COMS E6998-9: Algorithmic Techniques for Massive Data — Sep 8, 2015

Lecture 1 – Introduction, Schedule, Approximate Counting

Instructor: *Alex Andoni* — Scribes: *Erik Waingarten*

1 Introduction and Course Description

The class is about algorithms. Historically, the golden standard for algorithms has been $O(n)$ (linear time and space). However, the data we work with has grown much larger than the computer resources we have. We need to think of algorithms which limit the access to data. These limitations can be either limited time and/or limited space.

1.0.1 A common scenario: limited space

Suppose we have a router with internet traffic passing through it. The router receives some information, along with a source and an IP address, and sends it to the corresponding machine. We would like to compute certain statistics with the information passing through the router. For example, we would like to know the number of times a certain IP address makes a request.

The amount of information which passes through the router greatly exceeds its available storage. Therefore, we cannot simply store copies of data passing through the router, and then compute based on the stored data.

Many of these tasks are impossible. In order to solve this problems, we relax the guarantees. Instead of exact computations, we approximate. Instead of requiring these approximations to work all the time, we require they work with large probability.

One way to approximate is to say:

$$\text{true answer} \leq \text{output} \leq \alpha * \text{true answer} \tag{1}$$

We often think of $\alpha = 1 + \epsilon$, where ϵ is very small. As ϵ becomes smaller, the output becomes closer to the true answer. We will require Equation 1 to hold with probability $1 - \delta$. We think of δ as being a very small number, so the probability is close to 1. In other words, the approximation holds often.

1.0.2 List of Topics

- Streaming algorithms. In this model, data passes through an algorithm. The algorithm sees all of it, but cannot store it.
- Dimension reduction, sketching. Consider the setting where we work with distributed data. No processor has access to the entire data. The goal is for each processor to compute small summaries of the data it holds. Algorithms use the summaries to obtain information on the entire dataset.
- High dimensional nearest neighbor search. We want to find the most similar objects in a database. The dimensionality of individual objects is very large.

- Sampling, property testing. In this model, we cannot even look at all the data. Instead, we will only look at small pieces of the data in hopes of deducing overall properties of the dataset.
- Parallel algorithms. In this model, many processors are available in the computation, but no processor has access to all the data.

1.0.3 Grading

The grading is divided into 3 portions:

- Scribing. 2-3 students write detailed notes on the material presented in Lecture. (%10).
- 5 Homeworks. The first assignment is worth %7. The rest of the assignments are worth %12 each. There are 5 total days of lateness (120 hours) to use.
- Research-based project. Projects can be one of the following: solve or make progress on an open problem, apply an algorithm to your research area, or survey and synthesize some important papers. (%35)

2 Probability Review

These are a few probability tools we will use in the analysis of algorithms.

Let X be a random variable.

Definition 1 (Expectation). *For a discrete random variable X, the expectation of X, $\boldsymbol{E}[X]$ is*

$$\boldsymbol{E}[X] = \sum_a a \Pr[X = a]$$

For a continuous random variable X, the expectation of X, $\boldsymbol{E}[X]$, is

$$\boldsymbol{E}[X] = \int a\phi(a)da$$

where ϕ is the probability density function of X.

Lemma 2 (Linearity of Expectation). *Let X and Y be two random variables. $\boldsymbol{E}[X+Y] = \boldsymbol{E}[X] + \boldsymbol{E}[Y]$.*

Lemma 3 (Markov's inequality). *Let X be a non-negative random variable. For all $\lambda > 0$,*

$$\Pr[X > \lambda] \leq \frac{\boldsymbol{E}[X]}{\lambda}$$

Definition 4 (Variance). *The variance of a random variable X, denoted $var[X]$, is*

$$var[X] = \boldsymbol{E}[(X - \boldsymbol{E}[X])^2] = \boldsymbol{E}[X^2] - \boldsymbol{E}[X]^2$$

Lemma 5 (Chebyshev Inequality). *For all $\lambda > 0$,*

$$\Pr[|X - \boldsymbol{E}[X]| > \lambda] \leq \frac{var[X]}{\lambda^2}$$

Corollary 6. *With probability* $1 - c$*,*

$$X = \boldsymbol{E}[X] \pm \sqrt{var[X]/c}$$

Proof. The proof follows from the Chebyshev inequality.

$$\Pr[|X - \mathbf{E}[X]| > \sqrt{\text{var}[X]/c}] \leq \frac{\text{var}[X]}{\sqrt{\text{var}[X]/c}^2}$$

So the probability that X is not within $\mathbf{E}[X] \pm \sqrt{\text{var}[X]/c}$ is at most c. □

3 Approximate Counting

We want to count the number of times an event happens. The problem is motivated by the scenario described in Section 1. The router needs to maintain the number of times a certain IP address made a request.

