Today

- randomized rounding of LPs
  - MAX 2SAT
  - congestion min

project info tonight.

$\boxed{\text{Optimization}}$

min/max $f(x_1,\dots,x_n)$
subject to constraints

When can we solve efficiently?

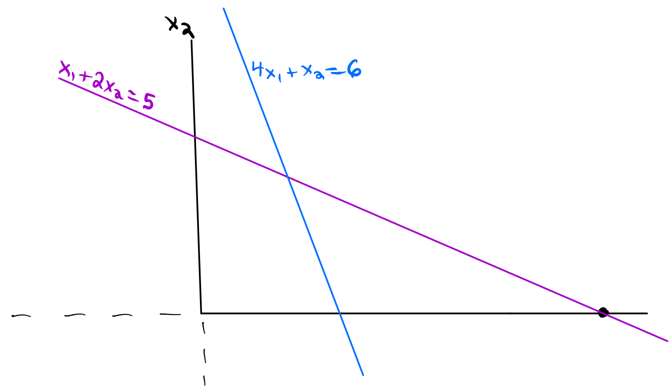
$\boxed{\text{Linear programming}}$ L

$f(x_1,\dots,x_n)$ linear fn.
constraints linear

Example: Diet Problem

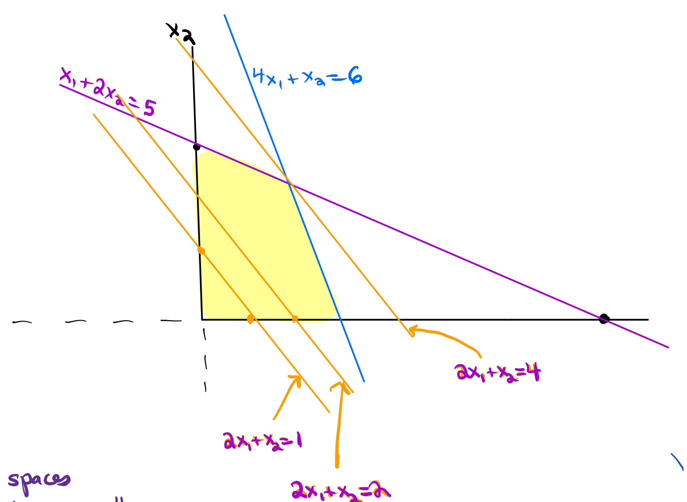
Athlete wants to max protein consumption
subject to ≤ 5 units of fat/day
$\leq \$6$ / day

	protein/lb	fat/lb	$/lb
steak	2	1	4
peanut butter (PB)	1	2	1



x_1 #lbs of steak/day
x_2 #lbs of PB/day

max $\boxed{2x_1+x_2}$ ← objective fn
subject to
$$\boxed{\begin{aligned}4x_1+x_2&\leq 6\\ x_1+2x_2&\leq 5\\ x_1,x_2&\geq 0\end{aligned}}$$
← feasible set
feasible region



feasible pt with max objective fn value is "optimal solution"

feasible set is "polyhedron": intersection of half spaces
if also bounded & nonempty ⇒ "polytope"

feasible set convex

linear cost fns define family of parallel hyperplanes
optimal feasible pt must occur at corner, a.k.a. vertex
[can't be expressed as convex comb of feasible pts]

Unfortunately too many vertices to enumerate $\approx m^{n/2}$
m constraints, n variables

Ex: $0\leq x_i\leq 1$ $1\leq i\leq n$
n dimensional hypercube

max $\boxed{2x_1 + x_2}$
subject to
$$\boxed{\begin{array}{l} 4x_1 + x_2 \le 6 \\ x_1 + 2x_2 \le 5 \\ x_1, x_2 \ge 0 \end{array}}$$

$c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

$A = \begin{pmatrix} 4 & 1 \\ 1 & 2 \end{pmatrix}$

$b = \begin{pmatrix} 6 \\ 5 \end{pmatrix}$

Input to LP problem

$$\boxed{\begin{array}{l} \vec{c} \in \mathbb{R}^n \\ A \in \mathbb{R}^{m \times n} \\ \vec{b} \in \mathbb{R}^m \end{array}}$$

find $x \in \mathbb{R}^n$ to
max $c \cdot x$
subject to $Ax \le b$
$x_i \ge 0 \ \forall i$

