Lecture 10

Semidefinite Programs and the Max-Cut Problem*

In this class we will finally introduce the content from the second half of the course title, Semidefinite Programs. We will first motivate the discussion of SDP with the Max-Cut problem, then we will formally define SDPs and introduce ways to solve them.

10.1 Max Cut

Let $G = (V, E)$ with edge weights $w_e > 0$ for all $e \in E$ where $\sum_{e \in E} w_e = 1$. Our goal is to maximize $\sum_{e \in \partial(A)} w_e$ over $A \subseteq V$, or equivalently maximize $\sum_{uv \in E} w_{uv} \mathbb{1}[A(u) \neq A(v)]$ over functions $A : V \to \{0, 1\}$.

Remark 10.1. Couple quick observations:

- $Opt = 1 \Leftrightarrow G$ is bipartite
- $Opt \geq \frac{1}{2}$

  Proof. (Algorithmic) Pick $A : V \to \{0, 1\}$ at random.

$$\begin{aligned}
\mathbf{E}\left[\text{cut val}\right] &= \mathbf{E}\left[\sum_{uv \in E} w_{uv} \mathbb{1}\left[A(u) \neq A(v)\right]\right] \\
&= \sum_{uv \in E} w_{uv} \mathbf{Pr}[A(u) \neq A(v)] \\
&= \sum_{uv \in E} w_{uv} \frac{1}{2} \\
&= \frac{1}{2}
\end{aligned}$$

□

*Lecturer: Ryan O'Donnell. Scribe: Franklin Ta.

- G complete $\Rightarrow Opt = \frac{1}{2} + \frac{1}{2(n-1)}$
- Max-cut is NP-hard.

Integer Programming Formulation Now lets look at the IP formulation for Max-Cut. For $\{x_v\}_{v\in V}, \{z_e\}_{e\in E} \in \{0,1\}$,

$$\begin{aligned} \max \quad & \sum_{uv\in E} w_{uv} z_{uv} \\ \text{s.t.} \quad & z_{uv} \leq x_u + x_v \\ & z_{uv} \leq 2 - (x_u + x_v) \end{aligned}$$

Where x encodes which partition the vertex is in, and z encodes whether the edge is cut. To see why this works, suppose that $x_u \neq x_v$, then $z_{uv} \leq 1, z_{uv} \leq 1$, so $z_{uv} = 1$. Otherwise, suppose $x_u = x_v$, then $z_{uv} \leq 2, z_{uv} \leq 0$, so $z_{uv} = 0$.

Linear Programming Relaxation To get the LP relaxation, just let x_v, $z_e \in [0,1]$. But unfortunately, this LP relaxation isn't very good. Set $x_v = \frac{1}{2}$ for all v, then $z_e = 1$ for all e, which makes $LPOpt = 1$ for all graph G.

Another Formulation Seeing that didn't work, let's try another formulation. For $y_v \in \mathbb{R}$,

$$\begin{aligned} \max & \sum_{uv\in E} w_{uv} \left(\frac{1}{2} - \frac{1}{2} y_u y_v\right) \\ \text{s.t.} \quad & y_v y_v = 1 \quad \forall v \in V \end{aligned}$$

Here we changed our indicator, y_v, to use 1 or -1 to encode which partition we are in. Note that if $y_u = y_v$, then $\left(\frac{1}{2} - \frac{1}{2} y_u y_v\right) = 0$, and if $y_u \neq y_v$, then $\left(\frac{1}{2} - \frac{1}{2} y_u y_v\right) = 1$.

SDP Relaxation[Delorme Poljak '90] Note that the previous formulation is still exactly Max-Cut (which is NP-hard) so we won't be able to solve it. So to relax it, we can allow $\vec{y}_v \in \mathbb{R}^d$: (this is the 'SDP')

$$\begin{aligned} \max & \sum_{uv\in E} w_{uv} \left(\frac{1}{2} - \frac{1}{2} \vec{y}_u \cdot \vec{y}_v\right) \\ \text{s.t.} \quad & \vec{y}_v \cdot \vec{y}_v = 1 \quad \forall v \in V \end{aligned}$$

To visualize what this SDP is doing, note that since $||\vec{y}_v|| = 1$, it is possible to embed $\{y_v\}_{v\in V}$ onto a unit sphere:

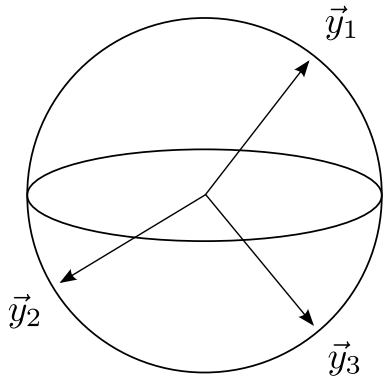

Figure 10.1: Vectors $\vec{y}_v$ embedded onto a unit sphere in $\mathbb{R}^d$.

