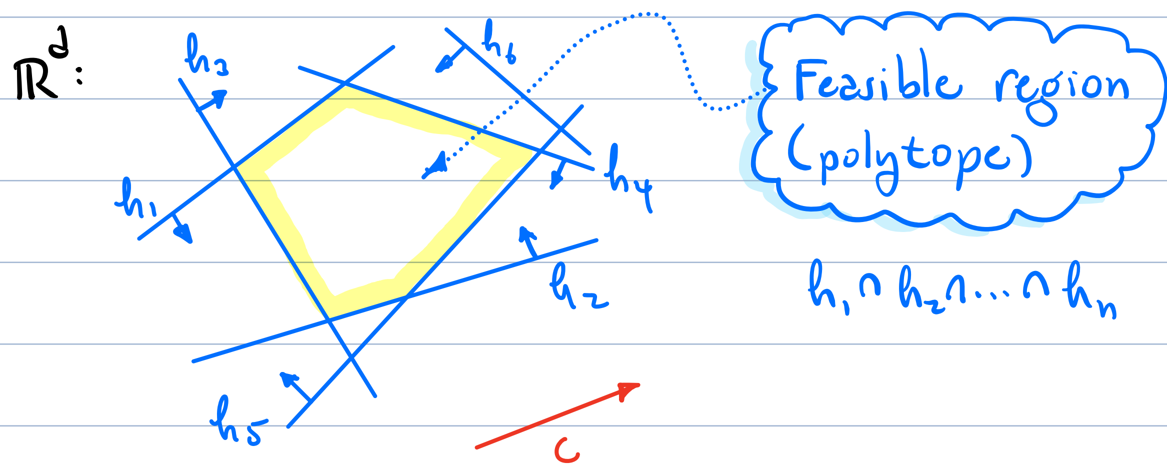
CMSC 754 – Computational Geometry
Lecture 7: Linear Programming (Preliminary)

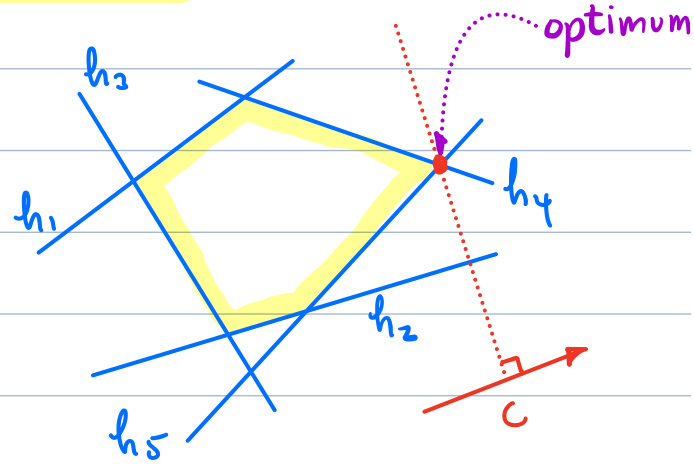
Linear Programming (LP):

- A fundamental optimization technique in $\mathbb{R}^d$
- Given a set of n linear constraints (halfspaces)
  $H = \{h_1, \ldots, h_n\}$, where $h_i : a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{id}x_d \leq b_i$



- also given a linear objective function
  $c(x) = c_1x_1 + \cdots + c_dx_d = c^Tx$ $\quad c, x \in \mathbb{R}^d$

Goal: Find the point of the feasible region that maximizes the objective function (equiv, maximizes the orthogonal projection on the objective vector)



Matrix Form:

Given an $n \times d$ matrix A, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}^d$, find $x \in \mathbb{R}^d$ to:

maximize: $c^T x$

subject to: $Ax \leq b$

i^{th} row of A corresponds to h_i

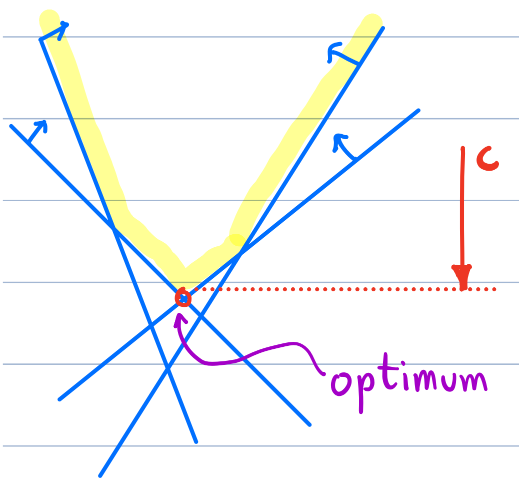
Three possible outcomes:

