CS 395T: Sublinear Algorithms, Fall 2020 November 3rd, 2020

Lecture 20: Graph Sketching

Prof. Eric Price *Scribe: Niels Kornerup, Joshua Prupes*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Overview

This lecture covers the probability that your vote will matter in an election, sampling a non-zero element from a turnstile stream, and an introduction to graph sketching.

2 Probability that a vote matters

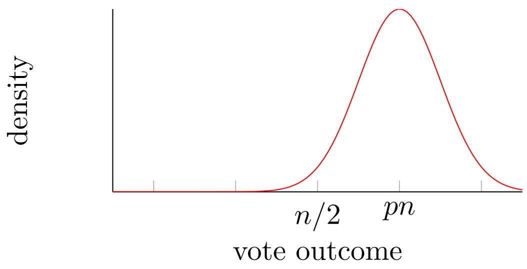
Lets suppose you are in a state with n residents and candidates A and B. What is the probability that your vote makes a difference? If all other votes are taken to be uniformly random, then there are $\binom{n}{n/2} = O(2^n/\sqrt{n})$ ways to have exactly $n/2$ votes for each candidate, giving us a probability of $\sim 1/\sqrt{n}$ to influence the election.

What if voters pick from some distribution other than uniform? Lets say that each voter chooses A with probability 0.52 and B with probability 0.48. Now your probability of deciding the election is $\mathbb{P}[\sum x_i = n/2]$, where x_i is 1 with probability 0.52. This gives us that

$$\begin{aligned}
\mathbb{P}\left[\sum x_i = n/2\right] &= \binom{n}{n/2}(.52)^{n/2}(.48)^{n/2} \\
&\approx \frac{2^n}{\sqrt{n}}\left(\frac{1}{2^n}\right)\left(1 - \frac{1}{25^2}\right)^{n/2} \\
&= \frac{1}{\sqrt{n}}\left(1 - \frac{1}{625}\right)^{n/2} \\
&= \frac{1}{\sqrt{n}}e^{-\Theta(n)}
\end{aligned}$$

The expected value of voting can be given by $\mathbb{P}[\text{change}](\text{effect of change})$. In general, the effect of change is $\Theta(n)$ since all voters are affected. This means that our first analysis gives us an expected value of $\Theta(\sqrt{n})$ for voting while the second gives $\sqrt{n}e^{-\Theta(n)}$.

In reality, we do not know what the true voting distribution is. Despite this, it is reasonable to assume that it looks something like this:



Given this model, we know that

$$\begin{aligned}\mathbb{P}[\#\text{ votes} = n/2] &\geq \frac{\mathbb{P}[\#\text{ votes} < n/2]}{n/2} \\ &\approx \frac{.2}{n}\end{aligned}$$

because our density is monotonic on the domain where the vote outcome is less than pn. This gives us that the expected value of voting is constant. In other words, it is equally important to vote, regardless of the size of the population.

3 Sampling from a turnstile stream

Lets say you have a turnstile stream $x \in \mathbb{R}^n$ and you want to output some $i \in \text{supp}(x)$ if any exists. In homework 3 we showed that if all values are integers and $\|x\|_0 = 1$, we can find the element by computing the following ratio:

$$\frac{\sum ix_i}{\sum x_i}$$

This ratio will be the non-zero index i^*. If we have that $\|x\|_0 = k$, we can instead sample at a rate of $\frac{1}{k}$. If we do this, we get that the probability that we sample exactly one non-zero index is:

$$k\frac{1}{k}\left(1 - \frac{1}{k}\right)^{k-1} \approx 1/e$$

Since we don't know the value of k ahead of time, we can sample at rate $1/k = 1, 1/2, 1/4, \ldots 1/n$. This implies that $\exists k$ with $k \leq \|x\|_0 \leq 2k$. This sampling method gives us that the chance exactly one non-zero index is sampled can be given by

$$\frac{\|x\|_0}{k}\left(1 - \frac{1}{k}\right)^{\|x\|_0 - 1} \geq e^{-2}$$

This method would work, except that it has one issue: there is no way to know if we actually only got one non-zero value in our sample. We can get around this by using a hash function $h : [n] \to [n^{10}]$. We can keep track of $\sum h(i)x_i$ and at the end make sure that it is equal to $h(i^*)\sum x_i$. Finally, we will repeat $\log(n)$ times to make sure we will succeed at the right scale. This results in a $O(\log^2(n))$ word algorithm.

4 Graph Sketching [Ahn, Guha, and Mcgregor '12]

General problem is to construct an $O(n \log^{O(1)}(n))$ space linear sketch of a graph. This means that one can handle edges being inserted and deleted. From our sketch we want to be able to recover the following things from the original graph (can't recover everything):

- Sample a random edge from a cut
- Estimate the size of a cut to $(1 \pm \epsilon)$
- Find a spanning forest
- Find a $(1 \pm \epsilon)$-approximate minimum spanning tree
- Test bipartiteness

Let A be the $n \log^2(n) \times m$ sampler sketch matrix defined in part 3. Let E be defined as the edge-vertex incident matrix which is $m \times n$ where every row corresponds to an edge that will have two nonzero values for vertices that it touches. One of the nonzero values is arbitrarily assigned 1 and the other -1.

The idea is to store $y = A \cdot E \in \mathbb{R}^{o(\log^2(n)) \times n} = n \log^3(n)$ space

- To sample an edge from each node v:
  run the sampler sketch algorithm on $y^{(v)} = A * (\sum_{\substack{\text{edges p} \\ (u,v) \in G}} \pm e_p)$
- To sample an edge from any cut $(S, \bar{S})$:
  $$\sum_{v \in S} y^{(v)} = A * (\sum_{\substack{\text{edges p} \\ (u,v) \in G \\ \text{p cross cut}}} \pm e_p)$$
  since the edges contained in S will cancel each other out.
- To estimate the size of the cut within $(1 \pm \epsilon)$:
  Run same algorithm as above accept make A the AMS sketch matrix
- To find a spanning tree...
  Will go over in next lecture

References

[AhnGM12] Kook Jin Ahn, Sudipto Guha, Andrew McGregor. Analyzing graph structure via linear measurements *ACM-SIAM*, 459–467, 2012.