Lecture 04 & 05 — October 9 & 12, 2020

Prof. Cyrus Rashtchian *Topics: Approximate Counting*

Overview. In the last few lectures, we surveyed the topics for this whole course, reviewed some probability theory, discussed various complexity measures for algorithms. Today we will provide our first sketching result: Robert Morris's method of counting large numbers in a small register [Mor78].

1 Approximate Counting

We will now discuss our first detailed example of a sketching algorithm. In the following, we discuss a problem first studied by Robert Morris [Mor78]. This is essentially the first streaming paper, from 1978, way before big data was a thing. The motivation for him was to understand space-bounded devices (back when space/memory was expensive).

Problem. Design an algorithm that monitors a sequence of events and upon request can output (an estimate of) the number of events thus far. More formally, create a data structure that maintains a single integer n and supports the following operations.

- `init();` sets $n \leftarrow 0$
- `update();` increments $n \leftarrow n + 1$
- `query();` outputs n or an estimate of n

A trivial algorithm stores n as a sequence of $\lceil \log n \rceil = O(\log n)$ bits (a counter).

Question. Can you solve this problem using fewer than $\log n$ bits??

Nope. If we want `query();` to return the exact value of n, this is the best we can do. Suppose for some such algorithm, we use $f(n)$ bits to store the integer n. There are $2^{f(n)}$ configurations for these bits. In order for the algorithm to be able to store the exact value of all integers up to n, the number of configurations must be greater than or equal to the number n. Hence,

$$2^{f(n)} \geq n \Rightarrow f(n) \geq \log n.$$

Approximate Solution. The goal is to use much less space than $O(\log n)$, and so we must instead answer `query();` with some estimate $\tilde{n}$ of n. We would like this $\tilde{n}$ to satisfy

$$\Pr(|\tilde{n} - n| > \varepsilon n) < \delta, \tag{1}$$

for some $0 < \varepsilon, \delta < 1$ that are given to the algorithm up front. For example, get the answer to a factor of $\varepsilon = 1/10$ with probability 90%.

Morris's algorithm provides such an estimator for some ε, δ that we will analyze shortly. We assume the algorithm has a perfect source of randomness. Then, the algorithm works as follows:

- `init();` sets $X \leftarrow 0$
- `update();` increments X with probability 2^{-X}
- `query();` outputs $\tilde{n} = 2^X - 1$

Intuitively, the variable X is attempting to store a value that is approximately $\log_2 n$. Before giving a rigorous analysis in Section 2, we first give a probability review.

2 Analysis of Morris's Algorithm

Let X_n denote X in Morris's algorithm after n updates. Let $\widetilde{n} = 2^{X_n} - 1$ be the output.

We first analyze the expectation, then the variance, and then use the concentration bounds from above to provide the overall analysis.

Claim 1. *For Morris's algorithm,* $\mathbb{E}2^{X_n} = n + 1$.

Proof. We will prove by induction. Consider the base case where $n = 0$. We have initialized $X \leftarrow 0$ and have yet to increment it. Thus, $X_n = 0$, and $\mathbb{E}2^{X_n} = n + 1$. Now suppose that $\mathbb{E}2^{X_n} = n + 1$ for some fixed n.

We have

$$
\begin{aligned}
\mathbb{E}2^{X_{n+1}} &= \sum_{j=0}^{\infty} \Pr(X_n = j) \cdot \mathbb{E}(2^{X_{n+1}} \mid X_n = j) \\
&= \sum_{j=0}^{\infty} \Pr(X_n = j) \cdot \left(2^j \left(1 - \frac{1}{2^j}\right) + \frac{1}{2^j} \cdot 2^{j+1}\right) \\
&= \sum_{j=0}^{\infty} \Pr(X_n = j) 2^j + \sum_{j=0}^{\infty} \Pr(X_n = j) \\
&= \mathbb{E}2^{X_n} + 1 \\
&= (n+1) + 1.
\end{aligned}
$$

This completes the inductive step. □

It is now clear why we output our estimate of n as $\tilde{n} = 2^X - 1$: it is an unbiased estimator of n. Moreover, since $X \approx \log n$, the expected amount of space we use is $O(\log \log n)$.

