ORIE 6334 Bridging Continuous and Discrete Optimization Nov 20, 2019

Lecture 20

Lecturer: David P. Williamson *Scribe: J. Carlos Martinez*

1 Recap

In the previous lectures, we explored the problem of finding spectral sparsifiers and saw both deterministic and randomized algorithms that, given a graph G with n vertices, find a spectral sparsifier with $O\left(\frac{n \log n}{\epsilon^2}\right)$ edges. In this lecture[1], we improve upon this result to find linear-sized spectral sparsifiers, or more concretely, spectral sparsifiers with $O\left(\frac{n}{\epsilon^2}\right)$ edges. This result is due to Batson, Spielman, and Srivastava [2].

This result is interesting since since we noted in the past that a spectral sparsifier is a generalization of the cut sparsifier of a graph. The best results known results for cut sparsifiers (prior to this one) were that a cut sparsifier with $(n \log^{O(1)} n)/\epsilon^2$ edges can be found in nearly linear time [1].

2 The Main Result

Recall that given a graph $G = (V, E)$, a weighted graph $H = (V, E')$ with weights $w(i, j)$ is an ϵ-spectral sparsifier of G if

$$(1 - \epsilon)L_G \preceq L_H \preceq (1 + \epsilon)L_G.$$

In the last lecture we showed that

$$(1 - \epsilon)L_G \preceq \sum_{(i,j)\in E} w(i, j)(e_i - e_j)(e_i - e_j)^T \preceq (1 + \epsilon)L_G$$

if and only if

$$(1 - \epsilon)I \preceq \sum_{(i,j)\in E} w(i, j)x(i, j)x(i, j)^T \preceq (1 + \epsilon)I$$

for $x(i, j) = L_G^{\dagger/2}(e_i - e_j)$ and $\sum_{(i,j)\in E} x(i, j)x(i, j)^T = I^*$. Recall that I^* is our continuing fudge of an identity matrix, which is actually $L_G L_G^{\dagger}$. For any vector v such that $v^T e = 0$, $I^* v = v$.

[1]This lecture is derived from Spielman 2015, Lecture 18, which can be found at `http://www.cs.yale.edu/homes/spielman/561/lect18-15.pdf`. These notes are partially based on Shijin Rajakrishnan's notes from the 2016 iteration of this class.

The goal is to show that given $v_1, \cdots, v_m$ such that $\sum_{i\in[m]} v_i v_i^T = I$ (i.e., the vectors are in *isotropic position*), there exists $S \subseteq [m]$ and weights $w_i \geq 0$ such that $|S| = \lceil n/\epsilon^2 \rceil$ and

$$(1-\epsilon)^2 \preceq \sum_{i\in S} w_i v_i v_i^T \preceq (1+\epsilon)^2 I.$$

For any symmetric A and vectors $v_1, \cdots, v_m$ in isotropic position we have

$$\sum_{i=1}^m v_i^T A v_i = \sum_{i=1}^m (v_i v_i^T) \cdot A = \left(\sum_{i=1}^m v_i v_i^T\right) \cdot A = I \cdot A = \mathrm{tr}(A).$$

In this lecture, we prove a weaker version of the theorem first and then mention how to extend it to the general case. The version we prove is as follows.

Theorem 1 *Given vectors $v_1, \cdots, v_m$ in isotropic position, we can find a subset $S \subseteq [m]$ and weights $w_i \geq 0$ such that $|S| \leq 6n$ and*

$$\frac{1}{\sqrt{13n}} I \preceq \frac{1}{\sqrt{13n}} \sum_{i\in S} w_i v_i v_i^T \preceq \sqrt{13} I.$$

3 The Algorithm

The basic idea behind the algorithm is that we greedily pick vectors and weights such that we control how the maximum and minimum eigenvalues change.

Algorithm 1: Linear sized spectral sparsifier.

