CS 395T: Sublinear Algorithms, Fall 2020 09/08/2020

Lecture 4: Distinct elements counting II: limited independence and turnstile

Prof. Eric Price *Scribe: Ajil Jalal*

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

1 Overview

In the last lecture we saw an algorithm that can estimate $\|x\|_2$ of a stream x under the turnstile model. Specifically, using $4-$wise independent hash functions $\{h_1, h_2, \cdots, h_m\}$, with each $h_j : [n] \to \{-1, +1\}$ and $m = O(1/\varepsilon^2)$, the estimator $\widehat{y} = \frac{1}{m}\sum_{j=1}^{m}(\sum_i h_j(i)x_i)^2$ guarantees $\widehat{y} \in (1 \pm \varepsilon)\|x\|^2$ with probability at least $3/4$.

In this lecture we will extend this algorithm to get a high probability bound under the assumption that each h_j is $4-$wise independent. The second half of this lecture will consider lower bounds for streaming algorithms.

2 High Probability Estimation of $\|x\|$

How do we get an estimator for $\|x\|$ which is good with $1 - \delta$ probability? We first consider the following options and their respective shortcomings:

1. *Chebyshev Inequality.* A similar application of the Chebyshev inequality from last class requires $m = O\left(\frac{1}{\varepsilon^2\delta}\right)$ samples.
2. *Mean of Means / Better concentration inequalities.* If each y_j has well behaved moments/concentrates well, we can get a sharper confidence bound via the Hoeffding inequality (and variations). In this case, $m = O\left(\frac{1}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)\right)$ space suffices to get $\widehat{y}^2 \in (1 \pm \varepsilon)\|x\|^2$ with probability $1 - \delta$.

   However, this only works if h is fully independent. In the previous lecture, we only assumed that h has $4-$wise independence. Alternatively, we only assume that the fourth moment of y exists, and make no further assumptions about its moments.

2.1 Median-of-Means estimator

We will now show that using an appropriate median-of-means algorithm gives us a high probability estimate of $\|x\|$ using $4-$wise independence. Consider the following variation of the previous algorithm:

1. Use $m = \frac{18}{\varepsilon}^2 \log_{3/2}(1/\delta)$ hash functions in order to compute $y_k^2 = (\sum_i h_k(i)x_i)^2$ for each $k \in [m]$.

2. Randomly divide these into $R = \log_{3/2}(1/\delta)$ batches, such that each batch has $B = \frac{18}{\varepsilon^2}$ samples.

3. For batch $j \in [R]$, define $\widehat{y}(j)$ to be the mean of all samples in the batch.

4. The final estimate of the mean is the median over all batches. That is,

$$\tilde{y} = \text{median}_{1 \le j \le R} \widehat{y}(j),$$

is our final estimate of $\|x\|^2$.

The analysis of this algorithm is as follows. Notice that since $\tilde{y}$ is the median of a bunch of $\widehat{y}$'s, $\tilde{y}$ is a good estimate if at least half of the $\widehat{y}$'s are good.

Since each $\widehat{y}$ is an average of $18/\varepsilon^2$ samples, we have $\widehat{y}(j) \notin (1 \pm \varepsilon)\|x\|^2$ w.p. $\le \frac{1}{9}$.

This implies that

$$\begin{aligned}\mathbb{P}\left[\tilde{y} \notin (1 \pm \varepsilon)\|x\|^2\right] = \mathbb{P}[\text{at least } \frac{R}{2} \ \widehat{y} \ \text{ are bad}] &\le \binom{R}{\frac{R}{2}} \left(\frac{1}{9}\right)^{\frac{R}{2}}, \\ &\le 2^R 3^{-R} = \left(\frac{2}{3}\right)^R = \delta.\end{aligned}$$

What goes wrong if we use only the median? The previous algorithm first divides the sketches into batches, computes the mean over each batch, and then takes the median. What goes wrong if we only use the median?

