Scribed by Rachel Lawrence

Scribe Notes of Lecture 4

In which we introduce semidefinite programming and apply it to Max Cut.

1 Overview

We begin with an introduction to Semidefinite Programming (SDP). We will then see that, using SDP, we can find a cut with the same kind of near-optimal performance for Max Cut in random graphs as we got from the greedy algorithm – that is,

$$cut > \frac{|E|}{2} + \Omega(n \cdot \sqrt{d})$$

in random graphs $G_n, \frac{d}{n}$. More generally, we will prove that you can always find a cut at least this large in the case that G is triangle-free and with maximum vertex degree $\geq d$, which will imply the bound in random graphs. We will also see how to use SDP to certify an upper bound:

$$max\ cut < \frac{|E|}{2} + O(n \cdot \sqrt{d})$$

with high probability in $G_{n, \frac{d}{n}}$

Methods using SDP will become particularly helpful in future lectures when we consider planted-solution models instead of fully random graphs: greedy algorithms will fail on some analogous problems where methods using SDP can succeed.

2 Semidefinite Programming

Semidefinite Programming (SDP) is a form of convex optimization, similar to linear programming but with the addition of a constraint stating that, if the variables in the linear program are considered as entries in a matrix, that matrix is positive semidefinite. To formalize this, we begin by recalling some basic facts from linear algebra.

2.1 Linear algebra review

Definition 1 (Positive Semidefinite) *A matrix $M \in \mathbb{R}^{n\times n}$ is positive semidefinite (abbreviated PSD and written $M \succeq \mathbf{0}$) if it is symmetric and all its eigenvalues are non-negative.*

We will also make use of the following facts from linear algebra:

1. If $M \in \mathbb{R}^{n\times n}$ is a symmetric matrix, then all the eigenvalues of M are real, and, if we call $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ the eigenvalues of M with repetition, we have

$$M = \sum_i \lambda_i \mathbf{v}^{(i)} (\mathbf{v}^{(i)})^T$$

   where the $\mathbf{v}^{(i)}$ are orthonormal eigenvectors of the λ_i.

2. The smallest eigenvalue of M has the characterization

$$\lambda_1 = \min_{\mathbf{y}\neq\mathbf{0}} \frac{\mathbf{y}^T M \mathbf{y}}{||\mathbf{y}||^2}$$

   and the optimization problem in the right-hand side is solvable up to arbitrarily good accuracy

This gives us the following lemmas:

Lemma 2 *$M \succeq \mathbf{0}$ if and only if for every vector $\mathbf{y}$ we have $\mathbf{y}^T M \mathbf{y} \ge 0$.*

Proof: From part (2) above, the smallest eigenvalue of M is given by

$$\lambda_1 = \min_{\mathbf{y}\neq\mathbf{0}} \frac{\mathbf{y}^T M \mathbf{y}}{||\mathbf{y}||^2}$$

Noting that we always have $||\mathbf{y}||^2 \ge 0$, then $\lambda_1 \ge 0$ if and only if the numerator $\mathbf{y}^T M \mathbf{y}$ on the right is always non-negative. □

Lemma 3 *If $A, B \succeq \mathbf{0}$, then $A + B \succeq \mathbf{0}$*

Proof: $\forall \mathbf{y}$, $\mathbf{y}^T(A+B)\mathbf{y} = \mathbf{y}^T A\mathbf{y} + \mathbf{y}^T B\mathbf{y} \ge 0$. By Lemma 2, this implies $A + B \succeq 0$. □

Lemma 4 *If $A \succeq 0$ and $a \ge 0$, then $aA \succeq 0$*

Proof: $\forall y$, $\mathbf{y}^T aA\mathbf{y} = a(\mathbf{y}^T A\mathbf{y}) \ge 0$. By Lemma 2, this implies $aA \succeq 0$. □

2.2 Formulation of SDP

With these characterizations in mind, we define a *semidefinite program* as an optimization program in which we have n^2 real variables $X_{i,j}$, with $1 \leq i, j \leq n$, and we want to maximize, or minimize, a linear function of the variables such that linear constraints over the variables are satisfied (so far this is the same as a linear program) and subject to the additional constraint that the matrix X is PSD. Thus, a typical semidefinite program (SDP) looks like

$$\begin{aligned}
&\max \sum_{i,j} C_{i,j} X_{i,j} \\
&s.t. \\
&\qquad \sum_{i,j} A^{(1)}_{i,j} X_{i,j} \leq b_1 \\
&\qquad \vdots \\
&\qquad \sum_{i,j} A^{(m)}_{i,j} X_{i,j} \leq b_m \\
&\qquad X \succeq \mathbf{0}
\end{aligned}$$

where the matrices $C, A^{(1)}, \ldots, A^{(m)}$ and the scalars $b_1, \ldots, b_m$ are given, and the entries of X are the variables over which we are optimizing.

