| **Advanced SDPs** | **Lecture 2 : February 16, 2017** |
|---|---|
| *Lecturers: Nikhil Bansal and Daniel Dadush* | *Scribe: Nikhil Bansal* |

# 2.1 Grothendieck's inequality and Maximizing Quadratic Forms

In the last class we saw the hyperplane rounding scheme for the max-cut problem. Another useful way to view it is the following. Given the unit vector $u_i$ for vertex $i$, we set it to $\text{sign}(\langle u_i, g\rangle)$ where $g$ is a random Gaussian. The expected contribution of edge $(i,j)$ turns out to be $(1-\text{sign}(\langle u_i, g\rangle)\text{sign}(\langle u_i, g\rangle))/2$, while the SDP contribution is $1/4\|u_i - u_j\|^2 = 1/2(1 - u_i \cdot u_j)$. The analysis then involved showing that for any two vectors $u_i, u_j$,

$$\mathbb{E}[(1 - \text{sign}(\langle u_i, g\rangle)\text{sign}(\langle u_j, g\rangle))/2] \geq \alpha_{GW} 1/2(1 - u_i \cdot u_j)$$

where $\alpha_{GW} = 0.878....$

We now consider the following problem. Given an $m \times n$ matrix A with possibly negative entries, Find $x \in \{-1,1\}^m$ and $y \in \{-1,1\}^n$ to maximize $x^T A y = \sum_{i,j} x_i a_{ij} y_j$ .

As usual we write the SDP (vector) relaxation: $\max \sum_{ij} a_{ij} u_i \cdot v_j$ such that $\|u_i\|_2^2 = 1$ for $i \in [m]$ and $\|v_j\|^2 = 1$ for $j \in [n]$.

How can we round this to a $\{\pm 1\}$ solution. As we discussed last time, the hyperplane rounding algorithm may not give any guarantee due to the negative and positive terms.

We consider three different approaches. We first make a simple but useful observation.

**Lemma 1** *Suppose we have a solution with objective value $B$, such that $x_i, y_j \in [-1,1]$. Then we can also obtain a $\{-1,1\}$ with value at least $B$.*

**Proof:** We can modify the variables one by one to $-1$ or 1 while never decreasing the objective as follows. Fix all variables except $x_1$. The objective is linear in $x_1$,

so there is a direction where the objective does not decrease, keep moving $x_1$ until it reaches $-1$ or 1. Repeat the same by all variables except $x_2$ and so on. ■

Note that this trick works whenever there are no terms likes $x_i^2$.

## 2.2 Algorithm 1

We can assume that the $u_i$ and $v_j$ lie in $\mathbb{R}^{m+n}$. Let $B$ denote SDP objective. Let $g$ be a random gaussian vector (iid $N(0,1)$ entries). Let $x_i = \langle g, u_i \rangle$ and $y_j = \langle v_j, g \rangle$.

Recall from Exercise 1 that $x_i$ and $x_j$ also distributed as $N(0,1)$ (but correlated due to the common $g$) and also satisfy

**Lemma 2** $\mathbb{E}[x_i y_j] = u_i \cdot v_j$.

**Proof:** Let $d$ denote the dimension where the vectors lie. Let $g(1), \ldots, g(d)$ be the coordinates of $g$. Then,

$$\mathbb{E}[x_i y_j] = \mathbb{E}[\langle g, u_i \rangle \langle g, v_j] = \mathbb{E}[(\sum_{k=1}^{d} g(k) u_i(k))(\sum_{k'=1}^{d} g(k') v_j(k'))]$$

