CS 761: Randomized Algorithms Fall 2019

Lecture 10 — November 15, 2019

Prof. Gautam Kamath *By: Joshua McGrath, Kam Chuen Tung*
Edited by Vedat Levi Alev

Throughout the notes, $\|\cdot\|$ denotes the 2-norm.

1 Spectral Sparsification

We recall the definition of a cut approximator.

Definition 1. *(cut approximator) A graph $H = (V, E')$ is a ε-cut approximator of $G = (V, E)$ if $(1-\varepsilon)w_G(\delta(S)) \leq w_H(\delta(S)) \leq (1+\varepsilon)w_G(\delta(S))$ for any partition of the graph into $S, V \setminus S$.*

Spectral approximation is a generalization of cut approximation. First we will need to review some Linear Algebra and Graph Theory.

Definition 2. *(Positive Semidefinite Matrices) A matrix $M \in R^{n\times n}$ is positive semidefinite if for any $x \in R^n$, $x^\top Mx \geq 0$.*

We denote a positive semidefinite matrix M by writing $M \succcurlyeq 0$.

Definition 3. *(Graph Laplacian) Given a weighted graph $G = (V, E, w)$ we define the Laplacian matrix of G as*

$$L = D - A$$

Where A is the weighted adjacency matrix of G and D is the diagonal matrix with $D_i = \sum_{k=1}^{n} w(i,k)$

Observation 4. *For any graph G its Laplacian L_G is positive semidefinite.*

Proof. Begin by decomposing $L_G = \sum_{e\in E} L_e$ where L_e is the Laplacian of the graph G modified to have only the edge $e = (i, j)$. $L_e = b_e b_e^\top$ where b_e is 0 everywhere except entries i, j where b_e is $-\sqrt{w(i,j)}$ and $\sqrt{w(i,j)}$ respectively. Using this decomposition of the matrix we consider $x^\top L_G x$

$$\begin{aligned} x^\top L_G x &= x^\top (\sum L_e)x \\ &= \sum x^\top b_e b_e^\top x \\ &= \sum \left(x^\top b_e\right)^2 \quad = \sum w(i,j)(x_i - x_j)^2 \geq 0 \end{aligned}$$

Therefore L_G is positive semidefinite.

□

Given these definitions we generalize the idea of a cut approximator.

Definition 5. *(Spectral Approximator) A graph H is a ε-spectral approximator of G for error paramter $\varepsilon \leq 1$*

$$(1-\varepsilon)L_G \preccurlyeq L_H \preccurlyeq (1+\varepsilon)L_G$$

or equivalently, for all $x \in R^n$

$$(1-\varepsilon)x^\top L_G x \leq x^\top L_H x \leq (1+\varepsilon)x^\top L_G x$$

Fact 6. *Any spectral approximator is also a cut approximator*

Proof. Let $x_S(i) = 1$ if $i \in S$ and 0 otherwise for any $S \subseteq V$. Then

$$x_S^\top L_G x_S = \sum_{(i,j)\in E} w(i,j)(x_S(i) - x_S(j))^2 = \sum_{(i,s)\in E(S)} w(i,j)(x_S(i) - x_S(j))^2 = W_G(\delta(S))$$

Where $E(S)$ is the edges of G with exactly one endpoint in S. Then any graph which satisfies the spectral condition satisfies the cut approximator condition, because we asked for the spectral condition to hold for any $x \in R^n$. □

We can now prove the following Theorem from Benczur and Karger ([3]):

Theorem 7. *Given a graph G and an error parameter $\varepsilon \leq 1$ there exists an ε-cut approximation G' which has $O(n \log n/\varepsilon^2)$ edges.*

We will prove this by the following reduction. Suppose we have vectors $v_1, \dots, v_m \in \mathbb{R}^n$ such that $\sum\limits_{i=1}^{m} v_i v_i^\top = I_n$, then there exists $s_1, \dots s_m \in \mathbb{R}$ with $O\left(n \log n/\varepsilon^2\right)$ non-zero entries such that

$$(1-\varepsilon)I_n \leq \sum_{i=1}^{m} s_i v_i v_i^\top \leq (1+\varepsilon)I_n$$

