CS 224: Advanced Algorithms Spring 2017

Lecture 14 — March 9, 2017

Prof. Jelani Nelson *Scribe: Demi Guo*

1 Overview

In this lecture, we will talk about:

1. Semi definite programming (SDP)
2. Goemans-Williamson MaxCut
3. Approximation in other settings (streaming algorithms)

2 Semi Definite Programming

2.1 Initial Motivation

A linear programming problem is one in which we wish to maximize or minimize a linear objective function of real variables over a polytope. In semidefinite programming, we instead use real-valued vectors and are allowed to take the dot product of vectors; nonnegativity constraints on real variables in LP are replaced by semidefiniteness constraints on matrix variables in SDP. Specifically, a general semidefinite programming problem can be defined as any mathematical programming problem of the form:

$$\begin{aligned} \min_{x^1,\ldots x^n\in\mathbb{R}^n} & \sum_{i,j\in[n]} c_{i,j}(x^i\cdot x^j) \\ \text{subject to} & \sum_{i,j\in[n]} a_{i,j,k}(x^i\cdot x^j) \le b_k \ \forall k \end{aligned}$$

2.2 Linear Algebra

Fact 1. $tr(A^TB) = \sum_{i,j} A_{i,j}B_{i,j}$

Fact 2. X is Positive Semi Definite (PSD) if $\forall z \in \mathbb{R}^n$, $z^TXz \ge 0$.

Fact 3 (Loewner Ordering)**.** If $A \succeq B$, then $A - B$ is PSD.

Fact 4. For a real symmetric matrix X, we have the following properties:

1. X is PSD.
2. All eigenvalues of X are ≥ 0.
3. $\exists M$ s.t. $X = M^TM$

2.3 Formal Definition

$$\min tr(C^T X)$$
$$\text{subject to } tr(A_k^T X) = b_k, k = 1, 2, ...m$$
$$X \succeq 0$$

where entry i, j in C is given by $c_{i,j}$ from the previous section and A_k is an $n \times n$ matrix having i, jth entry $a_{i,j,k}$ from the previous section. Note that originally, it should be $tr(A_k^T X) = b_k$, but after adding slack variables the inequality can become equality.

2.4 Relationship to Vector Programming

SDP is equivalent to vector programming. By Fact 4, we can write $X = M^T M$. If we have:

$$M = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^n \end{bmatrix} \tag{1}$$

where $x^i \in \mathbb{R}^n$, we can write $X_{i,j} = \langle x^i, x^j \rangle = x^i \cdot x^j$. According to Section 2.1, we can easily see that SDP can be interpreted as vector programming.

2.5 Properties

(see book "semi-definite programming" by Vandem Berghe, Boyd) Given any L, we can solve SDP up to L bits of precision in time poly(n,m,L).

3 Goemans -Williamson MaxCut

3.1 Definition

$$\max \sum_{(i,j)\in E} \frac{1 - x_i x_j}{2}$$
$$\text{s.t. } \forall i \in [n], x_i \in \{-1, 1\}$$

We color every vertex i, x_i, using two colors $-1, 1$.

3.2 VP Relaxation

We can relax the definition above into a VP or SDP problem:

$$\max \sum_{(i,j)\in E} \frac{1 - \langle v_i, v_j \rangle}{2}$$
$$\text{s.t.} \forall i \in [n], \langle v_i, v_i \rangle = \|v_i\|_2^2 = 1$$

In fact, there's an integrality gap between the two problems. In HW5, you are asked to demonstrate that sometimes the optimal solution of VP Relaxation is better than the optimal solution of MaxCut.

3.3 Goemans-Wiliamson

The best algorithm so far, by (JACM'95), is a randomized algorithm using hyperplane rounding technique, which in expectation cuts $\alpha \cdot OPT$ edges. More formally, we have:

Theorem 5. $\forall \varepsilon$, we can obtain an $(\alpha + \varepsilon)-$approximation to MaxCut in time $poly(n, lg\frac{1}{\varepsilon})$, where

$$\alpha = \inf_{0 \le \theta \le \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos(\theta)} = 0.87856$$

The above rounding can be derandomized. [1]
In fact, the integrality gap can be exactly 0.87856. You can make graphs arbitrarily close to that value. The rough idea of the graph is to throw a lot of random points, and we draw edges depending on whether two points lie within an angle θ. [2]
In later section, we get rid of ε by assuming that we can solve SDP exactly (keeping track of the errors).

Algorithm.

1. Solve SDP relaxation to get $v_1, \ldots, v_n$
2. Pick random vector g on sphere.
3. Set $x_i = sign(\langle g, v_i \rangle) \in \{-1, 1\}$.

