Sketching and streaming — Notes 1

Jelani Nelson
minilek@seas.harvard.edu

July 18, 2016

1 Intro

Sketching and streaming. A *sketch* $C(X)$ of some data set X with respect to some function f is a *compression* of X that allows us to compute, or approximately compute, $f(X)$ given access only to $C(X)$. Sometimes f has 2 (or multiple) arguments, and for data X and Y, we want to compute $f(X, Y)$ given $C(X), C(Y)$. For example, if two servers on a network want to compute some similarity or distance measure on their data, one can simply send the sketch to another (or each to a third party), which reduces network bandwith.

As a trivial example, consider the case that Alice has a data set X which is a set of integers, and Bob has a similar data set Y. They want to compute $f(X, Y) = \sum_{z \in X \cup Y} z$. Then each party can let the sketch of their data simply be the sum of all elements in their data set.

When designing *streaming* algorithms, we want to maintain a sketch $C(X)$ on the fly as X is updated. In the previous example, if say Alice's data set is being inserted into on the fly then she can of course maintain a sketch by keeping a running sum. The streaming setting appears in many scenarios, such as for example an Internet router monitoring network traffic, or a search engine monitoring a query stream.

2 Approximate counting

In the following, we discuss a problem first studied in [Mor78].

Problem. Our algorithm must monitor a sequence of events, then at any given time output (an estimate of) the number of events thus far. More formally, this is a data structure maintaining a single integer n and supporting the following two operations:

- **update():** increments n by 1
- **query():** must output (an estimate of) n

Before any operations are performed, it is assumed that n starts at 0. Of course a trivial algorithm maintains n using $\lceil \log n \rceil$ bits of memory (a counter). Our goal is to use much less space than this. It is not too hard to prove that it is impossible to solve this problem exactly using $o(\log n)$ bits of space. Thus we would like to answer **query()** with some estimate $\tilde{n}$ of n satisfying

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

for some $0 < \varepsilon, \delta < 1$ that are given to the algorithm up front.

The algorithm of Morris provides such an estimator for some ε, δ that we will analyze shortly. The algorithm works as follows:

1. Initialize $X \leftarrow 0$.
2. For each update, increment X with probability $\frac{1}{2^X}$.
3. For a query, output $\tilde{n} = 2^X - 1$.

Intuitively, the variable X is attempting to store a value that is $\approx \log_2 n$. Before giving a rigorous analysis in Section 4, we first give a probability review.

3 Probability Review

We are mainly discussing discrete random variables. For the remainder of this section we consider random variables taking values in some set $S \subset \mathbb{R}$. Recall the expectation of X is defined to be

$$\mathbb{E}\, X = \sum_{j \in S} j \cdot \mathbb{P}(X = j).$$

We now state a few basic lemmas and facts without proof.

Lemma 1 (Linearity of expectation)**.**

$$\mathbb{E}(X+Y) = \mathbb{E}\,X + \mathbb{E}\,Y \tag{2}$$

Lemma 2 (Markov)**.** *If X is a nonnegative random variable, then*

$$\forall \lambda > 0,\ \mathbb{P}(X > \lambda) < \frac{\mathbb{E}\,X}{\lambda}$$

Lemma 3 (Chebyshev)**.**

$$\forall \lambda > 0, \mathbb{P}(|X - \mathbb{E}\,X| > \lambda) < \frac{\mathbb{E}(X - \mathbb{E}\,X)^2}{\lambda^2} \tag{3}$$

Proof. $\mathbb{P}(|X - \mathbb{E}\,X| > \lambda) = \mathbb{P}((X - \mathbb{E}\,X)^2 > \lambda^2)$, and thus the claim follows by Markov's inequality. □

Rather than the second mmoment, one can also consider larger moments to obtain:

$$\forall p \geq 1, \forall \lambda > 0,\ \mathbb{P}(|X - \mathbb{E}\,X| > \lambda) < \frac{\mathbb{E}\,|X - \mathbb{E}\,X|^p}{\lambda^p}. \tag{4}$$

By a calculation and picking p optimally (or by an argument using the moment-generating function which we will not cover here; see e.g. `http://people.seas.harvard.edu/~minilek/cs229r/fall13/lec/lec1.pdf`), one can also obtain the following "Chernoff bound".

