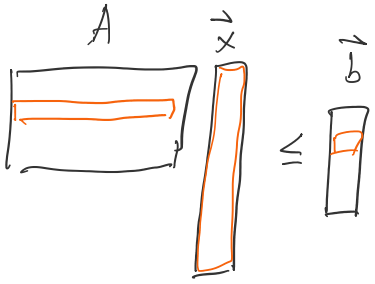
Lec25-linear-programming

Thursday, November 9, 2023 9:59 AM

Very big topic with entire textbooks devoted, but we will simply introduce basics.

Let $A \in \mathbb{R}^{m \times n}$ matrix.
$\vec{b} \in \mathbb{R}^m$
$\vec{c} \in \mathbb{R}^n$ } vectors

Choose $\vec{x} \in \mathbb{R}^n$ to
maximize $\vec{c} \cdot \vec{x}$ (dot product $\sum_{j=1}^{n} c_j x_j$)
s.t. $A\vec{x} \leq \vec{b}$.



Each row gives
$\vec{a_i} \cdot \vec{x} \leq b_i$
$\sum_j a_{ij} x_j \leq b_i$

Generalizing Minimize? Rewrite to maximize $\sum_j (-c_j) x_j$.

$\vec{a_i} \cdot \vec{x} \geq b_i$? Use $-\vec{a_i} \cdot \vec{x} \leq -b_i$ instead

$\vec{a_i} \cdot \vec{x} = b_i$? Use both $\geq$ & $\leq$ constraints

So minimizing or other constraint types $=, \geq$ allowed.

Example

Constraints: (rewrite →)

$2x_1 + x_2 \leq 4$ → $2x_1 + x_2 \leq 4$ •
$x_2 \geq 1$ → $-x_2 \leq -1$ •
$x_1 - x_2 \leq -2$ → $x_1 - x_2 \leq -2$ •

(as matrix →)

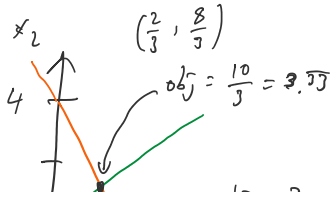
$$A = \begin{bmatrix} 2 & 1 \\ 0 & -1 \\ 1 & -1 \end{bmatrix} \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \leq \quad \vec{b} = \begin{bmatrix} 4 \\ -1 \\ -2 \end{bmatrix}$$

Objective:

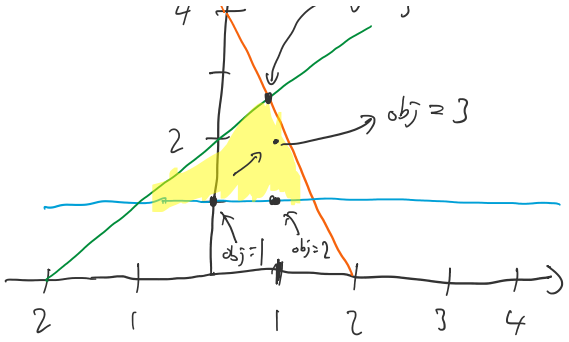
minimize $-x_1 - x_2$ maximize $x_1 + x_2$

obj $\vec{c} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Visual

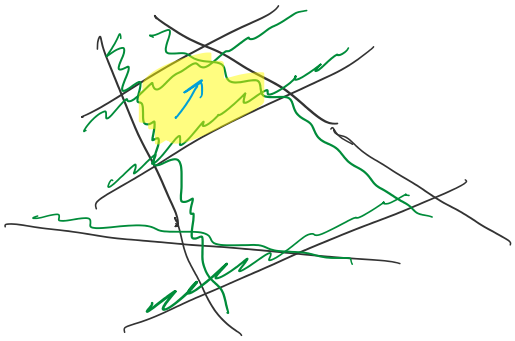


Geometric view:
lots of constraints form a feasible region.



lots of constraints form a feasible region.

We have a direction that increases objective.



History of LP algorithms

- Simplex method - George Dantzig 1940s

  Not poly-time, exponential time

  Fast in practice.

- Ellipsoid method, 1970s

  First poly-time alg for LP

  Horribly slow in practice; never used

- Interior Point methods

  - polynomial & practical 1980s in US - Karmarkar

    (1967 in Soviet union) Dikin

Lots of efficient implementations now: CPLEX, GLPK, Gurobi, etc (even Excel)

Applications of Linear Programming

We solved unrelated problems by converting to flow. We can do the same for LP.