- ☺ **Feasible:** A (finite) optimal point exists

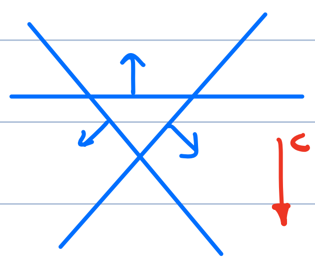
Assuming genl. pos. this is a vertex of feasible polytope

- ☹ **Infeasible:** No solution because feasible region is empty
- ☹ **Unbounded:** No optimum finite solution exists because feasible region is unbounded in direction of objective function

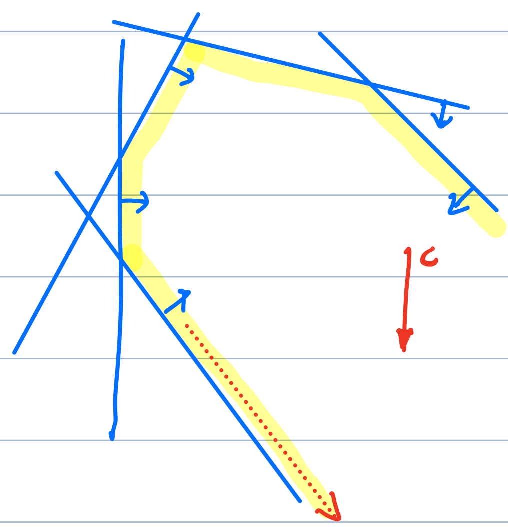
Feasible



Infeasible

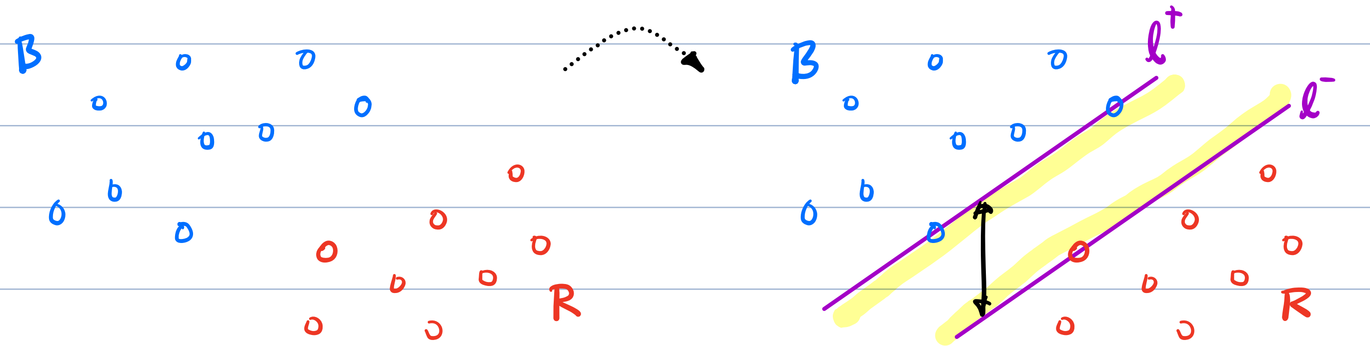


Unbounded



Example:

- Given two point sets, B + R, in $\mathbb{R}^2$, find two parallel lines ℓ^+ + ℓ^- that are maximally vertically separated, with ℓ^+ above ℓ^-, s.t. B lies on or above ℓ^+ + R lies on or below ℓ^-.



Formulated as an LP: **Points:** $b_i = (b_{ix}, b_{iy}) \in B$ $\quad r_j = (r_{jx}, r_{jy}) \in R$

Lines: $\ell^+: y = e \cdot x + f^+ \qquad \ell^-: y = e \cdot x + f^-$

Constraints: for $i = 1, \dots, |B|$ $\quad b_{iy} \geq e \cdot b_{ix} + f^+$ $\quad$ (b_i above ℓ^+)

for $j = 1, \dots, |R|$ $\quad r_{iy} \leq e \cdot r_{ix} + f^-$ $\quad$ (r_j below ℓ^-)

$f^+ \geq f^-$ $\quad$ (ℓ^+ above ℓ^-)