$\equiv$ max $c_1x_1 + c_2x_2 + \dots + c_nx_n$

s.t. $a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \le b_1$

$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \le b_2$

$\vdots$

$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \le b_m$

$x_1, x_2, \dots, x_n \ge 0$

$a \cdot x \ge b \quad \equiv \quad -a \cdot x \le -b$

$a \cdot x = b \quad \equiv \quad \begin{array}{l} a \cdot x \ge b \\ a \cdot x \le b \end{array}$

x unconstrained $\quad x = x^+ - x^-$, $x^+, x^- \ge 0$

There are efficient algs for LP.

- simplex — exp worst case, "smoothed" poly time
- ellipsoid
- interior pt methods

(ellipsoid and interior pt methods: polynomial time)

$\boxed{\text{ellipsoid}}$: sometimes even exponential-sized LPs can be solved in poly time

$\boxed{\text{Bottom line:}}$ If you can formulate your problem as LP, it can be solved in time $\text{poly}(m, n, \log(\max_{i,j}(a_{ij}, b_i, c_j)))$

$\boxed{\text{Using "linear-programming relaxations" to get approx algs for NP-hard problems \& then apply randomized rounding}}$

MAXSAT

Given a Boolean formula in CNF (and of ors)
- n Boolean vars $x_1, \dots, x_n$
- m clauses $C_1, \dots, C_m$
  - each clause is OR of some # of $\underbrace{\text{vars \& negations}}_{\text{literals}}$ e.g. $x_3 \vee \bar{x}_5 \vee x_{11}$
- w_j weight of j^{th} clause

max-version even when all clauses have ≤ 2 literals

MAXSAT: find assignment that maximizes wt of satisfied clauses

MAX kSAT: all clauses have $\le k$ literals

MAX EkSAT: all clauses have exactly k literals

Random assignment

k	Pr(clause sat)
1	
2	
3	
4	

We will consider **MAX 2-SAT** all clauses have 1 or 2 literals

$\boxed{\text{Step 1:}}$ Formulate MAX-2SAT as integer linear program where we require all vars to be $\in \{0,1\}$

Suppose input formula has n vars, m clauses

Define vars $x_1, \dots, x_n$ $\forall$ var in formula

$z_1, \dots, z_m$ $\forall$ clause in formula

where $x_i = \begin{cases} 1 & \text{var } i \text{ T} \\ 0 & \text{o.w.} \end{cases}$ $z_j = \begin{cases} 1 & \text{if clause } j \text{ satisfied} \\ 0 & \text{o.w.} \end{cases}$

Ex

$C_j = (\overline{x_1} \vee x_2)$

$(1-x_1) + x_2 \geq z_j$

$C_j = (x_5)$

$x_5 \geq z_j$

integer linear programming is NP-hard though!

Step 2: Relax to a LP and solve $\Rightarrow$ $\underbrace{z_1^*, \ldots, z_m^*, x_1^*, \ldots, x_n^*}_{\mathrm{OPT}_g\,(2)}$ fractional

$$\max \sum_{j=1}^{m} w_j z_j$$

$$\sum_{i \in C_j} y_{ij} \geq z_j \qquad j \in 1, \ldots, m$$

where $y_{ij} = \begin{cases} x_i & \text{if var } i \text{ pos in } C_j \\ 1 - x_i & \text{'' neg in } C_j \end{cases}$

$$x_i \in \{0,1\} \qquad i = 1 \ldots n$$

$$z_j \in \{0,1\} \qquad j = 1 \ldots m$$

Key observation:

Step 3: Round the soln $\Rightarrow$ integer soln

Corollary:

- Similar analysis shows randomized rounding gives $1-\frac{1}{e}$ approx alg for MAXSAT
- Choosing better of
  - randomized rounding soln
  - unbiased randomized alg

  $\Rightarrow \frac{3}{4}$ approx alg
- Solving LP & then setting x_i: True w.p. $f(x_i^*)$, False o.w., where $f:[0,1]\to[0,1]$, $1-4^{-x} \le f(x) \le 4^{x-1}$

  gives $\frac{3}{4}$ approx alg.

