# CS-GY 6763: Lecture 10
# Dimension Dependent Optimization, Linear Programming

NYU Tandon School of Engineering, Prof. Christopher Musco

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

We were only able to improve the $\epsilon$ 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:

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

## DIMENSION DEPENDENT BOUND

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

**Theorem (Dimension Dependent Convex Optimization)**

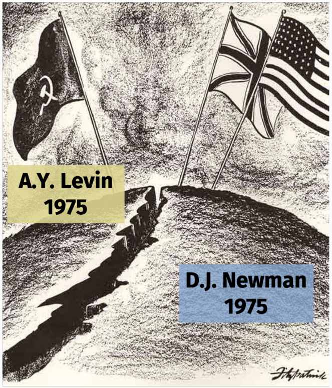
*There is an algorithm (the Center-of-Gravity Method) which finds $\hat{\mathbf{x}}$ satisfying $f(\hat{\mathbf{x}}) \leq \min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x}) + \epsilon$ using $O(d\log(B/\epsilon))$ calls to a function and gradient oracle for convex $f$.*

**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 $\mathbf{x}^{(0)}$. If $\|\nabla f(\mathbf{x})\|_2 \leq G$, we always have $B = O(RG)$.

## 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 popular algorithms, including the famous

## CENTER OF GRAVITY METHOD

A few basic ingredients:

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

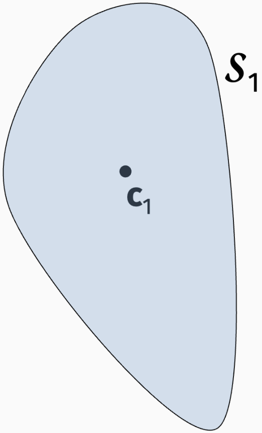
$$c = \frac{\int_{x\in\mathcal{S}} x\,dx}{\mathrm{vol}(\mathcal{S})} = \frac{\int_{x\in\mathcal{S}} x\,dx}{\int_{x\in\mathcal{S}} 1\,dx}$$

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.

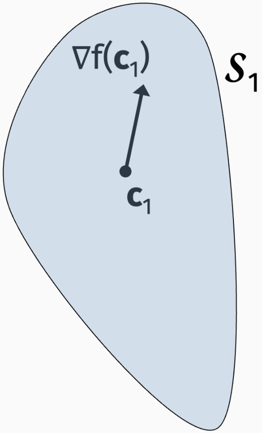
- $\mathcal{S}_1 = \mathcal{S}$
- For $t = 1, \ldots, T$ :
  - $\mathbf{c}_t$ = center of gravity of $\mathcal{S}_t$.
  - Compute $\nabla f(\mathbf{c}_t)$.
  - $\mathcal{H} = \{\mathbf{x} | \langle \nabla f(\mathbf{c}_t), \mathbf{x} - \mathbf{c}_t \rangle \leq 0\}$.
  - $\mathcal{S}_{t+1} = \mathcal{S}_t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



## CENTER OF GRAVITY METHOD

Natural "cutting plane" method.

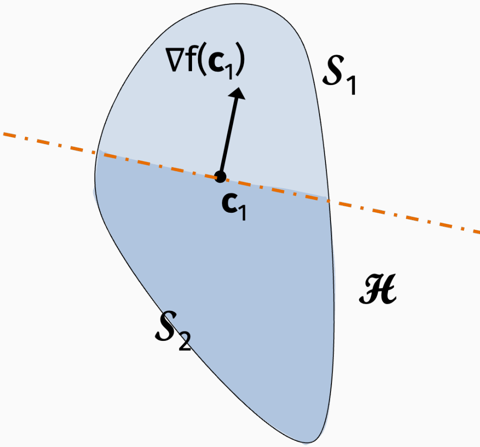
- $\mathcal{S}_1 = \mathcal{S}$
- For $t = 1, \ldots, T$ :
    - $\mathbf{c}_t$ = center of gravity of $\mathcal{S}_t$.
    - Compute $\nabla f(\mathbf{c}_t)$.
    - $\mathcal{H} = \{\mathbf{x} | \langle \nabla f(\mathbf{c}_t), \mathbf{x} - \mathbf{c}_t \rangle \leq 0\}$.
    - $\mathcal{S}_{t+1} = \mathcal{S}_t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



## CENTER OF GRAVITY METHOD

Natural "cutting plane" method.

