Lecture 7 - Linear programming

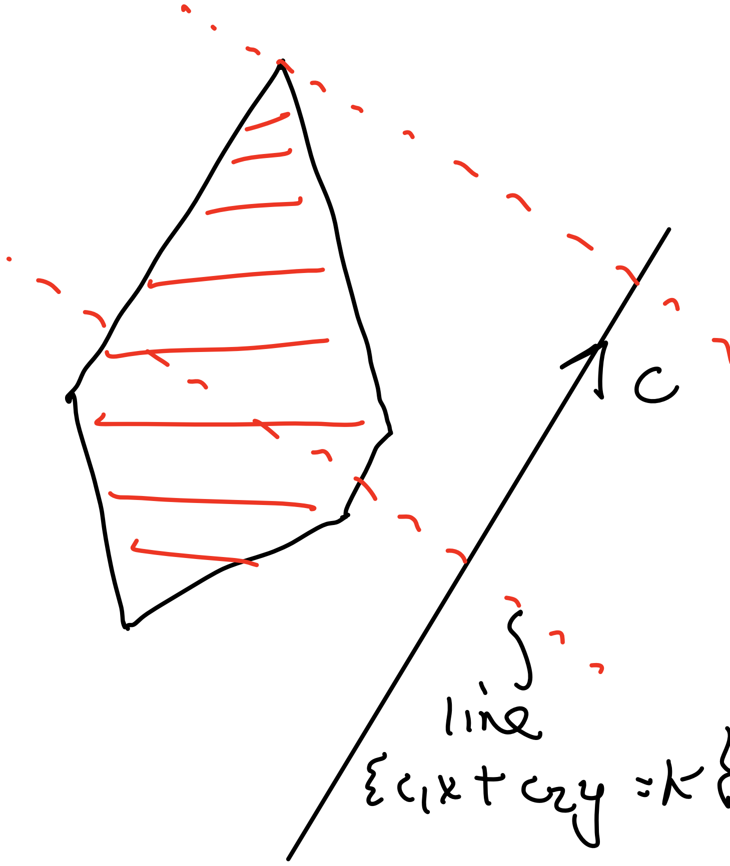
- Consider $f: \mathbb{R}^2 \to \mathbb{R} : (x,y) \mapsto c_1 x + c_2 y$ where $(c_1, c_2) \neq (0,0)$.
  & a set $H = \{h_1, \dots, h_n\}$ of half-planes.
- Goal: find a point $(x,y) \in \bigcap_{h_i \in H} h_i = \bigcap H$ at which f attains max value.
- We will write
  $h_i : a_{i1} x + a_{i2} y \leq b_i$ for $i = \{1, \dots, n\}$

Geometric significance

- f determined by vector $\vec{c} = (c_1, c_2)$

- As we move in the direction of $\vec{c}$, the value of f increases:

$$f((x,y) + t(c_1,c_2)) = f(x,y) + t f(c_1,c_2) = f(x,y) + t(c_1^2 + c_2^2) > f(x,y).$$



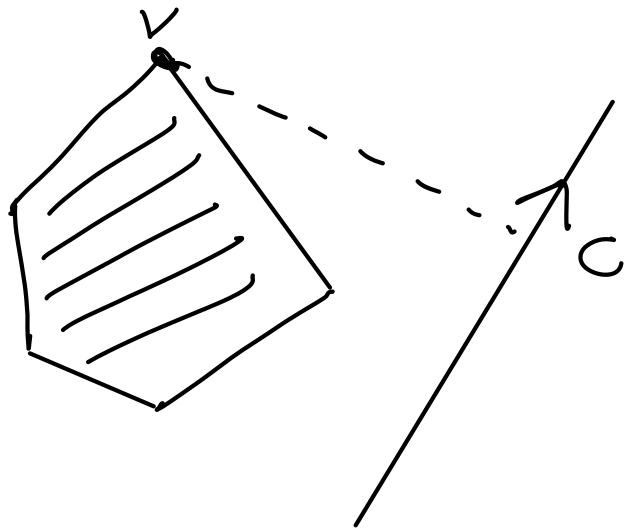
- At lines $\{(x,y) : c_1 x + c_2 y = k\}$ f has constant value. These are lines perpendicular to $\vec{c}$.

