CS 229r Spectral Graph Theory in Computer Science, Lecture 7 Sep 24, 2020

Lecture 7

Instructor: Salil Vadhan *Scribe: Elbert Du*

0.1 Correction From last time:

When we convert a random walk matrix to Jordan normal form, the first row of the matrix looks like this:

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & \mu_2 & * & * \\ 0 & 0 & \ddots & * \\ 0 & 0 & \dots & \mu_n \end{pmatrix}$$

that is, we know that the first row has all 0s other than the first 1. In particular, this means that when we take large powers of this matrix, it converges to a matrix with a 1 in the top left entry and 0s everywhere else.

1 Markov Chain Monte Carlo (MCMC)

Goal: We want to efficiently sample from a desired probability distribution where the support is too large to sample from efficiently.

Where does this appear?
In TCS: Having efficient algorithms for sampling often leads to fast algorithms for counting combinatorial objects. [JVV86][DG99][ALGV19]

1.1 Sampling Random Forests

Given an undirected graph $G = (V, E)$, we want to count the number of spanning forests of G. That is, the number of acyclic subgraphs.

We know that it is $\#P$ complete to compute this exactly, but to get to within a $1 \pm \epsilon$ multiplicative factor, it's actually equivalent to sampling spanning forests (nearly) uniformly at random.

The question now is how do we actually sample these approximately uniformly at random? The answer is with MCMC. What we do is create a larger graph where the set of vertices in this graph is the set of spanning forests in the original graph. This graph might be exponentially large so we can't actually write the entire thing down. However, the idea is that to do a random walk, we only need to know the vertices adjacent to our current vertex, which is a small set, and if we have a uniform stationary distribution all we have to do is walk for long enough do get an almost uniform random forest.

The way we do this walk is as follows: if we start at a forest F, we pick a random edge $e \in E$. If $e \in F$, we move to $F \setminus \{e\}$. If $e \notin F$, and it does not create a cycle, we move to $F \cup \{e\}$. If we choose an edge that creates a cycle, then we just self-loop. Note: we don't need every vertex to have an edge leaving it for these forests. Note that this is just a random walk on an $|E|$ regular graph where E is the set of edges.

We wish to show that the mixing time of this graph is $poly(n)$ where n is the number of vertices in our original graph. This result was proven last year here:[ALGV19], so we will not prove it.

A generalization of this problem is for weighted spanning forests, where we wish to sample each forest with probability proportional to its weight. If we let w_e be the weight of each edge e, then the weight of the forest F is $w(F) = \prod_{e \in F} w_e$

Other examples:

- Perfect matchings in bipartite graphs
- Weighted matchings
- Independent sets
- Colorings

For a lot of these problems, we get polynomial time algorithms by studying rapidly mixing Markov chains and bounding their mixing times.

Important takeaways: estimating mixing time for random walks is equivalent up to log factors to bounding eigenvalues on undirected graphs.

However, directed graphs make this more complicated. We can no longer get the stationary distribution from the degrees of the vertices. The mixing time can be bounded by eigenvalues or singular values, but can differ by more than a log factor. In fact, the mixing time can even get exponentially large.

2 Graph Coloring

In this setting, we suppose our graph G is simple and undirected, and let $\chi(G) =$ the minimum number of colors necessary to color the vertices of G such that no adjacent vertices have the same color. It is $\#P$ complete to compute or approximate this to within a multiplicative factor of $n^{1-\epsilon}$. Even so, there are still polynomial time algorithms which can give us useful bounds. If we let $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_n$ be the eigenvalues of the adjacency matrix M,

Claim 1. *$\mu_n < 0$ for non-empty graphs G.*

Proof. G is simple so it has no self loops. Thus, $\sum_{i=1}^{n} \mu_i = Tr(M) = 0$. Since the graph G is non-empty, our matrix M has to have some nonzero eigenvalue so it must have a negative eigenvalue. □

Theorem 2. *Hoffman Lower Bound:*

$$\left\lceil \frac{\mu_1}{-\mu_n} \right\rceil + 1 \leq \chi(G)$$

Proof. Proof of Hoffman: if $k = \chi(G)$, then we can re-order the vertices to group each of the k colors together. Then, in block form, each of the matrices along the diagonal will be all 0s because we cannot have any edge connecting two vertices of the same color. That is, we can write

$$M = \begin{pmatrix} 0 & M_{12} & \dots & M_{1k} \\ M_{12}^T & 0 & \ddots & \vdots \\ \vdots & & \ddots & \\ M_{1k}^T & \dots & M_{k-1k}^T & 0 \end{pmatrix}$$

Now, we claim that

$$\sum_{j=1}^{k} \lambda_{max}(M_{jj}) \geq \lambda_{max}(M) + (k-1)\lambda_{min}(M)$$

This is basically just the block analog of the statement that the trace is at least $\lambda_{max}(M) + (n-1)\lambda_{min}(M)$ which follows directly from the fact that the trace is also the sum of the eigenvalues. The proof of this claim is still nice, but we won't be going over it.

In our case, the LHS of this lemma is 0 and the RHS is $\mu_1 + (k-1)\mu_n$. Therefore, we get $k \geq 1 + \frac{\mu_1}{-\mu_n}$. Since we know k has to be an integer, we can bound this a little tighter by taking the ceiling.

