These lecture notes have not undergone rigorous peer-review. Please email quanquan.liu@yale.edu if you see any errors.

1 Introduction

In the first two lectures, we covered techniques for estimating the frequency of events in a data stream using $o(\log n)$ space; specifically, we discussed techniques for obtaining a $(1+\varepsilon)$-approximation of the number of bits seen in a stream using $O\left(\frac{\log\log n}{\varepsilon}\log\left(\frac{1}{\delta}\right)\right)$ space with probability at least $1-\delta$. Our discussion centers around the Morris algorithm, which uses a single counter to estimate the number of increments in a stream. Then, we use the Median Trick, which uses multiple IID estimators, to improve the accuracy and reduce the probability of failure.

2 The Morris Algorithm

The Morris algorithm is a simple and elegant algorithm for answering queries asking for the estimating the number of 1 bits seen in a stream using only $O(\log\log n)$ bits of space, where n is the true total number of bits seen thus far. Here, and throughout, we label each one bit as an *increment*. The algorithm maintains a counter X that is initially zero. Whenever an increment occurs, the algorithm increases X by one with probability $1/2^X$. The final estimate is $\tilde{C} = 2^X - 1$.

A query can occur after any number of increments. Hence, we denote the counter X that occurs after i increments by X_i and the estimate by $\tilde{C}_i$.

Algorithm 1 Morris Algorithm

```
X ← 0
for each bit in the stream do
    p ← 1/2^X
    r ← random number in [0, 1]
    if r < p then
        X ← X + 1
    end if
    Upon a query, return 2^X − 1
end for
```

The intuition behind the algorithm is that the counter X grows exponentially with the number of increment; therefore, the estimate $\tilde{C}$ is also an exponential function of X. The algorithm trades off space for accuracy, as it uses fewer bits to store X than to store n, but introduces some error in the estimation.

2.1 Analysis of the Morris Algorithm

We will first analyze the expected value and the variance of the estimate $\tilde{C}_n$ (after n increments) produced by the Morris algorithm. We first prove the expectation.

Lemma 2.1. *The expected value of $\tilde{C}_n$ is equal to n, i.e., $\mathbb{E}[\tilde{C}_n] = n$.*

Proof. First, we note that $\mathbb{E}[\tilde{C}_n] + 1 = \mathbb{E}[\tilde{C}_n + 1] = \mathbb{E}[2^{X_n}]$. Hence, proving $\mathbb{E}[2^{X_n}] = n + 1$ directly proves our lemma. We will prove this simpler expression.

We will use induction on the number of increments. For the base case, when there are no increments, we have $X_0 = 0$ and $2^{X_0} = 1$, so $\mathbb{E}[2^{X_0}] = 1$. For the induction step, assume that after $n-1$ increments, we have $\mathbb{E}[2^{X_{n-1}}] = n$. Then, after the n-th increment, we have

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

where we sum over all possible values of 2^{X_n}, condition on the possible values of X_{n-1} (where 2^{X_n} is a deterministic function of X_n), and the induction hypothesis in the last step. Therefore, by induction, we have $\mathbb{E}[2^{X_n}] = n + 1$ for any number of n and $\mathbb{E}[\tilde{C}_n] = n$. □

Now, we would like to provide concentration bounds on the value of $\tilde{C}_n$. Using the Markov bound, we can obtain an easy upper bound with constant probability.

Theorem 1 (Markov's Inequality)**.** *Markov's Inequality states that for any non-negative random variable X and any positive $\lambda > 0$, the probability that X is at least λ is less than or equal to the expected value of X divided by λ. Formally, it is expressed as:*

$$P(X \geq \lambda) \leq \frac{\mathbb{E}[X]}{\lambda}$$

where $\mathbb{E}[X]$ denotes the expected value of X.

Using Markov's inequality and Lemma 2.1, if we let $\lambda = 10\mathbb{E}[\tilde{C}_n]$, then with probability at least 0.9, $\tilde{C}_n$ does nto exceed $10n$. However, we also want to lower bound the estimate. To lower bound the estimate, we can use Chebyshev's inequality that also gives a tighter bound.

Theorem 2 (Chebyshev's Inequality)**.** *Chebyshev's Inequality states that for any (not necessarily positive) random variable X with finite expected value μ and finite non-zero variance σ^2, the probability that X is more than k standard deviations away from μ is at most $1/k^2$. Formally, it is expressed as:*

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

for all $k > 0$.

To use Chebyshev's inequality, we must first compute the variance, $\mathbf{Var}[\tilde{C}_n]$.