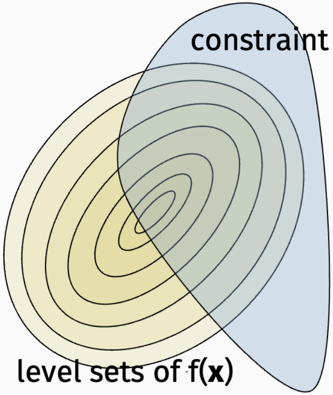
- $\mathcal{S}_1 = \mathcal{S}$
- For $t = 1, \ldots, T$ :
    - $\mathbf{c}_t$ = center of gravity of $\mathcal{S}_t$.
    - Compute $\nabla f(\mathbf{c}_t)$.
    - $\mathcal{H} = \{\mathbf{x} \,|\, \langle \nabla f(\mathbf{c}_t), \mathbf{x} - \mathbf{c}_t \rangle \leq 0\}$.
    - $\mathcal{S}_{t+1}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



## CENTER OF GRAVITY METHOD

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

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



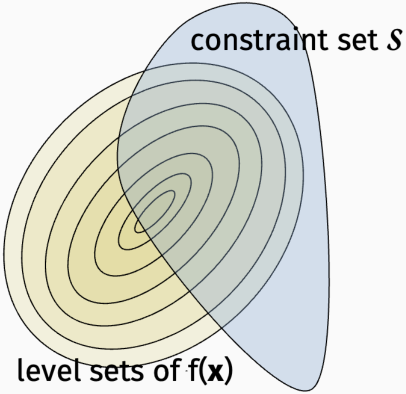
Intuitively, why does it make sense to search in $\mathcal{S}_t \cap \mathcal{H}$ where:

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

By convexity,

$f(\mathbf{y}) \geq f(\mathbf{c}_t) + \langle \nabla f(\mathbf{c}_t), \mathbf{y} - \mathbf{c}_t \rangle$.

If $\mathbf{y} \notin \{\mathcal{S}_t \cap \mathcal{H}\}$ then $\langle \nabla f(\mathbf{c}_t), \mathbf{y} - \mathbf{c}_t \rangle$ is positive, so $f(\mathbf{y}) > f(\mathbf{c}_t)$.



## CONVERGENCE THEOREM

**Theorem (Center-of-Gravity Convergence)**

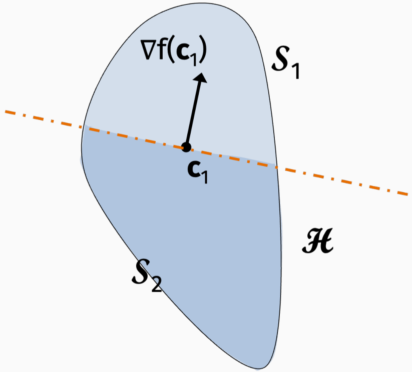
*Let $f$ be a convex function with values in $[-B, B]$. Let $\hat{\mathrm{x}}$ be the output of the center-of-gravity method run for $T$ iterations. Then:*

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

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

## 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* $\mathcal{Z} = \{\mathbf{x} \big| \langle \mathbf{a}, \mathbf{x} - \mathbf{c} \rangle \leq 0\}$ *then:*

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

## 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:*

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

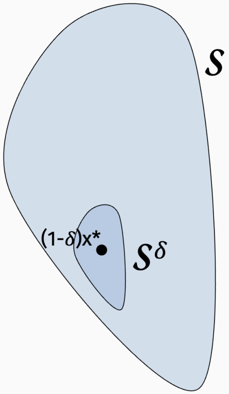
Let $\mathcal{Z}$ be the compliment of $\mathcal{H}$ from the algorithm. Then we cut off at least a $1/e$ fraction of the convex body on every iteration.

**Corollary:** After $t$ steps, $\mathrm{vol}(\mathcal{S}_t) \leq \left(1 - \frac{1}{e}\right)^t \mathrm{vol}(\mathcal{S})$.

## CONVERGENCE PROOF

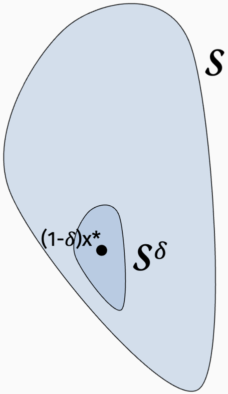
Let $\delta$ be any small error parameter.

