Lecture 19: Semi-definite programming, and the Goemans-Williamson algorithm.

Recall: linear programming

Variables : $x \in \mathbb{R}^n$

constraints : $a_1, \dots, a_m \in \mathbb{R}^n$
$b_1, \dots, b_m \in \mathbb{R}$
$A = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}$ $b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$

Objective : $c \in \mathbb{R}^n$.

$$\min \quad c^T x \quad \text{s.t.} \quad \langle a_i, x \rangle \geq b_i \quad \forall i = 1, \dots, m$$

$\hookrightarrow \; Ax \geq b$

LPs are convex, and can be solved efficiently.

What about more powerful classes of algorithms?

Semi definite programming.

Variables: A symmetric matrix $X \in \mathbb{R}^{n \times n}$

Constraints : $A_1, \dots, A_m \in \mathbb{R}^{n \times n}$
$b_1, \dots, b_m$

Objective : $C \in \mathbb{R}^{n \times n}$

$\langle A, B \rangle = \sum_{ij} A_{ij} B_{ij}$

$$\min \quad \langle C, X \rangle$$

$$\text{s.t.} \quad \langle A_i, X \rangle \geq b_i \quad \forall i$$

$$X \succeq 0$$

this is what makes it SDP!

this means X is positive semidefinite, i.e.

all of its eigenvalues ≥ 0.

Fact: This is still convex, so can be solved efficiently.

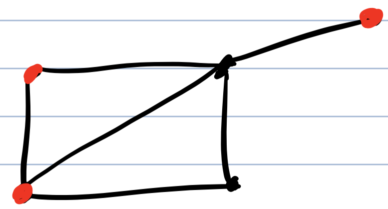
Why is this useful??

- generalizes LPs.
- allows you to encode geometric relationships efficiently via convex relaxation

example: Max-Cut.

given a graph $G = (V, E)$, find a partition of vertices into 2 parts s.t. as many edges are cut as possible.

e.g.



NP-hard to find best partition!

But maybe we can find a good approximation?

Def: we say a partition (S, T) is a c-approx solution if

$$\# \text{ edges cut by } S, T \geq c \cdot OPT,$$

$$OPT = \text{value of optimal cut.}$$

Fact: $c = 1/2$ is "trivial"

But can we do better?

Step 1: Write Max-Cut as an integer program:

Let $V = \{1, \dots, n\}$, and let $x_i \in \{\pm 1\}$, $i = 1, \dots, n$.

Then

$$\text{Max-Cut} = \max_{x_1, \dots, x_n} \sum_{(i,j) \in E} \frac{1 - x_i x_j}{2}$$

0 if $x_i = x_j$

-1 if $x_i \neq x_j$

Step 2: Relax the integer constraint on x_i, so that we can solve it efficiently.

How to relax?

It turns out the following is a good idea:

$x_i \in \{\pm 1\}$ just means $x_i \in \mathbb{R}$ and $|x_i| = 1$.

relax this!

$x_i \in \mathbb{R}^n$, and $\|x_i\|_2 = 1$.

$x_i x_j \longrightarrow \langle x_i, x_j \rangle$

So new problem is

$$\max_{\substack{x_1, \dots, x_n \in \mathbb{R}^n \\ \|x_i\|_2 = 1}} \sum_{(i,j) \in E} \frac{1 - \langle x_i, x_j \rangle}{2}.$$

Claim: This is an SDP!

How?

Write $X = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$

Then $X^T X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$

$$= \begin{bmatrix} \langle x_1, x_1 \rangle & \langle x_1, x_2 \rangle & \cdots & \langle x_1, x_n \rangle \\ \langle x_1, x_2 \rangle & \langle x_2, x_2 \rangle & \cdots & \langle x_2, x_n \rangle \\ \vdots & & \ddots & \vdots \\ \langle x_1, x_n \rangle & & \cdots & \langle x_n, x_n \rangle \end{bmatrix}$$

i^{th} diagonal entry is $\langle x_i, x_i \rangle = \|x_i\|_2^2 = 1.$

$(i,j)^{th}$ entry, $i \neq j$ $= \langle x_i, x_j \rangle$

Fact: $X^T X$ is PSD.

