CS-448 Sublinear Algorithms for Big Data Analysis February 26, 2025

Lecture 2

Lecturer: Michael Kapralov

1 Approximate counting continued

In the previous lecture we considered is the *counting problem*: maintain an approximate counter that counts up to at most n using a small amount of space. Morris' algorithm, presented below, maintains an $O(\log\log n)$ bit counter X that, after some modifications that we discuss below, leads to an (ε, δ)-approximate counter:

Definition 1 ((ϵ, δ)-approximate counter) *A (randomized) algorithm provides an (ϵ, δ)-approximation $\widehat{n}$ for a counter n if*

$$\Pr\left[(1-\varepsilon)n \leq \widehat{n} \leq (1+\varepsilon)n\right] \geq 1-\delta,$$

where the probability is over the internal randomness of the algorithm.

(1) We maintain a counter X: $X \leftarrow 0$

(2) For each event, increment X w.p. 2^{-X}

(3) Output $2^X - 1$

Why is the space complexity of Morris' algorithm $O(\log\log n)$? Of course, the counter in the algorithm above will be incremented a linear in n number of times with some positive probability, so in the worst case we will need $\Omega(\log n)$ bits to store X. However, whenever the algorithm provides a useful bound, i.e. whenever,

$$2^X - 1 \leq Cn$$

for some absolute constant $C > 1$, say, we have $2^X \leq Cn + 1$ and therefore $X \leq \log_2(Cn+1)$ and thus X can be represented by

$$\log_2\lceil \log_2(Cn+1)\rceil = \log_2\log_2 n + O(1)$$

bits.

We now turn to analyzing the algorithm. In the previous lecture we proved that our estimate is unbiased:

Claim 2 *Let X_n be the value of X after the occurence of n events, then*

$$\mathbb{E}[2^{X_n}] = n+1$$

To show that our estimator is indeed good, we need to show that our estimate is close to n with high probability. Given that our estimate is unbiased, we use Chebyshev's inequality to obtain a bound on the probability of failure by bounding the variance of our estimator. Let $\tilde{n}$ be the output of the algorithm after n events occur, i.e

$$\tilde{n} = 2^{X_n} - 1.$$

By Chebyshev's inequality we get

$$\Pr[|\tilde{n} - \mathbb{E}[\tilde{n}]| > \epsilon n] = \Pr[|\tilde{n} - n]| > \epsilon n] \leq \frac{Var[\tilde{n}]}{n^2\epsilon^2}$$

Note that first equality follows from the fact that our estimate is unbiased. In order to bound the variance, we will use the following bound on the second moment of 2^{X_n}, whose proof is left as an exercise:

Claim 3 $\mathbb{E}[2^{2X_n}] = \frac{3}{2}n^2 + \frac{3}{2}n + 1$ (exercise set I)

Using claims (2) & (3), we show that the variance of $\tilde{n}$ is on the order of n^2.

$$\begin{aligned} Var[\tilde{n}] = \mathbb{E}[(\tilde{n} - n)^2] &= \mathbb{E}[(2_n^X - (n+1))^2] \\ &= \mathbb{E}[2^{2X_n}] - (n+1)^2 \leq \frac{n^2}{2} \end{aligned}$$

Thus, the probability of failure of Morris' Algorithm is

$$\Pr[|\tilde{n} - n]| > \epsilon n] \leq \frac{Var[\tilde{n}]}{n^2\epsilon^2} = \frac{1/2n^2}{\epsilon^2 n^2} = \frac{1}{2\epsilon^2} \quad (1)$$

Isn't this more than 1?

If ε is small, then the right hand side is more than 1, which is not good as we are upper bounding the probability of failure. Still, we note that the above bound provides some useful information: if $\varepsilon \gg 1$, it shows that our estimator $\tilde{n}$ is unlikely to grossly overestimate the true answer n. On the other hand, the bound above leaves open the possibility that $\tilde{n}$ is always 0, for example, so we need to do better.

For sufficiently large ϵ? :)

The problem with (1) is the the variance of $\tilde{n}$ is too large – we need to reduce it. To reduce variance we will run a number t (to be determined later) of independent rounds of Morris Algorithm, and then output the average as an estimate. Given that the outputs are i.i.d random variables, the variance goes down, but the mean doesn't change. Guided by this observation, we introduce Morris+, a modified version of Morris.