Given any matrix M we can represent it by its eigendecomposition $M = \sum\limits_{i=1}^{n} \lambda_i u_i u_i^\top$ and then we can define the pseudoinverse of M as $M^\dagger = \sum\limits_{i=1}^{n} \frac{1}{\lambda_i} u_i u_i^\top$. Finally we can represent I_n in terms of the laplacian of a graph by the equation

$$\begin{aligned} I &= L_G^{-1/2} L_G L_G^{-1/2} \\ &= \sum_{e\in E} \left(L_G^{-1/2} b_e\right)\left(b_e^\top L_G^{-1/2}\right) \\ &= \sum_{e\in E} v_e v_e^\top, \end{aligned}$$

where $v_e = L_G^{-1/2} b_e$.

We can now solve the problem with sampling, where we sample non-uniformly in order to preserve I_n. Pick the vector v_i with probability $p = \|v_i\|^2$, if it is chosen set $s_i = \frac{1}{\|v_i\|^2}$ else set $s_i = 0$. We then have that in $E\left[s_i v_i v_i^\top\right] = \frac{v_i v_i^\top}{\|v_i\|^2} \cdot \Pr(v_i \text{ is chosen}) = v_i v_i^\top$. Hence by linearity of expectation we have the correct sum, I_n. We will do it many rounds so that, using the Chernoff bound we can obtain tight concentration for $\sum_i s_i v_i v_i^\top$ and that $O(n \log n/\varepsilon^2)$ of the s_i are nonzero. Because only the smaller vectors have been removed for the summation we can show we have tight concentration.

Now we describe the algorithm for finding s_i's in detail.

1. Set $\mathcal{F} := \emptyset, s_i := 0$.
2. Repeat $C = 6 \log n/\varepsilon^2$ times:
   - For $i = 1..m$, choose v_i w.p. $\|v_i\|^2$.
   - In case v_i is chosen, set $F := F \cup \{i\}$ and increase s_i by $\frac{1}{C\|v_i\|^2}$.

To analyse the algorithm, we prove the following two lemmas:

Lemma 8. $|\mathcal{F}| = O(\frac{n \log n}{\varepsilon^2})$ *with probability* ≥ 0.9.

Proof. We upper-bound $\mathbb{E}[|\mathcal{F}|]$ by $C \cdot \sum_i Pr[v_i \text{ is chosen}]$. Computing the latter boils down to calculating $\sum_i \|v_i\|^2$. Since

$$\begin{aligned}\sum_i \|v_i\|^2 &= \sum_i v_i^\top v_i \\ &= \sum_i \operatorname{tr}(v_i^\top v_i) \\ &= \sum_i \operatorname{tr}(v_i v_i^\top) \\ &= \operatorname{tr}(\sum_i v_i v_i^\top) \\ &= \operatorname{tr}(I_n) \\ &= n,\end{aligned}$$

it follows that $\mathbb{E}[|\mathcal{F}|] \leq Cn = O(n \log n/\varepsilon^2)$. The lemma follows by Markov's inequality. ☐

Lemma 9. $(1-\varepsilon)I_n \preceq \sum_i s_i v_i v_i^\top \preceq (1+\varepsilon)I_n$ *with high probability.*

To prove this lemma, we need to introduce Matrix Chernoff bound. Below is the statement we need.

Lemma 10. *Given random independent (symmetric)* $n \times n$ *matrices* $M_1, \ldots, M_k$ *with* $0 \preceq M_i \preceq R \cdot I_n$. *Suppose* $\mu_{\min} \cdot I_n \preceq \sum_i \mathbb{E}[M_i] \preceq \mu_{\max} \cdot I_n$. *Then, for* $0 \leq \varepsilon \leq 1$,

$$Pr\left[\lambda_{\max}(\sum_i M_i) \geq \mu_{\max} + \varepsilon\right] \leq n \cdot \exp\left(\frac{-\varepsilon^2 \cdot \mu_{\max}}{3R}\right)$$

and

$$Pr\left[\lambda_{\min}(\sum_i M_i) \leq \mu_{\min} - \varepsilon\right] \leq n \cdot \exp\left(\frac{-\varepsilon^2 \cdot \mu_{\min}}{2R}\right).$$

For proof, refer to standard references on matrix Chernoff bound, for example [1] and [5] (Section 5). Now we return to proving the lemma.