Objective: maximize $f^+ - f^-$

Matrix form:

$$x = \begin{bmatrix} e \\ f^+ \\ f^- \end{bmatrix} \qquad c = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} b_{1x} & 1 & 0 \\ \vdots & & \\ b_{mx} & 1 & 0 \\ -r_{1x} & 0 & -1 \\ \vdots & & \\ -r_{nx} & 0 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} e \\ f^+ \\ f^- \end{bmatrix} \leq \begin{bmatrix} b_{1y} \\ \vdots \\ b_{my} \\ -r_{1y} \\ \vdots \\ -r_{ny} \\ 0 \end{bmatrix}$$

LP Formulation:

Find $(e, f^+, f^-) \in \mathbb{R}^3$

maximizing: $f^+ - f^-$

subject to:

$b_{iy} \geq e \cdot b_{ix} + f^+$

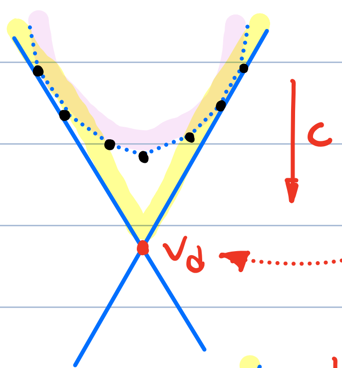
$r_{jy} \leq e \cdot r_{jx} + f^-$

$f^+ \geq f^-$

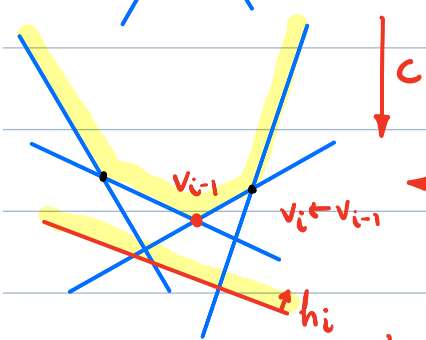
LP in spaces of constant dimension:

- Assume n is large (asymptotic quantity)
  - & d is constant (we'll ignore even exponential depedence)
- We'll present a (randomized) algorithm with (expected) running time $O(d!\,n) = O(n)$.

Incremental Approach:

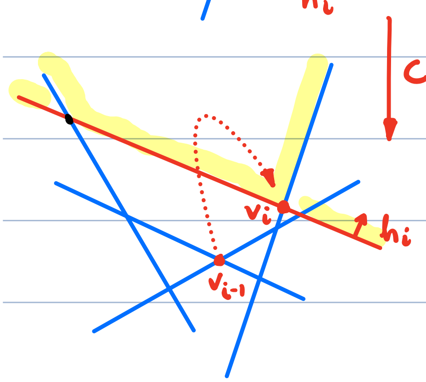


Initialization: Find d-halfspaces that define a bounded solution v_d (or report that LP is unbounded) $\rightarrow O(dn)$ time



Incremental:

for $i \leftarrow d+1$ to n
- add halfspace h_i
- if $v_{i-1} \in h_i$
  set $v_i \leftarrow v_{i-1}$ (no change)

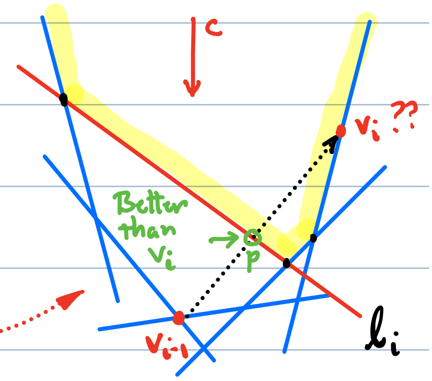


- else
  $v_i \leftarrow$ new optimal vertex
  (How? Later)

Lemma: If $v_{i-1} \notin h_i$, the new optimum v_i lies on h_i's bounding hyperplane.

Proof: Let ℓ_i denote h_i's bounding hyperplane

- Assume (for illustration) c is directed downwards.
- $v_{i-1} \in h_i \Rightarrow v_{i-1}$ is below ℓ_i
- Suppose to contrary $v_i \notin \ell_i$
  - $\Rightarrow v_i$ lies above ℓ_i
  - $\Rightarrow$ Segment $\overline{v_{i-1} v_i}$ intersects ℓ_i
- Let $p = \overline{v_{i-1} v_i} \cap \ell_i$