Consider the following stream $x = (1, 1, 0, \cdots, 0)$. In this case, if we consider $y = \sum_i h(i)x_i$, then we have $y = \begin{cases} 2 & w.p.\ 0.25, \\ -2 & w.p.\ 0.25, \\ 0 & w.p.\ 0.5. \end{cases}$

Hence, y^2 is either 0 or 4, and both values are equally likely. If one were to take the median of these values, then the final estimate will be either 0 or 4 with overwhelming probability. The only way for this estimator to succeed is if there are an equal number of 0s and 4s, but this situation is extremely unlikely.

This concludes the first half of the lecture, and we will now move on to lower bounds for streaming algorithms.

3 Lower bounds for streaming algorithms

Most lower bounds for streaming algorithms are shown by reduction to appropriate problems in communication complexity. The intuition for this is that communication complexity studies the limits of what can be computed with limited communication between parties. Streaming algorithms have to compress large streams using the minimum amount of resources. This shows that these two problems are connected, since being able to compress large streams should imply protocols for efficient communication.

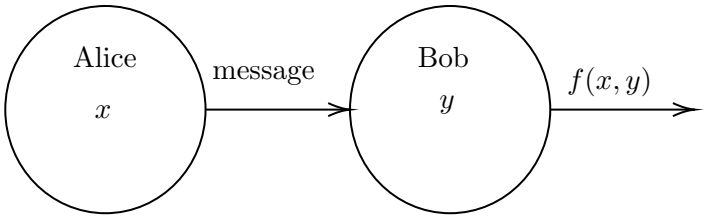

Figure 1: Alice has a binary string $x \in \{0,1\}^n$ and Bob has a binary string $y \in \{0,1\}^b$. Alice can communicate with Bob and in the end, they need to compute some function $f(x, y)$.

We now show a lower bound for counting distinct elements in the insertion-only streaming model using a reduction from the Indexing problem in 2-Party 1-Way communcation model.

In 2-Party 1-Way communication, Alice and Bob receive inputs x and y respectively. Alice is allowed to send a message to Bob, and Bob must compute a function $f(x, y)$. We now define the communication model and indexing problem.

Definition 1 (2-Party 1-Way communication)**.** *Alice has a binary string $x \in \{0,1\}^n$ and Bob has a binary string $y \in \{0,1\}^b$. Alice can communicate with Bob and in the end, they need to compute some function $f(x, y)$.*

Definition 2 ($INDEX$)**.** *Given a string $x \in \{0,1\}^n$ and an index $i \in [n]$, compute INDEX$(x, i) = x_i \in \{0,1\}$.*

We will use the following theorem from communication complexity, and defer its proof to the end of class.

Theorem 3. *The randomized 1-way communication complexity of $INDEX$ is $\Omega(n)$.*

Randomized communication complexity means that Bob outputs $f(x, y)$ correctly with $1 - c$ probability, and $c \in [0, 0.5)$ is a constant. Alice and Bob are allowed to use public coins, i.e., they can agree on all hash functions and other randomized tools they use in their algorithms.

We will now show the following Theorem using Theorem 3.

Theorem 4. *Solving distinct elements with $\varepsilon = 0$ and $\delta = 0.25$ requires $\Omega(n)$ space.*

Proof. Let $\mathcal{A}$ be an algorithm for distinct elements. Alice first inserts her message x into a stream, then runs $\mathcal{A}$ on the stream. The state of the algorithm is her message to Bob.

Once Bob receives this message from Alice, Bob asks for the number of distinct elements, call this v_1. Since he has the complete state of the algorithm, he can insert i into the stream and ask $\mathcal{A}$ for the number of distinct elements, call this v_2.

Finally, Bob outputs 1 iff $v_2 = v_1 + 1$.

The reason this works is: if $x_i = 0$, then inserting i into the stream increases the number of distinct elements by 1. Otherwise if $x_i = 1$, then inserting i does not change the number of distinct elements.

This implies that if there exists a $S-$space $\delta-$failure algorithm for distinct elements, then there exists a $S-$communication $2\delta-$failure algorithm for $INDEX$ (since Bob runs it twice, the probability of error becomes 2δ.)