Proof. Apply matrix Chernoff bound on the random matrices $M_{i,j}$, with $i \in [C]$ and $j \in [m]$. $M_{i,j}$ equals $v_j v_j^\top / C \|v_j\|^2$ if v_j is chosen in round i, and equals 0 otherwise. Then,

$$\begin{aligned}\textstyle\sum_{i,j} \mathbb{E}[M_{i,j}] &= \textstyle\sum_{i,j} \|v_j\|^2 \times (v_j v_j^\top / C \|v_j\|^2) \\ &= C \times \textstyle\sum_j (v_j v_j^\top / C) \\ &= \textstyle\sum_j (v_j v_j^\top) \\ &= I_n.\end{aligned}$$

Note also that $0 \preceq v_j v_j^\top \preceq I_n$, and so $0 \preceq M_{i,j} \preceq \frac{1}{C} \cdot I_n$. Apply matrix Chernoff bound with $\mu_{\min} = \mu_{\max} = 1$ and $R = 1/C$, we have

$$\begin{aligned}Pr[(1-\varepsilon)I_n \preceq \textstyle\sum_i M_i \preceq (1+\varepsilon)I_n] &\geq 1 - n \cdot \exp\left(\frac{-\varepsilon^2}{3R}\right) - n \cdot \exp\left(\frac{-\varepsilon^2}{2R}\right) \\ &= 1 - n \cdot n^{-2} - n \cdot n^{-3} \\ &\geq 1 - 2/n.\end{aligned}$$

The proof follows by noting that s_i's are defined exactly so that

$$\sum_{i,j} M_{i,j} = \sum_i s_i v_i v_i^\top.$$

□

Therefore, the algorithm works as desired.

Remark 11. *For construction of graph spectral sparsifier with $O(n/\varepsilon^2)$ edges, refer to [2].*

2 Faster Linear Algebra

The least squares problem is: given a matrix $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^d$, we want to find $x = \arg\min_{x \in \mathbb{R}^d} \|Ax - b\|$. A textbook solution takes $\Omega(n \operatorname{poly}(d))$. We will analyze an $\tilde{O}\left(nd + \operatorname{poly}(d/\varepsilon)\right)$ solution using the sketch-and solve paradigm. First we will compress A into $A' \in \mathbb{R}^{k \times d}$ where $k = \operatorname{poly}(d/\varepsilon)$. This compression will make use of the subspace embedding.

Definition 12. *(Subspace Embedding) A ε-ℓ_2 subspace embedding for the column space of $A \in \mathbb{R}^{n \times d}$ is a matrix S such that*

$$(1-\varepsilon)\|Ax\|^2 \leq \|(SA)x\|^2 \leq (1+\varepsilon)\|Ax\|^2$$

for all $x \in \mathbb{R}^d$. Here

Suppose $S \in \mathbb{R}^{k \times n}$ is a subspace embedding (and sketch matrix) where k is $\operatorname{poly}(d/\varepsilon)$. Then we can solve least squares in $O(\operatorname{poly}(d/\varepsilon))$ time (and the projection takes $O(nd)$ time). If x is the optimal solution for the sketched least squares and x^* in the optimal solution for the original least squares. Then by definition of the subspace embedding we have that

$$\|Ax - b\| \leq \frac{1}{1-\varepsilon}\|S(Ax-b)\| \leq \frac{1+\varepsilon}{1-\varepsilon}\|Ax^* - b\|$$

Now we need to find a cover, which is basically a ε-lattice over the unit ball in $\mathbb{R}^d$. We can find a $O(d/\varepsilon^2)$ embedding using the Johnson-Lindenstrauss Lemma.

3 Algebraic Method

Certain tasks involve checking whether two objects are identical. The algebraic method consists of three steps:

1. Construct polynomials (with coefficients in finite field $\mathbb{F}$) from the two objects,
2. Evaluate the polynomials at random points, and
3. Compare their values.