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



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

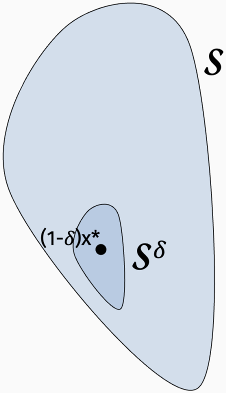
## CONVERGENCE PROOF



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

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

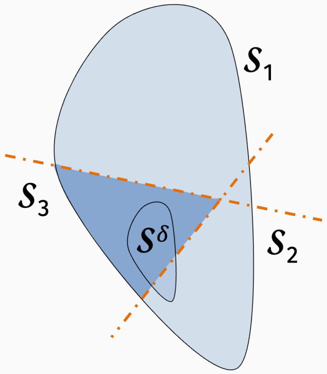
## CONVERGENCE PROOF



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

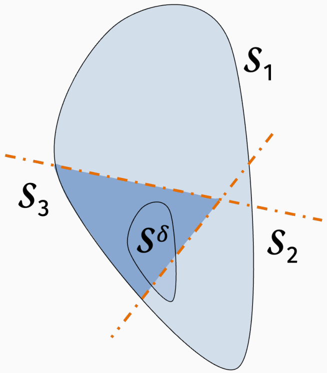
$$\begin{aligned} f(\mathbf{y}) &= f\left((1-\delta)\mathbf{x}^* + \delta \mathbf{x}\right) \\ &\leq (1-\delta) f(\mathbf{x}^*) + \delta f(\mathbf{x}) \\ &\leq f(\mathbf{x}^*) - \delta f(\mathbf{x}^*) + \delta f(\mathbf{x}) \\ &\leq f(\mathbf{x}^*) + 2B\delta. \end{aligned}$$

## CONVERGENCE PROOF



**Claim 1:** Let $\delta = (1 - \frac{1}{e})^{T/d}$. After $T$ steps, we either have that $\mathcal{S}_T$ equals $S^\delta$ exactly, or we "chopped off" at least one point $\mathbf{y}$ in $\mathcal{S}^\delta$. Why?

## CONVERGENCE PROOF



**Claim 2:** If $\mathcal{S}_T$ equals $S^\delta$ or we "chopped off" at least one point $\mathbf{y}$ in $\mathcal{S}^\delta$, then:

$$f(\hat{\mathbf{x}}) = \min_{t=1,\ldots,T} f(\mathbf{c}_t) \leq 2B\delta.$$

Why?

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

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

## RUNTIME ISSUE

**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 60, basic questions remained regarding computational complexity.

We will see how to resolve this issue with an elegant cutting plane methods called the **Ellipsoid Method** that was introduced by Naum Shor in 1977.

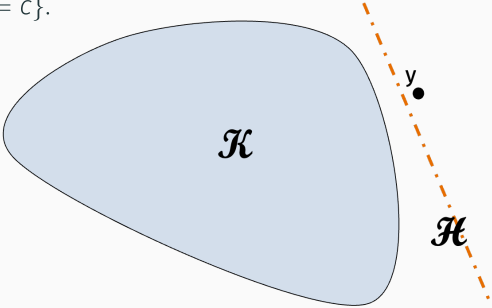
## FORMALIZATION

To talk about runtime efficiency we need to be more concrete about how our (convex) constraint set is even specified.

**Seperation Oracle:** For a convex set $\mathcal{K} \subset \mathbb{R}^d$, a seperation oracle $\mathsf{S}_{\mathcal{K}}$ is a function that takes in points in $\mathbb{R}^d$ and returns:

$$\mathsf{S}_{\mathcal{K}}(\mathbf{y}) = \begin{cases} \emptyset & \text{if } \mathbf{y} \in \mathcal{K}. \\ \text{separating hyperplane } (\mathbf{a}, c) & \text{if } \mathbf{y} \notin \mathcal{K}. \end{cases}$$

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



## SEPARATION ORACLE

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

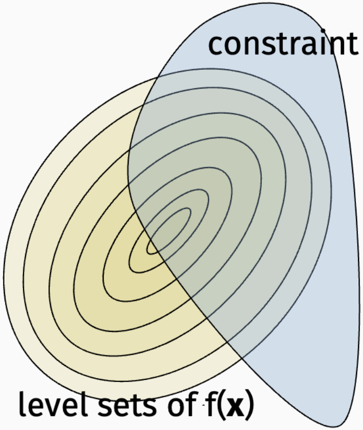
## PROBLEM SIMPLIFICATION

**Instead of directly solving a constrained optimization problem, solve the membership problem.** Given a separation oracle $S_{\mathcal{K}}$ for a convex set $\mathcal{K}$, determine if $\mathcal{K}$ is empty, or otherwise return any point $\mathbf{x} \in \mathcal{K}$.