We will also use the following alternative characterization of PSD matrices

Lemma 5 *A matrix $M \in \mathbb{R}^{n \times n}$ is PSD if and only if there is a collection of vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ such that, for every i, j, we have $M_{i,j} = \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle$.*

Proof: Suppose that M and $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ are such that $M_{i,j} = \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle$ for all i and j. Then M is PSD because for every vector $\mathbf{y}$ we have

$$\mathbf{y}^T M \mathbf{y} = \sum_{i,j} y_i y_j M_{i,j} = \sum_{i,j} y_i y_j \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle = \left\| \sum_i y_i \mathbf{x}^{(i)} \right\|^2 \geq 0$$

Conversely, if M is PSD and we write it as

$$M = \sum_k \lambda_k \mathbf{v}^{(k)} (\mathbf{v}^{(k)})^T$$

we have

$$M_{i,j} = \sum_k \lambda_k v_i^{(k)} v_j^{(k)}$$

and we see that we can define n vectors $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}$ by setting

$$x_k^{(i)} := \sqrt{\lambda_k} \cdot v_i^{(k)}$$

and we do have the property that

$$M_{i,j} = \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle$$

□

This leads to the following equivalent formulation of the SDP optimization problem:

$$\begin{aligned}
\max \sum_{i,j} & C_{i,j} \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle \\
s.t. & \\
& \sum_{i,j} A_{i,j}^{(1)} \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle \leq b_1 \\
& \vdots \\
& \sum_{i,j} A_{i,j}^{(m)} \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle \leq b_m
\end{aligned}$$

where our variables are vectors $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}$. This is the statement of the optimization problem that we will most commonly use.

2.3 Polynomial time solvability

From lemmas 3 and 4, we recall that if A and B are two matrices such that $A \succeq \mathbf{0}$ and $B \succeq \mathbf{0}$, and if $a \geq 0$ is a scalar, then $a \cdot A \succeq \mathbf{0}$ and $A + B \succeq 0$. This means that the set of PSD matrices is a convex subset of $\mathbb{R}^{n \times n}$, and that the above optimization problem is a convex problem.

Using the ellipsoid algorithm, one can solve in polynomial time (up to arbitrarily good accuracy) any optimization problem in which one wants to optimize a linear function over a convex feasible region, provided that one has a *separation oracle* for the feasible region: that is, an algorithm that, given a point,

1. Checks whether it is feasible and, if not,
2. Constructs an inequality that is satisfied by all feasible point but not satisfied by the given point.

In order to construct a separation oracle for a SDP, it is enough to solve the following problem: given a matrix M, decide if it is PSD or not and, if not, construct an inequality $\sum_{ij} a_{ij} x_{ij} \geq 0$ that is satisfied by the entries of all PSD matrices but that is not satisfied by M. In order to do so, recall that the smallest eigenvalue of M is

$$\min_{\mathbf{y}} \frac{\mathbf{y}^T M \mathbf{y}}{||\mathbf{y}||^2}$$

and that the above minimization problem is solvable in polynomial time (up to arbitrarily good accuracy). If the above optimization problem has a non-negative optimum, then M is PSD. If it is a negative optimum $\mathbf{y}^*$, then the matrix is not PSD, and the inequality

$$\sum_{i,j} X_{i,j} y_i^* y_j^* \geq 0$$

is satisfied for all PSD matrices X but fails for $X := M$. Thus we have a separation oracle and we can solve SDPs in polynomial time up to arbitrarily good accuracy.

3 SDP Relaxation of Max Cut and Random Hyperplane Rounding

The Max Cut problem in a given graph $G = (V, E)$ has the following equivalent characterization, as a quadratic optimization problem over real variables $x_1, \ldots, x_n$, where $V = \{1, \ldots, n\}$:

$$\begin{aligned} \mathit{max\ cut}(G) = \max & \sum_{(i,j)\in E} \frac{1}{4}(x_i - x_j)^2 \\ s.t. & \\ & x_i^2 = 1 \qquad \forall i \in V \end{aligned}$$

We can interpret this as associating every vertex v with a value $x_v = \pm 1$, so that the cut edges are those with one vertex of value $+1$ and one of value -1.

While quadratic optimization is NP-hard, we can instead use a relaxation to a polynomial-time solvable problem. We note that any quadratic optimization problem has a natural relaxation to an SDP, in which we relax real variables to take vector values and we change multiplication to inner product:

$$\begin{aligned} \mathit{max\ cut}(G) \leq \max & \sum_{(i,j)\in E} \frac{1}{4}||\mathbf{x}_i - \mathbf{x}_j||^2 \\ s.t. & \\ & ||\mathbf{x}_i||^2 = 1 \qquad \forall i \in V \end{aligned}$$

Solving the above SDP, which is doable in polynomial time up to arbitrarily good accuracy, gives us a unit vector $\mathbf{x}_i$ for each vertex i. A simple way to convert this collection to a cut $(S, V - S)$ is to take a random hyperplane through the origin, and then define S to be the set of vertices i such that $\mathbf{x}_i$ is above the hyperplane. Equivalently, we pick a random vector $\mathbf{g}$ according to a rotation-invariant distribution, for example a Gaussian distribution, and let S be the set of vertices i such that $\langle \mathbf{g}, \mathbf{x}_i \rangle \geq 0$.