$l \leftarrow -n$;
$u \leftarrow n$;
$\Delta l \leftarrow 1/3$;
$\Delta u \leftarrow 2$;
$w_i \leftarrow 0$ for all i;
$A \leftarrow 0$;
for $k = 1, \cdots, 6n$ **do**
  Pick v_i, c such that $\lambda_{\min}(A + c v_i v_i^T) \geq l + \Delta l$ and
   $\lambda_{\max}(A + c v_i v_i^T) \leq u + \Delta u$;
  $w_i \leftarrow w_i + c$;
  $A \leftarrow A + c v_i v_i^T$;
  $l \leftarrow l + \Delta l$;
  $u \leftarrow u + \Delta u$;
end
return A

Initially, $A = 0$, $\lambda_{\min}(0) = 0 \geq -n$ and $\lambda_{\max}(0) = 0 \leq n$. At termination, $\lambda_{\min}(A) \geq -n + 1/3(6n) = n$ and $\lambda_{\max}(A) \leq n + 2(6n) = 13n$. We scale down the matrix by $\sqrt{13}n$, which gives the result.

4 Barrier Functions

We will not work with $\lambda_{\max}$ and $\lambda_{\min}$ directly. Rather, we will use *barrier functions*[2]. Let

$$U(A, u) = \sum_{i=1}^{n} \frac{1}{u - \lambda_i} = \text{tr}((uI - A)^{-1})$$

and

$$L(A, l) = \sum_{i=1}^{n} \frac{1}{\lambda_i - l} = \text{tr}((A - lI)^{-1})$$

for $l \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n \leq u$, where λ_i is the ith eigenvalue of A.

Initially, $l_0 \leq \lambda_{\min}(A_0) \leq \lambda_{\max}(A_0) \leq u_0$. Then, $U(A_0, u_0)$ and $L(A_0, l_0)$ are positive and bounded. We will make sure the barrier functions do not increase as the algorithm progresses. We also need to make sure that the barrier functions don't go through the discontinuities at $u = \lambda_{\max}$ and $l = \lambda_{\min}$. Then, $l \leq \lambda_{\min}(A) \leq \lambda_{\max}(A) \leq u$ throughout. Clearly,

$$U(A_0, u_0) = \sum_{i=1}^{n} \frac{1}{n - 0} = 1$$

and

$$L(A_0, l_0) = \sum_{i=1}^{n} \frac{1}{0 - (-n)} = 1.$$

In each iteration, we want to find c, v_i in each step such that

$$U(A + cv_i v_i^T, u + \Delta u) \leq U(A, u) \leq 1$$

and

$$L(A + cv_i v_i^T, l + \Delta l) \leq L(A, l) \leq 1,$$

so that the barrier functions do not increase as we update A, u, l through the run of the algorithm. We also need to make sure, as pointed out, that the barrier functions don't go through the discontinuities. However, we will focus on making sure that the functions do not increase.

[2]In the original paper due to Batson, Spielman, and Srivastava, as well as other notes on their result, the notation $\Phi^u(A)$ is used for $U(a, A)$ and $\Phi_l(A)$ is used for $L(l, A)$.

5 The Analysis

We now turn towards proving that such a selection of v and c can be made in each iteration of the loop. We will show that there exists matrices U_A and L_A such that the following two lemmas hold.

Lemma 2 *There are matrices U_A, L_A such that the barrier functions do not increase in an iteration. Namely, the barrier functions do not increase for c, v if $v^T U_A v \leq \frac{1}{c} \leq v^T L_A v$.*

Lemma 3 *For the choices of U_A, L_A from Lemma 2,*

$$\sum_{i=1}^{m} v_i^T U_A v_i \leq \frac{1}{\Delta u} + U(A,u) \leq \frac{1}{\Delta l} - \frac{1}{1/L(A,l) - \Delta l} \leq \sum_{i=1}^{m} v_i^T A v_i.$$

From our choice of parameters we obtain

$$\frac{1}{\Delta u} + U(A,u) \leq \frac{1}{2} + 1 = \frac{3}{2}$$

and

$$\frac{1}{\Delta l} - \frac{1}{1/L(A,l) - \Delta l} \leq 3 - \frac{1}{1 - 1/3} = 3 - \frac{3}{2} = \frac{3}{2}.$$

Then, by Lemma 3 we have

$$\sum_{i=1}^{m} v_i^T U_A v_i \leq \frac{3}{2} \leq \sum_{i=1}^{m} v_i^T A v_i.$$

This implies that $\exists v_i, c$ such that

$$v_i^T U_A v_i \leq \frac{1}{c} \leq v_i^T A v_i.$$

Then, Lemma 2 ensures that there exists a vector v_i and weight c so that we can add $c v_i v_i^T$ to A without increasing the barrier functions. It only remains to prove the two lemmas.