## FROM MEMBERSHIP TO OPTIMIZATION

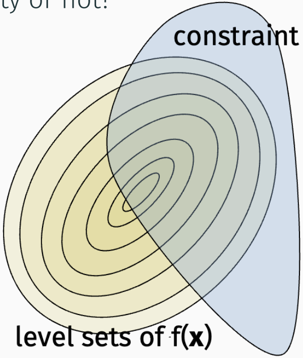
Original problem: $\min_{x \in \mathcal{S}} f(x)$.

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



## FROM MEMBERSHIP TO OPTIMIZATION

**Original problem:** $\min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x})$. How to reduce to determining if a convex set $\mathcal{K}$ is empty or not?

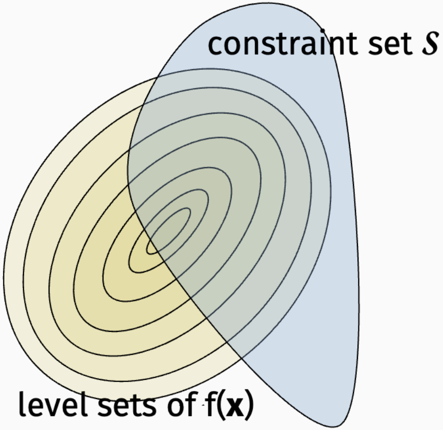


**Claim:** Given any fixed value $c$, can check if $f(\mathbf{x}^*) \leq c$ and, if it is, find some $\mathbf{x}$ with $f(\mathbf{x}) \leq c$.

**Approach:** Solve membership problem on $\mathcal{K} = \mathcal{S} \cap \mathcal{C}$ where $\mathcal{C} = \{\mathbf{x} : f(\mathbf{x}) \leq c\}$. $\mathcal{C}$ and $\mathcal{S}$ are convex, so $\mathcal{K}$ is as well.

## FROM MEMBERSHIP TO OPTIMIZATION

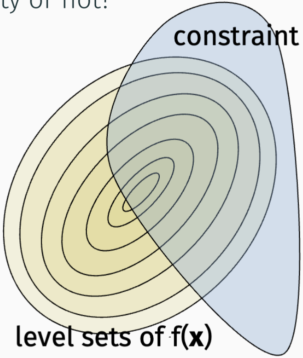
**Prove on homework**: Given efficent separation oracles for $\mathcal{C}$ and $\mathcal{S}$, I can construct an efficient separation oracle for $\mathcal{K}$.



How do I get a seperation oracle for $\mathcal{C}$?

## FROM MEMBERSHIP TO OPTIMIZATION

**Original problem:** $\min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x})$. How to reduce to determining if a convex set $\mathcal{K}$ is empty or not?



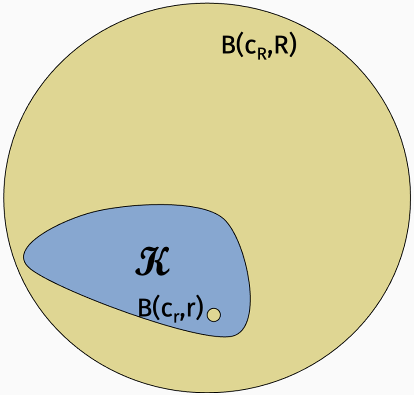
**Claim:** Given any fixed value $c$, can check if $f(\mathbf{x}^*) \leq c$ and, if it is, find some $\mathbf{x}$ with $f(\mathbf{x}) \leq c$.

**Final algorithm:** Assuming $f$ is positive, just run exponential/binary search to find $\tilde{c} \leq f(\mathbf{x}^*) + \epsilon$!