Max-Flow Given directed graph $G=(V,E)$, with capacities c_e $\forall e \in E$,
find a flow from $s \in V$ to $t \in V$ of max value.

Constraints: $0 \le f(e) \le c_e \quad \forall e \in E$

$\forall$ except s & t $v \in V$, $\sum_{(u,v)\in E} f(u,v) = \sum_{(v,w)\in E} f(v,w)$

Maximize $\sum_{(s,w)\in E} f(s,w)$.

(or $\sum_{(u,t)\in E} f(u,t)$)

Rewrite as LP:

maximize $\sum_{(u,t)\in E} x_{ut}$

subject to $0 \le x_{uv} \le c_{uv} \quad \forall (u,v) \in E$

and $\sum_{(u,v)\in E} x_{uv} = \sum_{(v,w)} x_{vw} \quad \forall v \in V \setminus \{s,t\}$.

Minimum-cost flow

Given directed graph $G=(V,E)$ with capacities c_e & costs q_e on each edge s.t. sending α units of flow on edge e costs $\$\alpha q_e$.

Find a flow of value r from s to t that minimizes cost.

minimize $\sum_{(u,v)} q_{uv} x_{uv}$

s.t. $0 \le x_{uv} \le c_{uv} \quad \forall (u,v) \in E$

$\sum_{(u,v)\in E} x_{uv} = \sum_{(v,w)\in E} x_{vw} \quad \forall v \in V \setminus \{s,t\}$

$\sum_{(u,t)\in E} x_{ut} \ge r$

Shortest path: Given directed graph $G=(V,E)$, with positive edge lengths q_e, find the shortest path from s to t.

Let $x_{uv} = \begin{cases} 1 & \text{if we use edge } (u,v) \\ 0 & \text{else.} \end{cases}$

minimize $\sum_{(u,v)\in E} x_{uv} q_{uv}$

s.t. $\sum_{(u,v)\in E} x_{uv} = \sum_{(v,w)\in E} x_{vw} \quad \forall v \in V \setminus \{s,t\}$

$\sum_{(u,t)\in E} x_{ut} = 1$

equivalent to minimum cost flow of value 1

Solution might not be integer, but there always is a 0/1 solution

Converting to 0-1 solution takes a little more work to select a single path from the flow.

Maximum Bipartite Matching

Given bipartite graph $G = (A \cup B, E)$, choose as large a subset of edges $M \subseteq E$ as possible that form a matching.

large a subset of edges: objective

that form a matching: constraints

maximize $\sum_{u,v} x_{uv}$

s.t. $\sum_{u} x_{uv} \le 1$

$\sum_{v} x_{uv} \le 1$

similar problem that needs converting to 0-1 solution

Integer Linear Programming

Sometimes we can convert a continuous sol to a 0-1 solution like above.
Other times we need to explicitly add that as a constraint.

maximize $\vec{c} \cdot \vec{x}$

subject to $A\vec{x} \le \vec{b}$

and $x_i \in \{0,1\} \quad \forall i$

(or $x_i \in \mathbb{Z}$ in more general case)

Unfortunately, ILP is NP-hard, so no poly-time solution unless $P = NP$.
But lots of optimized code available & heuristics for the problem.

Minimum Vertex Cover Given graph $G = (V, E)$, choose $C \subseteq V$ s.t. every edge in E is incident to some $v \in C$, while minimizing $|C|$.

Ex. Choose people in social network so that everyone is friends with one of the people in the set.

Ex. Placing sensors on nodes in an electricity grid to monitor every wire.

As ILP: Let x_u $\forall u \in V$ specify if we choose u in our set.

$$\text{minimize} \quad \sum_{v \in V} x_v$$

$$\text{subject to} \quad x_u + x_v \geq 1 \quad \forall \{u, v\} \in E$$

$$x_u \in \{0, 1\} \quad \forall u \in V \quad \leftarrow \text{integrality constraint}$$

hard to solve in general.

Still useful to write problems as ILP, because lots of optimized solvers despite NP-hardness.

Turns out many useful problems can be converted to ILPs.
(even better if we can convert to LP)