Linear programming

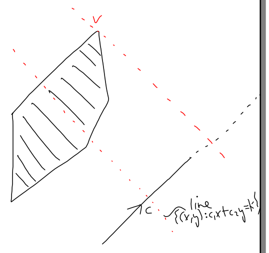
Consider $f : \mathbb{R}^2 \to \mathbb{R} : (x,y) \longmapsto c_1x + c_2y$ where $(c_1,c_2) \neq (0,0)$
& a set $H = \{h_1, \dots, h_n\}$ of halfplanes.

- Goal: find a point $(x,y) \in \bigcap_{h_i \in H} h_i = \bigcap H$ at which f attains a maximal value.
- We will write $h_i : a_{i1}x + a_{i2}y \leq b_i$ for $i \in \{1, \dots, n\}$.

lis 1-15:54

Geometric significance?

- f determined by vector $\vec{c} = (c_1,c_2)$
- As we move in the $\vec{c}$-direction f increases: i.e. for $t > 0$
$f((x,y) + t(c_1,c_2))$
$= f(x,y) + tf(c_1,c_2)$
$= f(x,y) + c_1^2 + c_2^2 > f(x,y)$
- At lines $\{(x,y) : c_1x + c_2y = k\}$ f has constant value. These lines are those perpendicular to $\vec{c}$.
- Hence f obtains maximal value at point v which is extreme in the direction of v

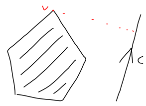


lis 1-16:08

Different possibilities

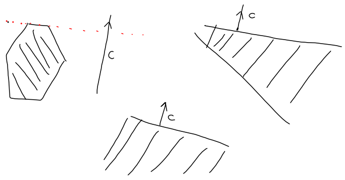
1) $\bigcap H$ is empty. No solution - problem is infeasible.

2)



Just have 1 pt at which f obtains max. value.

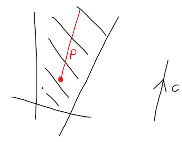
3)



Infinitely many solutions - these form a segment, or half-line, a line.

lis 1-16:18

4)



The function f is unbounded on the intersection: in this case there exists a half-line in the intersection along which f increases (p in picture).

Input to algorithm: vector $\vec{c}$ + $\{h_1, \dots, h_n\}$ a set of halfplanes.

Output: - If problem is infeasible, provide 3 halfplanes w/ empty intersection.
- Point in intersection at which f achieves its max: if more than one point exists, choose least point with respect to a chosen lex ordering.
- If f not bounded above on intersection = provide half-line in intersection along which f is increasing.

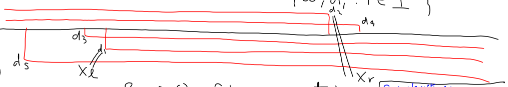
lis 1-16:24

Firstly, 1-d case: - $fx = cx$ for $c \neq 0$.
- half planes $a_ix \leq b_i$ for $a_i \neq 0$ & $i = 1, \dots, n$.

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

- Let $I = \{i : a_i > 0\}$, $J = \{j : a_i < 0\}$
- 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 $x_l = \max\{-\infty, d_j : j \in J\}$, $x_r = \min\{\infty, d_i : i \in I\}$

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



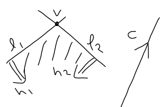
- Cases (1) $x_l \leq x_r < \infty$ & $c > 0$. f has max at x_r.
(2) $-\infty < x_l \leq x_r$ & $c < 0$. f has max at x_l.
(3) $x_r = \infty$ (I is empty) & $c > 0$ - $[x_l, \infty]$ is half-line along which f increases.
(4) $x_l = -\infty$ & $c < 0$ - then $(-\infty, x_r)$ is half-line along which f increases.