Denote $\sigma_{uv} = \vec{y}_u \cdot \vec{y}_v = \cos(\angle(\vec{y}_u, \vec{y}_v))$. Then for $(u, v) \in E$, we want $\sigma_{uv} \approx -1$ as possible since this will maximize the sum. Translating to the figure above, we want to have vectors pointing away from each other as much as possible. Also note that if d is 1, the solution is exact, so $SDPOpt \geq Opt$. Also note that $d \leq n$ in general.

Example 10.2. Let's see some examples of how this SDP compares with Opt.

Let G be K_3. Clearly we can embed $\vec{y}_1, \vec{y}_2, \vec{y}_3$ 120 degrees apart in the unit circle in $\mathbb{R}^2$, so we get:

$$\begin{aligned} SDPOpt(G) &\geq \sum_{uv \in E} w_{uv} \left(\frac{1}{2} - \frac{1}{2} \vec{y}_u \vec{y}_v \right) \\ &= \sum_{i=1}^{3} \frac{1}{3} \left(\frac{1}{2} - \frac{1}{2} \cos\left(\frac{2\pi}{3} \right) \right) \\ &= \frac{3}{4} \end{aligned}$$

This bound can be shown to be tight, so $SDPOpt(G) = \frac{3}{4}$. It can also be shown that $Opt(G) = \frac{2}{3}$ and $LPOpt(G) = 1$. Thus $\frac{8}{9} SDPOpt(G) = Opt(G)$.

Another example is G being C_5. In this case we have $Opt(G) = .88\, SDPOpt(G)$

Remark 10.3. Ellipsoid algorithm can (weakly) solve SDP in polytime. So assume you can find optimal $\vec{y}_v$.

Randomized Rounding [Goemans Williamson '94] At this point, we know we can relax Max-Cut into a SDP, and we know we can solve SDPs, but we still need to somehow convert that back to a solution for Max-Cut. We can do this using Goemans-Williamson randomized rounding:

We want to cut the vectors $\{y_v\}_{v \in V}$ with a random hyperplane through zero where everything on one side of the hyperplane is in one partition, and everything on the other side

of the hyperplane is in the other. We do this by choosing $\vec{g} \in \mathbb{R}^d \setminus \{0\}$ (where $\vec{g}$ is normal to hyperplane) from any rotationally symmetric distribution. Then set $y_u = \text{sgn}(\vec{g} \cdot \vec{y}_u) \in \{-1, 1\}$.

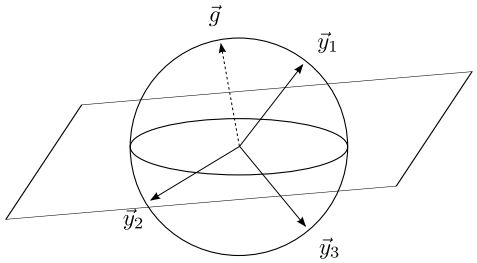

Figure 10.2: Vectors being separated by a hyperplane with normal $\vec{g}$.

Analysis of the rounding Fix $(u, v) \in E$. Then the probability that an edge (u, v) gets cut is the same as the probability that the hyperplane splits $\vec{y}_u$ and $\vec{y}_v$.

So consider just the 2-D plane containing $\vec{y}_u, \vec{y}_v$. Since the hyperplane was chosen from a rotationally symmetric distribution, the probability that the hyperplane cuts these two vectors is the same as the probability that a random diameter lies in between the angle θ of the two vectors.

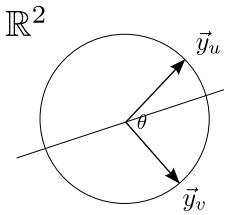

Figure 10.3: The plane of two vectors being cut by the hyperplane

Thus:

$$\begin{aligned}
\mathbf{Pr}[(u, v) \text{ gets cut}] &= \frac{\theta}{\pi} \\
&= \frac{\cos^{-1}(\vec{y}_u \cdot \vec{y}_v)}{\pi} \\
&= \frac{\cos^{-1}(\sigma_{uv})}{\pi} \\
\mathbf{E}[\text{cut val}] &= \sum_{uv \in E} w_{uv} \frac{\cos^{-1}(\sigma_{uv})}{\pi}
\end{aligned}$$

Now recall that $SDPOpt = \sum_{uv \in E} w_{uv}(\frac{1}{2} - \frac{1}{2}\sigma_{uv}) \geq Opt$.

