Outline:

How to solve most convex optimization problems, IRL.

Polynomial time.

Convex problems we can solve in polynomial time.

$\min_x f(x)$ s.t. $g(x) \le 0$, f, g convex

$\exists \lambda \ge 0$ st. $h_\lambda(x) = f(x) + \lambda g(x)$

global min of $h_\lambda(x)$ is solution

But which λ?

$\lambda = 0$ just min f

$\lambda \to \infty$ just min g

x_λ be sol to $h_\lambda(x)$

$0 < \lambda < \infty$ $f(x_\lambda)$ gets bigger and $g(x_\lambda)$ smaller

binary search on λ? $g(x_\lambda) = 0$

Just hope for $-\varepsilon \le g(x_\lambda) \le 0$

$(1-t) f(x) + t g(x)$ search on $t \in [0,1]$

$\sim f(x) + \frac{t}{1-t} g(x)$ $\lambda = \frac{t}{1-t}$

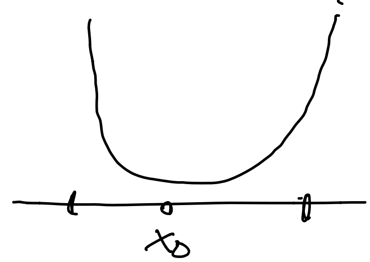
Interior Point Methods (aka Barrier Methods)

Consider $\min_x f(x) + \varepsilon \frac{1}{-g(x)} = f(x) - \frac{\varepsilon}{g(x)}$

s.t. $g(x) \le 0$

$g(x)$ convex $\Rightarrow$ $-g$ concave $\Rightarrow$ $\frac{1}{-g(x)}$ convex

Problem is still convex but goes to ∞ at boundary



local approach to global which stays inside $g(x) \le 0$

Start with ε big – so just want a feasible point.
Lower ε, and track solutions as you go

Really use $f(x) - \varepsilon \log(-g(x))$

$-\log(-g(x)) \to \infty$ as $g(x) \to 0$

Version 1: $\min f(x)$ s.t. $g_1(x) \le 0, \dots, g_k(x) \le 0$

$$f(x) - \underbrace{\varepsilon \sum_{i=1}^{k} \log(-g_i(x))}_{\text{Barrier}}$$

Start with ε

decrease $\varepsilon \leftarrow \varepsilon/2$ or $\varepsilon(1 - 1/\sqrt{m})$

x_{ε_t} find $x_{\varepsilon_{t+1}}$ start working from x_{ε_t}.

Usually only need high accuracy when $\varepsilon \to 0$

Usually use Newton's Method, not gradients.

$\min F(x)$ Taylor approx to F

small δ

$$F(x+\delta) \approx F(x) + \nabla F(x)^T \delta + \tfrac{1}{2}\delta^T (\underbrace{\nabla^2 F(x)}_{\text{Hessian...}})\delta$$

$$\delta = -\underline{(\nabla^2 F(x))^{-1}} \nabla F(x)$$

Usually go to $\underline{F}(x + t\delta)$ some t

Karmarkar solves LPs in polynomial time.

Solve for very small ε, and can then round to exact solution.

Generalized Ineqs $x \in K_i$ $\mathbb{R}^n_+$, S^n_+

Polynomial Time.

input size N. poly time always performs
at most $\le O(N^c)$
for some fixed constant c ($c=2$ or 3, $c=6$)

Gaussian Elim n-by-n matrix $N=n^2$ #s.
run time $\le O(n^3) = O(N^{3/2})$

Add two b bit #s, time $\le O(b)$.
comparison $\le O(b)$.
Multiplying b bit #s time $\le O(b \lg_2 b)$

$x_0 = 3$
for i in 1 to n
$x_i = x_{i-1}^2$

$x_n = 3^{2^n}$ need $\Theta(2^n)$ bits to write down.

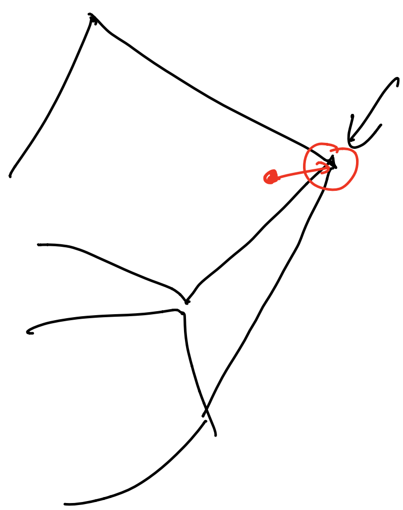
LU factorization is poly time, even when
count bits.

LP is poly time, but only when counting bits.
# of ops actually depends on # of bits...
Running time really depends on K,

poly in $\log(K)$.
and K can be $\exp(\#\text{bits})$

Approx Solve LP: given c, A, b, $\varepsilon > 0$
we can find an x s.t. $Ax \le b$
and $c^T x \ge c^T x_* - \varepsilon$
Time is poly in # ~~bits~~ used to write A, b, c and ε

once ε small enough, can round to a solution



Confirm solution has poly in the input size.

True because true for $Ax = b$

Use rationals. Write inputs as p/q,
p and q are integers.

# of bits to write $p \le 1 + 2\lceil \log_2 p \rceil$ bits

size(p) = # bits need to write p

size(x) = # bits to write x.

