CS7880: Special Topics in Theoretical Computer Science Spring 2019

Lecture 10 — February 11, 2019

Prof. Huy Nguyen *Scribe: David Stalfa*

1 Review from Previous Class

The problem we are studying is graph streaming: we are given a set of vertices and the streaming updates consist of the addition or removal of edges. In this setting, we would like to compute some function on the final graph.

The general question we are interested in answering is:

Question 1. *If we are given a graph with many edges, can we store a smaller graph that maintains certain properties of the original?*

More specifically, we want to maintain cut properties of the original graph.

Question 2. *If G is a graph on n vertices, is there some graph H with $O(n)$ weighted edges such that, for every cut $(S, V \setminus S)$*

$$(1-\varepsilon)E_G(S, V \setminus S) \leq E_H(S, V \setminus S) \leq (1+\varepsilon)E_G(S, V \setminus S) \ ?$$

We ended class with a special case of this problem.

Question 3. *If G is a complete graph on n vertices, is there some graph H with $O(n)$ weighted edges such that, for every cut $(S, V \setminus S)$*

$$(1-\varepsilon)E_G(S, V \setminus S) \leq E_H(S, V \setminus S) \leq (1+\varepsilon)E_G(S, V \setminus S) \ ?$$

In this class, we will answer questions 2 and 3 in the affirmative. This will end the portion of the course on streaming algorithms, and we will begin discussing the second topic: compressive sensing.

2 Graph Sparsifiers

We might tray to answer question 3 by using a generic graph sampling algorithm.

Generic graph sampling algorithm:

- sample each edge e with probability p_e
- if e is sampled, then assign weight $\frac{1}{p_e}$ to e

In our case, we might use $p_e = \frac{1}{n}$, which assigns each sampled edge weight n.

However, this does not work for our purpose. We can see that

$$\Pr[\text{vertex } v \text{ has no neighbors in } H] = \left(1 - \frac{1}{n}\right)^{n-1} \approx e^{-1}$$

So the chance of some vertex having no neighbors is high.

One way to remedy this problem is to use a different value for p_e. Since

$$\Pr[\text{vertex } v \text{ has no neighbors in } H] = (1 - p_e)^{n-1} \approx \exp(-p_e n)$$

setting $p_e = \frac{\log n}{n}$ yeilds

$$\Pr[\text{vertex } v \text{ has no neighbors in } H] \approx \frac{1}{n}$$

This solution, however, does not address question 3 since the sampling provides a graph with $O(n \log n)$ edges.

In fact, if we allow H to have $O(n \log n)$ edges, there are several solutions to the problem, e.g. using expander graphs. There is also a result from Karger which states

Theorem 4. *If $p_e \geq \min\{1, \frac{c \log n}{\lambda \varepsilon^2}\}$ then H is a sparsifier of G, where c is some constant and λ is the size of the min cut in G.*

A Different Approach. We can answer questions 3 and 2 in the affirmative using the following approach. Instead of sampling each edge independently with some probability, we sample from the distribution over all subgraphs of G such that all vertices have at least degree 10 (or some similar constant). This method results in a graph H that is a sparsifier for G and has $O(n)$ edges, though we do not prove this result here.

Theorem 4 can help us answer questions about the min cut of a graph G, like the following.

Question 5. *What is the approximate value of the min cut in G?*

First, we will use a technique called 'wishful thinking', and then use Theorem 4 to complete the algorithm.

Wishful thinking. In this model, we suppose someone has told us the value of the min cut, and it is our job to verify what they've said. For instance, suppose our informant has told us that the value of the min cut is $\frac{n}{1000}$. In this case, we can apply Theorem 4 and construct a new graph H by sampling G with

$$p_e = \frac{c \log n}{\frac{n}{1000}\varepsilon^2}$$

We can then check if, in the resulting H, the min cut is equal to

$$p_e \frac{n}{1000} = \frac{c \log n}{\varepsilon^2}$$

Finding the min cut in H might be nearly as hard as finding the min cut in G (i.e. p_e might be close to 1) so we can use a tool from last class: we check if H is k-connected.

Min-Cut Approximation Instead of having someone tell us the answer, we simply try all answers (to within a factor of 2). The final algorithm is as follows:

- For each guess $\lambda = 0, 1, 2, 4, 8, \ldots$
  - Sample edges with probability $p_e = \min\{1, (c \log n)/(\lambda \varepsilon^2)\}$
  - Check if min cut in the resulting graph is $(c \log n)/\varepsilon^2$

Since k-connectivity takes $O(kn \cdot \text{polylog}(n))$ space, this algorithm takes

$$(\log n)(n \log n) \cdot \text{polylog}(n) = O(\text{polylog}(n))$$

Open question. Here we cover an open problem in the space of graph sparsification. We define the *Laplacian* of a graph G as follows.

- If there is an edge (u, v) in G, represent this as a matrix

$$\begin{array}{c} \\ u \\ v \end{array} \begin{array}{c} \begin{array}{cc} u & v \end{array} \\ \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \end{array}$$

- The Laplacian of G is the sum over all edge matrices

We also define the spectral sparsifier of G.

Definition 6. *Let $L_{G'}$ be the Laplacian of any graph G'. Then H is the spectral sparsifier of G if, for all vectors x,*

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

This concept of a sparsifier generalizes the notion we used before.

In classical settings, we can find H in almost linear time (depending on how few edges you want in H). The open question, is

Question 7. *How can we find H in a streaming setting?*

– This ends the section of the course on streaming algorithms. –

3 Compressive Sensing

The problem here will be similar to the one studied at the end of the streaming section, but the guiding question will be slightly different.