□

Theorem 3. *Wilf's Theorem:*

$$\chi(G) \leq \lfloor \mu_1 \rfloor + 1$$

Exercise 4. *Using this theorem, compute the bounds for $\chi(G)$ for $G = K_n$ (complete graph without self loops) and $G = C_n$ (the undirected n-cycle)*

For the complete graph K_n, we have $\mu_1 = n-1$ and $\mu_j = -1$ for all $j > 1$ so theorem 2 just tells us that $n \leq \chi(G) \leq n$
For C_n, the theorem gives us $2 \leq \chi(G) \leq 3$ if n is even and $3 \leq \chi(G) \leq 3$ if n is odd.

Now, let's look at some proof sketches for theorem 2:

Note: when we do graph coloring, we're basically just trying to decompose the graph into independent sets.

3 Isoperimetry and Conductance

We again let G be an undirected graph. We ask the question: how well connected is G?

Intuitively, if G is well-connected, we expect the number of edges leaving a set S to be proportional to the number of vertices in S. This leads to some definitions:

Definition 5. *We define the boundary of S,*

$$\delta S := \{(a,b) \in E \mid a \in S, b \notin S\}$$

Definition 6. *We define the isoperimetric ratio*

$$\Theta(S) = \frac{|\delta S|}{|S|}$$

Note that this particular expression only really makes sense for $|S| \leq \frac{n}{2}$ so we can alternatively let $s = \frac{|S|}{n}$ and define the symmetrized version of this:

$$\Theta(S) = \frac{|\delta S|}{ns(1-s)}$$

which changes it by at most a factor of 2.

On the other hand, conductance measures the ratio of the weight of the cut to the total degree of the set.

Definition 7. *If we let $d(S) = \sum_{a \in S} d(a)$ (the sum of the degrees of the vertices in the graph) and $w(\delta S) = \sum_{e \in \delta S} w(e)$, then the conductance is*

$$\phi(S) = \frac{|w(\delta S)|}{d(S)}$$

Some interesting things to note:

For a bit of notation, let $\pi|_S$ be the stationary distribution conditioned on S. Since G is unweighted, this just means we start at each vertex in S with probability proportional to its degree.

The conductance is equivalent to the probability that if we start a random walk in $\pi|_S$, we leave S in one step. To see this, imagine each edge as two directed edges in opposite direction. Because our starting distribution is $\pi|_S$, we walk along each directed edge with the same probability when we take our first step. The total number of directed edges originating from vertices in S is the sum of the degrees of the vertices of S and the number of them that leave S is just the number of edges leaving S. Thus,

$$\phi(S) = P[\text{r.w. starts from the distribution } \pi|_S, \text{and leaves S in one step}]$$

and if we let $\pi(S) = \sum_{a \in S} \pi(a) \leq \frac{1}{2}$, we know that this is at most twice the value of

$$P[\text{r.w. starting from the stationary distribution starts at S and leaves S in one step}]/(\pi(S) \cdot \pi(V - S))$$

Now, note that $\pi(S) \cdot \pi(V - S)$ is just the probability that if we choose two points from the stationary distribution, the first is in S and the second is in $V - S$. Thus, if this second quantity is equal to 1, then the probability that we leave S in one step when we start at S is the same as the probability that any point chosen from the stationary distribution is in $V - S$. In particular, if it's true for all sets S, then where we end up after the second step is independent of where we ended up after the first and the random walk mixes in 1 step. This intuitively tells us why we care about conductance: high conductance tells us the random walk mixes quickly. This also works for directed graphs.

Now, we've seen before that a lot of things become equivalent with nice graphs, and this is no exception. If we have an unweighted regular graph, then

$$\phi(S) = \frac{\Theta(S)}{d}$$

We can now look at the conductance and isoperimetric number of a graph G:

$$\Theta(G) := \min_{S:|S| \leq n/2} \Theta(S)$$

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

Theorem 8.

1. $\Theta(G) \geq \lambda_2/2$
2. $\phi(G) \geq \nu_2/2$

These are similar proofs, and we won't go through all of the calculations.

Proof. 1. If we let x_S be the indicator vector for the set S projected onto the space orthogonal to $\mathbf{1}$, it turns out that

$$\overline{\Theta}(S) = \frac{|\delta S|}{ns(1-s)} = \frac{x_S^T L x_S}{x_S^T x_S}$$

If we minimize this over all sets S, this is

$$\geq \min_{x \perp \mathbf{1}} \frac{x^T L x}{x^T x} = \lambda_2$$

So

$$\Theta(S) \geq \frac{1}{2}\overline{\Theta}(S) \geq \lambda_2/2$$

Now, let's do something similar for the conductance

2. If we let y_s be the indicator vector for S projected onto the space orthogonal to the degree vector $\overrightarrow{d}$, then it turns out

$$\frac{w(\delta S)/d(V)}{d(S)/d(V) \cdot d(S-V)/d(V)} = \frac{y_S^T L y_S}{y_S^T D y_S}$$

So now if we let $z_S = D^{1/2} y_S$ projected onto $\overrightarrow{d}^{1/2}$ we get that this is equal to

$$\frac{z_S^T N z_S}{z_S^T z_S}$$

and minimizing over S, this is

$$\geq \min_{z \perp \overrightarrow{d}^{1/2}} \frac{z^T N z}{z^T z} = \nu_2$$

□

Note that to exactly compute the minimum over S, it's basically an integer quadratic program which is NP-hard. What we do here is a convex relaxation of this to turn it into computing λ_2, which can be done in polynomial time. This theorem is therefore basically giving us a poly time approximation of a hard value to compute.

References

[ALGV19] Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials ii: High-dimensional walks and an fpras for counting bases of a matroid, 2019.

[DG99] Martin Dyer and Catherine Greenhill. Random walks on combinatorial objects. In *Surveys in Combinatorics 1999*, pages 101–136. University Press, 1999.

[JVV86] Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169 – 188, 1986.