Lecture 14: How do LP solvers work?

e.g. LP: maximize $5x_1 + x_2$

s.t. $x_1 + x_2 \le 7$ ⓐ

$-2x_1 + 3x_2 \ge -4$ ⓑ

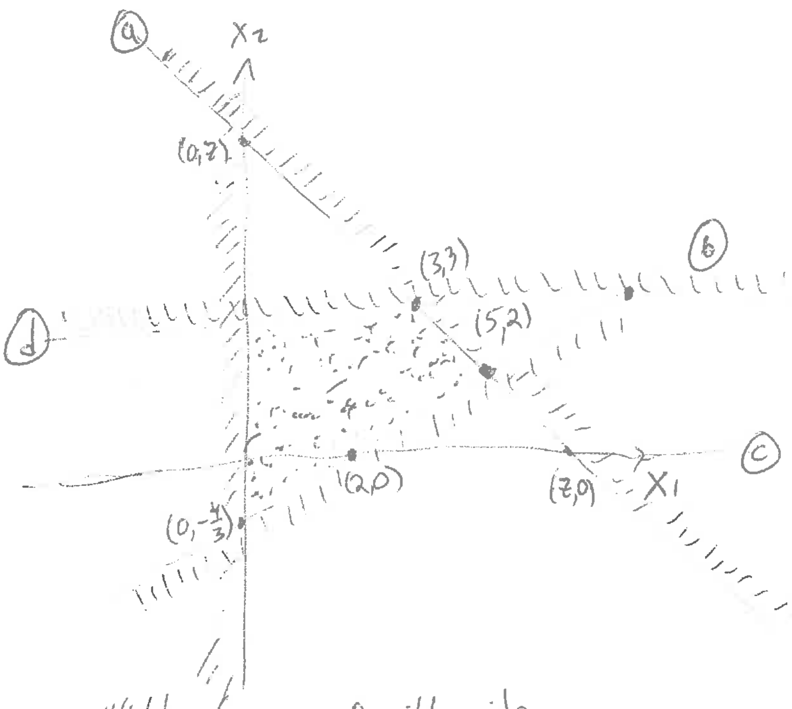
$x_1 \ge 0$ ⓒ

$x_2 \le 3$ ⓓ

Geometric viewpoint: plot the "feasible points" $x = (x_1, \dots, x_n)$ in $\mathbb{R}^n$. [n=2 in our example] [satisfying all constraints]

[Think first of each inequality as an equality. The points satisfying as equality are "on the boundary" of satisfying ineq. In n=2 case, equality is a line. In n=3: plane. Generally: "hyperplane"]

[Each inequality says you're on one side of line/plane/hyperplane. It's a "halfspace".] "Feasible region" is intersection of halfspaces



[shading shows infeasible side of halfspace]

[dotted area is feasible]

Stupid/annoying possibility 1:
region is empty
"LP is infeasible"

e.g. "$x_1 \ge 2$, $x_1 \le 1$".

[Automatically not the case for many natural problems. E.g. Max-flow: can always set all flow vals $f_e = 0$.]

Stupid/annoying possibility 2:
region is unbounded
e.g. If our example didn't have $x_1 \ge 0$

[Also usually not an issue; don't mind including $x_i \ge 2^{-1000}$ & $x_i \le 2^{1000}$ for all $i = 1 \dots n$.
Then everything inside a big box. usually.]

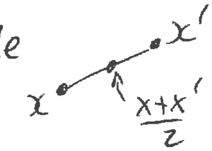
[[There are stupid/annoying tricks to get around these stupid/annoying possibilities. Let's not worry about it for now.]]

Ignoring stupidities:

- Feasible region is a "convex polytope"

  convex: given any two feasible pts x, x', their average $\frac{x+x'}{2}$ also feasible

  polytope: [[high-dim. generalization of a "polygon"]] vertices, edges, flat faces...



[[actually, all points on line segment feasible]]

Observation: Every "vertex" (corner) is the intersection/solution of n constraint-equations. [[linearly indep]] [[In our $n=2$ example, of 2 lines.]]

⤳ If # of constraints is m, then $\leq \binom{m}{n}$ vertices [[some intersects may not be feasible]]

Given any n out of m constraints, can efficiently find intersection, [[solving linear system]] check if feasible

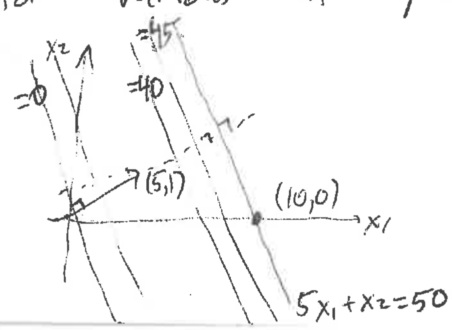
But... $\binom{m}{n}$ is exponentially large! E.g. if $m=2n$, $\binom{m}{n} = \binom{2n}{n} \geq 2^n$.