Best possible if we compare our solution against OPT_{LP}

Ex: $(x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$

Randomized rounding summary

important technique for obtaining approx algs for NP-hard problems

Recipe

1. Set problem up as integer linear program. vars $\in \{0,1\}$
2. Relax ILP $\rightarrow$ LP vars $\in [0,1]$
3. Solve LP to optimality $\rightarrow$ x_i^* $i=1,..,n$
4. Construct solution by randomly rounding vars $\rightarrow \{0,1\}$
   treat x_i^* as probability
5. bound quality of soln by comparing to LP OPT

Another example: **Congestion minimization**

Input: directed graph $G=(V,E)$
set of pairs (s_i,t_i) $i=1..k$
Output: path P_i from $s_i \rightsquigarrow t_i$ $\forall i=1..k$
s.t. $\underbrace{\text{congestion}}_{\text{max \# paths that intersect any edge}}$ is minimized

NP-hard

Approx alg via randomized rounding

1. Set up ILP [multi commodity flow]
   vars $f_i(e) \in \{0,1\}$ flow from $s_i \rightsquigarrow t_i$ on edge e
   C congestion

$$\min \ C$$

subject to

$$\sum_{\substack{e \text{ s.t.} \\ e=(u\to v) \\ \text{for some } u}} f_i(e) = \sum_{\substack{e \text{ s.t.} \\ e=(v\to w) \\ \text{for some } w}} f_i(e) \qquad \begin{matrix}\forall v \neq s_i,t_i \\ \forall i\end{matrix} \qquad \text{conservation of flow}$$

$$\sum_{\substack{e \text{ s.t.} \\ e=(s_i\to u) \\ \text{for some } u}} f_i(e) = 1 \qquad \forall i \qquad \text{route 1 unit of flow from } s_i \to t_i$$

$$\sum_i f_i(e) \le C \qquad \text{congestion bound}$$

$$f_i(e) \in \{0,1\} \qquad \forall i,e$$

2. Relax to LP

3. Solve LP

4 Round soln

5 Bound result.

<u>Theorem</u> If $C^* \geq c \ln n$ for some const c, then whp the max congestion on any edge $\leq C^* + \sqrt{c C^* \ln n}$

$\Rightarrow$ approx ratio ≤ 2

<u>Proof</u> Fix $e = (u \to v)$

Let $X_i(e) = \begin{cases} 1 & \text{if } e \in P_i \\ 0 & \text{o.w.} \end{cases}$

$$E(X_i(e)) = \sum_{\substack{P \in \mathcal{P}_i \text{ s.t.} \\ e \in P}} f_P^i = f_i^*(e)$$

If only know $C^* \geq 1$

Theorem $\Pr\left(\text{any edge has congestion} \geq \boxed{\frac{6 \ln n}{\ln \ln n}}\, C^*\right) \leq \frac{1}{n}$

$\uparrow \alpha$

gives approx ratio of α

Proof Fix $e = (u \to v)$

Let $X_i(e) = \begin{cases} 1 & \text{if } e \in P_i \\ 0 & \text{o.w.} \end{cases}$

$$E(X_i(e)) = \sum_{\substack{P \in \mathcal{P}_i \text{ s.t.} \\ e \in P}} f_P^i = f_i^*(e)$$

Let $X(e) = \sum_{i=1}^{k} X_i(e) \implies \forall e \quad E(X(e)) = \sum_{i=1}^{k} f_i^*(e) \leq C^*$

$$\Pr\left(X(e) \geq \underbrace{(1+\delta)}_{\alpha} C^*\right) \leq e^{-C^*[(1+\delta)\ln(1+\delta) - \delta]} \qquad \text{[Chernoff bound]}$$

$$\Pr\left(X(e) \geq \alpha C^*\right) \leq e^{-C^*[\alpha \ln \alpha + 1 - \alpha]} \underset{C^* \geq 1}{\leq} e^{-\alpha(\ln \alpha - 1)}$$

$\alpha(\ln \alpha - 1)$

$= \frac{6 \ln n}{\ln \ln n}\left[\ln \ln n + \ln 6 - \ln \ln \ln n - 1\right]$

$\geq 3 \ln n$

$$\leq e^{-3 \ln n} = \frac{1}{n^3}$$

$$\implies \Pr\left(\text{any edge has congestion} \geq \alpha C^*\right) \underset{\text{union bound}}{\leq} \sum_e \Pr(X(e) \geq \alpha C^*)$$

$$\leq n^2 \cdot \frac{1}{n^3} = \frac{1}{n}$$

Optimal!