Theorem 2.2. *The variance of $\tilde{C}_n$ is $\frac{n\cdot(n-1)}{2}$, i.e., $\mathbf{Var}[\tilde{C}_n] = \frac{n\cdot(n-1)}{2}$.*

Proof. We will use a similar induction argument as before for the expectation. Again, we first simplify our expression by noting $\mathbf{Var}[\tilde{C}_n] = \mathbf{Var}[2^{X_n} - 1] = \mathbf{Var}[2^{X_n}]$. Also, we further know that $\mathbf{Var}[2^{X_n}] = \mathbb{E}[2^{2X_n}] - \left(\mathbb{E}[2^{X_n}]\right)$. By Lemma 2.1, $\left(\mathbb{E}[2^{X_n}]\right) = (n+1)^2$. Hence, all that's left for us to compute is $\mathbb{E}[2^{2X_n}]$. We will prove via induction that $\mathbb{E}[2^{2X_n}] = \frac{3n(n+1)}{2} + 1$.

For the base case, when there are no increments, we have $X_0 = 0$ and $\mathbb{E}[2^{2X_0}] = 1$. For the induction step, assume that after $n-1$ increments, we have $\mathbb{E}[2^{2X_{n-1}}] = \frac{3n(n-1)}{2} + 1$. Then, after the n-th increment, we have (following the same calculations we performed for the expectation),

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

Therefore, by induction, we have $\mathrm{Var}[\tilde{C}_n] = \mathbb{E}[2^{2X_n}] - \left(\mathbb{E}[2^{X_n}]^2\right) = \frac{3n(n+1)}{2} + 1 - (n+1)^2 = \frac{n(n-1)}{2}$ for any number of increments n. □

Setting $k = \sqrt{10}$ to obtain that with probability at least 0.9, $\tilde{C}_n \leq (\sqrt{10}+1)\cdot n$. We also obtain a lower bound guarantee, but the lower bound guarantee is trivial since $\tilde{C}_n$ is non-negative. Hence, our remaining task is to obtain better approximation bounds as well as probability of success. We'll first tackle getting better approximation bounds.

3 Improving to $(1+\varepsilon)$-Approximation

We now introduce the *median-of-means* trick or otherwise colloquially known as the *median trick*. The technique works roughly as follows for appropriate settings of b and T:

- **Median-of-Means Trick**: The Median-of-Means trick is a method for reducing the variance of an estimator produced by an algorithm $\mathcal{A}$ by running T IID instances of $\mathcal{A}$, dividing the result into b blocks, computing the mean of each block, and then taking the median of these means. This can significantly improve the accuracy of the estimate, especially when the distribution of the estimator is skewed or heavy-tailed.
- **Algorithm**:
  1. Divide T IID estimators into b blocks.

2. Compute the mean of each block.
3. Take the median of these b means.

We'll first discuss the first two steps as applied to the Morris Algorithm. We'll call this new algorithm the Morris+ algorithm. The Morris+ algorithm is a simple extension of the Morris. It takes t IID instances of the Morris algorithm and outputs the mean of the returned values. We'll set t appropriately later to obtain our desired $(1+\varepsilon)$-approximation. We denote the i-th instance of Morris+ after n increments as X_n^i and $\tilde{C}_n^i$.

Algorithm 2 Morris+ Algorithm

Require: Stream of elements S, number of instances t
Ensure: Estimate of the count of elements in S

for $i = 1$ to t **do**
  Initialize counter X_i to 0.
end for
for each element e in S **do**
  for $i = 1$ to t **do**
    Generate a random number r in $[0, 1]$
    if $r < \frac{1}{2^{X_i}}$ **then**
      Increment X_i
    end if
  end for
  On a query, compute the mean of $2^{X_1} - 1, 2^{X_2} - 1, \ldots, 2^{X_t} - 1$ and **return** the mean as the estimate of the count of elements in S
end for

Let $\tilde{A}_n = \frac{1}{t}\sum_{i=1}^{t} \tilde{C}_n^i$. Then, by linearity of expectations, $\mathbb{E}[\tilde{A}_n] = n$ by Lemma 2.1.' We'll now compute the variance.

Lemma 3.1. *The variance of $\tilde{A}_n$ is $\frac{n\cdot(n-1)}{2t^2}$, i.e., $\mathbf{Var}[\tilde{A}_n] = \frac{n\cdot(n-1)}{2t^2}$.*

Proof. The calculation proceeds quite simply because all $\tilde{C}_n^i$ are IID:

$$\begin{aligned}\mathbf{Var}[\tilde{A}_n] &= \mathbf{Var}\left[\frac{1}{t}\sum_{i=1}^{t}\tilde{C}_n^i\right] \\ &= \frac{1}{t^2}\cdot\sum_{i=1}^{t}\mathbf{Var}[\tilde{C}_n^i] \\ &= \frac{1}{t^2}\cdot\frac{n(n-1)}{2}.\end{aligned}$$

□

So we see that simply by repeating the algorithm t times and taking the mean, we've reduced the variance by a factor of $\frac{1}{t}$. Setting, $t = \frac{\sqrt{10}}{\varepsilon^2}$ gives our desired $(1+\varepsilon)$-approximation using Chebyshev's inequality with at least 0.9 probability when $k = \sqrt{10}$.