- Hence f obtains maximal value at any point v in intersection which is extreme in the direction of c.

Different possibilities

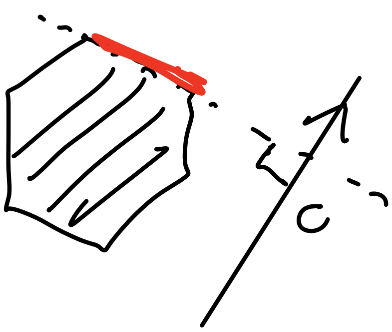
1) $\cap H$ is empty. No solution - problem is infeasible.

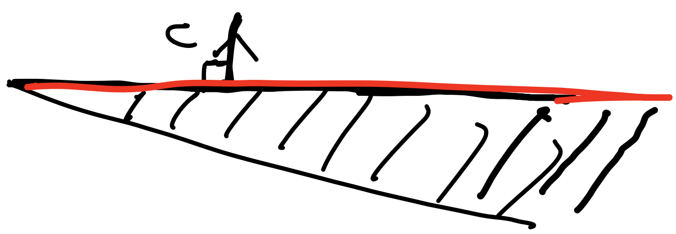
2)

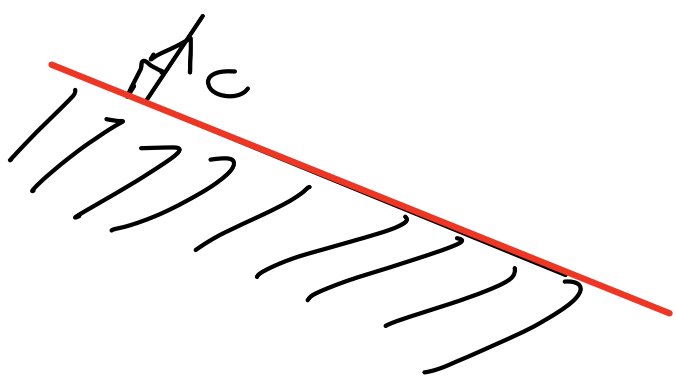


Just 1 point v at which f obtains a max. value.

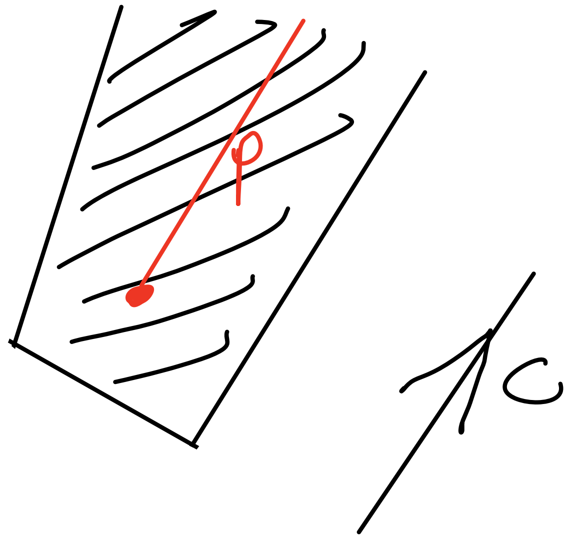
3) **Infinitely many solutions: these form a segment, line or half-line.**







4)



The function f is unbounded on intersection:

there exists a half-line in the intersection along which f is increasing. (p in picture)

Input to algorithm

Vector $\vec{c}$ & $H = \{h_1, \ldots, h_n\}$ a set of half-planes.

Output

- If problem is infeasible, provide 3 halfplanes with empty intersection.
- If f achieves a maximum, provide such a maximum in intersection (if more than one point, choose smallest with respect to lex ordering)
- If f not bounded above in intersection, provide half-line in intersection along which f is increasing.

