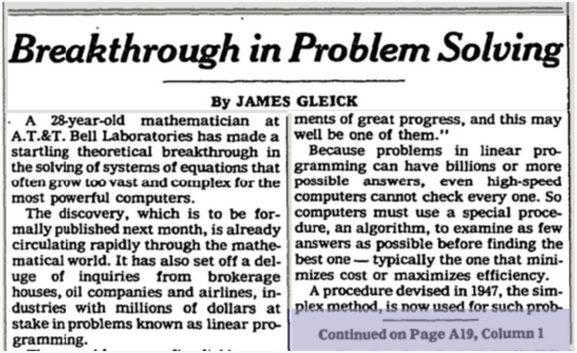
The method works for any convex program.

For LPs, we have an $O(nd)$ time seperation oracle, and ellipsoid update take $O(d^2)$ time.

Careful analysis of the binary search method, how to set B_r and B_R, etc. leads to the final runtime bound.

Theorem (Karmarkar, 1984)

Assume $n = d$. The <u>interior point method</u> solves any linear program with L-bit integer valued constraints in $O(n^{3.5}L)$ time.



Breakthrough in Problem Solving

By JAMES GLEICK

A 28-year-old mathematician at A.T.&T. Bell Laboratories has made a startling theoretical breakthrough in the solving of systems of equations that often grow too vast and complex for the most powerful computers.

The discovery, which is to be formally published next month, is already circulating rapidly through the mathematical world. It has also set off a deluge of inquiries from brokerage houses, oil companies and airlines, industries with millions of dollars at stake in problems known as linear programming.

ments of great progress, and this may well be one of them."

Because problems in linear programming can have billions or more possible answers, even high-speed computers cannot check every one. So computers must use a special procedure, an algorithm, to examine as few answers as possible before finding the best one — typically the one that minimizes cost or maximizes efficiency.

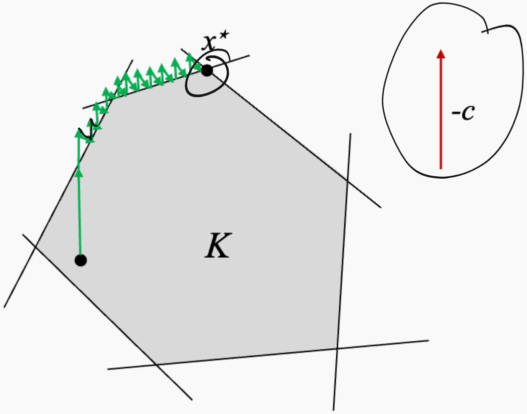
A procedure devised in 1947, the simplex method, is now used for such prob-

Continued on Page A19, Column 1

Front page of New York Times, November 19, 1984.

INTERIOR POINT METHODS

Will post lecture notes on the website (optional reading).

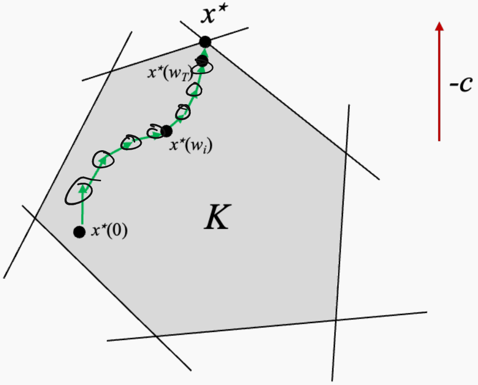


Projected Gradient Descent Optimization Path

INTERIOR POINT METHODS

Will post lecture notes on the website (optional reading).

$\sqrt{d}$ iterations.



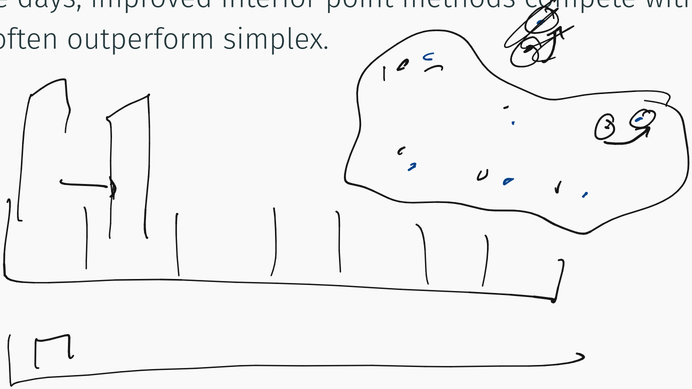
Ideal Interior Point Optimization Path

POLYNOMIAL TIME LINEAR PROGRAMMING

Both results had a huge impact on the theory of optimization, although at the time neither the ellipsoid method or interior point method were faster than a heuristic known at the Simplex Method.

$n^2 \varepsilon^{-2} \log(n) + n^{2}$

These days, improved interior point methods compete with and often outperform simplex.