Say n is the true number of events, so the particular event we are counting happens n times. How much space does it take to maintain the count of the events? Well, we need to store n, which we can do in $O(\log n)$ bits. Suppose n is very large, so $O(\log n)$ bits cannot fit in the router's storage. Can we maintain the count in less space?

In general, no. We relax the requirements. The algorithm should keep an approximation of the count. We will see an algorithm which uses $O(\log \log n)$ bits and computes an approximation of the count.

3.0.4 Morris Algorithm

We present an algorithm which returns a value which is $(1 \pm O(1))n$ of the actual value with high constant probability. More specifically, we will say that with probability at least $\frac{9}{10}$,

$$(1 - O(1))n \leq \text{ algorithm output } \leq (1 + O(1))n$$

3.0.5 The Algorithm

We maintain a counter X.

1. Initialize $X = 0$.
2. When the event happens: Update $X \leftarrow X + 1$ with probability $\frac{1}{2^X}$ and do nothing with probability $1 - \frac{1}{2^X}$.
3. When done, output $2^X - 1$.

3.0.6 Analysis

We prove the algorithm works in three steps:

1. We compute the expected value of our estimator, $\mathbf{E}[2^X - 1]$, and show the expectation is n (what it should be).

2. We compute the variance of the estimator, $\text{var}[2^X - 1]$.

3. We use Corollary 6 and the variance computed to say that the estimator is close to its expected value.

In order to prove $\mathbf{E}[2^X - 1] = n$, we will analyze how X evolves as events happens. We'll let $X_0 = 0$, be the original value, and in general, X_i will be the value of X after i events happen. At the end of the execution of the algorithm, $X = X_n$. So the algorithm's output is $2^{X_n} - 1$.

Claim 7. *Let X_n be the random variable equal to X after n increments. Then $\boldsymbol{E}[2^{X_n} - 1] = n$.*

Proof. We will write $\mathbf{E}[2^{X_n}]$ in terms of $\mathbf{E}[2^{X_{n-1}}]$ and then use induction to get our desired claim. In the base case, $\mathbf{E}[2^{X_0}] = 1$.

$$\mathbf{E}[2^{X_n}] = \sum_i 2^i \Pr[X_n = i] \tag{2}$$

$$\begin{aligned}
\Pr[X_n = i] &= \Pr[X \text{ is incremented and } X_{n-1} = i-1] + \Pr[X \text{ is not incremented and } X_{n-1} = i] && (3)\\
&= \frac{1}{2^{i-1}} \Pr[X_{n-1} = i-1] + (1 - \frac{1}{2^i}) \Pr[X_{n-1} = i] && (4)
\end{aligned}$$

Where the probability that X is incremented at step n is $\frac{1}{2^{X_{n-1}}}$. So

$$\begin{aligned}
\mathbf{E}[2^{X_n}] &= \sum_i 2^i \left(\frac{1}{2^{i-1}} \Pr[X_{n-1} = i-1] + (1 - \frac{1}{2^i}) \Pr[X_{n-1} = i] \right) && (5)\\
&= \sum_i \left(2 \Pr[X_{n-1} = i-1] + 2^i \Pr[X_{n-1} = i] - \Pr[X_{n-1} = i] \right) && (6)\\
&= \sum_i 2^i \Pr[X_{n-1} = i] + 2 \sum_i \Pr[X_{n-1} = i-1] - \sum_i \Pr[X_{n-i} = i] && (7)\\
&= \mathbf{E}[2^{X_{n-1}}] + 2 - 1 && (8)\\
&= \mathbf{E}[2^{X_{n-1}}] + 1 && (9)
\end{aligned}$$

Where in the final steps, we used the fact that the sum of all possible values for a discrete random variable is 1. So by induction, and noting that the X_0 case adds 1,

$$\mathbf{E}[2^{X_n}] = n + 1 \tag{10}$$

This completes the proof. □

Claim 8. *var*$[2^X - 1] \leq \dfrac{3n(n+1)}{2} + 1 = O(n^2)$

Proof. The first thing to note is that $\text{var}[2^X - 1] = \text{var}[2^X]$ by the definition of variance and linearity of expectation.

$$\begin{aligned}
\text{var}[2^{X_n}] &= \mathbf{E}[2^{2X_n}] - \mathbf{E}[2^{X_n}]^2 && (11)\\
&\leq \mathbf{E}[2^{2X_n}] && (12)
\end{aligned}$$

since $\mathbf{E}[2^{X_n}]^2 \geq 0$.