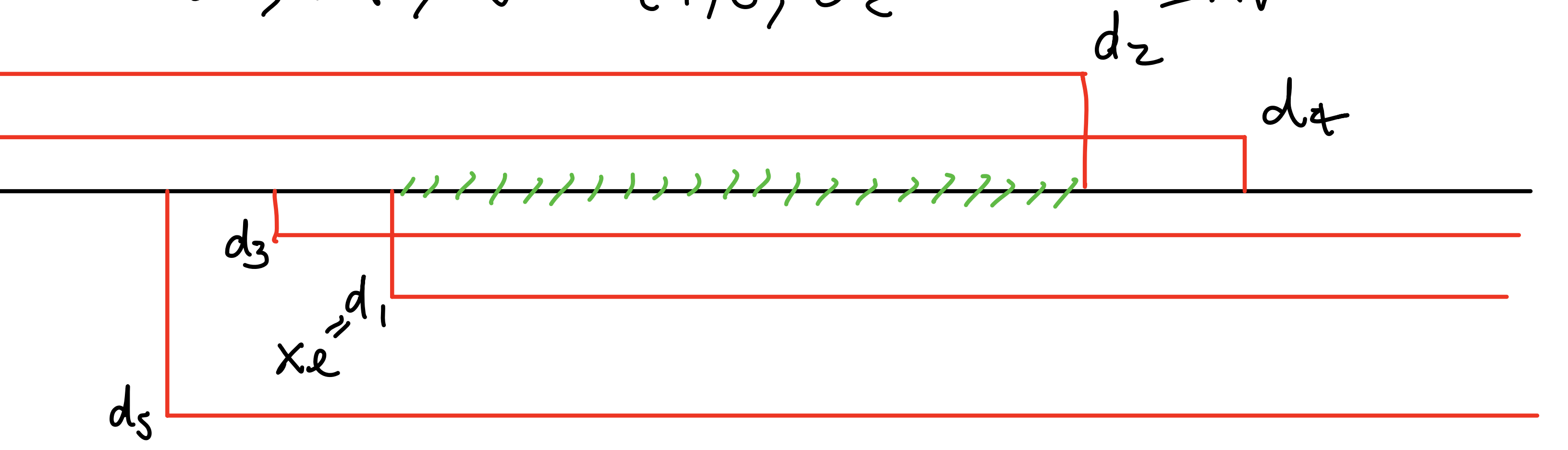
Firstly solve 1-d case

- $f(x) = cx$ for $c \neq 0$
- half-planes $a_i x \leq b_i$ for $a_i \neq 0$, $i = 1, \dots, n$.

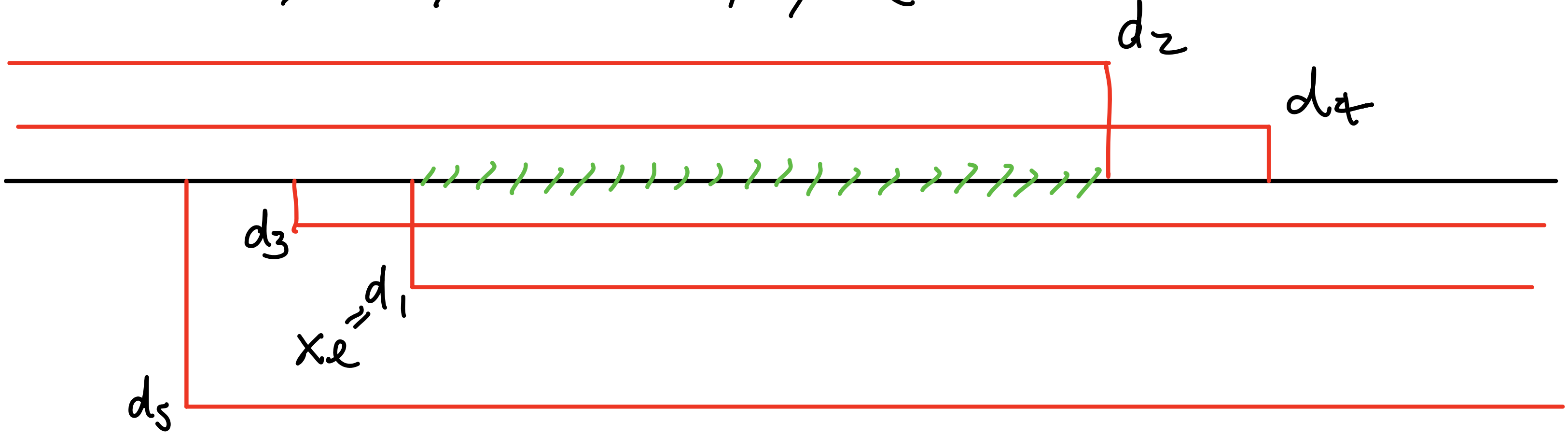
Goal: find point in intersection at which f attains max value.