1.0.1 Morris+

Morris+ Algorithm provides an (ϵ, δ)-approximation by repeating Morris Algorithm many times and then returning the average.

(1) For any t, maintain t copies of Morris Algorithm, let $X_n^1, X_n^2 \cdots X_n^t$ be the output of the t copies.

(2) Output $\frac{1}{t}\sum_{j=1}^{t}(2^{X_n^j} - 1)$

By linearity of expectation, the expected value of Morris+'s estimate is still

n. However, the variance becomes smaller. In particular, the variance of the estimate is now

$$Var\left[\frac{1}{t}\sum_{j=1}^{t}(2^{X_n^j}-1)\right] = \frac{1}{t^2}\sum_{j=1}^{t}Var[(2^{X_n^j}-1)] = \frac{Var(X_n^1)}{t}$$

where the first equality follows from the independence of the random variables, and the second equality follows by noting that X_n^j have identical distributions. By Chebyshev's inequality,

$$\Pr[|\tilde{n}-n]| > \epsilon n] \leq \frac{Var[\tilde{n}]}{n^2\epsilon^2} = \frac{n^2}{2t\epsilon^2 n^2} = \frac{1}{2t\epsilon^2}$$

To get a δ probability of error, we chose t to be at least $\frac{1}{2\epsilon^2\delta}$. Note that the space required by Morris+ becomes $O(\epsilon^{-2}\delta^{-1}\log\log n)$. Thus, we now obtain an algorithm that requires $O(\log\log n)$ space and returns $(1\pm 1/2)$-approximation with high constant probability (say 9/10). Ideally, we would like to design an algorithm with better than linear dependency on $1/\delta$, so we improve this algorithm even further by employing the "median trick".

The 'median trick' is really as much of a trick as the birthday phenomenon is a paradox

1.1 Morris++

The idea is to run $t = O(\log\frac{1}{\delta})$ copies of Morris+ (the choice of t will be justified in the proof below) with failure probability probability 1/3 (any constant less than half would suffice), and output the median answer. The space requirement of Morris++ now becomes $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\log\log n)$.

Note that Morris++ fails if and only if at least 1/2 of estimates fail (since we are returning the median as an estimate). We define for $i \in [1,t]$

$$Y_i = \begin{cases} 0 & \text{if the } i\text{-th estimate fails} \\ 1 & \text{o.w.} \end{cases}$$

Note that $\mathbb{E}[Y_i] \geq 2/3$. So, the probability that Morris++ fail is

$$\begin{aligned}
\Pr\left[\sum_{i=1}^{t}Y_i \leq \frac{t}{2}\right] &= \Pr\left[\sum_{i=1}^{t}(Y_i - \mathbb{E}[Y_i]) \leq \frac{t}{2} - \sum\mathbb{E}[Y_i]\right] \\
&= \Pr\left[\sum_{i=1}^{t}(Y_i - \mathbb{E}[Y_i]) \leq t(1/2-2/3)\right] \\
&\leq \Pr\left[\left|\sum_{i=1}^{t}(Y_i - \mathbb{E}[Y_i])\right| \geq (2/3-1/2)t\right] \\
&\leq \Pr\left[\left|\sum_{i=1}^{t}(Y_i - \mathbb{E}[Y_i])\right| \geq (2/3-1/2)\sum_{i=1}^{t}\mathbb{E}[Y_i]\right] \\
&= \Pr\left[\left|\sum_{i=1}^{t}(Y_i - \mathbb{E}[Y_i])\right| \geq (1/6)\sum_{i=1}^{t}\mathbb{E}[Y_i]\right]
\end{aligned}$$

By the Chernoff bound we now get

$$\Pr\left[\sum_{i=1}^{t}Y_i \leq \frac{t}{2}\right] \leq 2e^{\frac{-(1/6)^2(2t/3)}{3}}$$

To get a δ probability of failure, we choose $t = C \log \frac{1}{\delta}$ for some large enough $C > 0$.

1.2 Summary

We used the approximate counting problem to illustrate the following two ideas that are commonly used in the design of streaming algorithms:

$1/\epsilon^2$ factor in space complexity

1. If we have an estimator with a large variance, then we can reduce the variance and thereby improve the precision of the estimator by averaging (linear operation) several independent copies of the estimator.