As the coordinates $g(k)$ are independent and $N(0,1)$, we have $\mathbb{E}[g(k)g(k')] = 0$ if $k \neq k'$ and $\mathbb{E}[g(k)^2] = 1$. So the above sum is

$$\sum_{k=1}^{d} u_i(k) v_j(k) = u_i \cdot v_j$$

■

So, we have that

$$\mathbb{E}[\sum_{ij} a_{ij} x_i y_j] = \sum_{ij} a_{ij} u_i \cdot v_j = B$$

Now if $x_i$ and $y_j$ would lie in $[-1, 1]$ we would be done by Lemma 1. Of course, there is no reason why this should hold (indeed if it would, we would have solved the problem exactly). But note that $x_i$ and $y_j$ are distributed as $N(0,1)$, so the probability that any of them exceeds $c\sqrt{\log n}$ is at most say $1/n^4$ for some constant $c$. So morally, one can view all the $x_i$ and $y_j$ lying in $[-c\sqrt{\log n}, c\sqrt{n}]$.

More formally, consider the following algorithm. Let $M = c\sqrt{\log n}$. If some $x_i$ or $y_j$ lies outside $[-M, M]$ return the all 0 solution. Otherwise return $\tilde{x}_i = x_i/M$ and $\tilde{y}_j = y_j/M$. Indeed, as one would expect, this truncation does not lose much and the solution with $\tilde{x}$ and $\tilde{y}$ has value about $B/M^2 = \Omega(B/\log n)$. We skip the (relatively) details of bounding the effects of truncation here, as we will see a much better guarantee using this approach later. But the reader may wish to try doing this formally.

### 2.2.1 Algorithm 2

Before we begin, recall the problem from Exercise 1 to show that if $p(x) = p_0 + p_1 x + \cdots + p_k x^k$ is a univariate polynomial with non-negative coefficients, and $A$ is a PSD matrix with $a_{ij} = u_i \cdot u_j$. Then the matrix with entries $p(a_{ij})$ is also PSD.

Here is a cool way of showing this. Define

$$u_i' = p_0^{1/2} e_0 \oplus \sum_{h=1}^{k} \oplus p_h^{1/2} u_i^{\otimes h}$$

Here $\oplus$ means that we take a direct sum of the different vector spaces that arise as tensor powers (and thus view them as contributing separate coordinates), and $e_0 = u_i^{\otimes 0}$ is a direction (coordinate) that is orthogonal to other power of $u_i$.

Then the key observation is that,

$$u_i' \cdot u_j' = \sum_{h=0}^{k} p_h \langle u_i^{\otimes h}, u_j^{\otimes h} \rangle = \sum_{h=1}^{k} p_h (u_i \cdot u_j)^h = p(a_{ij})$$

which gives a Gram decomposition of $p(A)$, hence implying that it is PSD.

## 2.3 Krivine's proof of Grothendieck's inequality

Amazingly, already in the 50's Groethendieck showed that integrality gap of the SDP above is at most $2\ln(1+\sqrt{2})/\pi \approx 0.56$ (this is long before SDPs were invented!). Below we give Krivine's proof of this, which also gives an algorithm.

The idea will be the following. Let $B = \sum_{ij} a_{ij} u_i \cdot v_j$ be the SDP objective. Suppose that given $u_i$ and $v_j$, we would construct some other unit vectors $u_i'$ and $v_j'$ such that

applying the hyperplane rounding of $u'_i$ and $v'_j$ gives contribution $u_i \cdot v_j$ in expectation, i.e.

$$\mathbb{E}[\text{sign}(\langle g, u'_i\rangle)\text{sign}(\langle g, v'_j\rangle)] = c' u_i \cdot v_j$$

for some fixed $c'$, then we would get a $c'$ approximate solution.

How can we accomplish this?

Suppose $u'_i$ and $v'_i$ are two unit vectors at angle $\theta'$. Then a random hyperplane separates then with probability $\theta'/\pi$ and otherwise they have the same sign. So,

$$\mathbb{E}[\text{sign}(\langle g, u'_i\rangle)\text{sign}(\langle g, v'_j\rangle)] = (-1)\frac{\theta'}{\pi} + (1)\frac{\pi - \theta'}{\pi} = 1 - \frac{2\theta'}{\pi} = \frac{2}{\pi}(\frac{\pi}{2} - \theta')$$

As $u'_i \cdot v'_j = \cos\theta' = \sin(\pi/2 - \theta')$, the rhs above is simply $\frac{2}{\pi}\arcsin(u'_i \cdot v'_j)$.