[[Why are we so interested in corners?]]

Fact: Optimum solution always occurs at a vertex.

[[Unfortunately, expon. many, so can't just check them all.]]

[[Why?]] E.g. objective is $5x_1+x_2$. Plot $5x_1+x_2 = \beta$ for various #'s β...



Objective value at $x=(x_1,x_2)$

$= (5,1) \cdot (x_1, x_2)$

= proportional to how far x is in direction $(5,1)$

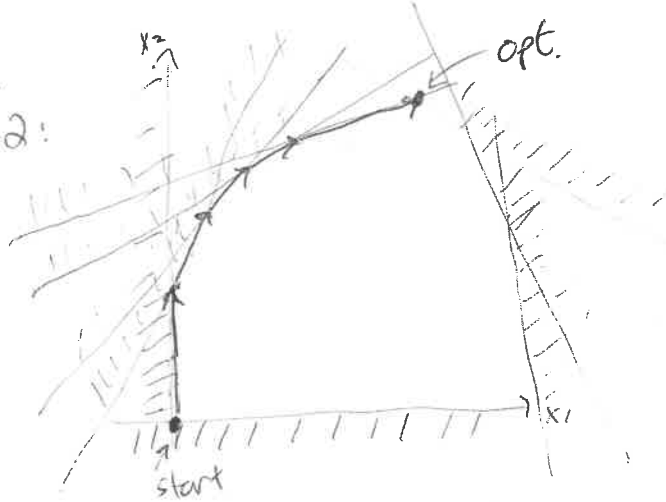
(proj. to (5,1))

[[Intuition/proof for why optimum always occurs at a vertex: keep sliding $5x_1+x_2=\beta$ line more & more, larger and larger β, till last time it still intersects feasible region. Last point of contact always a vertex (could also be a whole edge/facet).]]

Suggests a ("physical") algorithm:

- start anywhere in feasible region
- keep moving in objective direction [[eg, (5,1)]] till you hit a wall
- keep trying to move as much in obj. direction as you can, hugging wall, till onther wall
- keep going till stuck. ← (beauty of LP: ∄ bad "local optima". Every local opt. is a global opt! :))

e.g. $n=2$:



objective: x_2 $(= 0x_1 + 1x_2)$

[[Moving in this direction is like a helium balloon always trying to float up]]

Note: - this ends up walking from vertex to vertex, along edges.

- in 2d, only one choice at each vtx, but for $n \geq 3$ can have many improving edges.

This is how the "Simplex Algorithm" solves LP.

Pros: • Simple alg. To figure out an acceptable next vertex is some simple linear algebra calculation

- Often efficient in practice. [[heuristically]]

Cons: • How to find a starting vertex?

[Often not a big deal; e.g., $x_i = 0\ \forall i$ is often a vertex. This is true for flows: $f_e = 0\ \forall e$ is a vertex. In general, though, have to do an annoying stupidity that involves setting up and solving another artificial LP!]

• Provably exponential time in worst case.

[Can get stuck doing expon. many exponentially small improvements]

First known ~~provably~~ poly-time alg: [but very much not practically efficient]

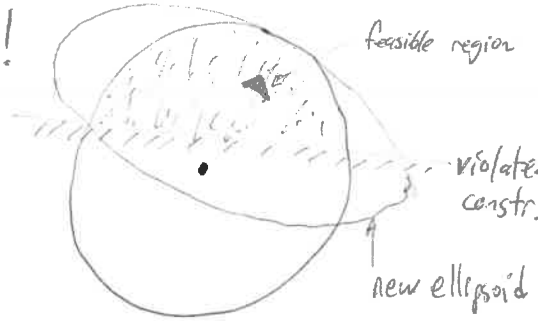
"Ellipsoid Alg."

[Can always binary search for optimum, adding "objective $\geq \beta$" as a constraint [for various β], so suffices to solve "find a feasible point"]

• Searches for feasible point from outside

1. Start with giant sphere, sure to contain feasible region.
2. Check if center is feasible. If so, done! Else, get a "violated constraint".
3. Compute [not hard] smallest ellipsoid containing ~~[crossed out]~~ cut-off chunk.
4. Go to 2.