To prove Lemma 2, we first analyze what the addition of cvv^T does to the matrix A, by using the following formula.

Theorem 4 (Sherman-Morrison formula) *For X non-singular and symmetric, and a vector v,*

$$(X - vv^T)^{-1} = X^{-1} + \frac{X^{-1} v v^T X^{-1}}{1 - v^T X^{-1} v}.$$

The formula expresses a rank-1 update to the inverse of a matrix. We now prove Lemma 2.

Proof:

Note how $U(A,u)$ changes as we add cvv^T to A. Namely,

$$\begin{aligned}
U(A+cvv^T,u) &= \operatorname{tr}\left((uI-A-cvv^T)^{-1}\right) \\
&= \operatorname{tr}\left((uI-A)^{-1}\right) + c\frac{\operatorname{tr}\left((uI-A)^{-1}vv^T(uI-A)^{-1}\right)}{1-cv^T(uI-A)^{-1}v} \\
&= U(A,u) + c\frac{\operatorname{tr}\left((uI-A)^{-1}vv^T(uI-A)^{-1}\right)}{1-cv^T(uI-A)^{-1}v} \\
&= U(A,u) + c\frac{v^T(uI-A)^{-2}v}{1-cv^T(uI-A)^{-1}v}.
\end{aligned}$$

The second inequality follows from the Sherman-Morrison formula, the third by the definition of $U(u,A)$, and the fourth by the cyclic property of the trace.

So as we add cvv^T, the barrier function increases. To counteract this, we increase the value of u so to keep the barrier function constant. Let $\hat{u} = u + \Delta u$. Then,

$$\begin{aligned}
U(A+cvv^T,\hat{u}) &= U(A,\hat{u}) + c\frac{v^T(\hat{u}I-A)^{-2}v}{1-cv^T(\hat{u}I-A)^{-1}v} \\
&= U(A,u) - (U(A,u)-U(A,\hat{u})) + c\frac{v^T(\hat{u}I-A)^{-2}v}{1-cv^T(\hat{u}I-A)^{-1}v}.
\end{aligned}$$

We want that $U(A+cvv^T,\hat{u}) \leq U(A,u)$, so we want that $U(A,u)-U(A,\hat{u}) \geq c\frac{v^T(\hat{u}I-A)^{-2}v}{1-cv^T(\hat{u}I-A)^{-1}v}$. In turn, this means that we want

$$\begin{aligned}
\frac{1}{c} &\geq \frac{v^T(\hat{u}I-A)^{-2}v}{U(A,u)-U(A,\hat{u})} + v^T(\hat{u}I-A)^{-1}v \\
&= v^T U_A v,
\end{aligned}$$

where we let

$$U_A = \frac{(\hat{u}I-A)^{-2}}{U(A,u)-U(A,\hat{u})} + (\hat{u}I-A)^{-1}.$$

The proof for L is similar with

$$L_A = \frac{(A-\hat{l}I)^{-2}}{L(A,\hat{l})-L(A,l)} - (A-\hat{l}I)^{-1}.$$

□

(Extended aside: Although we didn't cover this in lecture, the following argument was provided by Jason Gaitonde about why the barrier functions don't cross their discontinuous points. We show above that

$$U(A + cvv^T, \hat{u},) = U(A, \hat{u}) + \frac{cv^T(\hat{u}I - A)^{-2}v}{1 - cv^T(\hat{u}I - A)^{-1}v}.$$

As a function of c for a given v, the discontinuity happens when the denominator hits zero. Thus, the discontinuity happens when

$$1/c = v^T(\hat{u}I - A)^{-1}v.$$

But $U_A \succ (\hat{u}I - A)^{-1}$ as it is the sum of that and another positive definite matrix, so

$$v_i^T(\hat{u}I - A)^{-1}v_i < v_i U_A v_i \leq 1/c.$$

Thus as you vary c from 0 to the value we found, you never cross the pole, or equivalently the largest eigenvalue doesn't cross $\hat{u}$.