So if we find α such that

$$\frac{\cos^{-1}(\sigma_{uv})}{\pi} \geq \alpha\left(\frac{1}{2} - \frac{1}{2}\sigma_{uv}\right) \quad \forall \sigma_{uv} \in [-1, 1]$$

then we can conclude $\mathbf{E}[\text{cut val}] \geq \alpha\, SDPOpt \geq \alpha Opt$

Plotting the above,

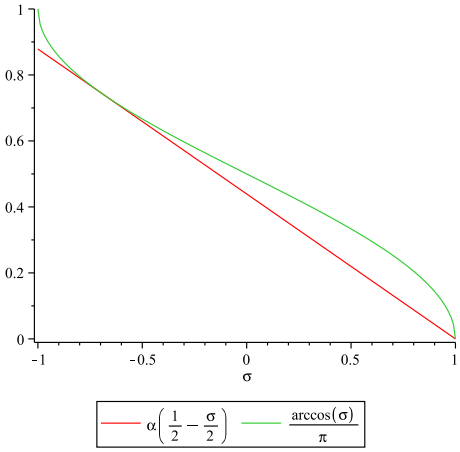

Figure 10.4: $\alpha\left(\frac{1}{2} - \frac{1}{2}\sigma\right)$ vs $\frac{\cos^{-1}(\sigma_{uv})}{\pi}$

we see that $\alpha = .87856...$ works.

Theorem 10.4. $\mathbf{E}[\textit{Goemans Williamson cut}] \geq .87856\, SDPOpt \geq .87856 Opt$

Note that this gives a polytime ".878-factor approximation algorithm for Max-Cut"

10.2 Semidefinite Programming

10.2.1 Positive Semidefinite Matrices

Theorem 10.5. *$S \in \mathbb{R}^{n \times n}$ symmetric is positive semidefinite (P.S.D.) iff equivalently*

1. *$S = Y^\top Y$ for some $Y \in \mathbb{R}^{d \times n}, d \leq n$.*
2. *S's eigenvalues are all greater than or equal to 0.*
3. *$x^\top S x = \sum_{i,j} x_i x_j s_{ij} \geq 0$ for all $x \in \mathbb{R}^n$.*
4. *$S = LDL^\top$ where D diagonal, $D \geq 0$, L unit lower-triangular (i.e., $L = \begin{pmatrix} 1 & & & & 0 \\ * & 1 & & & \\ * & * & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ * & * & \dots & * & 1 \end{pmatrix}$).*

5. *There exist joint real random variables* $Z_1, ..., Z_n$ *such that* $\mathbf{E}[Z_i Z_j] = s_{ij}$

6. $(S \in \mathbb{R}^{n \times n}) \in$ *convex-hull of* $\{xx^\top : x \in \mathbb{R}^n\}$

Remark 10.6. 3 and 6 from Theorem 10.5 implies $\{S \in \mathbb{R}^{n^2} : S \text{ PSD}\}$ is convex

Remark 10.7. Recall the SDP of Max-Cut is:

$$\begin{aligned} \max \sum_{uv \in E} w_{uv} \left(\frac{1}{2} - \frac{1}{2} \sigma_{uv} \right) \\ \text{s.t.} \quad \sigma_{uv} = \vec{y}_u \cdot \vec{y}_v \\ \sigma_{vv} = 1 \quad \forall v \in V \end{aligned}$$

Thus,

$$\begin{aligned} \exists Y = \begin{pmatrix} | & | & & | \\ \vec{y}_{v_1} & \vec{y}_{v_2} & \cdots & \vec{y}_{v_n} \\ | & | & & | \end{pmatrix} \\ \text{s.t.} \quad (\sigma_{uv})_{u,v \in V} = YY^\top \end{aligned}$$

$\Leftrightarrow$ matrix $S = (\sigma_{uv})$ is PSD by Theorem 10.5.1.

