CPSC 665 : **An Algorithmist's Toolkit** Lecture 1 : 2 Feb 2015

Max-Cut and Semidefinite Programming

Lecturer: Sushant Sachdeva *Scribe: Alex Reinking*

1. Introduction

In this lecture we talked about the Max-Cut Problem and its approximation via Semidefinite Programming. We started off by defining the Max-Cut problem and presenting a naive algorithm to approximate it. The invention of Semi-Definite Programming produces a far superior approximation algorithm, as we saw by the end of lecture.

2. Max-Cut

Definition 2.1 (**Graph Cut**). *A* ***cut*** *of a graph $G = (V, E)$ is a bi-partition of V, given by $S \subseteq V$. The cut is the pair $(S, V \setminus S)$. We say that an edge $(i, j) \in E$ is* ***cut*** *when exactly one of i or j is in S. The* ***size*** *of a graph cut is the number of edges cut. An illustration of this can be seen in the figure below.*

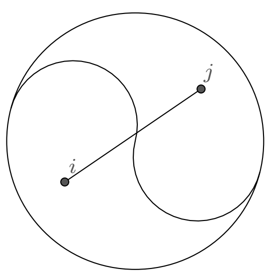

FIGURE 1. Illustration of a graph cut; i and j are to be construed as lying in separate components.

Then, the Max-Cut problem is to find the largest cut of a graph G. This is equivalent to finding a maximum bipartite subgraph in G. It is also one of the 21 classic NP-complete problems, as showed by Karp [Kar72].

2.1. **Randomized Approximation.** It is possible to derive a 1/2-approximation to Max-Cut via a randomized algorithm. The process we follow couldn't be simpler – sample every edge independently and uniformly at random with probability 1/2 to be included in the cut S.

This can be seen by observing that $\mathbf{Pr}[(i, j)\text{is cut}] = 1/2$, which implies that the expected value of the cut is just half of the total number of edges.

This is a dead-simple randomized algorithm, yet people tried fruitlessly for a long time to improve on this. The first breakthroughs were by Michel Goemans and David Williamson [GW94]. They used semidefinite programming to achieve a 0.87856-approximation. This was the first result better than 1/2. So what is semidefinite programming?

Remark 2.2. *A good exercise would be to derive a deterministic, greedy 1/2 approximation*

3. Semidefinite Programming

We begin with some necessary preliminary results. The proofs of these results can be found in any standard text on Linear Algebra, but are restated here for completeness.

Theorem 3.1. *The Spectral Theorem Given $X \in \mathcal{M}_n(\mathbb{R})$ symmetric, let $\lambda_1, \ldots, \lambda_n$ and $u_1, \ldots, u_n$ be the n eigenvalues and eigenvectors respectively, counted with multiplicity.*

Then, the following statements are equivalent:

(1) For all i, $Xu_i = \lambda_i u_i$.
(2) The set $\{u_i\}$ is an orthogonal (normal) basis for $\mathbb{R}^n$
(3) $X = \sum_{i=1}^{n} \lambda_i u_i u_i^T = U\Lambda U^T$ with $U = [u_1 \cdots u_n]$ and $\Lambda = \mathbf{Diag}\,(\lambda_1, \ldots, \lambda_n)$.

(4) By orthogonality, $UU^T = U^T U = I$.

Definition 3.2. *We say that $X \in \mathcal{M}_n(\mathbb{R})$ is symmetric positive semidefinite (written $X \succeq 0$) when X is symmetric and all of its eigenvalues are non-negative.*

Theorem 3.3. *The following statements are equivalent:*

(1) $X \succeq 0$
(2) $\forall y \in \mathbb{R}^n$, $y^T X y = \sum_{i,j} X_{ij} y_i y_j \geq 0$. That is to say, the quadratic forms are all non-negative.
(3) $X = V^T V$, ie. $X_{ij} = V_i^T V_j$ for some $V \in \mathcal{M}_n(\mathbb{R})$. It is also said that X has a "Gram Matrix" representation corresponding to $\{v_1, \ldots, v_n\}$.