The reason $L(A, l)$ doesn't have continuity issues is actually easier and for a different reason. Inductively, $L(A, l) \leq 1$; in particular, this implies that $\frac{1}{\lambda_{\min}(A) - l} \leq 1$, or equivalently, $\lambda_{\min}(A) - l \geq 1$ (the inequality doesn't flip as $\lambda_{\min}(A) - l > 0$). But $\hat{l} = l + 1/3$ in the algorithm, so $\lambda_{\min}(A) - \hat{l} \geq 2/3$, so as a function of l, updating l does not cause you to cross the pole at $\lambda_{\min}(A)$. The point is the Δl jumps are chosen cleverly enough to ensure that it doesn't cause the lower barrier to jump the smallest eigenvalue, even without updating by cvv^T. Updating A to $A + cvv^T$ only increases the eigenvalues for all nonnegative c, so you move in the right direction with no potential continuity issue in the potential function. Without this continuity issue, the choice of c then bumps down $L(A + cvv^T, \hat{l})$ by enough to retain the invariant that the barrier be at most 1, so you'll be safe to update l in the same way in the next step, and so on.)

We now turn to the proof of Lemma 3.

Proof: To bound the first term, note that

$$\frac{d}{du}U(A, u) = \frac{d}{du}\sum_{i=1}^{n}\frac{1}{u - \lambda_i} = -\sum_{i=1}^{n}\frac{1}{(u - \lambda_i)^2} = -\mathrm{tr}((uI - A)^{-2})$$

and so the function is decreasing. Since the vectors v_i are in isotropic position, we then get that

$$\begin{aligned}\sum_{i=1}^{m} v_i^T U_A v_i &= \mathrm{tr}(U_A) \\ &= \frac{\mathrm{tr}((\hat{u}I - A)^{-2})}{U(A, u) - U(A, \hat{u})} + \mathrm{tr}((\hat{u}I - A)^{-1}) \\ &\leq \frac{\mathrm{tr}((\hat{u}I - A)^{-2})}{U(u, A) - U(\hat{u}, A)} + U(u, A).\end{aligned}$$

Likewise,

$$\frac{d^2}{du^2}U(A,u) = 2\sum_{i=1}^{n}\frac{1}{(u-\lambda_i)^3} > 0$$

and so the function is convex. In particular, convexity implies that

$$U(u,A) - U(\hat{u},A) \geq (-\Delta u)\frac{d}{du}U(\hat{u},A) = \Delta u \cdot \mathrm{tr}((\hat{u}I - A)^{-2}),$$

which yields

$$\frac{\mathrm{tr}((\hat{u}I - A)^{-2})}{U(u,A) - U(\hat{u},A)} \leq \frac{1}{\Delta u}.$$

Putting everything together we get that

$$\sum_{i=1}^{m} v_i^T U_A v_i \leq \frac{\mathrm{tr}((\hat{u}I - A)^{-2})}{U(u,A) - U(\hat{u},A)} + U(u,A) \leq U(A,u) + \frac{1}{\Delta u}.$$

The proof of the lower bound is similar, but slightly uglier and more involved. While we did not cover it in class, we include it here for completeness. In this case, we have

$$\begin{aligned}\sum_{i=1}^{m} \mathbf{v_i}^T L_A \mathbf{v_i} &= \mathrm{tr}(L_A) \\ &= \frac{\mathrm{tr}((A - \hat{l}I)^{-2})}{L(\hat{l},A) - L(l,A)} - \mathrm{tr}((A - \hat{l}I)^{-1}) \\ &= \frac{\mathrm{tr}((A - \hat{l}I)^{-2})}{L(A,\hat{l}) - L(A,l)} - L(A,\hat{l}).\end{aligned}$$