In order to show (1) however, we will also control on the variance of our estimator. This is because, by Chebyshev's inequality,

$$\Pr(|\tilde{n} - n| > \varepsilon n) < \frac{1}{\varepsilon^2 n^2} \cdot \mathbb{E}(\tilde{n} - n)^2 = \frac{1}{\varepsilon^2 n^2} \cdot \mathbb{E}(2^{X_n} - 1 - n)^2.$$

When we expand the above square, we find that we need to control $\mathbb{E}2^{2X_n}$. The proof of the following claim is by induction, similar to that of Claim 1.

Claim 2. *For Morris's algorithm, we have*

$$\mathbb{E}2^{2X_n} = \frac{3}{2}n^2 + \frac{3}{2}n + 1. \tag{2}$$

Proof. We again prove this by induction. It is clearly true for $n = 0$. Then

$$\begin{aligned}
\mathbb{E}2^{2X_{n+1}} &= \sum_{j=0}^{\infty} \Pr(2^{X_n} = j) \cdot \mathbb{E}(2^{2X_{n+1}} \mid 2^{X_n} = j) \\
&= \sum_{j=0}^{\infty} \Pr(2^{X_n} = j) \cdot \left(\frac{1}{j} \cdot 4j^2 + \left(1 - \frac{1}{j}\right) \cdot j^2\right) \\
&= \sum_{j=0}^{\infty} \Pr(2^{X_n} = j) \cdot (j^2 + 3j) \\
&= \mathbb{E}2^{2X_n} + 3 \cdot \mathbb{E}2^{X_n} \\
&= \left(\frac{3}{2}n^2 + \frac{3}{2}n + 1\right) + (3n + 3) \\
&= \frac{3}{2}(n+1)^2 + \frac{3}{2}(n+1) + 1
\end{aligned}$$

This completes the inductive step. □

Bounding the failure probability. Now note $\mathrm{Var}[Z]$ in general is equal to $\mathbb{E}Z^2 - (\mathbb{E}Z)^2$. Also, $\mathrm{Var}[2^{X_n} - 1] = \mathrm{Var}[2^{X_n}]$. These together imply that

$$\mathrm{Var}[2^{X_n}] = \mathbb{E}[2^{2X_n}] - (\mathbb{E}[2^{2X_n}])^2 = \frac{3}{2}n^2 + \frac{3}{2}n + 1 - (n+1)^2 = \frac{1}{2}n^2 - \frac{1}{2}n < \frac{1}{2}n^2$$

and thus

$$\Pr(|\tilde{n} - n| > \varepsilon n) < \frac{1}{\varepsilon^2 n^2} \cdot \frac{n^2}{2} = \frac{1}{2\varepsilon^2},$$

which is not particularly meaningful since the right hand side is only smaller than 1 when $\varepsilon > 1/\sqrt{2}$, and otherwise it says nothing. But we really want it to work for any ε.

2.1 Morris+

To decrease the failure probability of Morris's basic algorithm, we instantiate s independent copies of Morris's algorithm and average their outputs. That is, we obtain independent estimators $\tilde{n}_1, \ldots, \tilde{n}_s$ from independent instantiations of Morris's algorithm, and our output to a query is

$$\tilde{n}^+ = \frac{1}{s} \cdot \sum_{i=1}^{s} \tilde{n}_i$$

Since each $\tilde{n}_i$ is an unbiased estimator of n, so is their average. Furthermore, since variances of independent random variables add, and multiplying a random variable by some constant $c = 1/s$ causes the variance to be multiplied by c^2, the right hand side of (2) becomes

$$\Pr(|\tilde{n}^+ - n| > \varepsilon n) < \frac{1}{2s\varepsilon^2} < \delta$$

for $s = 1/(2\varepsilon^2\delta) = \Theta(1/(\varepsilon^2\delta))$. This is pretty good, but we can do even better.