- Let $I = \{i : a_i > 0\}$, $J = \{j : a_j < 0\}$
- Half-plane equations become
  $x \leq b_i / a_i = d_i$ for $i \in I$
  & $x \geq b_j / a_j = d_j$ for $j \in J$.
- Let $x_\ell = \max\{-\infty, d_j : j \in J\}$
  $x_r = \min\{d_i, \infty : i \in I\}$.

$I = \{2, 4\}$, $J = \{1, 3, 5\}$



$I = \{2,4\}$, $J = \{1,3,5\}$



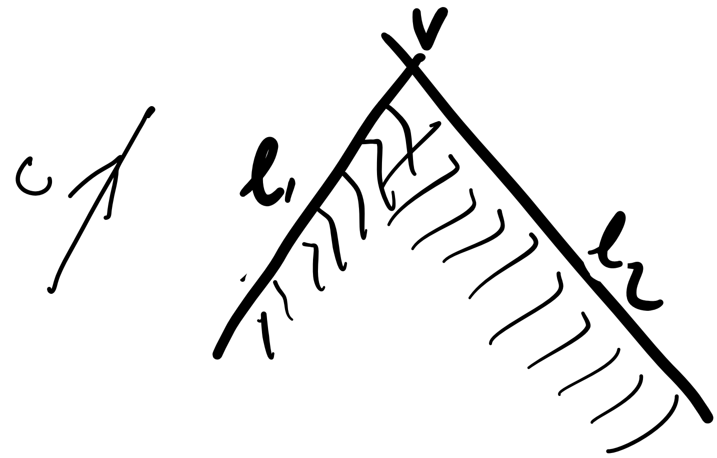
Cases

1. $x_r < x_\ell$ (Empty intersection - infeasible)
2. $x_\ell \leq x_r < \infty$ & $c > 0$. (f has max at x_r)
3. $x_r = \infty$ (I is empty) & $c > 0$ (f increases along $[x_\ell, \infty]$)
4. $-\infty < x_\ell \leq x_r$ & $c < 0$ (f has max at x_ℓ)
5. $x_\ell = -\infty$ (J is empty) & $c < 0$ (f increases along $[-\infty, x_r]$)

Complexity : $O(n)$

2-d case (bounded)

- In bounded case, we are given 2 half-planes h_1, h_2 such that f is bounded from above on $h_1 \cap h_2$.



- Then $h_1 \cap h_2$ has maximum at $l_1 \cap l_2 = v$
  - If there is more than 1 solution - choose least solⁿ with resp. to lex. ordering.