Now,

$$\frac{d}{dl}L(A,l) = \frac{d}{dl}\sum_{i=1}^{n}\frac{1}{\lambda_i - l} = \sum_{i=1}^{n}\frac{1}{(\lambda_i - l)^2} = \mathrm{tr}((A - lI)^{-2}),$$

and

$$\frac{d^2}{dl^2}L(A,l) = 2\sum_{i=1}^{n}\frac{1}{(\lambda_i - l)^3} > 0,$$

and thus $L(A,l)$ is increasing and convex in l. Then by convexity,

$$L(A,l) - L(A,\hat{l}) \geq -\Delta l\,\frac{d}{dl}L(A,\hat{l}) \geq -\Delta l\ \mathrm{tr}((A - \hat{l}I)^{-2}).$$

Rearranging terms, we get

$$\frac{\mathrm{tr}((A-\hat{l}I)^{-2})}{L(A,\hat{l})-L(A,l)} \geq \frac{1}{\Delta l}.$$

To bound the second term, we claim that

$$\frac{L(A,\hat{l})-L(A,l)}{\Delta l} \leq L(A,l)L(A,\hat{l}).$$

If the claim is true, then by rearranging terms we get

$$L(A,\hat{l}) \leq \frac{1}{1/L(A,l)-\Delta l},$$

as desired. To prove the claim we observe that

$$\begin{aligned}
\frac{L(A,\hat{l})-L(A,l)}{\Delta l} &= \frac{1}{\Delta l}\sum_{i=1}^{n}\left[\frac{1}{\lambda_i-\hat{l}}-\frac{1}{\lambda_i-l}\right] \\
&= \frac{1}{\Delta l}\sum_{i=1}^{n}\left[\frac{\lambda_i-l-(\lambda_i-\hat{l})}{(\lambda_i-l)(\lambda_i-\hat{l})}\right] \\
&= \frac{1}{\Delta l}\sum_{i=1}^{n}\left[\frac{\Delta l}{(\lambda_i-l)(\lambda_i-\hat{l})}\right] \\
&= \sum_{i=1}^{n}\left[\frac{1}{(\lambda_i-l)(\lambda_i-\hat{l})}\right] \\
&\leq \left[\sum_{i=1}^{n}\frac{1}{\lambda_i-l}\right]\left[\sum_{i=1}^{n}\frac{1}{\lambda_i-\hat{l}}\right] = L(A,l)L(A,\hat{l}),
\end{aligned}$$

as long as $l \leq \lambda_{\min}$, as we have been guaranteeing. □

To get the stronger result, we change the parameters Δu and Δl and we end up proving that

$$\mathrm{tr}(L_A) \geq \frac{1}{\Delta l} - L(A,l),$$

instead of

$$\mathrm{tr}(L_A) \geq \frac{1}{\Delta l} - \frac{1}{1/L(l,A)-\Delta l}.$$

That proof is even messier.

Having seen the correctness of the algorithm, we turn towards the question of its run time. The general usage of a spectral sparsifier is to reduce the dependence of

the runtime of algorithms that depend on the number of edges by creating a sparse approximation. The algorithm described runs far too slowly to realize gains using a sparsifier for cut algorithms (for instance). The faster algorithm due to Lee and Sun [3], constructs a sparsifier with $O(n/\epsilon^2)$ edges in $\tilde{O}(m/\epsilon^{O(1)})$ time. Their barrier functions are

$$U(A,u) = \sum_{i=1}^{n} \exp\left(\frac{1}{u-\lambda_i}\right)$$

and

$$L(A,l) = \sum_{i=1}^{n} \exp\left(\frac{1}{\lambda_i - l}\right).$$

An open research question is if we can get a $O(n/\epsilon^2)$ cut sparsifier by a greedy algorithm.

References

[1] Wai Shing Fung, Ramesh Hariharan, Nicholas JA Harvey, and Debmalya Panigrahi, *A General Framework for Graph Sparsification.* Proceedings of the Forty-Third Annual ACM Symposium on the Theory of Computing, pp 71-80, 2011.

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

[3] Lee, Yin Tat and Sun, He. *An SDP-based algorithm for linear-sized spectral sparsification.* Proceedings of the Forty-Ninth Annual ACM Symposium on the Theory of Computing, pp 678-687, 2017.