Fact 3.4. *Positive Semidefinite matrices are partially ordered. Equivalently, $A \succeq B$ iff $A - B \succeq 0$ iff $A - B$ is positive semidefinite.*

Fact 3.5. *The set of positive semidefinite matrices forms a convex cone.*

$$\{X : X \succeq 0\} = \left\{ \sum_i \lambda_i u_i u_i^T : u_i \in \mathbb{R}^n, \lambda_i \geq 0 \right\}$$

Proof. (Sketch) To prove $\subseteq$ containment, note that if X is positive semidefinite, then it is contained in the right hand side by the spectral theorem.

To prove $\supseteq$ containment, note that if you have any such combination, you can verify that it gives a non-negative quadratic form.

$$y^T (u_i u_i^T \lambda_i) y = \lambda_i (u_i^T y_i)^2 \geq 0$$

$\square$

3.1. **Linear Programming.** We will now put this machinery to use in to give the definition of a semidefinite program.

Definition 3.6. *The Frobenius Dot Product over matrices $X, Y \in \mathcal{M}_n(\mathbb{R})$ is given by the formula*

$$X \bullet Y = \sum_{i,j} X_{ij} Y_{ij} = \mathbf{Tr}(X^T Y) = \mathbf{Tr}(X Y^T) \tag{1}$$

This can be thought of as projecting X, Y to $\mathbb{R}^{n^2}$ and taking the regular dot product there.

So, if we have a linear program on n^2 variables, we can give the program as

$$LP \left[\begin{array}{lll} \text{min } C \bullet X & & C \in \mathcal{M}_n(\mathbb{R}) \\ \text{s.t.} \forall i \in [m] & A_i \bullet X = b_i & b_i \in \mathbb{R} \\ & = \sum_{j,k} (A_i)_{jk} X_{jk} & A_i \in \mathcal{M}_n(\mathbb{R}) \end{array} \right. \tag{2}$$

Semidefinite programs just add $X \succeq 0$ as a constraint! So they're easy to understand in terms of what we already know!

3.2. **An Example SDP.** Imagine we had some matrix (see the figure below for an illustration of the constraints)

$$A = \begin{pmatrix} x & z \\ z & y \end{pmatrix} \Leftrightarrow \begin{array}{c} x \geq 0 \\ y \geq 0 \end{array} \text{ and } xy \geq z^2 \tag{3}$$

You can use this specialized case to ask the question: what is the minimum of

$$\min_X \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \bullet X = X_{12} + X_{21} \tag{4}$$

such that $X_{11} = 1$ and $X_{22} = 2$. These constraints may be encoded by the following matrix equations:

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \bullet X = 1 \text{ and } \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \bullet X = 2 \tag{5}$$

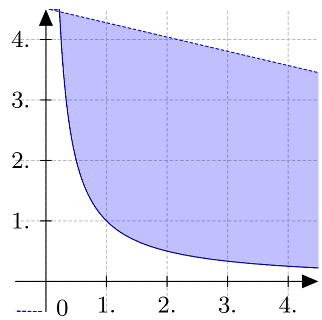

FIGURE 2. Illustration a level set of a positive semi-definite form, $xy \geq 1$.