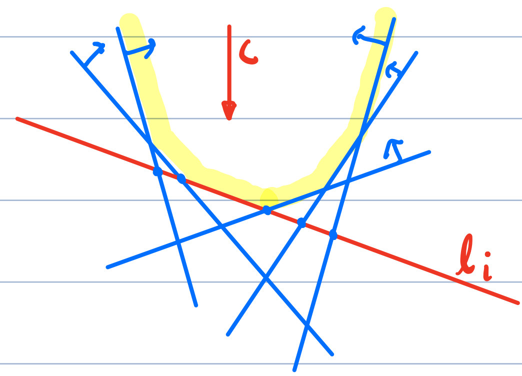


- By convexity, p is feasible
- By linearity, objective fn. decreases from $v_{i-1} \to v_i$
  - $\Rightarrow p$ is a better solution than v_i

$\rightarrowtail\!\!\!\leftarrowtail$ contradiction! $\square$

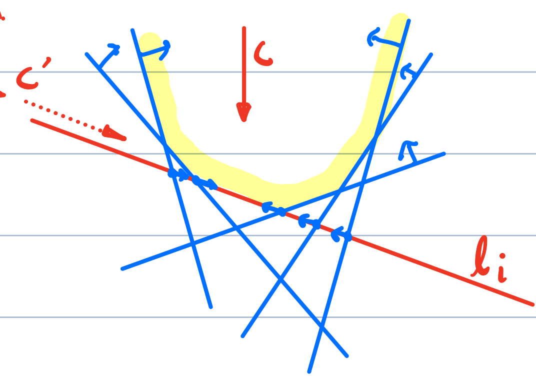
How to update optimal vertex?

- Given prior halfspaces $h_1, \dots h_{i-1}$, objective vector c, and current hyperplane ℓ_i



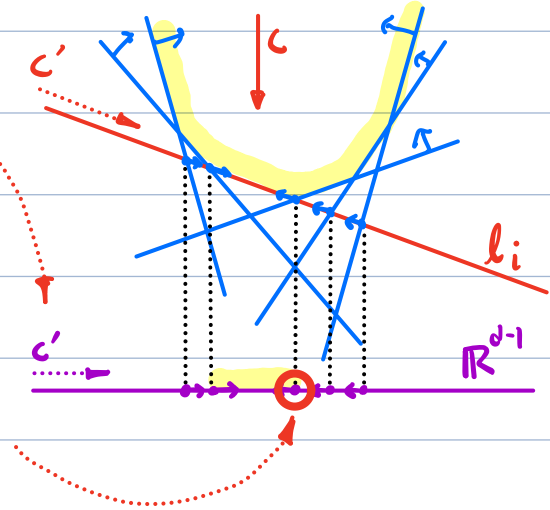
① Intersect $h_1, \dots, h_{i-1}$ with ℓ_i

② $c' \leftarrow$ Project c onto ℓ_i



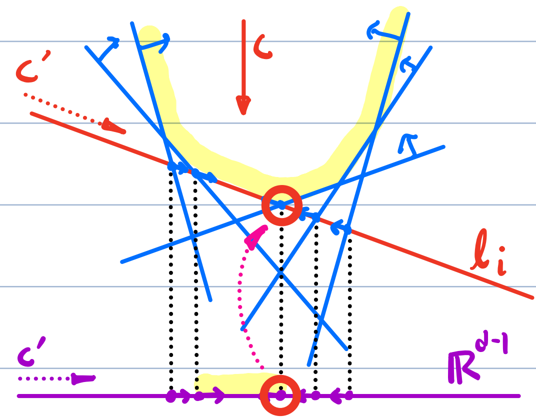
③ Project all this to $\mathbb{R}^{d-1}$

(Gauss elimination)



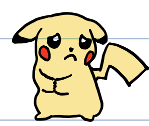
④ Recursively solve this (d-1)-dim LP on i-1 constraints

⑤ Project this back onto l_i

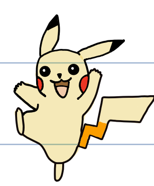


Running Time:

- Worst case - Bad! $O(n^d)$



- Expected case (Random insertion order) - Good! $O(n)$



Worst-Case Analysis:

- Let $W_d(n)$ = worst-case time for n constraints in $\mathbb{R}^d$
- View algorithm recursively, rather than incremental

- Recurrence:

$$T_d(n) = \underbrace{T_d(n-1)}_{\text{Solve LP on first } n-1 \text{ constraints}} + \underbrace{d}_{\text{Test feasibility of } v_{i-1}} + \Big[\underbrace{dn}_{\text{Project onto } \ell_i} + \underbrace{T_{d-1}(n-1)}_{\text{Solve recursively}}\Big]$$

By induction (exercise) can show $T_d(n) \leq c \cdot n^d$

$\Rightarrow$ $T_d(n) = O(n^d)$

(To be continued)