Complexity: find max & min from sets of n points. Complexity O(n).

lis 1-16:33

Bounded 2-d case

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



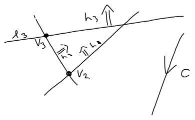
- Then $h_1 \cap h_2$ has maximum at $l_1 \cap l_2 = v$.
- If there is more than one soln - choose least one with respect to given lex. ordering.
-

lis 1-16:50

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

(Optimal point v_i: f reaches max at v_i in C_i, & v_i least among such points w.r.t. lex order)

- If $v_{i-1} \in C_i$ then $v_i = v_{i-1}$.
- Otherwise, C_i is empty or v_i lies on the boundary ℓ_i of the half-plane h_i.
- How to find v_i in this case? Let its co-ordinates be (x,y) – then $a_{i_1}x + a_{i_2}y = b_i$
- Assuming $a_{i_2} \neq 0$ (otherwise $a_{i_1} \neq 0$) we have $y = \frac{b - a_{i_1}x}{a_{i_2}}$.
- We search for max value of f on this line.



lis 1-16:54

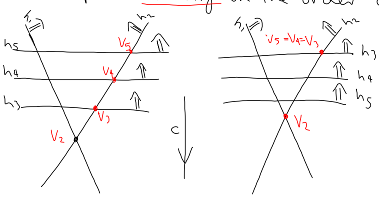
On this line, consider f as a function of one variable:

- $g(x) = c_1 x + c_2\left(\frac{b_i - a_{i_1}x}{a_{i_2}}\right) = \left(c_1 - \frac{c_2 a_{i_1}}{a_{i_2}}\right)x + c_2\left(b_i/a_{i_2}\right)$
- Want max of g on this line – does not depend on constant
  So we are seeking max of $(*)_1$ $\hat{g}(x) = \left(c_1 - \frac{c_2 a_{i_1}}{a_{i_2}}\right)x$
- We are looking for max of f on $\ell_i \cap C_{i-1}$, & this is given
  $a_{j_1}x + a_{j_2}\left(\frac{b_i - a_{i_1}x}{a_{i_2}}\right) \leqslant b_j$ for $j = 1, 2, \dots, i-1$.
- Rewrite as $(*_2)$ $\left(a_{j_1} - \frac{a_{j_2}a_{i_1}}{a_{i_2}}\right)x \leqslant b_j - \frac{a_{j_2}b_i}{a_{i_2}}$.
- Now we find v_i (or that C_i is empty) by solving 1-d linear program $(*_1, *_2)$.
- See code - lines 7-17 (on E-learning - except line 9)

lis 1-17:08

Running time – If $v_{i-1} \in h_i$ constant time required to set $v_i = v_{i-1}$.
– Otherwise time to calculate v_i is linear in i - so $O(i)$

- Complexity $O(3) + O(4) + \dots O(n) = O(3+4+\dots+n)$
- This is quite high running time, $= O(n^2)$.
  & depends heavily on the order of the half-planes.



Introduce randomization into algorithm – see L9 of code.

lis 1-17:22

- Randomized expected time of algorithm is much lower – average time of calc. taking into account all possible orders.
- Calculation of randomized expected time:
  X_i a random variable defined 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 algorithm estimated by
  $\sum_{i=3}^{n} O(i) X_i$.
- Expected time $E(X) = \sum_{i=3}^{n} O(i) E(X_i)$
  where $E(X_i)$ = probability $(X_i = 1) = \text{prob}(v_{i-1} \notin h_i)$.
- In fact $\text{prob}(v_{i-1} \notin h_i) = 2/i$.
- Hence $E(X) = \sum_3^n O(i) \cdot 2/i = \sum_3^n O(1) = O(n)$.

Expected time is linear $O(n)$.

lis 1-17:33

- Must calculate $\text{prob}(v_{i-1} \notin h_i)$.
- Now $v_i = \ell_j \cap \ell_k$ for $j, k \leqslant i$ & j, k min w' these props.
  $p(v_{i-1} \notin h_i) = p(j = i \text{ or } k = i)$
- 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 j or $k = i$.
  So $\text{prob}(j = i \text{ or } k = i)$ $\frac{2(i-1)}{i(i-1)} = 2/i$.

$\Longrightarrow$ Expected randomized complexity is $O(n)$.

See E-Learning for unbounded case.

lis 1-17:45