proof: Recall a matrix M is PSD $\Leftrightarrow$ $v^T M v \geq 0$ $\forall v \in \mathbb{R}^n$.

$$v^T (X^T X) v = (Xv)^T \cdot Xv = \|Xv\|_2^2 \geq 0.$$

Fact: Any matrix M s.t. $M \succeq 0$, and $M_{ii} = 1$ $\forall i$, can be written as $X^T X$ for some X.

proof: By spectral theorem:

$$M = \begin{bmatrix} & & \\ & U & \\ & & \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \begin{bmatrix} & & \\ & U^T & \\ & & \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} & & \\ & U & \\ & & \end{bmatrix} \begin{bmatrix} \lambda_1^{1/2} & & \\ & \ddots & \\ & & \lambda_n^{1/2} \end{bmatrix}}_{X^T} \underbrace{\begin{bmatrix} \lambda_1^{1/2} & & \\ & \ddots & \\ & & \lambda_n^{1/2} \end{bmatrix} \begin{bmatrix} & & \\ & U^T & \\ & & \end{bmatrix}}_{X}$$

$$= X^T X.$$

So:

linear objective in M.

$$\max_{\substack{x_1, \dots, x_n \in \mathbb{R}^n \\ \|x_i\|_2^2 = 1}} \sum_{(i,j) \in E} \frac{1 - \langle x_i, x_j \rangle}{2} \iff \max_{\substack{M \succeq 0 \\ M_{ii} = 1 \ \forall i}} \frac{|E|}{2} - \frac{\langle C, M \rangle}{2}$$

$$C_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{o.w} \end{cases}$$

linear constraint on M

The SDP captures "covariance" information about the x_i variables.

Ok, so we can solve this relaxed problem efficiently, but how do we use this to solve MAX-CUT?

Step 3: Rounding.

Find some way of converting SDP sol'n back to $\{+1, -1\}$.

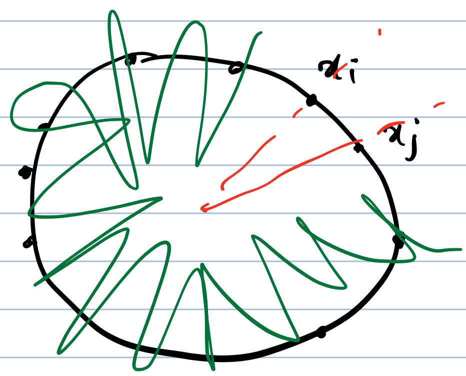
Observe: $x_i = \begin{pmatrix} \pm 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ is valid, so the value of the SDP soln is $\geq$ value of MAX-CUT solution.

So all we need to do is round in a way that doesn't decrease value of SDP solution too much.

Key idea: Gaussian rounding: sample $g \sim \mathcal{N}(0, I)$, and for every x_i, let

$$y_i = \text{round}(x_i) = \text{sign}(\langle g, x_i \rangle).$$

Intuition: if x_i, x_j are close, you should round them to same value more often

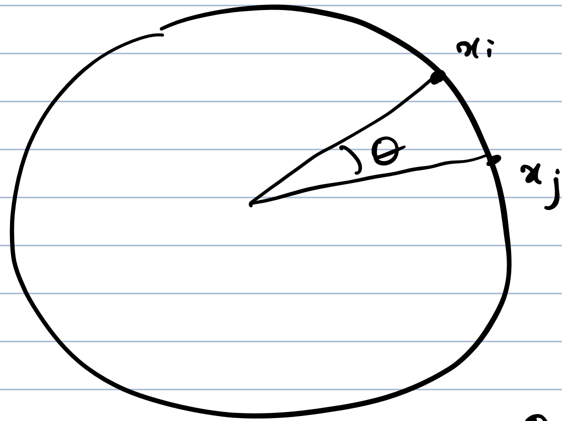


Claim: $\mathbb{E}\left[\frac{1 - y_i y_j}{2}\right] = \cos^{-1}(\langle x_i, x_j \rangle)$.