Given our desired approximation, we now need to amplify the probability of success.

3.1 Amplifying the Probability of Success by Taking the Median

The idea is to run the algorithm multiple times independently and take the median of the outputs as the final output. This method can reduce the probability of failure to any desired level by increasing the number of repetitions logarithmically in terms of the inverse of the probability of failure.

We will analyze the median trick using the Chernoff bound, which is a useful tool for bounding the tail probabilities of sums of independent random variables.

Theorem 3 (Chernoff Bound)**.** *The Chernoff Bound is a probabilistic inequality that provides an upper bound on the tail distribution of sums of independent random variables. There are many variants of the bound; we present the common multiplicative version. Formally, it is expressed as:*

$$\textbf{\textit{Upper Tail:}}\ \textit{for any}\ \psi > 0, \quad P(X \geq (1+\psi)\mu) \leq e^{-\frac{\psi^2\mu}{3}}$$

and

$$\textbf{\textit{Lower Tail:}}\ \textit{for any}\ \psi \in (0,1), \quad P(X \leq (1-\psi)\mu) \leq e^{-\frac{\psi^2\mu}{2}},$$

where X is a random variable representing the sum of independent Bernoulli trials, and μ is the expected value of X.

Now, we can extend Morris+ quite simply by running r IID copies of the algorithm and then taking the median of the results. We call this the Morris++ algorithm.

Algorithm 3 Morris++ Algorithm using Median

Require: Stream of elements S, number of instances t, number of runs r
Ensure: Median of the estimates from r runs of Morris+

```
Initialize an empty list L
for j = 1 to r do
    for i = 1 to t do
        Initialize counter X_i to 0
    end for
end for
for each element e in S do
    for j = 1 to r do
        for i = 1 to t do
            Generate a random number p in [0, 1]
            if p < 1/2^{X_i} then
                Increment X_i
            end if
        end for
        Compute the mean of 2^{X_1} − 1, 2^{X_2} − 1, ..., 2^{X_t} − 1
        Add the mean to the list L
    end for
    On a query, sort the list L and return the median of the list L
    Empty L
end for
```

Let $Y_n^i = 1$ if $\tilde{A}_n^i$ gives a $(1+\varepsilon)$-approximation of n and $Y_n^i = 0$, otherwise. Clearly, the Y_n^i variables are Bernoulli variables. Then, it follows that $\mathbb{E}[Y_n^i] = \mathsf{Pr}(Y_i = 1) \geq 0.9$. Let $\tilde{J}_n = \sum_{i=1}^{r} Y_n^i$. This means

that $\mathbb{E}[\tilde{J}_n] = 0.9r$. When $\tilde{J}_n > 0.5r$, we'll obtain a $(1+\varepsilon)$-approximation of n if we take the median!

Now, we only need to upper bound the probability that $\tilde{J}_n \leq 0.5r$.

Lemma 3.2. *The probability that $\tilde{J}_n \leq 0.5r$ is at most $e^{-0.4^2 \cdot 0.9 \cdot r/2}$.*

Proof. We directly use the lower tail bound of the Chernoff bound. Substituting $\psi = 0.4$, we get

$$\mathsf{Pr}(\tilde{J}_n \leq (1-0.4) \cdot 0.9r) \leq e^{-0.4^2 \cdot 0.9 \cdot r/2}.$$

□

In order to obtain at most a probability of failure of δ, we set r such that $e^{-0.4^2 \cdot 0.9 \cdot r/2} \leq \delta$. Solving, we obtain $r \geq \frac{2\ln(1/\delta)}{0.4^2 \cdot 0.9} = O(\log(1/\delta))$.

Altogether, we see that Morris++ gets a $(1+\varepsilon)$-approximation of n with probability at least $1-\delta$ when the space used is $O\left(\frac{\log(1/\delta)\log\log n}{\varepsilon^2}\right)$.

This does not quite get us the high probability bound in $o(\log n)$ space, but very recently, Nelson and Yu [NY22] showed an (optimal) space bound of $O(\log\log n + \log(1/\varepsilon) + \log(1/\delta))$ that gets a $(1+\varepsilon)$-approximation and does obtain the desired high probability bound.

References

[NY22] Jelani Nelson and Huacheng Yu. Optimal bounds for approximate counting. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 119–127, 2022.