□

Proof. Let v_i and g be unit vectors on a sphere sphere located at the origin. Then, $\langle v_i, v_j \rangle = \cos \theta_{i,j}$, where $\theta_{i,j}$ is the angle between v_i, v_j. Now, we want to show that $\frac{\mathbb{E}\, val(x)}{OPT(SDP)} \ge \alpha$, or equivalently:

$$\frac{\sum_{e \in E} P(e\ cut)}{\sum_{(i,j) \in E} \frac{1 - \cos \theta_{i,j}}{2}} \ge \alpha \quad (1)$$

Now, let's analyze what is $P(e\ cut)$ for $e = (i, j)$. We can consider the 2D plane that v_i, v_j span. Since iff e is cut, the sign of $\langle g, v_i \rangle$ will be different from that of $\langle g, v_j \rangle$, or equivalently here the sign of $\cos(\theta_{g,v_i})$ is different from that of $\cos(\theta_{g,v_j})$. By considering the geometric meaning, it's easy to see that $P(e\ cut) = \frac{2\theta_{i,j}}{2\pi} = \frac{\theta_{i,j}}{\pi}$. Now, simplifying, we have $\forall e$, $\frac{P(e\ cut)}{\frac{1 - \cos \theta_{i,j}}{2}} = \frac{2}{\pi} \frac{\theta_{i,j}}{1 - \cos \theta_{i,j}} \ge \alpha$, or $P(e\ cut) \ge \alpha(\frac{1 - \cos \theta_{i,j}}{2})$. Plugging in this inequality into the LHS of (1), we can easily see the inequality (1) follows. □

4 Approximation in Other Settings

4.1 Approx for Streaming Algos

1. See Sequence of items in stream (e.g. router sees seq of destination IPs in packets of routes)

2. Want to compute some function of input sequence.
3. Goal: use very little memory
4. Example:
   (a) stream is $i_1, i_2, \dots i_m, \forall j, i, j \in [n]$
   (b) f(stream) = number of distinct integers in stream.
   (c) Easy Solutions:
      i. n-bit vector
      ii. mlgn bits (remember entire input)
5. Can we solve the problem in much smaller than $min\{n, m\}$ memory using following kinds of algorithms?
   (a) Exact, Deterministic
   It is shown that it can't be done. See Claim 6.
   (b) Exact, Random
   It is shown that it can't be done. (Alon, Matias, Szegedy, JCSS'99)
   (c) Approx, Deterministic
   It is shown that it can't be done. (Alon, Matias, Szegedy, JCSS'99)
   (d) Approx, Random
   This can be done. See later section. [3].

Claim 6. Any exact.,det. alg needs to use at least $min\{n, m\}$ bits of space.

Proof. Encoding Argument.

1. If A is exact/det, using S bits memory, we will show $\exists$ injection from $\{0,1\}^n$ into $\{0,1\}^S$. Thus, we can prove our claim.
2. Encoding($x \in \{0,1\}^n$):
   (a) create a stream containing all i s.t. $x_i = 1$
   (b) run A on stream
   (c) output mem content of A
3. Now, we prove it's an injection by giving a decoding algorithm which is guaranteed to cover x. If M stands for mem footprint of A, we have Decoding(M):
   (a) init A with mem contents M
   (b) T $\leftarrow$ A.query(). Here, T = support size of x.
   (c) $x \leftarrow (1, 1, \dots, 1) \in \{0,1\}^n$
   (d) for $i = 1$ to n: 1.Tack on i to stream 2.If A.query() == T + 1: T ++, $x_i \leftarrow 0$. 3.Return x

□

Now, let's show the general idea of the Approx, Random Algorithm in (FM'85).

Algorithm. The idealized algorithm is following:

1. Pick a random hash function $h : [n] \to [0,1]$ (This is idealized)
2. Store in memory $z = \min_{\text{i in stream}} h(i)$
3. output $\frac{1}{z} - 1$.

The intuition is following: let's say $t =$ number of distinct elements. You expect the numbers are evenly spaced. The min number is thus expected to be $z = \frac{1}{t+1}$.
Expectation might not be reality. To decrease variance, we can come up with a better algorithm:

1. Pick $h_1, \ldots h_k : [n] \to [0,1]$ ($k = \theta(\frac{1}{\varepsilon^2})$)
2. Store $z_k = \min_{\text{i in stream}} h_k(i)$
3. Output $\frac{1}{\frac{1}{k}(\sum_i z_i)} - 1$

□

Now, let's analyze the above algorithm.

Claim 7. $\mathbb{E}(z) = \frac{1}{t+1}$.