Remark 3.7. *For a good exercise, you should try to show that the minimizing X gives $C \bullet X = -2\sqrt{2}$ with*

$$X = \begin{pmatrix} 1 & -\sqrt{2} \\ -\sqrt{2} & 2 \end{pmatrix} \tag{6}$$

Theorem 3.8. *You can find an X that is optimal up to $\epsilon > 0$ in time* $\mathbf{poly}(n, m, L, \log \frac{1}{\epsilon})$ *where n is the number of variables, m is the number of constraints, and L is the bit complexity.*

Remark 3.9. *Of course, these examples only give equality constraints. Really, we would like to show that SDPs generalize LPs. This makes for a good exercise. As an outline, note that we have roughly the following correspondence:*

LP		**SDP**	*(slower thanks to n^2 vars)*
$\min c^T x$		$\min C \bullet X$	$A_i = \mathbf{Diag}\,(a_i)$
subj. to $Ax = b$,	$\Leftrightarrow$	*subj. to* $A_i \bullet X = b_i$	$C = \mathbf{Diag}\,(c)$
$x \geq 0$		$X \succeq 0$	

4. Max-Cut Approximation via SDP

At last, we are prepared to return to the Max-Cut problem. We will construct a Quadratic Integer Program that expresses our *intent*. If you think of

$$x_i = \begin{cases} +1 & \text{if } i \in S \\ -1 & \text{if } i \notin S \end{cases} \tag{7}$$

then looking at an edge (i, j), we don't really get a linear constraint. One suitable non-linear constraint is given by: (i, j) is cut $\Leftrightarrow x_i x_j = -1 \Leftrightarrow \frac{1}{4}(x_i - x_j)^2 = 1$

So to maximize the cut size, we really want to maximize

$$\max \sum_{(i,j)\in E} \frac{1}{2}(1 - x_i x_j) = \sum_{(i,j)\in E} \frac{1}{4}(x_i - x_j)^2 \tag{8}$$

subject to the constraints that $x_i \in \{1, -1\}$, or, equivalently, that $x_i^2 = 1$.

Lemma 4.1. *The optimal solution of this QIP = OPT Max-Cut*

4.1. **Vector Relaxation.** Unfortunately, quadratic integer programs (like linear integer programs) are nearly intractable. So, we relax the constraints to ignore *direction* and instead care only about unit magnitude. The correspondence for these programs is natural (let $X_{ij} = v_i^T v_j$):

$$\begin{pmatrix} \textbf{QIP} \\ \max \sum_{(i,j)\in E} \frac{1}{4}(x_i - x_j)^2 \\ \text{subj. to } x_i^2 = 1 \end{pmatrix} \Leftrightarrow \begin{pmatrix} \textbf{Relaxed} \\ \max \sum_{(i,j)\in E} \frac{1}{4}\left\|v_i - v_j\right\|^2 \\ \text{subj. to } \left\|v_i\right\|^2 = 1 \end{pmatrix} \Leftrightarrow \begin{pmatrix} \textbf{SDP} \\ \max \sum_{(i,j)\in E} \frac{1}{4}(X_{ii} - 2X_{ij} + X_{jj}) \\ \text{subj. to } X_{ii} = 1, X \succeq 0 \end{pmatrix}$$

So this is a really useful tool!

Theorem 4.2. *Given X, an optimal SDP solution up to ϵ, you can obtain $\{v_i\}$ in $O(n^3)$ arithmetic operations.*

And that's it for today. We'll close with a few remarks about SDPs

Remark 4.3. *It is best to think of SDPs as vector relaxations of QIPs.*

Remark 4.4. *All constraints and objectives must be linear in the Gram Matrix $(v_i^T v_j)_{ij}$, e.g. $\|v_i\| = 1 \Leftrightarrow v_i^T v_i = 1$.*

Remark 4.5. *You can't assume v_i lie in any particular dimension (other than n), and you cannot enforce any such dimension constraint. This means we cannot enforce an upper bound on the rank of X.*

Remark 4.6. *Since we only care about the magnitudes, $v_i^T v_j$, the $\{v_i\}$ need not be unique even if X is unique.*

We'll continue with more on semidefinite programming next week!

References

[GW94] Michel X Goemans and David P Williamson. .879-approximation algorithms for max cut and max 2sat. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 422–431. ACM, 1994.

[Kar72] Richard M Karp. *Reducibility among combinatorial problems.* Springer, 1972.