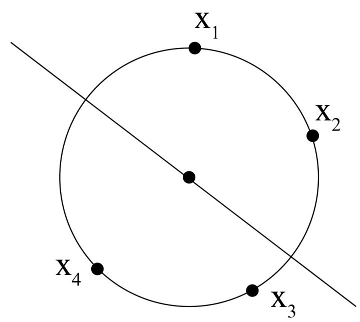

Figure 1: The hyperplane through the origin defines a cut partitioning the vertices into sets $\{x_1, x_2\}$ and $\{x_3, x_4\}$.

Let (i, j) be an edge: One sees that if θ is the angle between $\mathbf{x}_i$ and $\mathbf{x}_j$, then the probability (i, j) is cut is proportional to θ:

$$\mathbb{P}[(i,j) \text{ is cut }] = \frac{\theta}{\pi}$$

and the contribution of (i, j) to the cost function is

$$\frac{1}{4}||\mathbf{x}_i - \mathbf{x}_j||^2 = \frac{1}{2} - \frac{1}{2}\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \frac{1}{2} - \frac{1}{2}\cos\theta$$

Some calculus shows that for every $0 \leq \theta \leq \pi$ we have

$$\frac{\theta}{\pi} > .878 \cdot \left(\frac{1}{2} - \frac{1}{2}\cos\theta\right)$$

and so

$$\mathbb{E}[\text{ number of edges cut by } (S, V - S)] \geq .878 \cdot \sum_{(i,j)\in E} \frac{1}{4}||\mathbf{x}_i - \mathbf{x}_j||^2$$

$$= .878 \cdot \text{SDPMaxCut}(G) \geq .878 \cdot \text{MaxCut}(G)$$

so we have a polynomial time approximation algorithm with worst-case approximation guarantee .878.

Next time, we will see how the SDP relaxation behaves on random graphs, but first let us how it behaves on a large class of graphs.

4 Max Cut in Bounded-Degree Triangle-Free Graphs

Theorem 6 *If $G = (V, E)$ is a triangle-free graph in which every vertex has degree at most d, then*

$$MaxCut(G) \geq \left(\frac{1}{2} + \Omega\left(\frac{1}{\sqrt{d}}\right)\right) \cdot |E|$$

Proof: Consider the following feasible solution for the SDP: we associate to each node i an n-dimensional vector $\mathbf{x}^{(i)}$ such that $x_i^{(i)} = \frac{1}{\sqrt{2}}$, $x_j^{(i)} = -1/\sqrt{2deg(i)}$ if $(i,j) \in E$, and $x_j^{(i)} = 0$ otherwise. We immediately see that $||\mathbf{x}^{(i)}||^2 = 1$ for every i and so the solution is feasible.

For example, if we have a graph such that vertex 1 is adjacent to vertices 3 and 5:

	1	2	3	4	5	$\cdots$
$x^{(1)}$:	$\frac{1}{\sqrt{2}}$	0	$\frac{1}{\sqrt{2deg(1)}}$	0	$\frac{1}{\sqrt{2deg(1)}}$	
$x^{(2)}$:	0	$\frac{1}{\sqrt{2}}$	0	0	0	
$x^{(3)}$:	$\frac{1}{\sqrt{2deg(3)}}$	0	$\frac{1}{\sqrt{2}}$	0	0	
$\vdots$						$\vdots$
$x^{(n)}$:	0	0	0	0	0	$\cdots$

Let us transform this SDP solution into a cut $(S, V-S)$ using a random hyperplane.

We see that, for every edge (i,j) we have

$$\langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle = -\frac{1}{\sqrt{2d(i)}} - \frac{1}{\sqrt{2d(j)}} \leq -\frac{1}{\sqrt{d}}$$

The probability that (i,j) is cut by $(S, V-S)$ is

$$\frac{\arccos\left(\frac{1}{2} - \frac{1}{2\sqrt{d}}\right)}{\pi}$$

and

$$\frac{\arccos\left(\frac{1}{2} - \frac{1}{2\sqrt{d}}\right)}{\pi} = \frac{1}{2} + \frac{\arcsin\left(\frac{1}{2\sqrt{d}}\right)}{\pi} \geq \frac{1}{2} + \Omega\left(\frac{1}{\sqrt{d}}\right)$$

so that the expected number of cut edges is at least $\left(\frac{1}{2} + \Omega\left(\frac{1}{\sqrt{d}}\right)\right) \cdot |E|$. $\Box$