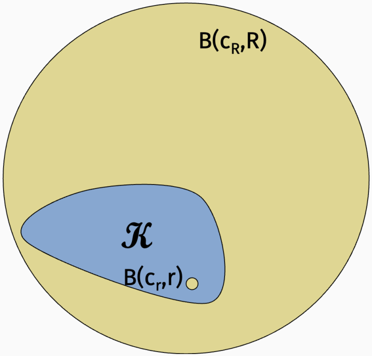
## ELLIPSOID METHOD SKETCH

**Goal of ellipsoid algorithm:** Solve "Is $\mathcal{K}$ empty or not?" given a separation 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

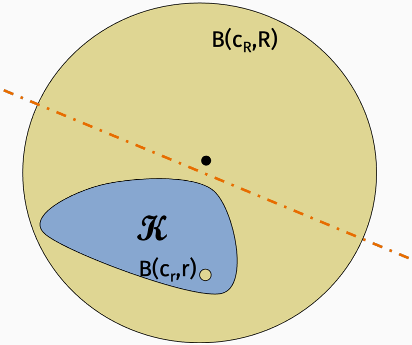


**Application to original problem:** Lots of details to consider. Assume for simplicty we known $f(\mathbf{x}^*)$ and that have no constraint set. Goal is to solve membership problem on $\tilde{C} = \{\mathbf{x} : f(\mathbf{x}) \leq f(\mathbf{x}^*) + \epsilon\}$. For a convex function $f$ such that $\|\nabla f(\mathbf{x})\|_2 \leq G$, it can be checked that $\tilde{C}$ contains a ball of radius $\epsilon/G$.

## ELLIPSOID METHOD SKETCH

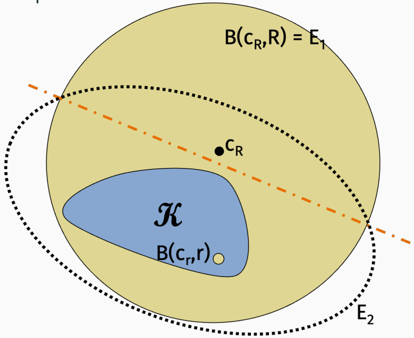
Iterative method similar to center-of-gravity:

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 separating hyperplane.



## ELLIPSOID METHOD SKETCH

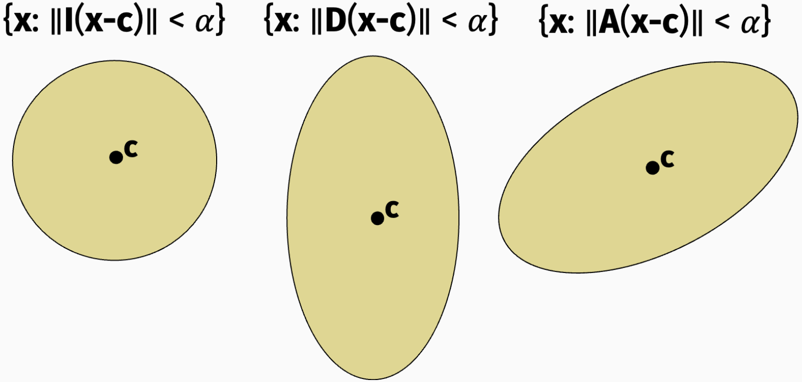
**Key insight:** Before moving on, approximate new search region by something that we can easily compute the centroid of. Specifically an ellipse!



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 $\mathbf{E}_i$ have parameters $\mathbf{Q}_i, \mathbf{c}_i$ and consider the half-ellipse:

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

Then $\mathbf{E}_{i+1}$ is the ellipse with parameters:

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

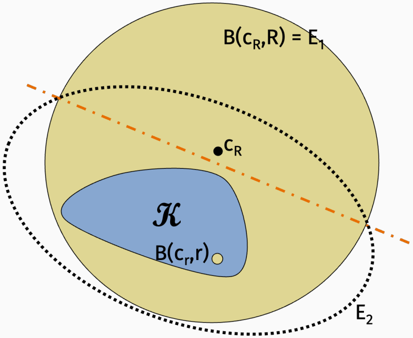
where $\mathbf{h} = \sqrt{\mathbf{a}_i^T \mathbf{Q}_i \mathbf{a}_i} \cdot \mathbf{a}_i$.