So, let us try to find a transformation $u_i \to u'_i$ and $v_j \to v'_j$ such that $c(u_i \cdot u_j) = \arcsin(u'_i \cdot v'_j)$ or equivalently $u'_i \cdot v'_j = \sin(c(u_i \cdot v_j))$ for some $c$. By the Taylor expansion of sin,

$$\sin(cu_i \cdot v_j) = \sum_{k=0}^{\infty}(-1)^k \frac{c^{2k+1}}{2k+1!}(u_i \cdot v_j)^{2k+1}$$

So, using the trick we saw before, we can define the vectors $u'_i$ and $v'_j$ as functions of $u_i$ and $v_j$ as

$$u'_i = \sum_{k=0}^{\infty} \oplus \left(\frac{c^{2k+1}}{2k+1!}\right)^{1/2} u_i^{\otimes 2k+1} \text{ and } v'_j = \sum_{k=0}^{\infty} \oplus (-1)^k \left(\frac{c^{2k+1}}{2k+1!})\right)^{1/2} v_j^{\otimes 2k+1}$$

This does the job as $u'_i \cdot v'_j = \sin(c(u_i \cdot v_j))$. It remains to find $c$. As $u'_i$ and $v'_j$ unit vectors, we note that their length is

$$\|u'_i\|^2 = \|v'_j\|^2 = \sum_{k=0}^{\infty} \frac{c^{2k+1}}{2k+1!} = \frac{1}{2}(e^{cx} - e^{-cx}) = \sinh(c).$$

So we set $c = \sinh^{-1}(1) = \ln(1 + \sqrt{2})$.

Putting everything together this gives the overall $c' = \frac{2}{\pi}c = \frac{2}{\pi}\ln(1 + \sqrt{2})$ approximation guarantee.

It is important to note that we applied different transformations to $u_i$ and $v_j$ (as the Taylor expansion for sin involved both positive and negative coefficient). The above approach would not work if our quadratic form was $\sum_{ij} a_i j u_i \cdot u_j$ for example. Indeed this turns out to be a harder question and is called minimizing quadratic forms on

graphs. We will not discuss this here, but the technique we will show next is more robust, and can be used to obtain guarantees for this more general question also. Interestingly, the approximation ratio depends roughly as $\log(\chi(G))$ where $\chi(G)$ is the chromatic number of the support graph $G$ of $A$. Note (and show) that the problem we are discussing corresponds to quadratic form on a bipartite graph.

### 2.3.1 Algorithm 3: Truncation + Bounding error

Let us consider the truncation approach again, where $M$ is a constant, say $M = 5$. Let $X_i = \langle g, u_i \rangle$ and $Y_j = \langle g, v_j \rangle$ be the gaussian projects. Let us define $\tilde{X}_i = X_i$ if $|X_i| \leq M$ and 0 otherwise, and similarly define $\tilde{Y}_j$.

Let $x_i = \tilde{X}_i/M$ and $y_j = \tilde{Y}_j/M$. Clearly, $x_i, y_j \in [-1, 1]$. By Lemma 1 this gives a solution of value at least $(1/M^2) \sum_{ij} a_{ij} \tilde{X}_i \tilde{Y}_j$, so we need to bound the error due to truncation on the objective.

We know that $\mathbb{E}[\sum_{ij} a_{ij} X_i Y_j] = B$, the SDP value. Let $\tilde{B} = \mathbb{E}[\sum_{ij} \tilde{X}_i \tilde{Y}_j]$. Then,

$$B - \tilde{B} = \sum_{ij} a_{ij}(X_i Y_j - \tilde{X}_i \tilde{Y}_j) = \sum_{ij} a_{ij}((X_i - \tilde{X}_i)Y_j + \tilde{X}_i(Y_j - \tilde{Y}_j)) = \sum_{ij} a_{ij}(U_i Y_j + \tilde{X}_i V_j)$$

where $U_i = X_i - \tilde{X}_i$ and $V_j = Y_j - \tilde{Y}_j$.