Definition 10.8. A *semidefinite program* is an LP over n^2 variables σ_{ij} with extra constraints $S := (\sigma_{ij})$ is PSD. (This is really $\frac{n(n+1)}{2}$ variables since it is symmetric)

Theorem 10.9. *Omitting technical conditions SDP can be solved in polytime.*

10.2.2 Strong Separation Oracle for PSDness

Given symmetric matrix $S \in \mathbb{Q}^{n \times n}$, we want to assert S is PSD or find $x \in \mathbb{Q}^n$ s.t. $x^\top S x < 0$.

Idea: Compute smallest eigenvalue of S. If it is greater than or equal to 0, then S is PSD, otherwise we can use the corresponding eigenvector z to show that $z^\top S z < 0$.

Theorem 10.10. $\exists$ *Strongly polytime algorithm such that if* S *PSD, returns* $S = LDL^\top$, *and if* S *not PSD, returns* x *s.t.* $x^\top S x < 0$. $(L, D, x \in \mathbb{Q})$

Bonus: Since can compute $\sqrt{D}$ to any accuracy (square root each term in D) and $S = Y^\top Y$, we compute $Y = \sqrt{D} L^\top$. Where columns of Y are "vectors".

Proof. Do (symmetric) Gaussian Elimination on S:

$$S = \begin{pmatrix} * & * & \dots & * \\ * & * & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \dots & * \end{pmatrix}$$

Clear out first column with L_1 (which adds multiples of first row to other rows).

$$\underbrace{\begin{pmatrix} 1 & & & 0 \\ * & 1 & & \\ \vdots & & \ddots & \\ * & & & 1 \end{pmatrix}}_{L_1} S = \begin{pmatrix} * & * & \dots & * \\ 0 & * & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \dots & * \end{pmatrix}$$

Since symmetric, clear out first row with $L_1^\top$.

$$L_1 S L_1^\top = \begin{pmatrix} * & 0 & \dots & 0 \\ 0 & * & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \dots & * \end{pmatrix}$$

Then clear out second row and column with L_2 and $L_2^\top$, and so on.

$$L_2 L_1 S L_1^\top L_2^\top = \begin{pmatrix} * & 0 & 0 & \dots & 0 \\ 0 & * & 0 & \dots & 0 \\ 0 & 0 & * & \dots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & * & \dots & * \end{pmatrix}$$

$$\vdots$$

$$\underbrace{L_n \dots L_2 L_1}_{L} S \underbrace{L_1^\top L_2^\top \dots L_n^\top}_{L^\top} = D$$

$$L S L^\top = D$$

$$S = L^{-1} D L^{-\top}$$

This will run to completion unless we hit the following cases:

If you just finished pivoting with a negative number, stop and output not PSD:

$$LSL^\top = \begin{pmatrix} * & 0 & 0 & \dots & 0 \\ 0 & -a & 0 & \dots & 0 \\ 0 & 0 & * & \dots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & * & \dots & * \end{pmatrix}$$

$$\begin{aligned} e_i^\top LSL^\top e_i &= e_i^\top \begin{pmatrix} * & 0 & 0 & \dots & 0 \\ 0 & -a & 0 & \dots & 0 \\ 0 & 0 & * & \dots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & * & \dots & * \end{pmatrix} e_i \\ &= -a \\ &< 0 \end{aligned}$$

Output $x = L^\top e_i$

If pivot is zero, and rows/cols of that pivot is not zero, stop and output not PSD:

$$LSL^\top = \begin{pmatrix} & & \\ 0 & \dots & b \\ \vdots & & \vdots \\ b & \dots & c \\ & & \end{pmatrix}$$

$$\begin{aligned} \begin{pmatrix} c & \dots & -b \end{pmatrix} LSL^\top \begin{pmatrix} c \\ \vdots \\ -b \end{pmatrix} &= \begin{pmatrix} c & \dots & -b \end{pmatrix} \begin{pmatrix} & & \\ 0 & \dots & b \\ \vdots & & \vdots \\ b & \dots & c \\ & & \end{pmatrix} \begin{pmatrix} c \\ \vdots \\ -b \end{pmatrix} \\ &= -cb^2 \\ &\leq 0 \end{aligned}$$

If $c \neq 0$, output $x = L^\top \begin{pmatrix} c \\ \vdots \\ -b \end{pmatrix}$, else output $x = L^\top \begin{pmatrix} 1 \\ \vdots \\ -b \end{pmatrix}$.

In all other cases we can run to completion and output $S = L^{-1}DL^{-\top}$

□