$$\mathbf{E}[2^{2X_n}] = \sum_i 2^{2i} \Pr[X_n = i] \tag{13}$$

$$= \sum_i 2^{2i} \left(\frac{1}{2^{i-1}} \Pr[X_{n-1} = i-1] + (1 - \frac{1}{2^i}) \Pr[X_{n-1} = i] \right) \tag{14}$$

$$= \sum_i 2^{i+1} \Pr[X_{n-1} = i-1] + \sum_i 2^{2i} \Pr[X_{n-1} = i] - \sum_i 2^i \Pr[X_{n-1} = i] \tag{15}$$

$$= \mathbf{E}[2^{2X_{n-1}}] + 4 \sum_i 2^{i-1} \Pr[X_{n-1} = i-1] - \sum_i 2^i \Pr[X_{n-1} = i] \tag{16}$$

$$= \mathbf{E}[2^{2X_{n-1}}] + 3 \sum_i 2^i \Pr[X_{n-1} = i] \tag{17}$$

$$= \mathbf{E}[2^{2X_{n-1}}] + 3\mathbf{E}[2^{X_{n-1}}] \tag{18}$$

So by induction and noting that $\mathbf{E}[2^{2X_0}] = 1$, it follows that $\mathbf{E}[2^{2X_n}] = 3\sum_{i=0}^{n-1} \mathbf{E}[2^{X_i}] + 1$. By Claim 7, this value is at most $\dfrac{3n(n+1)}{2} + 1$. □

Now we apply the Chebyshev bound. Although we don't have the variance computed exactly, an upper bound on the variance will still give us a Chebyshev bound. Our estimator, $2^X - 1$ is close to its expectation $\pm\sqrt{O(n^2)}$ with high constant probability. Specifically, we can say that with probability at least $\frac{9}{10}$, we have that

$$2^X - 1 = n \pm O(n)$$

This is OK on the upper bound (but not on the lower bound, which is a vacuous statement at the moment), but we would like to obtain an $1 + \epsilon$ approximation.

3.1 Morris+

In order to achieve a $(1 + \epsilon)$-approximation. We will repeat Morris's algorithm many times, and output the average. So in Morris+,

1. Run k independent copies of Morris's algorithm. Keeping $(X_1, ..., X_k)$.
2. At the end, output $\frac{1}{k}\sum_{i=1}^{k}(2^{X_i} - 1)$.

For the analysis, we'll let $Y_i = 2^{X_i} - 1$ and $Y = \frac{1}{k}\sum Y_i$. We'll also follow a similar strategy as before. We will prove the expectation of Y is n (what it should be), and then we will compute the variance to apply Corollary 6.

Claim 9. $\boldsymbol{E}[Y] = n$.

Proof. The proof follows from linearity of expectation.

$$\mathbf{E}[Y] = \mathbf{E}\left[\frac{1}{k}\sum Y_i\right] \tag{19}$$

$$= \frac{1}{k}\sum_{i=1}^{k} \mathbf{E}[Y_i] \tag{20}$$

$$= \frac{1}{k}\sum_{i=1}^{k} n \tag{21}$$

$$= n \tag{22}$$

□

Claim 10. *var*$[Y] = O(n^2/k)$

Proof.

$$\text{var}[Y] = \text{var}\left[\frac{1}{k}\sum Y_i\right] \tag{23}$$

$$= \sum_{i=1}^{k} \text{var}[Y_i/k] \tag{24}$$

$$= \sum_{i=1}^{k} \frac{1}{k^2} O(n^2) \tag{25}$$

$$= O(n^2/k) \tag{26}$$

□

We use Corollary 6. We have that with constant probability,

$$Y = n \pm \frac{1}{\sqrt{k}} O(n)$$

Letting $k = O(\frac{1}{\epsilon^2})$ gives us the $(1+\epsilon)$-approximation.

This algorithm works with high constant probability, for example, only $\frac{9}{10}$ probability. We would like to improve the probability bound to $1-\delta$, for any $\delta > 0$.

3.1.1 Morris++

In order to achieve the $(1+\epsilon)$-approximation from Morris+ with probability $1-\delta$, we can do run Morris+ with $k = O(\frac{1}{\epsilon^2}\frac{1}{\delta})$. Using the Chebyshev Inequality, we have that

$$Y = (1 \pm \epsilon)n$$

except with failure probability at most $\frac{\text{var}[Y]}{\epsilon^2 n^2} \leq O\left(\frac{n^2\epsilon^2\delta}{\epsilon^2 n^2}\right) = O(\delta)$. So with the appropriate constants, we achieve the approximation with probability at least $1-\delta$. The algorithm above suggests a $O(\frac{\log\log n}{\epsilon^2\delta})$ space bound (although this was not proven).

In fact, we have algorithms which take $O(\frac{\log\log n}{\epsilon^2}\log(\frac{1}{\delta}))$ space bounds. The idea is to maintain $O(\log(1/\delta))$ counters like Y from Morris+ and take the median of the Ys. For the analysis, we will need Chernoff/Hoefding bounds. This is done in the next lecture.