COMS E6998-9: Algorithms for Massive Data (Spring'19) Feb 28, 2019

Lecture 12: Streaming for dynamic graph problems: connectivity

Instructor: *Alex Andoni* Scribes: *Roland Maio*

1 Triangle Counting

We continue triangle counting from the previous lecture and recall the set up. Let T be the number of triangles in an undirected and unweighted graph G and we assume that T is bounded from below by some t so that $T \geq t$. Using the notion of triangles we define a measure of clusterability: $F = \frac{3T}{\sum_i \binom{\deg(i)}{2}}$[1]. We defined a vector x where each coordinate x_S is uniquely indexed by a size-3 subset S of vertices in G, and the value of x_S is the number of edges between vertices of S. The p-th moment of F is defined to be $F_p = \sum_S x_S^p$.

Recall that in the last lecture we showed that:

$$T = F_0 - 1.5F_1 + \frac{1}{2}F_2$$

The algorithm that we developed estimates each F_p, for $p \in \{0, 1, 2\}$, up to a $(1+\gamma)$-factor using space $O(\frac{1}{\gamma^2})$ words.

$$\hat{T} = \hat{F_0} - 1.5\hat{F_1} + \frac{1}{2}\hat{F_2}$$

We can use any F_p estimation algorithm as a black box. Unfortunately, the estimate $\hat{T}$ is not a $(1+\epsilon)$-approximation to T because of the sign and multiplicative factor approximations to F_0 and F_1 but $\hat{T}$ is still good. In particular we have an inequality on $|\hat{T} - T|$. Recall that F_0 is the number of distinct elements, that is, the number of sets S such that at least one edge is in G. If we start from an empty graph and add an edge, then about m coordinates in x will be non-zero. So $\gamma F_0 = O(\gamma mn)$.

$$\begin{aligned}
|\hat{T} - T| &\leq |F_0 - \hat{F_0}| + 1.5|F_1 - \hat{F_1}| + \frac{1}{2}|F_2 - \hat{F_2} \\
&\leq \gamma F_0 + 1.5\gamma F_1 + \frac{1}{2}\gamma F_2 \\
&\leq O(\gamma mn)
\end{aligned}$$

Our desired bound gives a condition on γ, specifically, we would like $O(\gamma mn) \leq \epsilon t \leq \epsilon T$. Setting $\gamma = \Theta(\frac{\epsilon t}{mn})$ yields a $(1 \pm \epsilon)$-approximation to T using space $O\big(\frac{1}{\gamma^2}\big) = O(\frac{(mn)^2}{\epsilon^2 t^2})$.

[1] Fun fact, from the theory seminar earlier in the day, the clusterability of Facebook per F is 0.16

2 Approach 2 to Triangle Count

We will now see another algorithm for triangle counting. The big idea is to take a different, more "natural" approach to the problem from what we initially considered; we will try to estimate the number of triangles by sampling.

The idea is to sample k sets $S, |S| = 3$, and keep $x_{S_1}, \ldots, x_{S_k}$ where the S_i are the sampled sets S.

Algorithm: Keep $x_{S_1}, \ldots x_{S_k}$.

Estimator: $E = \frac{M}{K}|\{i : x_{S_i} = 3\}|, M = \binom{n}{3}$

Analysis: The space is clearly k. The analysis of this estimator will be similar to our analysis of the Morris algorithm in the first lecture. The strategy will be to compute the expectation, bound the variance from above, and apply Chebyshev's inequality. We can take the expectation using the characteristic function $\chi[\cdot]$.

$$\mathbb{E}[E] = \frac{M}{k}\mathbb{E}_{S_1,\ldots,S_k}\sum_{i=1}^{k}\chi[x_{S_i} = 3] = \frac{M}{k}\sum_{i=1}^{k}\frac{T}{M} = T$$

The variance can be obtained by observing that the estimator is the sum of random variables.

$$\mathrm{Var}[E] = \frac{M^2}{k^2}\sum_{i=1}^{k}\mathrm{Var}[\chi[x_{S_i} = 3]] \leq \frac{M^2}{k^2}k\frac{T}{M} = \frac{M}{k}T$$

Now we can apply Chebshyev's inequality, $E \in T \pm O(\sqrt{\mathrm{Var}})$ with probability 90% choosing k so as to make the variance less than ϵT, $O(\sqrt{\mathrm{Var}}) < \epsilon T$, implying that we have:

$$\frac{M}{k}T < O(\epsilon T)^2 \implies k > \Omega(\frac{M}{\epsilon^2 T}) \geq \Omega(\frac{M}{\epsilon^2 t}) = \Omega(\frac{n^3}{\epsilon^2 t})$$

So depending on n and t the bound can be very small or large, so a priori it is not clear how good this is. The problem is the space directly depends on M, where M is all possible sets of size 3, and this seems very loose. If the graph is very sparse, then M will be much less than $\binom{n}{3}$. A random set in a sparse graph is unlikely to be a triangle.