LP RELAXATION

SET COVER PROBLEM

Given:

- n ground elements $\{1, \ldots, n\}$
- m sets $S_1, S_2, \ldots, S_m$ where $S_j \subseteq \{1, \ldots, n\}$
- non-negative weights $c_1, \ldots, w_m \geq 0$

Find:

$$\min_{Z \subseteq \{1,\ldots,m\}} \sum_{j \in Z} c_j \qquad \text{subject to} \qquad \cup_{j \in Z} S_j = \{1, \ldots, n\}.$$

This is an NP-Complete combinatorial optimization problem!
Likely impossible to find an efficient <u>exact</u> algorithm.

APPLICATIONS

- Finding efficient sets of test cases for code testing and verification.
- Complex employee shift scheduling problems (e.g. in the airline industry).
- Motif selection in computational biology.

APPLICATION: VERTEX COVER

Given:

- ground elements are edges
- sets are nodes so

$$S_j = \{\text{edges adjacent to } j\text{th node}\}$$

- Could have $c_j = 1$ for all $j \in 1, \ldots, m$ or could have different costs per node.

What vertices should we choose so that all edges are connected to at least one chosen vertex?

LINEAR PROGRAMMING

Let $\mathbf{x} \in \mathbb{R}^m$ be a vector of decision variables, $\mathbf{c}$ be a cost vector, and $\mathbf{b} \in \mathbb{R}^n$ be a vector of constraints. As before a linear program has the form the linear programs is:

$$\text{Minimize} \quad \mathbf{c}^T\mathbf{x} \quad \text{subject to} \quad \mathbf{A}\mathbf{x} \geq \mathbf{b},$$

Goal: Show that set cover can be written as a linear program, except with the additional constraint that $\mathbf{x}$ is a binary random vector. This is called an integer linear program.

LP FOR SET COVER: OBJECTIVE

Let $x_j = 1$ iff $j \in Z$. 0 otherwise

The objective is to minimize the sum of weights in Z:

$$\min_{Z \subseteq \{1,\ldots,m\}} \sum_{j \in Z} c_j \quad \Leftrightarrow \quad \min_{\mathbf{x}} \sum_{j=1}^{m} c_j x_j \quad \Leftrightarrow \quad \min_{\mathbf{x}} \mathbf{c}^T \mathbf{x}$$

LP FOR SET COVER: CONSTRAINT

The constraint is that C covers the ground elements. I.e. for all $i \in 1, \ldots, n$, we should have:

$$i \in \cup_{j \in Z} S_j \quad \Leftrightarrow \quad \sum_{j : i \in S_j}^{m} x_j \geq 1 \quad \Leftrightarrow \quad \mathbf{A}\mathbf{x} \geq \mathbf{c}$$

Question: What are $\mathbf{A}$ and $\mathbf{b}$?

LP FOR SET COVER: STATEMENT

$$\text{Minimize} \quad \mathbf{c}^\mathsf{T}\mathbf{x} \quad \text{subject to} \quad \mathbf{A}\mathbf{x} \geq \mathbf{b}, \quad \mathbf{x} \in \{0,1\}^m$$

$$\text{Minimize} \quad \sum_{j=1}^{m} c_j x_j \quad \text{subject to} \quad \sum_{j:i\in S_j} x_j \geq 1, \quad \mathbf{x} \in \{0,1\}^m$$

Without the constraint $\mathbf{x} \in \{0,1\}^m$, we can solve linear program in polynomial time with e.g. Interior Point, Ellipsoid Method. But we will get a fractional solution.

LP RELAXATION

Definition (Relaxation)

A linear program (where $\mathbf{x} \in \mathbb{R}^m$) is a *relaxation* of an integer program (where $\mathbf{x} \in \{0,1\}^m$) if

- a feasible solution to the ILP is a feasible solution to the LP and
- the value of the feasible solution in the ILP has the same value in the LP.

We always have that $OPT_{LP} \leq OPT_{ILP}$.

LP ROUNDING

Common approach for <u>approximately</u> solving ILPs:

Find a solution to an LP relaxation for the IP and then "round" the solution to be binary or integer. E.g round numbers close to 0 to 0, numbers close to 1 to 1. Often randomization is used in the rounding process.

LP TO SET COVER

Theorem

Let $\mathbf{x}^*$ *be optimal solution to LP. Define*

$$f = \max_{i \in 1,\ldots,n} |\{j : i \in S_j\}|.$$

Rounding procedure: *Put* $j \in Z$ *if and only if* $x_j^* \geq 1/f$.

Then Z *is feasible and gives an* f*-approximation to set cover.*

Question: What approximation do we get for vertex cover?

LP TO SET COVER: PROOF

Claim: C is feasible.

Fix i. We know

$$\sum_{j : i \in S_j} x_j^* \geq 1.$$

Then

LP TO SET COVER: PROOF

Claim: C gives an f-approximation to set cover.

$$Cost_{LP} = \sum_{j=1}^{m} x_j^* c_j \qquad\qquad Cost_{ILP} = \sum_{j \in Z} c_j$$