This shows that for $\delta < 0.25$, we need $\Omega(n)$ space for any $\delta-$failure distinct elements algorithm. $\square$

3.1 Proof of Theorem 3 for deterministic algorithms

Before we analyze randomized algorithms, let us first consider deterministic algorithms for $INDEX$. If Alice sends a message using fewer than n bits, then there exist inputs $x, \widehat{x}$ that produce the same message but such that $x_i \neq \widehat{x}_i$. Hence Bob will fail on one of these inputs.

An information-theoretic version of the above paragraph will prove more useful for analyzing randomized algorithms. If x is a uniform random string drawn from $\{0,1\}^n$, and Bob can compute x_i for any i, then this implies that

$$I(x;\widehat{x}) = H(x) - H(x|\widehat{x}) = n - 0 = n,$$

where $\widehat{x}$ is Bob's estimate of x, and $H(x|\widehat{x}) = 0$ since Bob can compute x exactly.

The communication model satisfies the Markov chain $x \to m \to \widehat{x}$, where m is the message Alice sends to Bob. This gives

$$\begin{aligned} I(x;\widehat{x}) &\leq I(x;m), \\ &= H(m) - H(m|x), \\ &\leq H(m), \\ &\leq S, \end{aligned}$$

where the first inequality follows from the Data Processing Inequality, the second line follows from the definition of mutual information, the third follows from non-negativity of discrete entropy and the fourth follows since the message has at most S bits.

This gives us $S \geq n$.

3.2 Yao's Principle

In the previous section, we considered a deterministic algorithm on a randomized input. This technique can be uesd to show lower bounds for randomized algorithms, via Yao's lemma.

Definition 5 (Yao's Prinicple(informal))**.** *To lower bound any randomized algorithm with failure probability δ on all inputs, it suffices to give a fixed distribution of inputs such that no deterministic algorithm has $\delta-$failure probability on the distribution.*

One way to understand this principle is to view a randomized algorithm as a deterministic algorithm using a random advice string. If the randomized algorithm works on all inputs, it must works on distributions. This implies that for each distribution, there exists a good advice string, and using this particular advice string makes the algorithm deterministic.

We now provide the mathematical setup for analyzing the previous intuition. Consider a matrix $F \in \{0,1\}^{\#\text{algorithms} \times \#\text{inputs}}$. Let $F_{ij} = 1$ if algorithm i fails on input j. For a distribution D over inputs, the failure probability of algorithm i is $(FD)_i$. Similarly, for a distribution σ over algorithms, the failure probability on input j is $(\sigma^T F)_j$.

Hence, if we can find a distribution D such that every entry of FD is $> \delta$, then any distribution σ over algorithms must have an average $> \delta$ for this fixed D.

Another way of saying this is through a min-max framework and Neumann's principle. For a distribution D over inputs and a distribution σ over inputs, the expected error is $\sigma^T FD$. By Neumann's principle, since this is bilinear, we have:

$$\min_{\sigma} \max_{D} \sigma^T FD = \max_{D} \min_{\sigma} \sigma^T FD.$$

We now consider the left hand side and right hand side of the above equation. Consider the left hand side. For any fixed σ chosen by the outer player, the inner player will choose the worst case input j deterministically. This gives

$$\min_{\sigma} \max_{D} \sigma^T FD = \min_{\sigma} \max_{j} (\sigma^T F)_j.$$

Similarly, for the right hand side, for any D chosen by the outer player, the inner player will choose the best algorithm i. This gives

$$\max_{D} \min_{\sigma} \sigma^T FD = \max_{D} \min_{i} (FD)_i.$$

From the above three equations, we have

$$\min_{\sigma} \max_{j} (\sigma^T F)_j = \max_{D} \min_{i} (FD)_i.$$

Hence considering randomized algorithms over worst case inputs (LHS) is the same as considering worst case distributions for deterministic algorithms (RHS).