// give up if current ellipsoid exponentially small.



Key theorem: after $3n$ slicings, ellipsoid's volume is halved (or smaller)

[So it's like binary search: geometric shrinking of volume]

[Why ellipsoids? No deep reason; just a simple kind of shape for which it's easy to compute the smallest enclosure of the previous chunk, and prove the volume decreases geometrically.]

Bonus: Alg doesn't need to "see/know" all constraints.

Just needs to solve "separation oracle" problem:

given x, → if feasible, say so (*)
→ if not, output violated constraint

[Sometimes you can set up an LP for a problem with n vbls & $\exp(n)$ many constraints, but where there's still an efficient alg for (*). Then these LPs can be solved in poly(n) time!]

~

"Interior point methods" - efficient in theory and in practice.

[As the name suggest, these algs walk thru the interior of the polytope.]

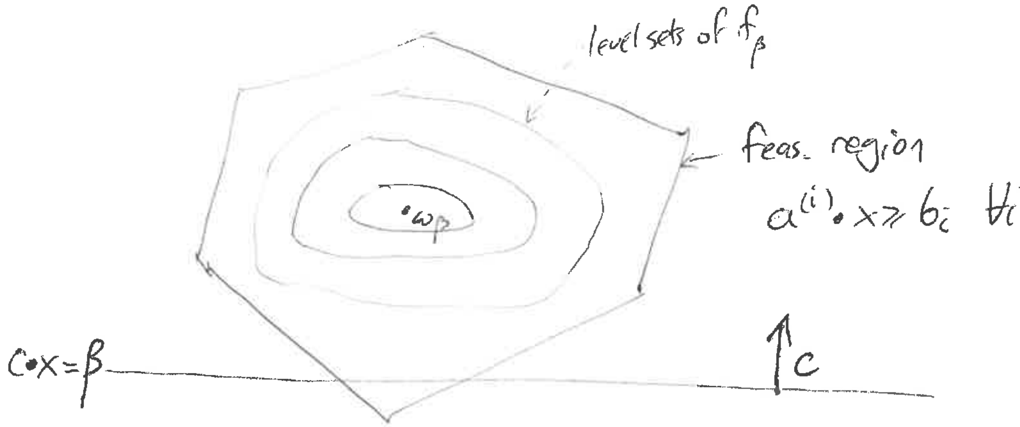
max $c \cdot x$

s.t. $a^{(1)} \cdot x \geq b_1$

⋮

$a^{(m)} \cdot x \geq b_m$

$P :=$ (the constraints above)



Define $P_\beta = P \cup \{c \cdot x \geq \beta\}$. Want to make β as big as possible while still having a point $x \in P_\beta$.

Idea: "hard constraints" → "soft constraints" ("barrier fcn")

$a \cdot x \geq b$ ⟿ "$\ln(a \cdot x - b)$ is large"

[Objective function is most important, so "take m copies of it".]

Define $f_\beta(x) = m \cdot \ln(c \cdot x - \beta) + \sum_{i=1}^{m} \ln(a^{(i)} \cdot x - \beta)$.

- Inside P_β, $f_\beta(x)$ is well-defined
- $f_\beta(x) = -\infty$ on boundary of P_β
- $f_\beta(x)$ is smooth & concave ⟹ has unique maximizer w_β where $\nabla f_\beta = 0$

  ↳ "analytic center of P_β"

Idea: • $t := 0$

- Start with very small β_t, some generic $x_t \in P_{\beta_0}$
- ~~Find w_{β_0}~~

  ~~Use gradient ascent to move from x_t towards w_{β_t}~~

  Use Newton's method " " " " " "

  [a "second order/derivative"-based faster method we can use because f_{β_0} is strictly concave, no saddle points, and 2nd-deriv (Hessian) of f_β can be explicitly given]

  but just a little bit [one step of Newton's method]

  x_{t+1} is new point
- $\beta_{t+1} := \beta_t$ + a little bit
- Repeat.

Analysis ideas:

- Newton's method converges rapidly // you basically get very close to w_{β_t}
- Since $c \cdot x \geq \beta$ so highly weighted (factor m), w_{β_t} must be at least a little bit far from $c \cdot x \geq \beta$

  ⟹ okay to increase β by a little

  $\left(\approx \frac{1}{\sqrt{m}}\right)$

  (⟹ alg time is $\approx \sqrt{m} \cdot$ time to compute matrix inverse of Hessian... $m^{3.5}$...?)