2.2 Morris++

There is a simple technique to reduce the dependence on the failure probability δ from $1/\delta$ down to $\log(1/\delta)$. This method is known as a **median-of-means** estimator. The technique is as follows.

We run t instantiations of Morris+, which we denote $\tilde{n}_1^+, \tilde{n}_2^+, \ldots \tilde{n}_t^+$. For each, we will achieve failure probability $\frac{1}{3}$ by taking the mean of $s = \Theta(1/\varepsilon^2)$ Morris estimators. We then output the median estimate from all the t Morris+ instantiations.

$$\tilde{n}^{++} = \text{median}(\tilde{n}_1^+, \tilde{n}_2^+, \ldots \tilde{n}_t^+).$$

We can calculate the expected space usage. Each Morris run takes space $O(\log\log n)$ in expectation. Each Morris+ run uses $s \approx 1/\varepsilon^2$ copies of Morris, leading to space roughly $\log\log(n)/\varepsilon^2$ for each. We will see shortly that there are $t \approx \log(1/\delta)$ copies of Morris+ in the overall Morris++ algorithm. Therefore, we have that the expected space of Morris++ will roughly be

$$s \cdot t \cdot \log\log n \;\simeq\; \frac{\log\log n}{\varepsilon^2} \cdot \log(1/\delta).$$

Analysis. Say that the ith Morris+ estimate **succeeds** if $|\tilde{n}^+ - n| < \varepsilon n$, and otherwise it fails.

The expected number of Morris+ instantiations that succeed is at least $2t/3$. For the median to be a bad estimate, less than half the Morris+ instantiations can succeed (or more than half must fail). This implies that number of succeeding instantiations deviated from its expectation by at least $t/6$.

To analyze Morris++, we define the following t indicator variables:

$$Y_i = \begin{cases} 1, & \text{if the } i\text{-th Morris+ instantiation succeeds.} \\ 0, & \text{otherwise.} \end{cases}$$

Then by the Chernoff bound,

$$\Pr\left(\sum_{i=1}^{t} Y_i \le \frac{t}{2}\right) \le \Pr\left(\left|\sum_{i=1}^{t} Y_i - \mathbb{E}\sum_{i=1}^{t} Y_i\right| \ge \frac{t}{6}\right) \le 2e^{-ct} < \delta,$$

for a constant c, where $c = 1/48$ seems to work. The final inequality "$< \delta$" holds by setting the number of estimators to be $t = \Theta(\log(1/\delta))$.

This implies that $|\tilde{n}^{++} - n| < \varepsilon n$ with probability at least $1 - \delta$, as desired.

Overall space complexity. Note the space is a ranadom variable. We will not show it here, but one can show that the total space complexity is, with probability $1 - \delta$, at most

$$O(\varepsilon^{-2} \log(1/\delta)(\log\log(n/(\varepsilon\delta))))$$

bits. In particular, for constant ε, δ (say each $1/100$), the total space complexity is $O(\log\log n)$ with constant probability. This is exponentially better than the $\log n$ space achieved by storing a counter.

An improvement. One issue with the above is that the space is $\Omega(\varepsilon^{-2} \log\log n)$ for $(1+\varepsilon)$-approximation, but the obvious lower bound is only

$$O(\log(\log_{1+\varepsilon} n)) = O(\log(1/\varepsilon) + \log\log n).$$

This can actually be achieved. Instead of incrementing the counter with probability $1/2^X$, we do it with probability $1/(1+a)^X$ and choose $a > 0$ appropriately. We leave it to the reader as an exercise to find the appropriate value of a and to figure out how to answer queries.

References

[Mor78] Robert Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, 10 1978.