But how do we bound this error?

The crucial insight is to connect the random variables to geometry. We note that given any collection of random vectors $X_1, \ldots, X_k$ on some probability space, we can associate vectors $w_1, \ldots, w_k$ to them (for those familiar, this the standard embedding of random variables in the Hilbert space), such that $\mathbb{E}[X_i X_j] = w_i \cdot w_j$ for all $i, j \in [k]$. In particular, $\mathbb{E}[X_i^2] = \|w_i\|^2$.

So to bound $\mathbb{E}[\sum_{ij} a_{ij} U_i V_j]$, it suffices to upper bound $\sum_{ij} a_{ij}(u_i \cdot v_j)$, where $u_i$ and $v_j$ are vectors with $\mathbb{E}[U_i^2] \leq \|u_i\|^2$

Now, it is not hard to check by the way $U_i$ was defined that $\mathbb{E}[U_i^2] \leq \gamma := 4M^2 e^{-M^2/2}$. For $M = 5$, $\gamma \leq 1/10$. Moreover, $\mathbb{E}[V_j^2] = 1$. So, this is just a scaled version of our original problem to begin with, and thus this error can be upper bounded by $\gamma B$. Similarly for the second error term.

So $\tilde{B} \geq B(1 - 2\gamma) \geq B/2$, giving an overall approximation of at least $B/(2M^2)$.

## 2.4 Laplacian, Conductance and Expansion

The Laplacian of a graph is defined as $D - A$ where $D$ is a diagonal matrix with entries as degrees. $L_G = D - A = \sum_{(i,j)\in E}(e_i - e_j)(e_i - e_j)^T$ which is $dI - A$ if $G$ is regular.

Note that $L_G$ is PSD, and that $x^T L_G x = \sum_{(u,v)\in E}(x_u - x_v)^2$. So we could write max-cut as $\max_{x\in\{0,1\}^n} x^T L_G x$.

Given a graph $G$, the conductance or edge expansion of a set $S$ is defineas

$$\phi(S) = \frac{E(S,\bar{S})}{\sum_{v\in S} d_v} = \frac{E(S,\bar{S})}{vol(S)}$$

which is the fraction of edges across cut divided by total possible edges from $S$.

The conductance of $G$ is defined as

$$\phi(G) = \min_{S:Vol(S)\leq Vol(G)/2} \phi(S)$$

For $G$ regular this is

$$\phi(G) = \min_{S:|S|\leq n/2} \frac{E(S,\bar{S})}{d|S|}$$

Examples: Conductance of cycle: $2/n$. Hypercube on $n$ vertices: $1/\log n$. Random graph $G_{n,p}$ with $p \geq 2\log n/n$ has $\Omega(1)$ expansion.

Understanding conductance of a graph plays a major role in graph partitioning, markov chain mixing, and various other areas in theoretical CS and optimization. A graph with conductance $\Omega(1)$ is called an expander and these are extremely useful objects with various applications.

How do we get a handle on the conductance of a graph?

In the next few lectures we will see three approaches for this. First is the so called Cheeger's inequality which relates the second eigenvalue of the normalized Laplacian to expansion. This is one of the most beautiful results in spectral graph theory.

In particular if $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ are the eigenvalues of the normalized Laplacian. Then

$$\lambda_2/2 \leq \Phi(G) \leq \sqrt{2\lambda_2}$$

In general both these inequalities are tight. So, if $\lambda_2 = \Omega(1)$, then $\lambda_2$ provides a good approximation of $\Phi(G)$.

Second, we will see an LP based $O(\log n)$ approximation. Finally, we will see a major breakthrough result of Arora-Rao and Vazirani on a roughly $O(\sqrt{\log n})$ approximation for sparsest cut based on SDPs.