CSCI 3100

LP: Simplex

Today:

- HW due
- Next HW up
- Oral grading signup on Monday

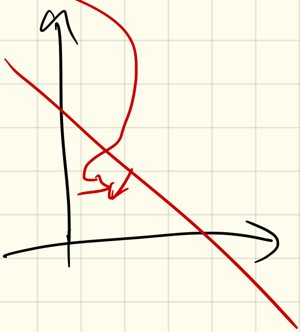
LP w/ d variables:

Each LP equality or inequality describes a *hyperplane* in $\mathbb{R}^d$.

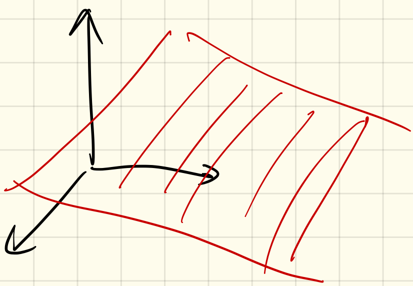
2d: $ax + by \le c$

$-\frac{a}{b}$ slope

y-intercept



3d: $ax + by + cz \le d$



$\mathbb{R}^d$: $a_1x_1 + \dots + c_d x_d \le c$

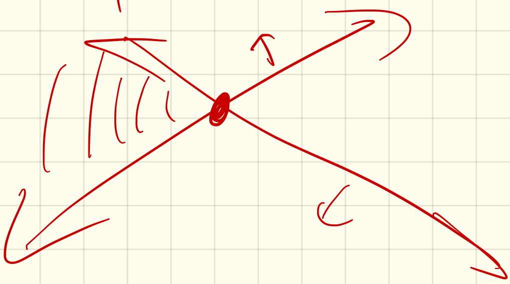
Vertices:

These happen when $\geq d$ hyperplanes meet in $\mathbb{R}^d$.

In $\mathbb{R}^2$:

$d = 2$

2 lines meet at a point



In $\mathbb{R}^3$:

Maximize $\quad x_1 + 6x_2 + 13x_3$ ~~$+ 2x_4$~~

s.t.

$x_1 \le 200$ ①

$x_2 \le 300$ ②

$x_1 + x_2 + x_3 \le 400$

$x_2 + 3x_3 \le 600$

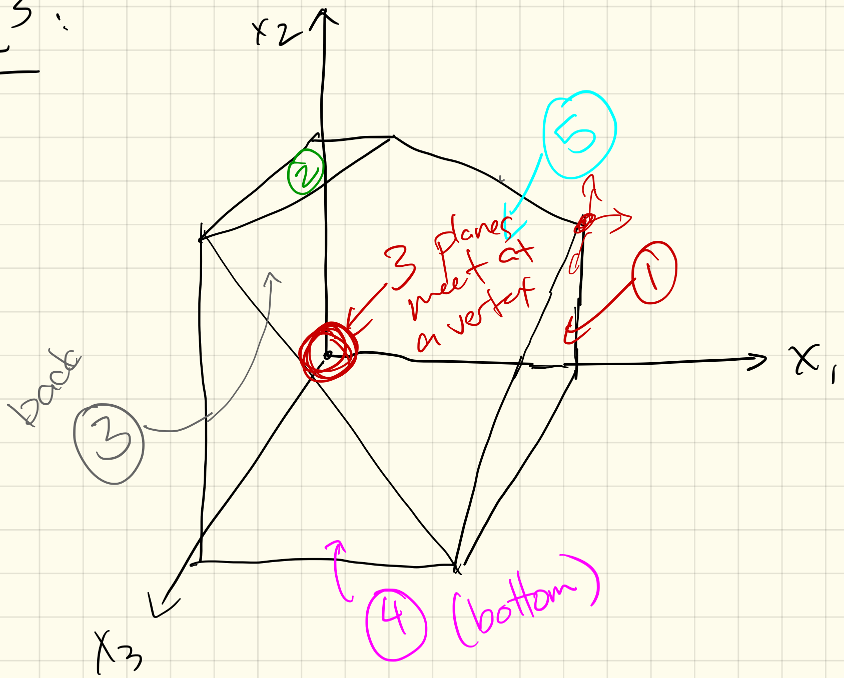
and

$x_1 \ge 0$ ③

$x_2 \ge 0$ ④

$x_3 \ge 0$ ⑤

$\mathbb{R}^3$:



<u>Dfn</u>: Pick a subset of inequalities.
If there is a unique point that satisfies all <u>with</u> equality, & it is feasible
↳ this is a <u>vertex</u> of the solution.