"proof": think about the 2D case, $x_i, x_j \in \mathbb{R}^2$.

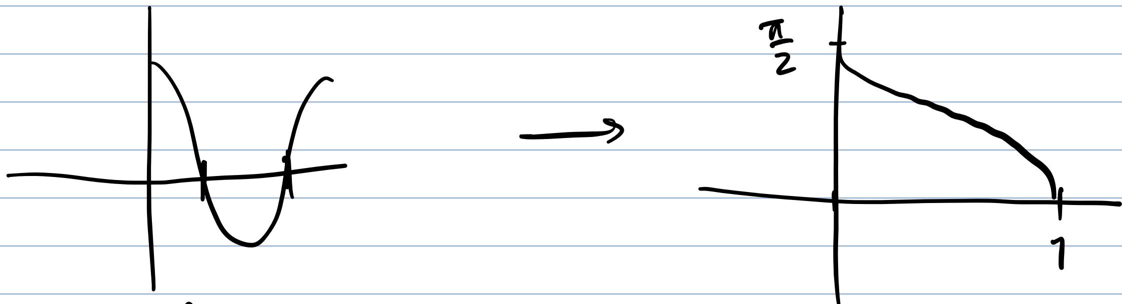
We only care about the angle of g, not the magnitude.

But the angle of g is uniformly random!

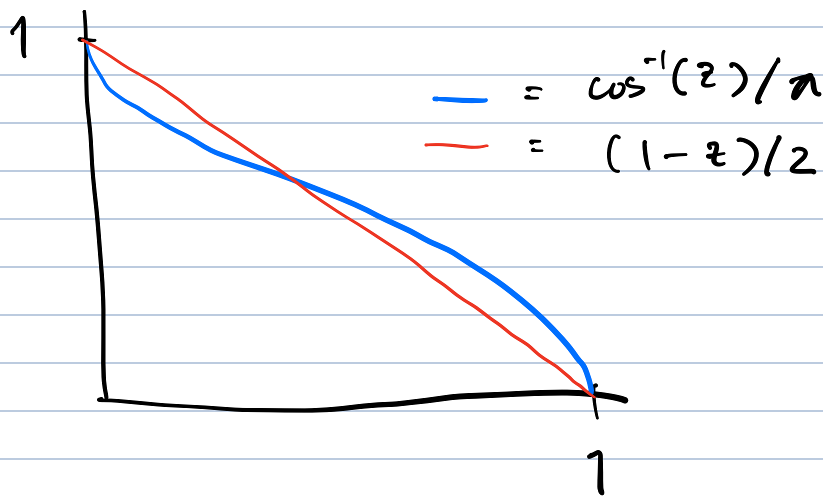


law of cosines.

$$\Pr[\, y_i \text{ and } y_j \text{ are cut} \,] = \frac{\theta}{\pi} = \frac{\cos^{-1}(\langle x_i, x_j \rangle)}{\pi}$$



$\cos(x)$ $\qquad\qquad$ $\cos^{-1}(x)$



Crucially: $\dfrac{\cos^{-1}(z)/\pi}{(1-z)/2} \geq 0.878\ldots$

$\forall z \in [0,1]$.

So :

$$\mathbb{E}\left[\sum_{(i,j)\in E} \frac{1-y_iy_j}{2}\right] = \sum_{(i,j)\in E} \mathbb{E}\left[\frac{1-y_iy_j}{2}\right]$$

$$= \sum_{(i,j)\in E} \frac{\cos^{-1}(\langle x_i, x_j\rangle)}{\pi}$$

$$\geq 0.878\ldots \sum_{(i,j)\in E} \frac{1-\langle x_i, x_j\rangle}{2}$$

$$= 0.878 \cdot \text{val of SDP sol}^{\text{n}}$$

$$\geq 0.878\ldots \cdot \text{OPT}$$

So we have shown:

Thm [Goemans-Williamson '94]: One can achieve a 0.878.... -approximation to MAX-CUT in polytime, using semidefinite programming

Is this optimal? Maybe!

Under a believable conjecture (Unique Games Conjecture) no polytime algo can do $0.878\ldots + \varepsilon$.