$x = p/q \qquad \text{size}(x) \le \text{size}(p) + \text{size}(q)$

$p_1 \ldots p_n$ integers

$$\text{size}\left(\prod p_i\right) \le \sum_i \text{size}(p_i)$$

$$\lceil \log \prod p_i \rceil \le \sum_i \lceil \log p_i \rceil$$

rationals

$$\text{size}\left(\sum_i x_i\right) \le 2 \sum_i \text{size}(x_i) \qquad x_i = p_i/q_i$$

$$\sum x_i = \frac{\sum_i p_i \prod_{j \ne i} q_j}{\prod q_i}$$

$$\text{size}\left(\prod q_i\right) \le \sum \text{size}(q_i)$$

$$\le \left(\sum p_i\right)\left(\prod q_i\right)$$

$$\le \text{size}\left(\sum p_i\right) + \text{size}\left(\prod q_i\right)$$

n-vector $(x_1 \ldots x_n)$ $\quad \text{size}(n) + \sum_i \text{size}(x_i)$

Lem for n-by-n matrix A

$$\text{size}(\det(A)) \le 2 \cdot \text{size}(A)$$

$$\det = \sum_{\text{perms } \pi} (-1)^{\text{sign}(\pi)} \cdot \prod_{i=1}^{n} A(i, \pi(i))$$

$$A(i,j) = p_{i,j} / q_{i,j}$$

$$\det = p/q \qquad q \le \prod q_{i,j}$$

$$|\det(A)| \le \prod_{i,j} (1 + |p_{i,j}|)$$

$$\text{size } d \le 2 \sum_{i,j} \text{size}(p_{i,j})$$

¿q?

Lem $Ax = b$, $\text{size}(x) \le 4n(\text{size}(A) + \text{size}(b))$

Cramer's rule says $x(i) = \dfrac{\det(A_i)}{\det(A)}$

$A_i = A$, but ith col replaced by b.

$$\text{size}(x(i)) \le 2(\text{size}(A) + (\text{size}(A) + \text{size}(b)))$$

$$\le 4n(\text{size}(A) + \text{size}(b))$$

LLL is poly time.

<u>Thm</u> The LP $c^T x$ s.t. $Ax \le b$
has solution vector x_* of size
$\le 4d(\text{size}(A) + \text{size}(b))$

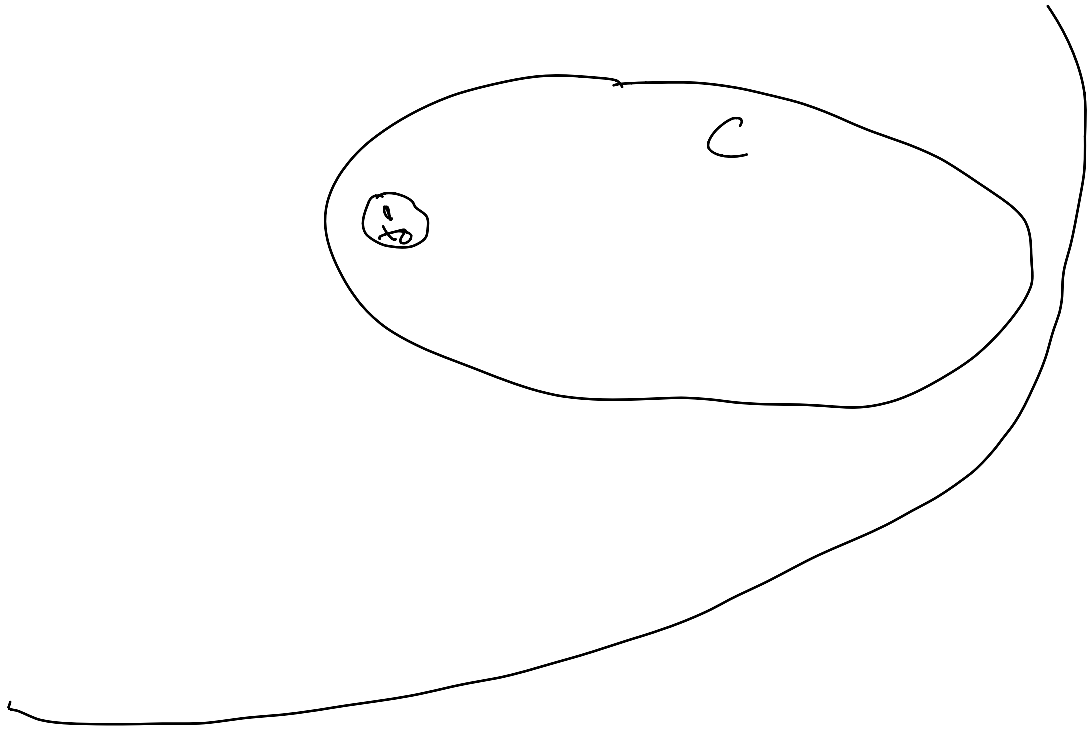
$\min_x f(x)$ s.t. $x \in C$ convex C, f

x is ε-approx $\exists$ optimal x_* s.t. $\|x - x_*\| \le \varepsilon$

Need to be able to tell if $x \in C$.
Assume can call a function to tell.
Membership Oracle

f given by a "function oracle".
$f(x) = c^T x$ is simple
On input x, δ get back rational t s.t.
$|t - f(x)| \le \delta$
and not too many bits.

need $x_0 \in C$
and r, R s.t. $B(x_0, r) \subseteq C \subseteq B(x_0, R)$



Can ε-approx solutions in time poly in
$\log(1/\varepsilon)$, $\log(R/r)$, size(x_0), n

$x \in \mathbb{R}^n$