In general: Each vertex is specified by exactly d equations
(in $\mathbb{R}^d$)

(Again, think 2 & 3d examples)

Neighbors:
Any vertices that share $d-1$ inequalities

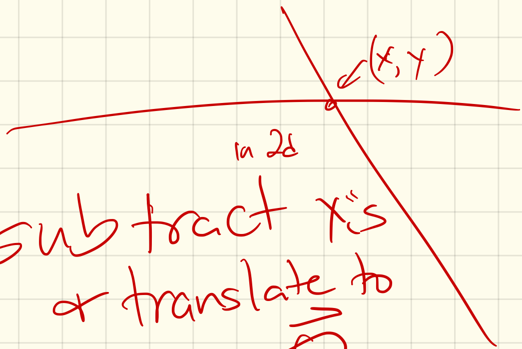
Simplex algorithm:

In each stage, 2 tasks:

1. Check if current vertex is optimal
2. If not, choose a nbr vertex that improves the result

Both are easy at the origin (next slide).

If not at $\vec{0}$:



subtract x's

or translate to $\vec{0}$

LP: max $c^T x = c_1 x_1 + \dots + c_d x_d$

s.t. $A\vec{x} \le \vec{b}$ → $a_{11}x_1 + x_{12}x_1 + \dots + x_{1d}x_1 \le b_1$

$\vdots$

$x_i \ge 0 \quad \forall i$

Note: $\vec{x} \in \mathbb{R}^d$, so

$x = (x_1, \dots, x_d)$

Start w/ origin, so our $\vec{x} = \vec{0}$ $\Rightarrow x_1 = 0$

$x_2 = 0$

$\vdots$

It is always a vertex! (Why?) $\underbrace{\text{all } x_i \ge 0}_{d}$

optimal only if:

all c_i's are negative

Conversely:

If <u>any</u> $c_i > 0$, we can increase the obj. function $c^T \vec{x}$.

How? Increase x_i

So: pick one & increase!

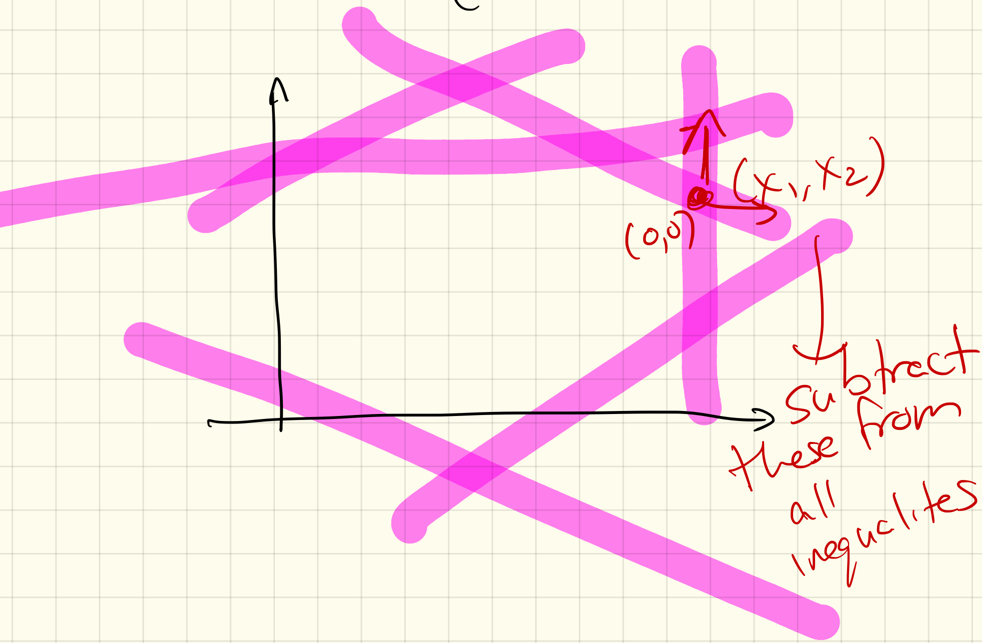
How much?

until we hit another constraint:

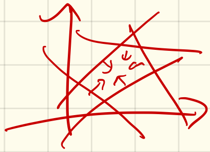
calculate x_i' intercepts

<u>Now</u>: What if <u>not</u> at origin?