So the idea is to say, what if we can sample from a smaller set? Rather than sample all the sets, sample instead from a bounded, more narrow set of sets of size 3.

Of all possible sets of size 3, we are actually only interested in a much smaller set. The first idea is basically doing a naive, brute force, Monte Carlo experiment. Clearly, we want many samples to fall inside T. So what if we can prune the size of the set from which we sample so that it still contains all the triangles? Then the set of triangles should be relatively larger compared to the pruned set. Let's call this set M'.

Define $M' = \{S \in [n]^3 : x_S \geq 1\}$, to be the set of all S such that $x_S \geq 1$. In a sparse graph this set should be much smaller than the universe. In particular, note that the size of M' is bounded above by a constant, that is: $|M'| = F_0 \leq mn$.

Algorithm: Sample $S_1, \ldots, S_k$ from M'; Keep x_{S_i}.

Estimator: $E = \frac{M'}{k}\sum_{i=1}^{k}\chi[x_{S_i} = 3]$

Claim: $k = O(\frac{M'}{\epsilon^2 T}) \leq O(\frac{mn}{\epsilon^2 t})$

In our previous analysis, the only thing that we really used was the dependence on M, so the exact same proof for the first idea works here too. We simply substitute the size of M' to obtain the new bound

and see the improvement.

So this is a better algorithm, but we must now figure out how to sample from M'. It is not possible to know what is M' before we observe the stream, so we need some kind of tool.

3 Tool (dynamic sampling)

We require the following to apply the tool:

1. There is a vector $x \in R^n$ that is maintained via a stream of updates.
2. A sketch such that at the end of the stream, it can sample $i \in [n]$ such that $x_i \geq 1$ and additionally outputs x_i

Once we build a tool, we may as well build one to solve more general problems, so think about this tool in the General Turnstile Streaming Model (GTSM).

We would also like the space of the tool to be polylogarithmic: $(\log(N))^{O(1)}$.

Note that this tool is weird in that its output is not deterministic, but rather a distribution. The tool observes the stream, does something, and spits out a coordinate saying: I promise you that this coordinate has been sampled uniformly at random from all non-zero coordinates.

Clearly, this is what we need for the triangle counting algorithm we have developed. We keep k i.i.d. dynamic sampling sketches where each sketch samples from M' by the construction of both M' and the tool.

Note that this tool is more powerful than what we need because it is in the GTSM. It returns a coordinate with absolute value greater than 0 which is more general than the triangle counting problem in which we know all coordinates are 0, 1, 2, or 3.

This version of the tool is called the ℓ_0-sampler. In general the ℓ_p-sampler will return (i, x_i) for fixed i with probability $\frac{|x_i|^p}{\sum_j |x_j|^p}$.

We will also use this tool for another graph streaming problem. We further desire that the sketch S be a linear random map, $S : \mathbb{R}^n \mapsto \mathbb{R}^k$ and that when the estimator E is applied to the sketch, $E(S(x))$, it produces (i, x_i) where $i \in_r \{i : x_i \neq 0\}$.

This is not necessary for triangle counting, but it will be for the next problem. In general, the idea is that if you design a sketch for the GTSM, it helps to make it linear.

We assume that $x \in \{-1, 0, 1\}^n$. This assumption is not too strong, because once we have a linear sketch it can be dropped.

Let $D = \{i : x_i \neq 0\}$ and now for the intuition, consider the coordinates that are non-zero. Suppose we know that at the end of the stream there are only 3 non-zero coordinates. This appears to be very difficult because a lot can go on in between, for a given coordinate, it may be that in the beginning there are a lot of adds, and toward the end there are a lot of subtracts, and the result is that it is zero. How can we find a non-zero coordinate?

The insight is to observe that we can use a count-min sketch. Observe that every $i \in D$ is a $(1/4)$-Heavy Hitter. Therefore, we can simply use the count-min sketch algorithm for $\phi \geq 1/4$ and $\epsilon = 0.01$.

Recall that count-min sketch will return precisely the set D when D is very small. But what if D is large? If the size of D is polynomial in n, then the space of count-min sketch becomes polynomial in n as well. The solution is to downsample the coordinates. Let $I \subset [n]$ be a random subset such that $\Pr[i \in I] = \frac{10}{\sqrt{n}}$. Now restrict our attention in the stream only to those coordinates of x in I, denoted

$x|_I$. Then the expected number of elements of D also in I is constant, specifically $\mathbb{E}[|D \cap I|] = 10$, and this returns us back to the case of small D.

