

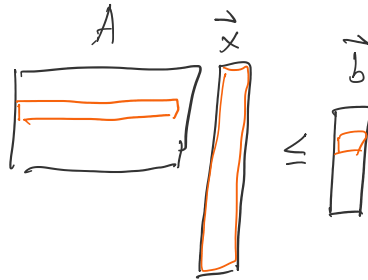
Lec26-linear-programming

Wednesday, November 6, 2024 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:

$$2x_1 + x_2 \leq 4$$

$$x_2 \geq 1$$

$$x_1 - x_2 \leq -2$$

Objective:

minimize $-x_1 - x_2$

rewrite ↘

$$2x_1 + x_2 \leq 4 \bullet$$

$$-x_2 \leq -1 \bullet$$

$$x_1 - x_2 \leq -2 \bullet$$

as matrix ↘

$$A \vec{x} \leq \vec{b}$$

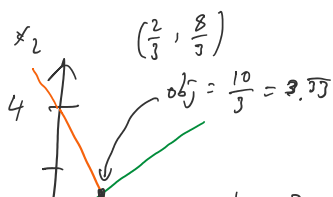
$$\begin{bmatrix} 2 & 1 \\ 0 & -1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 4 \\ -1 \\ -2 \end{bmatrix}$$

obj

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

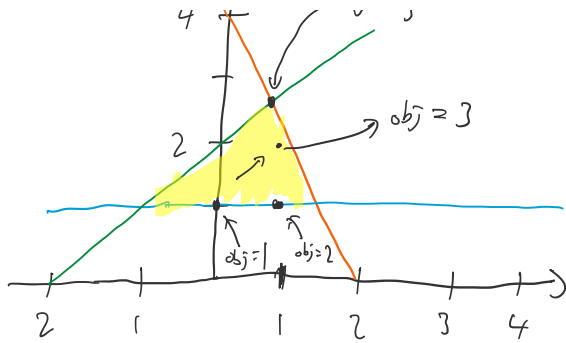
$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Visual



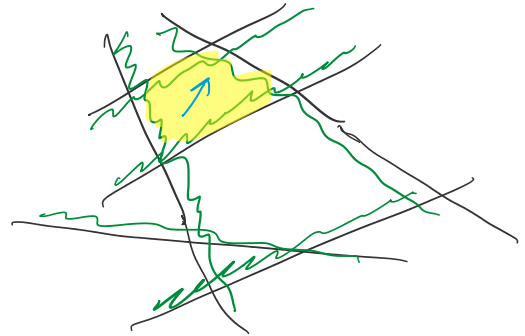
Geometric view:

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

Maximize

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

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

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

Rewrite as LP:

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

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

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

Minimum-cost Flow

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

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

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

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

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

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

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

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

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

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

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

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.
constraints objective

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

$$\text{s.t. } \sum_u x_{uv} \leq 1$$

$$\sum_v x_{uv} \leq 1$$

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.

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

$$\text{subject to } A\vec{x} \leq \vec{b}$$

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

(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 s 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.

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

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

$$x_u \in \{0,1\} \quad \forall u \in V$$

← 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)