Transform LP!
(ie shift all coords)



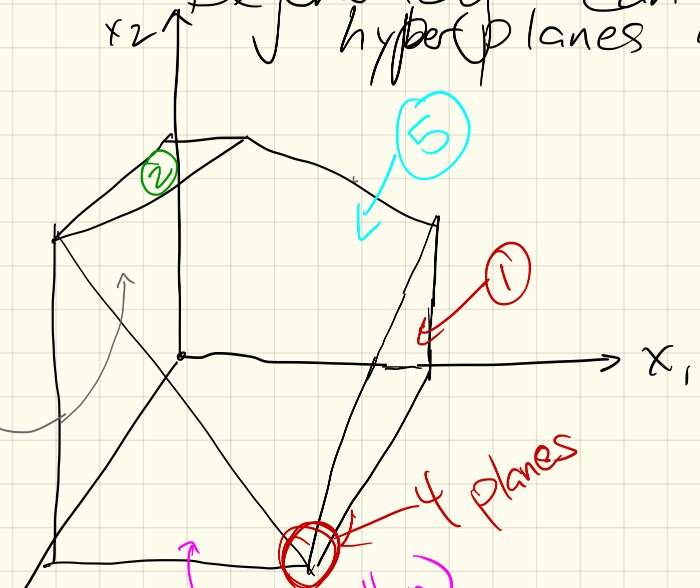
Some details



- Origin isn't always feasible,
  ↳ so must find a starting feasible point.
  (& reset to be $\vec{0}$)

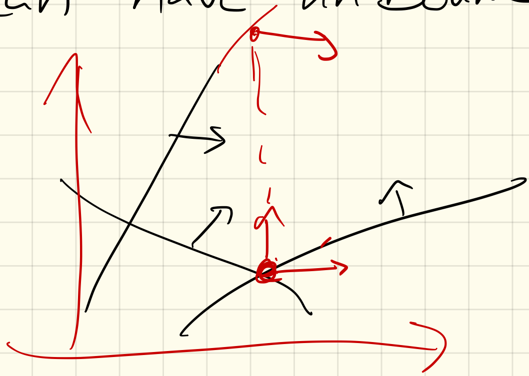
  Turns out, this is a (simpler) LP!
  (see notes)

- Degeneracy: Can have $>d$ hyperplanes at a vertex!



- Unboundedness:

Can have unbounded situation:



Detection:

When exploring for next vertex, swapping out an equality for another will not give a bound.

↳ Simplex stops & complains

Runtime:

Consider a vertex $u \in \mathbb{R}^n$, with m inequalities.

At most $n \cdot m$ nbrs:

choose one to drop & one to add: $\leq n(m-n)$ (could be smaller)

Checking for nbr:

Each is a dot product / matrix operation.

Gaussian elimination: $O(n^3)$ (basically)

$\Rightarrow$ Each iteration:

$O(mn^4)$

Can improve slightly:

- just need one $c_i > 0$
  & rescaling to $\overrightarrow{0}$ is easy.

$\Rightarrow$ Can improve to $O(mn)$ per iteration.

How many iterations?

- $m+n$ inequalities
- any n give a vertex

$$\Rightarrow \binom{m+n}{n} = O\left((m+n)^n\right)$$

$$\hookrightarrow O\left((m+n)^n \cdot mn\right)$$

Ick! Klee-Minty give examples that are actually this slow. (in 50's)

Alternatives

- Ellipsoid algorithm
  (Khachiyan '79)
- Interior point method
  (Karmarkar in '80's)

Polynomial

But:

In practice, simplex
does 'better'!