$\exists$ graphs with $\frac{OPT}{C^*} = \Omega\left(\frac{\ln n}{\ln \ln n}\right)$ integrality gap

so can't do better with such an approach

Hardness (directed graphs)

Every poly time alg has $\Omega\left(\frac{\log n}{\log \log n}\right)$ approx ratio

unless $NP \subseteq BPTIME\left(n^{O(\log \log n)}\right)$

MAXCUT

Recall assign each vertex at random to side

Input: $G=(V,E)$ $\quad w_{ij} \quad \forall (i,j)\in E$

Goal: partition vertex set so as to max weight of endpts crossing cut.

IP formulation of MAXCUT

$$x_i = \begin{cases} 0 & \text{on one side of partition} \\ 1 & \text{on other side} \end{cases}$$

$$z_{ij} = \begin{cases} 1 & \text{edge } (i,j) \text{ cut} \\ 0 & \text{o.w.} \end{cases}$$

$$\max \sum_{(i,j)\in E} w_{ij} z_{ij}$$

$$z_{ij} \le x_i + x_j \qquad \forall (i,j)\in E$$

$$z_{ij} \le 2-(x_i+x_j) \qquad \forall (i,j)\in E$$

$$x_i \in \{0,1\} \qquad i \in V$$

$$z_{ij} \in \{0,1\} \qquad \forall (i,j)\in E$$

Another approach

First, notation change

$\forall i$. $x_i \in \{-1, +1\}$

define $y_{ij} = x_i x_j \quad \forall i,j \in V$

$\boxed{\begin{array}{ll} \text{Want } \exists\, x_i \ \forall i \in V & \\ \text{s.t. } \quad y_{ij} = x_i x_j & \forall i,j \end{array}}$

$$\boxed{\begin{array}{lll} \max & \sum_{(i,j)\in E} w_{ij} \frac{1}{2}(1 - y_{ij}) & \\ & y_{ij} = y_{ji} & \forall i,j \in V \\ & y_{ii} = 1 & \forall i \in V \end{array}}$$

purple + brown is exact.

SDP rounding

Intro to semidefinite programming

linear programming where vars are entries in a PSD matrix

Defn If A is a symmetric n by n matrix
then A is a positive semidefinite (psd) matrix $\equiv A \succeq 0$
if any of the following equivalent conditions hold

1. $\forall \vec{c} \in \mathbb{R}^n$, $\quad c^T A c \geq 0$
2. A has nonnegative eigenvalues
3. $A = V^T V$ for some $m \times n$ matrix V, $m \leq n$
4. $A = \sum_{i=1}^{n} \lambda_i x_i x_i^T$ for some $\lambda_i \geq 0$ and orthonormal vectors $x_i \in \mathbb{R}^n$

Semidefinite program (SDP)

max or min $\sum_{i,j} c_{ij} x_{ij}$

subject to $\sum_{i,j} a_{ijk} x_{ij} = b_k$

$x_{ij} = x_{ji} \quad \forall i,j$

$X = (x_{ij}) \succeq 0$

$\equiv$ Vector program

max or min $\sum_{i,j} c_{ij} (v_i \cdot v_j)$

subject to $\sum_{i,j} a_{ijk} (v_i \cdot v_j) = b_k$

$v_i \in \mathbb{R}^n \quad i = 1, \ldots, n$

given $X \Rightarrow X = V^T V$;
set v_i to be i^{th} col of V

Key fact:

SDPs can be solved to within additive error ε
in time $\quad \text{poly}(\text{size of input}, \log(\frac{1}{\varepsilon}))$

in our discussions, we ignore additive error ε