The Problem. We are given a signal with a few meaningful parts and a lot of noise. We would like to store as little information as possible and still recover the important parts of the signal.

Formally, we are given a signal $x \in \mathbb{R}^n$ with k non-zeroes plus some noise. We would like to find linear measurements Πx in order to recover $\hat{x}$ such that

$$\|\hat{x} - x\|_p \leq f(k) \cdot \min_{y:k\text{-sparse}} \|y - x\|_q$$

Note that, as before, recovery error is proportional to the amount of noise. Specifically, if the signal is k-sparse, then we require that recover error is 0.

Count Sketch Result. We immediately get one result from count sketch.

Lemma 8. *For any x, if we construct Π according to Count Sketch with $O(\frac{k}{\varepsilon^2}\log n)$ measurements (rows), then we get $p = q = 2$ and $f(k) = 1 + \varepsilon$ with probability $(1 - \frac{1}{n})$.*

However, this result will not immediately answer the question we are interested in here.

A New Question. In the streaming setting, we allowed ourselves to construct a new set of measurements Π for each signal received. Here, we are concerned with a different setting.

Question 9. *Can we find a Π that performs the recover for all inputs?*

Note that, instead of finding a Π for each new signal we receive, we are fixing the set of measurements and hoping to show that this Π will be able to recover any given signal.

Toward answering this question, we have the following result.

Theorem 10. *There is some Π and an algorithm to find $\hat{x}$ such that Π has $O(k \log \frac{n}{k})$ rows, and*

$$\|\hat{x} - x\|_2 \leq \frac{1}{\sqrt{k}} \min_{y:k\text{-sparse}} \|x - y\|_1$$

Note, on the right hand of the inequality, we compare against the ℓ_1 norm instead of ℓ_2. So, Theorem 10 is a weaker guarantee than Lemma 8.

We introduce the following notation.

Definition 11. *Let $\|x_{tail}\|_1$ be the ℓ_1 norm of the noise in x. I.e. $\|x_{tail}\|_1$ is the same as x with the largest k indices set to 0.*

3.1 A Special Case: x is k-sparse

Suppose x contains no noise, i.e. x is k-sparse. In this case, we can always recover x with as few as $2k$ measurements: if we have a vector z and want to check if $\Pi(z - x) = 0$, this requires $2k$ measurements.

Ideally, we would like to solve the following optimization problem.

$$\text{minimize } \|z\|_0, \text{ subject to}$$
$$\Pi z = \Pi x$$

However, $\|.\|_0$ is not a convex space, so this problem is not efficiently solvable.

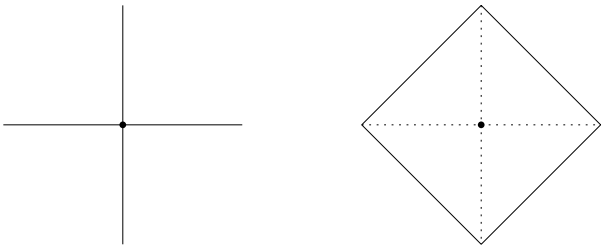

Figure 1: Left: $\|.\|_0$ unit ball. Right: ℓ_1 unit ball.

Approximating $\|.\|_0$. We can picture the $\|.\|_0$ unit ball in 2-dimensions as falling along the x- and y-axes. The closest convex approximation of this ball is the ℓ_1 unit ball (see Fig. 1). Since ℓ_1 is convex, the following optimization is solvable.

$$\text{minimize} \|z\|_1, \text{ subject to}$$
$$\Pi z = \Pi x$$

It turns out that solving such an optimization provides a close approximation for the original problem.

However, we would still like to characterize these Π's.

RIP Matrices.

Definition 12. *A matrix $\Pi \in \mathbb{R}^{n \times m}$ satisfies the (ε, k)-restricted isometry property (RIP) if, for all k-sparse x,*

$$(1-\varepsilon)\|x\|_2^2 \leq \|\Pi x\|_2^2 \leq (1+\varepsilon)\|x\|_2^2$$

Question 13. *How many rows are necessary to achieve this property?*

It might seem natural to try to answer question 13 by direct application of the JL Lemma. However, the JL Lemma provides guarantees only for a finite number of vectors, and there are an infinite number of k-sparse vectors.

In order to apply the JL Lemma, we must first reduce the space of possible vectors to a finite number. This reduction can be done by the following intuitive procedure, which we make precise later. Basically, we define a finite number of representative vectors in the possible space. For any non-representative vector, we calculate the error of its nearest representative, and guarantee that the additional error between the vector and its representative is also small.

This structure can be provided by an ε-net, and requires

$$\binom{n}{k} \cdot \left(\frac{10}{\varepsilon}\right)^{10}$$

representatives. The first term comes from the number of non-zeroes in the signal, and the second from the size of the ε-net for a k-dimensional unit ball. By the JL Lemma, the number of rows we require is

$$\left(\log \binom{n}{k} \cdot \left(\frac{10}{\varepsilon}\right)^{k}\right) \frac{1}{\varepsilon^2} \approx \frac{k}{\varepsilon} \log \frac{n}{k}$$

It remains to show that the error between vectors in the ε-net is sufficiently small.

Definition 14. *A matrix* Π *is* α*-incoherent if*

$$\left\|\Pi^i\right\| = 1 \quad \text{and} \quad |\langle \Pi^i, \Pi^j \rangle| \leq \alpha$$

for any $i \neq j$*, where* Π^i *is the* i^{th} *row of* Π*. I.e. any two rows of* Π *are almost orthogonal.*

The next step is to show that, if Π is sufficiently incoherent, then it satisfies the RIP property. However, we do not have time to finish this class.