**Computing the update takes $O(d^2)$ time.**

## GEOMETRIC OBSERVATION

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

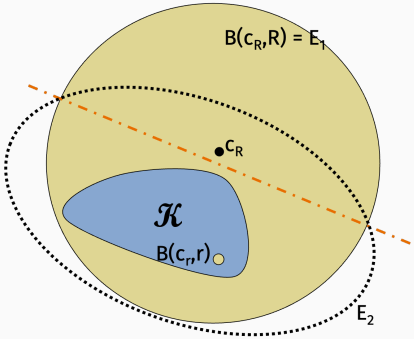
**Proof:** Via reduction to the “isotropic case”. I will post a proof on the course website if you are interested.



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



After $O(d)$ iterations, we reduce the volume by a constant. In total require $O(d^2 \log(R/r))$ iterations to solve the problem.

Complexity for solving $\min_{x \in \mathcal{S}} f(x)$ is roughly $\tilde{O}(d^4 \log(R/\epsilon))$, hiding logarithmic factors.

## 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.

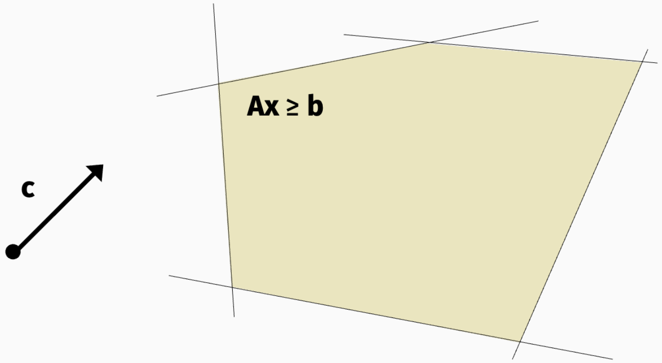
$$\min f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$$
$$\text{subject to } \mathbf{A}\mathbf{x} \geq \mathbf{b}.$$

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

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

## KILLER APPLICATION: LINEAR PROGRAMMING

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



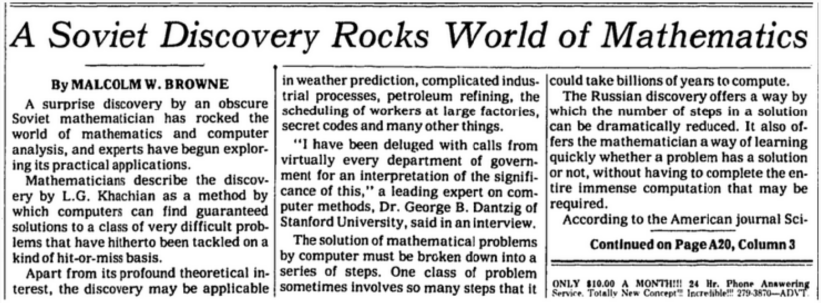
## LINEAR PROGRAMMING APPLICATIONS

- Classic optimization applications: industrial resource optimization problems were killer app in the 70s.
- Robust regression: $\min_x \|Ax - b\|_1$.
- $L1$ 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.

## LINEAR PROGRAMMING

**Theorem (Khachiyan, 1979)**

*Assume $n = d$. The ellipsoid method solves any linear program with $L$-bit integer valued constraints exactly in $O(n^4 L)$ time.*



# 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

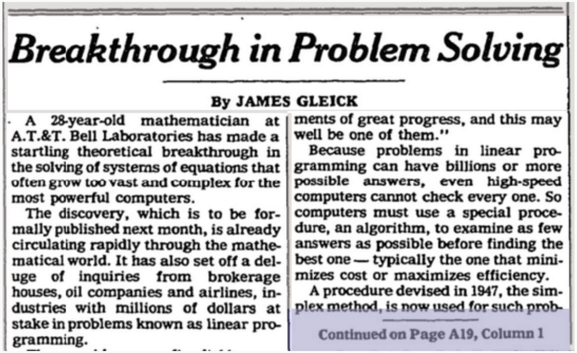
ONLY $10.00 A MONTH!!! 24 Hr. Phone Answering Service. Totally New Concept!! Incredible!!! 279-3870—ADVT.

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

## INTERIOR POINT METHODS

**Theorem (Karmarkar, 1984)**

*Assume $n = d$. The interior point method 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.