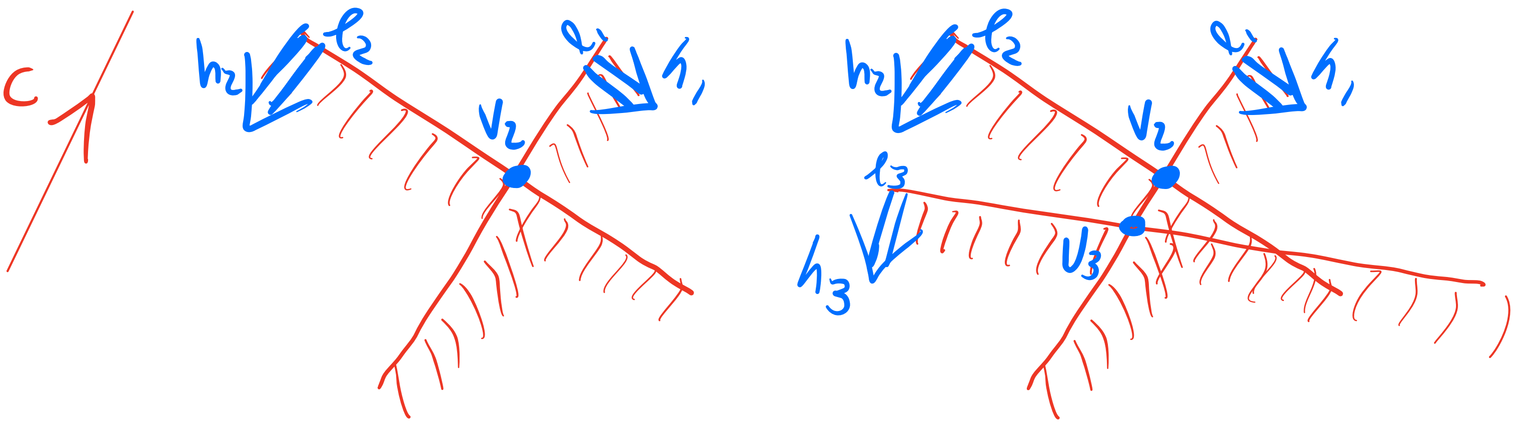
Incremental algorithm

- given optimal point $v_{i-1} \in C_{i-1} = h_1 \cap \dots \cap h_{i-1}$ we search for search for an optimal point $v_i \in h_i \cap C_{i-1} = C_i$.

- Call $v_i \in C_i$ optimal if f achieves maximum at $v_i \in C_i$ & v_i is least such with respect to lex ordering.

- If $v_{i-1} \in C_i$, then $v_i = v_{i-1}$.

- Else, C_i is empty or v_i lies on boundary ℓ_i of halfplane h_i.



- How to find v_i in this case?
- We can solve it using the 1-d algorithm: find a max of f on line $\ell_i \cap C_{i-1}$ in the 1-d halfplanes $\ell_i \cap C_j$ for $j \le i-1$.

- We can solve it using the 1-d algorithm: find a max of f on line $\ell_i \cap C_{i-1}$ in the 1-d halfplanes $\ell_i \cap C_j$ for $j \leq i-1$.

- ℓ_i is line $a_{i1}x + a_{i2}y = b_i$
- Assuming $a_{i2} \neq 0$ (else $a_{i1} \neq 0$) then $y = \dfrac{b_i - a_{i1}x}{a_{i2}}$
- Now consider f as a function of 1 variable on this line:

$$g(x) = f\left(x, \frac{b_i - a_{i1}x}{a_{i2}}\right) = c_1 x + c_2\big((b_i - a_{i1}x)/a_{i2}\big)$$

$$= \left(\frac{c_1 - c_2 a_{i1}}{a_{i2}}\right) x + c_2\left(\cancel{\frac{b_i}{a_{i2}}}\right)$$

- Its max does not depend on constant, so must find max of $g^*(x) = \left(\dfrac{c_1 - c_2 a_{i1}}{a_{i2}}\right) x$

in the 1-d halfplanes $\ell_i \cap C_j$:

$$a_{j1}x + a_{j2}\left(\frac{b_i - a_{i1}x}{a_{i2}}\right) \leq b_j \quad \text{For } j = 1, \dots, i-1.$$