4 Dynamic Sampling-Basic

We now build the tool, and do so in two steps, starting with the basic one. The difference between the two steps will be that the basic tool, DS-Basic, will fail sometimes. We will see that the failure will come when the expectation of the intersection of D and I is low or 0.

Let $g : [n] \mapsto [n]$ be a 2-wise independent random hash function. Let $h : \mathbb{N} \mapsto \mathbb{N}$ be a function that given an integer i, returns the number of zeroes in $g(i)$ in binary. For example, if $g(i) = 101000$, then $h(i) = 3$. In particular we note that $\Pr_g[h(i) = j] = 2^{-j-1}$, for $j \in [L]$ and $L = \log n$. Observe that the bounds on j are similar in principle to downsampling. We will use this approach because it will give a slightly cleaner estimator.

So we partition the full stream into L streams where the jth stream cares only about $x|_{I_j}$ where $I_j = \{i : h(i) = j\}$. So this means that we first bifurcate the coordinates by feeding them through the hash function g, and then we group them so that the groups correspond to the sets I_j. In particular, note that $\mathbb{E}[|I_j|] = n2^{-j-1}$ and $\mathbb{E}[|D \cap I_j|] = |D|2^{-j-1}$. So all the coordinates are split between L levels, and the different levels correspond to different levels of downsampling.

Sketch: for $j = 0, \ldots, L$:

1. Store CS_j: Count-min Sketch on stream for $x|_{I_j}, \phi, \epsilon = 0.01$.
2. Store N_j: ℓ_1-norm estimation for $x|_{I_j}, \epsilon = 0.01$

Run both with success probability $1 - \frac{1}{n}$.

Operationally this amounts to routing the observation (i, δ) into the appropriate sketches CS_j, N_j where $j = h(i)$.

Estimator:

- Find some j such that $E_N(N_j) \in [1, 20]$.
- If no such i exists, then report FAIL.
- Fetch all ϕ-Heavy Hitters from CS_j.
- Pick one at random.

Analysis

For the analysis, we need to prove that if the tool succeeds, that the output is in fact a random element of D, and that if it fails, then it will do so with small probability.

We begin with the case that D is small, say $1 \leq |D| \leq 10$. Because the I_j partition the coordinates, there must be some partition that contains an element of D, and the count-min sketch associated with that partition will recover that element. Formally, $\exists j$ such that $|D \cap I_j| \geq 1 \implies CS_j$ will recover this j.

Now, in the case that D is large, say $|D| > 10$, we want to prove that there will be a level with the right downsampling probability that the condition $E_N(N_j) \in [1, 20]$ will be true. Let k be such that $|D| \in [10 \times 2^k, 10 \times 2^{k+1}]$, then we have:

$$\mathbb{E}[|D \cap I_k|] = |D|2^{-k-1} \in [5, 10]$$

and

$$\text{Var}[|D \cap I_k|] \leq |D|2^{-k-1} \in [5, 10] \leq 10$$

so by Chebyshev's inequality we have:

$$\Pr_{I_k}[\big||D \cap I_k|\big| - \mathbb{E}[|D \cap I_k|] \geq 5] \leq \frac{10}{5^2} = \frac{2}{5}$$

So with probability at least 3/5 we will have that $|D \cap I_k| \in [1, 14]$. But the overall probability that the basic estimator fails is slightly bigger than this because the count-min sketch and ℓ_1-norm estimators may fail, specifically, $\Pr[\text{FAIL}] \leq \frac{2}{5} + O(\frac{1}{n})$.

Now consider j such that $E_N(N_j) \in [1, 20]$. This implies that $|x|_{I_j}| \leq 20|D| \implies \forall i \in D \cap I_j$ that i is a (1/40)-Heavy Hitter and therefore will be recovered by CS_j.

By symmetry, the output of the Dynamic Sampling-Basic algorithm is a random element of D.

5 Dynamic Sampling-Full

We conclude with the full algorithm, DS-Full. Run $k = O(\log n)$ i.i.d copies of DS-Basic using any one that does not FAIL. Thus, $\Pr[\text{DS-Full fails}] \leq O(\frac{2}{5})^{O(\log n)} < \frac{1}{n^2}$.

For the space requirements we consider:

- $k = O(\log n)$
- $L = O(\log n)$ substreams $I_0, \ldots, I_L$.
- Each uses CS_j, which requires $O(\log^2 n)$ space, and N_j which requires $O(\log n)$ space.

The total space requirements are therefore $O(\log^4 n)$.

Next time we will see how to use dynamic sampling to do connectivity in the case that the graph is dynamic, where edges can be inserted and deleted. And we will use this to get an algorithm that uses space $O(\log n)$.