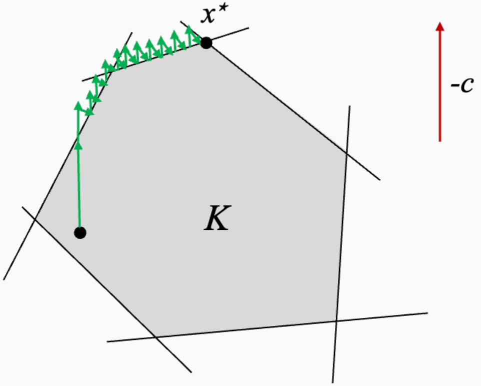
A procedure devised in 1947, the simplex method, is now used for such prob-

Continued on Page A19, Column 1

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

## INTERIOR POINT METHODS

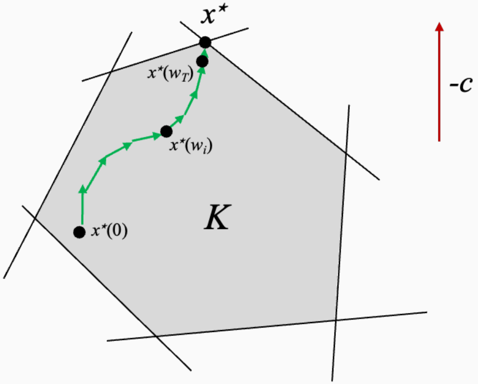
Lecture notes are posted on the website (optional reading).



Projected Gradient Descent Optimization Path

## INTERIOR POINT METHODS

Lecture notes are posted on the website (optional reading).



Ideal Interior Point Optimization Path

## POLYNOMIAL TIME LINEAR PROGRAMMING

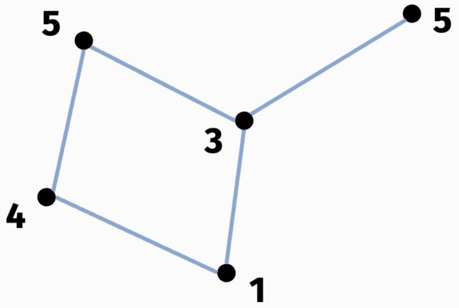
Both results had a huge impact on the theory of optimization, although at the time neither the ellipsoid method or interior point method were faster than a heuristic known at the Simplex Method.

These days, improved interior point methods often outperform simplex.

Polynomial time linear programming algorithms have also had a huge impact of combinatorial optimization. They are often the work-horse behind approximation algorithms for NP-hard problems.

## EXAMPLE: VERTEX COVER

Given a graph $G$ with $n$ nodes and edge set $E$. Each node is assigned a weight $w_1, \ldots, w_n$.



**Goal:** Select subset of nodes with minimum total weight that covers all edges.

## EXAMPLE: VERTEX COVER

Given a graph $G$ with $n$ nodes and edge set $E$. Each node is assigned a weight $w_1, \ldots, w_n$.

**Formally:** Denote if node $i$ is selected by assigning variable $x_i$ to 0 or 1. Let $\mathbf{x} = [x_1, \ldots, x_n]$.

$$\min_{\mathbf{x}} \sum_{i=1}^{n} x_i w_i \qquad \text{subject to} \qquad x_i \in \{0, 1\} \text{ for all } i$$

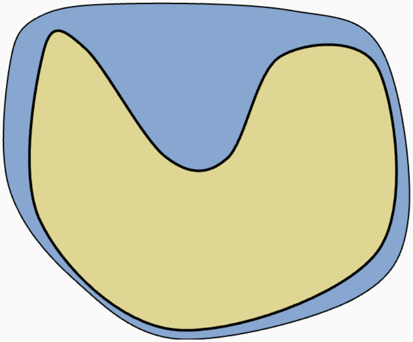
$$x_i + x_j \geq 1 \text{ for all } (i, j) \in E$$

**NP-hard to solve exactly.** We will use convex optimization give a 2-approximation in polynomial time.

Function to minimize is linear (so convex) but constraint set is not convex. Why?

High level approach:

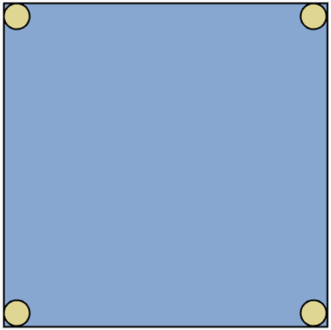
- Relax to a problem with convex constraints.
- Round optimal solution of convex problem back to original constraint set.