Consider the following example. Alice and Bob each has one binary string of length d. Call Alice's string $a = a_1 a_2 \dots a_d$ and Bob's string $b = b_1 b_2 \dots b_d$. They want to determine whether $a = b$ while minimizing communication.

Fact 13. *For any deterministic algorithm, they must communicate $\Omega(d)$ bits.*

Interestingly, there exists randomized algorithm that takes $O(\log d)$ bits of communication.

Alice and Bob first agree on a finite field $\mathbb{F} = \mathbb{F}_p$ for a large prime p. Alice forms the polynomial $A(x) := \sum_{i=1}^{d} a_i x^i \in \mathbb{F}[x]$ and Bob $B(x) := \sum_{i=1}^{d} b_i x^i \in \mathbb{F}[x]$. Alice chooses random $r \in \mathbb{F}$, computes r and $A(r)$, then sends r and $A(r)$ to Bob. Bob in turn computes $B(r)$ and tells Alice whether $A(r) = B(r)$.

To evaluate the algorithm, observe that:

Observation 14. *Each round of communication takes $O(\log p)$ bits.*

Observation 15. *If $A \neq B$, then $(A - B)$ is a non-zero polynomial in $F[x]$ having degree $\leq d$. So $(A - B)$ has at most d roots. That means the probability that $A(r) = B(r)$ is at most d/p.*

Suppose we choose $p = \Theta(d^2)$ (this is possible, because there exists prime number in $[n, 2n]$ for any $n > 0$). Then, it takes $O(\log d^2) = O(\log d)$ bits per round of communication, with a high probability $1 - O(1/d)$ of proving that $A \neq B$.

One advantage of using polynomials is that the number of zeros can be controlled by its degree. This is made precise in the following lemma:

Lemma 16. *(Schwartz-Zippel) Given a multivariate polynomial $Q \in \mathbb{F}[x_1, \dots, x_n]$ with $\deg Q \leq d$ (that is, for each monomial $x_1^{d_1} x_2^{d_2} \dots x_n^{d_n}$, one has $\sum_i d_i \leq d$). Fix $S \subseteq \mathbb{F}$ and choose $r_1, \dots, r_n$ uniformly at random from S. Then, if $Q \neq 0$,*

$$Pr[Q(r_1, \dots, r_n) = 0] \leq \frac{d}{|S|}.$$

Proof. The proof proceeds by induction on n.

(Base case: $n = 1$) A degree-d univariate polynomial has at most d roots, so the probability that the chosen number r_1 is a root of Q is at most $d/|S|$.

(Induction step) Let's write the polynomial Q as

$$\sum_{i=0}^{k} Q_i(x_2, \ldots, x_n) x_1^i$$

with $k \leq d$.

There are two cases: $Q_k(r_2, \ldots, r_n) = 0$ and $Q_k(r_2, \ldots, r_n) \neq 0$. In the second case, once the values $r_2, \ldots, r_n$ are fixed, we can consider Q to be a univariate polynomial in x_1, of degree k.

Note that Q_k is a $(n-1)$-variate polynomial of degree at most $d-k$. Therefore, (writing r for $(r_1, \ldots, r_n)$ and r_{-1} for $(r_2, \ldots, r_n)$ as shorthand),

$$\begin{aligned} & Pr[Q(r) = 0] \\ = \; & Pr[Q(r) = 0 \,|\, Q_k(r_{-1}) = 0] \cdot Pr[Q_k(r_{-1}) = 0] + \\ & Pr[Q(r) = 0 \,|\, Q_k(r_{-1}) \neq 0] \cdot Pr[Q_k(r_{-1}) \neq 0] \\ \leq \; & 1 \cdot (d-k)/|S| + k/|S| \cdot 1 \\ = \; & d/|S|. \end{aligned}$$

□

Using the algebraic method, we can design an efficient algorithm for determining if a bipartite graph admits a perfect matching. Given a bipartite graph G with node set $U \cup V$, $|U| = |V| = n$ and edge set $E \subseteq U \times V$. We can build the matrix $A = (a_{ij})$, where $a_{ij} = x_{ij}$ if $(i,j) \in E$ and $a_{ij} = 0$ otherwise. Then,