Proof. $\mathbb{E}(z) = \int_0^1 P(z > x)dx = \int_0^1 (1-x)^t dx = \ldots = \frac{1}{t+1}$ □

We can use a similar idea to prove the following claim:

Claim 8. $\mathbb{E}(z^2) = \frac{2}{(t+1)(t+2)}$

Analysis. Now, let's continue our analysis. $\bar{z} = \frac{1}{k}\sum_{i=1}^{k} z_i$. By markov or chebyshev inequality, $P(|\bar{z} - \mathbb{E}(z)| > \varepsilon\,\mathbb{E}(\bar{z})) < \frac{1}{\varepsilon^2(\mathbb{E}(\bar{z}))^2} \cdot \mathbb{E}((\bar{z} - \mathbb{E}(\bar{z}))^2)$.
$\mathbb{E}((\bar{z} - \mathbb{E}(\bar{z}))^2) = Var[\bar{z}] = Var[\frac{1}{k}\sum_{i=1}^{k} z_i] = \frac{1}{k}\sum_{i=1}^{k} Var[z_i] = \frac{1}{k}Var[z] = \frac{1}{k}(\mathbb{E}(z^2) - (\mathbb{E}(z))^2)$.
Combining all, we have $P(|\bar{z} - \mathbb{E}\,\bar{z}| > \frac{\varepsilon}{t+1}) < \frac{(t+1)^2}{\varepsilon^2}\frac{1}{k}\theta(\frac{1}{t^2}) = \theta(\frac{1}{k\varepsilon^2}) < \frac{1}{3}$ for $k = \Omega(\frac{1}{\varepsilon^2})$. □

Now, let's talk about a non-idealized algorithm. [4]

Algorithm.
1. Pick one random hash function h : $[n] \to [0,1]$ from a 2-wise family.
2. Store the k smallest hash values $h(i)$ ever seen. $z_1 < z_2 < \ldots < z_k$, where $k = \theta(\frac{1}{\varepsilon^2})$.
3. Output: $\frac{k}{z_k} - 1$.

□

4.2 More Streaming Algorithms on Other Problems

1. Turnstile Streaming:

   (a) Maintain $x \in \mathbb{R}^n$ subject to updates of the form "$x_i \leftarrow x_i + \Delta$ ", $\Delta \in \mathbb{R}$.

   (b) For query(), you should output approximately $f(x)$.

   Examples of f:

   (a) $|supp(x)| = |\{i, x_i \neq 0\}|$

   (b) $\|x\|_2 = (\sum_i x_i^2)^{\frac{1}{2}}$

   (c) $f_i(x) = x_i$ (plus or minus some error)

   (d) freq items: $\{i : |x_i|\ large\}$

   More information about the l2 estimation:

   (a) We want $(1 \pm \varepsilon)\|x\|_2^2$

   (b) Algorithm: (AMS Sketch) [5]:
   We use "linear sketching".

      i. maintain $y = \pi x$ in memory. (π is a $m \times n$ matrix, where m is much smaller than n. $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$)

      ii. will estimate $f(x)$ using y.

   Let the ith column in π be π_i. Then, we know $x_i \leftarrow x_i + \Delta \Leftrightarrow x \leftarrow x + \Delta \cdot e_i$, so $y \leftarrow y + \Delta \cdot \pi e_i$ where $\pi e_i = \pi_i$.
   Now, the AMS Sketch is:

      i. $\sigma : [m] \times [n] \rightarrow \{-1, 1\}$ from 4-wise family

      ii. $\pi_{i,j} = \frac{\sigma(i,j)}{\sqrt(m)}$

      iii. will estimate $\|x\|_2^2$ by $\|y\|_2^2$.

   Claim 9. $m = \Omega(\frac{1}{\varepsilon^2}) \Rightarrow P_\sigma(|\|y\|_2^2 - \|x\|_2^2| > \varepsilon\|x\|_2^2) \leq \frac{1}{3}$

4.3 Next Time

We will show $\mathbb{E}\,\|y\|_2^2 = \|x\|_2^2$ and $Var[\|y\|_2^2] \leq \frac{2}{m} \cdot \|x\|_2^4 \Rightarrow P(|\|y\|_2^2 - \|x\|_2^2| > \varepsilon\|x\|_2^2) < \frac{1}{\varepsilon^2\|x\|_2^4}\frac{2}{m}\|x\|_2^4$

References

[1] Sanjeev Mahajan, H. Ramesh, Derandomizing Semidefinite Programming Based Approximation Algorithms, *Random Struct. Algorithms*, 20(3):403440, 2002.

[2] U. Feige and G. Schechtman. On the optimality of the random hyperplane rounding technique for max cut. *Random Struct. Algorithms*, 20(3):403440, 2002.

[3] Flajolet, Philippe; Martin, G. Nigel. Probabilistic counting algorithms for data base applications (PDF). *Journal of Computer and System Sciences.*, 1985.

[4] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. *Random* 2002.

[5] Noga Alon, Yossi Matias, and Mario Szegedy. The Space Complex- ity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.*, 58(1):137147, 1999.