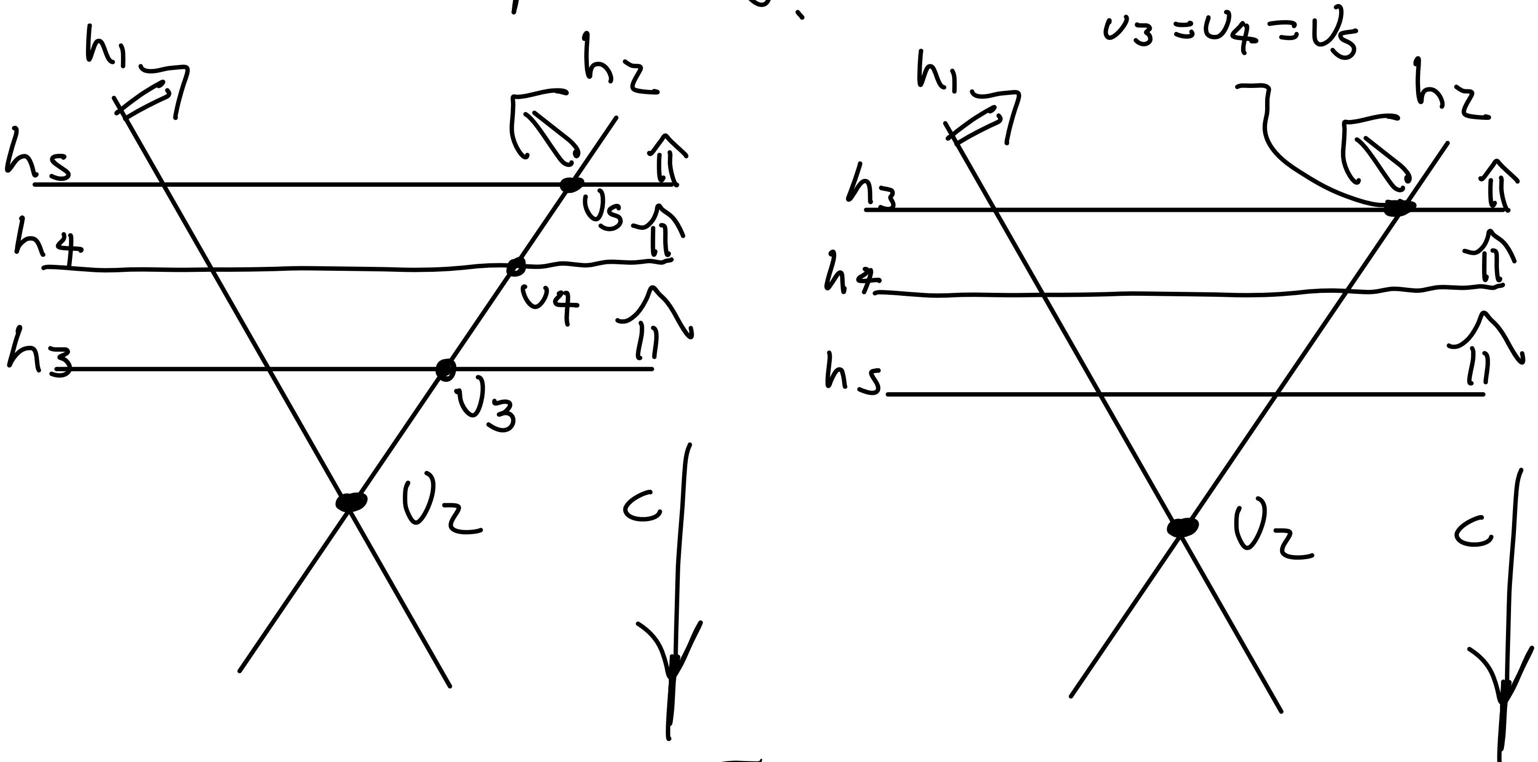
Rewrite as $\left(a_{j1} - \dfrac{a_{j2}a_{i1}}{a_{i2}}\right) x \leq b_j - \dfrac{a_{j2}b_i}{a_{i2}}$

- Now solve 1-d linear program for g^* at these half-planes $\rightarrow$ solution for 2-d case.

Running Time

- If $v_{i-1} \in h_i$, constant time to set $v_i = v_{i-1}$.
- Otherwise, time to calculate v_i is linear in i - so $O(i)$.
- Complexity: $O(3) + O(4) + \ldots + O(n)$
$$= O(3 + 4 + \ldots + n)$$
$$= O(n^2)$$

This is quite high running time & depends heavily on order of the half-planes.



- Introduce randomization to algorithm - randomize the order of the half-planes
(see L9 of pseudocode in E-Learning)

- Randomized expected time of alg. is much lower: average time of calculation taking into account all of possible orders

- Calculation of randomised exp. time:

X_i a random variable def. by $X_i = \begin{cases} 1 & \text{if } v_{i-1} \notin h_i \\ 0 & \text{if } v_{i-1} \in h_i \end{cases}$

- Time of alg. estimated by $\sum_{i=3}^{n} O(i)\, X_i$.