Observation 17. $\det(A)$ *can be regarded as a polynomial in* $\mathbb{F}[x_{11}, x_{12}, \ldots, x_{nn}]$ *(n^2 variables in total) and of degree* $\leq n$*. Moreover,* G *admits a perfect matching if and only if* $\det(A)$ *is not the zero polynomial.*

The latter follows from the formula

$$\det(A) = \sum_{\sigma \in S_n} (-1)^{sgn(\sigma)} \prod_i a_{i\sigma(i)},$$

that each σ *corresponds to a potential matching between* U *and* V*, and that the monomials will not cancel out each other.*

Then, we have a simple algorithm: choose $\mathbb{F} = \mathbb{F}_p$. For each x_{ij}, randomly assign to it a value in $\mathbb{F}$, then compute the determinant of the (scalar) matrix $\bar{A}$.

For $p = \Theta(n^2)$, the probability of false negative ($\det(A) \neq 0$ as polynomial but $\det(\bar{A}) = 0$) is $1/n$.

Using Gaussian elimination, this gives an $O(n^3)$ algorithm. An $O(n^\omega)$ algorithm also exists, using a technique called "blockwise inversion" to attempt to compute $\bar{A}^{-1}$ and, hence, determine whether it is singular.

(If we account for the cost of doing arithmetic in $\mathbb{F}$, then there is an additional polylog(n) factor.)

This is not good enough, as often we would like to find a perfect matching if it exists. The idea of "self-reducibility" is helpful here: remove an edge and check if the resultant graph still has a perfect matching. If yes, remove that edge. If no, add that edge to the matching and remove the two nodes that the edge connects. This has a time complexity of $O(m \cdot n^\omega)$.

We can do better than this, with the following two tools from linear algebra:

Fact 18. *For invertible $n \times n$ matrix M, let $c_{i,j}$ be the (i,j)-cofactor of M, i.e. $c_{i,j} := (-1)^{i+j} \det(M_{-i,-j})$. (We use the shorthand $M_{-i,-j}$ to denote the $(n-1) \times (n-1)$ matrix which is formed by removing the i-th row and the j-th column of from M.) Then,*

$$(M^{-1})_{ji} = \frac{c_{i,j}}{\det(M)}.$$

Now put $M := \bar{A}$. The significance is that, for edge (i,j), it can be included in the perfect matching if $c_{i,j} \neq 0$. This gives an $O(n(n^\omega + m)) = O(n^{\omega+1})$ algorithm, where $O(n^\omega)$ is the time for matrix inversion and $O(m)$ is the time to find an edge to include in the matching; note that we need to do both steps n times.

To achieve an even better time complexity, we use the following formula.

Fact 19. *(Fast rank-1 update) Using the Sherman-Morrison formula, we can compute $(\bar{A}+uv^\top)^{-1}$, given $\bar{A}^{-1}$, in $O(n^2)$ time. Let $\bar{B} := \bar{A}_{-i,-j}$. Then one can form*

$$\begin{bmatrix} 1 & 0 \\ 0 & \bar{B} \end{bmatrix}$$

from $\bar{A}$ using a constant number of rank-1 updates.

This allows us to compute $\bar{B}^{-1}$ from $\bar{A}^{-1}$ with $O(n^2)$ time instead of $O(n^\omega)$ time, bringing the overall time complexity down to $O(n^\omega + n \times n^2) = O(n^3)$.

Remark 20. *An $O(n^\omega)$ algorithm exists; see, for example, [4].*

References

[1] Rudolf Ahlswede and Andreas Winter. Strong converse for identification via quantum channels. *IEEE Transactions on Information Theory*, 48(3):569–579, 2002.

[2] Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.

[3] András A Benczúr and David R Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM Journal on Computing*, 44(2):290–319, 2015.

[4] Marcin Mucha and Piotr Sankowski. Maximum matchings via gaussian elimination. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–255. IEEE, 2004.

[5] Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.