$\log 1/\delta$ factor in space complexity

2. If we have an algorithm that gives a strictly greater than $1/2$ probability of success, then we can boost the success by repeating it a sufficient number of times (independently) and using the median (non-linear operation) as an estimate.

2 Distinct Elements Problem

In the *distinct elements problem*, we are given a stream of elements between 1 and n, and we wish to compute the number of distinct elements occurring in the stream. More formally, let x be a n-dimensional vector, where x_i denotes the frequency of i in the stream, we want to compute $||x||_0$ (the number of distinct elements). For example, think of computing the number of distinct queries on Google.com over a period of time, or counting the number of distinct cities on Instagram[1].

To obtain the exact solution, the best we can do is store all the distinct elements occurring in the stream. However, this can be the entire set i.e $\Omega(n)$, and our goal is to obtain an $o(n)$-space algorithm. To obtain a sublinear space algorithm we relax our requirements, and we ask for an (ε, δ)-approximation, namely a streaming algorithm that provides an approximation $\tilde{k}$ of $||x||_0$ such that

$$\Pr\left[||x||_0 \leq \tilde{k} \leq (1+\epsilon)k\right] \geq 1 - \delta \tag{2}$$

Before solving this problem, we reduce it to a simpler decision problem.

2.1 Distinct Elements: Decision Version

In this version of the problem, we are given a threshold t and we want to distinguish between the following two cases with "high" probability.

- **YES** : $||x||_0 \geq 2t$
- **NO** : $||x||_0 < t$

[1] HyperLogLog is a very practical distinct elements sketch used for such applications. The main ideas behind it are very similar to what we present here. A similar method was used by the Allied Forces during World War II – see the http://en.wikipedia.org/wiki/German_tank_problem

(1) Select a set $S \subseteq [n]$ by including every $i \in n$ independently with probability $\frac{1}{t}$

(2) Maintain $C = \sum_{i \in S} x_i$

(3) Output **YES** if $C > 0$, and **NO** otherwise.

Let $||x||_0 = k$, then the probability of outputting **YES** is as follows

$$\begin{aligned}\Pr[C > 0] &= \Pr[\text{S contains an element in the stream}] \\ &= 1 - \Pr[\text{S doesn't contain an element in the stream}] \\ &= 1 - \left(1 - \frac{1}{t}\right)^k\end{aligned}$$

which behaves differently under the different regimes.

- **NO** case: $\Pr[C > 0] = 1 - (1 - \frac{1}{t})^k \leq 1 - (1 - \frac{1}{t})^t \approx 1 - e^{-1} \approx 0.64$
- **YES** case: $\Pr[C > 0] = 1 - (1 - \frac{1}{t})^k \geq 1 - (1 - \frac{1}{t})^{2t} \approx 1 - e^{-2} \approx 0.86$

Proof by calculus!

Note that these bounds hold for t large enough since we used the approximation $(1 - \frac{1}{x})^x \approx e^{-1}$.

From the above analysis, we know that we can differentiate between the **YES** and the **NO** case. However, there is a small gap between the two cases, and this would result in a large error probability. To amplify the gap between them and obtain an arbitrarily small probability of error, we do the following. Let

$$Y_i = \begin{cases} 1 & \text{if the experiment output } \textbf{YES} \\ 0 & \text{o.w.} \end{cases}$$

(1) Repeat the experiment m times independently.

(2) Output **YES** if $\sum_{i=1}^{m} Y_i \geq 0.7m$, and **NO** otherwise.

One can easily show that we can distinguish between the two cases with probability at least $1 - \delta$ by setting $m = C \log \frac{1}{\delta}$ for some large enough $C > 0$. Similar to the Morris++ analysis, we select m by showing that the error probability decay exponentially in m. Note that there is nothing magical in the choice of 0.7, any constant in the range $(0.64, 0.86)$ would work. This gives an algorithm for distinguishing between the two cases with space complexity $O(\log n \log(1/\delta))$ that succeeds with probability at least $1 - \delta$, but our bound on the space complexity does not take into account the storage needed for S. If S is truly a random subset as defined above, however, it is not possible to store it compactly. We next design a version of the algorithm that uses pseudorandom S that is easy to store, but works almost as well for our purpose.