Lemma 4 (Chernoff)**.** *Suppose $X_1, \ldots, X_n$ are independent random variables with $X_i \in [0,1]$. Let $X = \sum_i X_i$. Then*

$$\forall 0 < \lambda < 1,\ \mathbb{P}(|X - \mathbb{E}\,X| > \lambda \cdot \mathbb{E}\,X) \leq 2 \cdot e^{-\lambda^2 \cdot \mathbb{E}\,X/3}. \tag{5}$$

4 Analysis of Morris' algorithm

Let X_n denote X in Morris' algorithm after n updates.

Claim 1.

$$\mathbb{E}\,2^{X_n} = n + 1.$$

Proof. We prove by induction on n. The base case is clear, so we now show the inductive step.

We have

$$\begin{aligned}\mathbb{E}\,2^{X_{n+1}} &= \sum_{j=0}^{\infty} \mathbb{P}(X_n = j)\cdot \mathbb{E}(2^{X_{n+1}} | X_n = j) \\ &= \sum_{j=0}^{\infty} \mathbb{P}(X_n = j)\cdot (2^j(1-\frac{1}{2^j}) + \frac{1}{2^j}\cdot 2^{j+1}) \\ &= \sum_{j=0}^{\infty} \mathbb{P}(X_n = j)2^j + \sum_j \mathbb{P}(X_n = j) \\ &= \mathbb{E}\,2^{X_n} + 1 \\ &= (n+1)+1\end{aligned} \tag{6}$$

□

It is now clear why we output our estimate of n as $\tilde{n} = 2^X - 1$: it is an unbiased estimator of n. In order to show (1) however, we will also control on the variance of our estimator. This is because, by Chebyshev's inequality,

$$\mathbb{P}(|\tilde{n} - n| > \varepsilon n) < \frac{1}{\varepsilon^2 n^2}\cdot \mathbb{E}(\tilde{n} - n)^2 = \frac{1}{\varepsilon^2 n^2}\,\mathbb{E}(2^X - 1 - n)^2. \tag{7}$$

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.

$$\mathbb{E}(2^{2X_n}) = \frac{3}{2}n^2 + \frac{3}{2}n + 1. \tag{8}$$

This implies $\mathbb{E}(\tilde{n} - n)^2 = (1/2)n^2 - (1/2)n - 1 < (1/2)n^2$, and thus

$$\mathbb{P}(|\tilde{n} - n| > \varepsilon n) < \frac{1}{\varepsilon^2 n^2}\cdot \frac{n^2}{2} = \frac{1}{2\varepsilon^2}, \tag{9}$$

which is not particularly meaningful since the right hand side is only better than $1/2$ failure probability when $\varepsilon \geq 1$ (which means the estimator may very well always be 0!).

4.1 Morris+

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

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

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 (9) becomes

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

for $s > 1/(2\varepsilon^2\delta) = \Theta(1/(\varepsilon^2\delta))$.

4.2 Morris++

It turns out there is a simple technique (which we will see often) to reduce the dependence on the failure probability δ from $1/\delta$ to $\log(1/\delta)$. The technique is as follows.

We run t instantiations of Morris+, each with failure probability $\frac{1}{3}$. Thus, for each one, $s = \Theta(1/\varepsilon^2)$. We then output the median estimate from all the s Morris+ instantiations. Note that the expected number of Morris+ instantiations that succeed is at least $2t/3$. For the median to be a bad estimate, at most half the Morris+ instantiations can succeed, implying the number of succeeding instantiations deviated from its expectation by at least $t/6$. Define

$$Y_i = \begin{cases} 1, & \text{if the } i\text{th Morris+ instantiation succeeds.} \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Then by the Chernoff bound,

$$\mathbb{P}(\sum_i Y_i \le \frac{t}{2}) \le \mathbb{P}(|\sum_i Y_i - \mathbb{E}\sum_i Y_i| \ge t/6) \le 2e^{-t/3} < \delta \tag{11}$$

for $t \in \Theta(\lg(1/\delta))$.

Overall space complexity. When we unravel Morris++, it is running a total of $st = \Theta(\lg(1/\delta)/\varepsilon^2)$ instantiations of the basic Morris algorithm. Now note that once any given Morris counter X reaches the value $\lg(stn/\delta)$, the probability that it is incremented at any given moment is at most $\delta/(nst)$. Thus the probability that it is incremented at all in the next n increments is at most $\delta/(st)$. Thus by a union bound, with probability $1-\delta$ none of the st basic Morris instantiations ever stores a value larger than $\lg(stn/\delta)$, which takes $O(\lg\lg(stn/\delta))$ bits. Thus the total space complexity is, with probability $1-\delta$, at most

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

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

References

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