## RELAX-AND-ROUND

High level approach:

- Relax to a problem with convex constraints.
- Round optimal solution of convex problem back to original constraint set.



## RELAX-AND-ROUND

**High level approach**:

- Relax to a problem with convex constraints.
- Round optimal solution of convex problem back to original constraint set.

Let $\bar{\mathcal{S}} \supseteq \mathcal{S}$ be the relaxed constraint set. Let $\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x})$ and let $\bar{\mathbf{x}}^* = \arg\min_{\mathbf{x}\in\bar{\mathcal{S}}} f(\mathbf{x})$. We always have that:

$$f(\bar{\mathbf{x}}^*) \leq f(\mathbf{x}^*).$$

So typically the goal is to round $\bar{\mathbf{x}}^*$ to $\mathcal{S}$ in such a way that we don't increase the function value too much.

## RELAXING VERTEX COVER

**Vertex Cover:**

$$\min_{\mathbf{x}} \sum_{i=1}^{n} x_i w_i \qquad \text{subject to} \qquad x_i \in \{0,1\} \text{ for all } i$$
$$x_i + x_j \geq 1 \text{ for all } (i,j) \in E$$

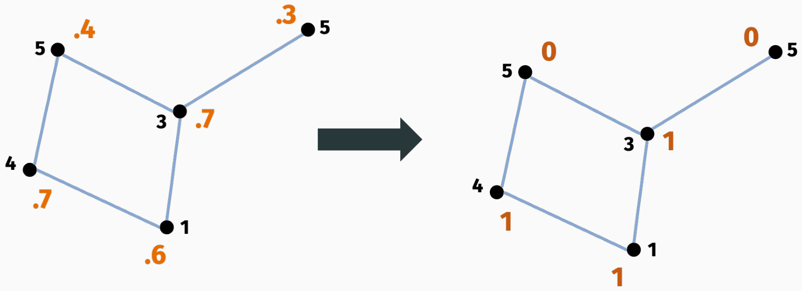
**Relaxed Vertex Cover:**

$$\min_{\mathbf{x}} \sum_{i=1}^{n} x_i w_i \qquad \text{subject to} \qquad 0 \leq x_i \leq 1 \text{ for all } i$$
$$x_i + x_j \geq 1 \text{ for all } (i,j) \in E$$

The second problem is a linear program! It can be solved in $\text{poly}(n)$ time!

## ROUNDING VERTEX COVER

**Simple rounding procedure:** If $\bar{x}_i^* \geq 1/2$, set $x_i = 1$, and set $x_i = 0$ otherwise.



**Observation 1:** All edges remain covered. I.e., the constraint $x_i + x_j \geq 1$ for all $(i, j) \in E$ is not violated.

## ROUNDING VERTEX COVER

**Observation 2:** Let $\mathbf{x}$ be the rounded version of $\bar{\mathbf{x}}^*$. We have $f(\mathbf{x}) \leq 2 \cdot f(\bar{\mathbf{x}})$, and thus $f(\mathbf{x}) \leq 2 \cdot f(\mathbf{x}^*)$.

**Proof:**

## VERTEX COVER

So, a polynomial time algorithm for solving LPs immediately yields a 2-approximation algorithm for the NP-hard problem of vertex cover.

- Proven that it is NP-hard to do better than a 1.36 approximation in [Dinur, Safra, 2002].
- Recently improved to $\sqrt{2} \approx 1.41$ in [Khot, Minzer, Safra 2018], which proved the 2-to-2 games conjecture.
- Widely believed that doing better than $2 - \epsilon$ is NP-hard for any $\epsilon > 0$, and this is implied by Subhash Khot's Unique Games Conjecture.

There is a simpler greedy 2-approximation algorithm that doesn't use optimization at all. Try coming up with it on your own!

## SPECTRAL METHODS

**Next section of course:** Spectral methods and numerical linear algebra.

Spectral methods generally refer to methods based on the "spectrum" of a matrix. I.e. on it's eigenvectors/eigenvalues and singular vectors/singular values. We will look at applications in:

- Low-rank approximation and dimensionality reduction.
- Data clustering and related problems.
- Constructing data embeddings (e.g. Word2Vec).