- Randomised expected time

$$E(X) = \sum_{i=3}^{n} O(i)\, E(X_i)$$

where $E(X_i) = \text{prob}(X_i = 1) = \text{prob}(v_{i-1} \notin h_i)$.

- As we will show,

$$\text{prob}(v_{i-1} \notin h_i) = 2/i.$$

Therefore $E(X) = \sum_3^n O(i) . 2/i = \sum_3^n O(1) = O(n)$.

Expected time is linear.

To show $\text{prob}(v_{i-1} \notin h_i) = 2/i$

- Now $v_i = l_j \cap l_k$ for $j, k \leq i$ & j, k minimal with these properties.

Then

$p(v_{i-1} \notin h_i) = p(v_i \neq v_{i-1})$

$= p(i = j \text{ or } i = k)$.

- There are $i(i-1)$ choices of pairs $j, k \leq i$.
- There are $i-1$ choices in which $j = i$.

— — — — $i-1$ — — — — — — $k = i$,

so $2(i-1)$ choices in which either j or k equals i.

So $\text{prob}(i = j \text{ or } i = k) = \dfrac{2(i-1)}{i(i-1)}$

$= 2/i$.

$\Rightarrow$ Expected randomised complexity is $O(n)$.

Unbounded case (sketch)

- Find half-line ρ in intersection along which f is increasing.
- $\rho = \{p + \lambda \vec{d} : \lambda \geq 0\}$.
- f increasing along ρ

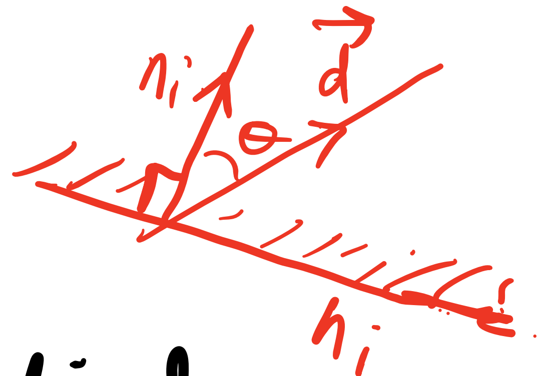
$\iff \theta_{\vec{c},\vec{d}} < 90°$

$\iff \vec{c} \cdot \vec{d} > 0$.

- ρ in h_i $\iff$

$\vec{n_i} \cdot \vec{d} \geq 0$

normal



- Hence to show it is unbounded, need to find

(*) $\vec{d}$ such that

$\vec{c} \cdot \vec{d} > 0$ & $\vec{n_i} \cdot \vec{d} \geq 0$ all h_i

& $\cap H$ non-empty.

- Can find such $\vec{d}$ using a 1-d linear program as well.

- Write $\vec{d} = \vec{c} + t\vec{e}$, $\vec{e}$ perp. to $\vec{c}$.
- $\vec{d}.\vec{c} = \vec{c}.\vec{c} + t\vec{e}.\vec{c} = \vec{c}.\vec{c} > 0$

  so must find t such that

  $(\vec{c} + t\vec{e}).\vec{n_i} \geq 0$ all i $\Leftrightarrow$

  $t(\vec{e}.\vec{n_i}) \geq -\vec{c}.\vec{n_i}$ (1-d linear program)
- If no sol, then there exists j, k for which system of two has no solution

  $\Rightarrow f$ bounded on $h_j \cap h_k$.

- One still needs to check whether $\cap H$ is non-empty.

  In fact, this holds $\Leftrightarrow$

  $H' = \{ h_i \in H : \vec{d}.\vec{n_i} = 0 \}$ is non-empty, which can again be solved using a 1-d linear program - see E-Learning.

- In the algorithm for 2-d linear programming, one first tests if it is unbounded.
  - If not, this produces 2 half-planes h_i, h_j on which it is
  → run bounded 2-d lin program.

- See E-Learning for more detailed pseudocode.

- Higher dimensional case can be handled recursively.