CMPUT 675: Algorithms for Streaming and Big Data **Fall 2019**

Lecture 1 (Sep 4, 2019): Introduction, background

Lecturer: Mohammad R. Salavatipour *Scribe: Mohammad R. Salavatipour*

1.1 Introduction

There is huge growth in data gathering and hence more demand for processing the data to extract useful information from the raw data in various applications. The goal of this course is to learn some of the tools and techniques to design efficient and fast algorithms that can extract useful information from raw data. Typically, these algorithms have to run fast (sometimes sublinear time) and often work with much smaller space than the data can possibly be stored. Below is a short list of topics we try to cover in this course

Streaming: There are situations when the data comes as a stream which is too large to be stored or processed later. We have only one pass (or sometimes a few passes) over the data and have to make decisions as the data comes.

sketching/sampling: the goal is to have a compressed form of data from which we can still answer queries and extract useful information.

Dimensionality reduction: In many applications data comes with very high dimensions (e.g. medical applications, spam filtering, etc). Designing algorithms to mange high dimension data is difficult. One goal is to reduce the dimension (e.g. via projection) while preserving (approximately) relevant structure/geometry of the problem.

property testing: Checking quickly with sufficiently high probability whether a given object has certain properties (e.g. if the result of a matrix computation is correct, if a large graph has certain graph properties, if a proof is valid, etc).

Sparse Fourier transform: There are old algorithms to compute discrete Fourier transform of a sequence of length n in time $O(n \log n)$. This has various applications (in signal processing, multiplication of large integers, etc). Sparse Fourier Transform is an algorithm to compute the DFT when the output is k-sparse in time $O(k \log n)$.

Approximate counting: Counting the number of objects with certain properties among a very large set (e.g. number of solutions to an equation, or the number of distinct items in a sequence, etc).

Below we do a quick overview of the basic probability background we use throughout the course.

1.2 Background on Probability

Most of the algorithms we discuss are randomized and heavily rely on basic tools from probability theory for their analysis. Let Ω be a discrete probability space. A probability function $\Pr : \Omega \to [0,1]$ has the property that $\sum_{x\in\Omega} \Pr(x) = 1$. A subset $A \subset \Omega$ is called an event. We define $\Pr(A) = \sum_{x\in A} \Pr(x)$. We say two events $A, B \subseteq \Omega$ are independent if $\Pr(A \cap B) = \Pr(A) \cdot \Pr(B)$. A random variable is a function $X : \Omega \to \mathbb{R}$. The expected value of X, denoted by $\mathrm{E}[X]$, is defined as $\mathrm{E}[X] = \sum_{i\in\Omega} i \cdot \Pr(X = i)$.

Lemma 1 $\mathrm{E}[X+Y]=\mathrm{E}[X]+\mathrm{E}[Y]$ *and for any constant* c*:* $\mathrm{E}[cX]=c\mathrm{E}[X]$.

Two random variables X, Y are independent if $\forall x, y \in \mathbb{R} : \Pr[X=x \wedge Y=y]=\Pr[X=x] \wedge \Pr[Y=y]$.

Lemma 2 *If* X, Y *are independent then* $\mathrm{E}[XY]=\mathrm{E}[X]\mathrm{E}[Y]$.

Variance of a random variable X is defined as $\mathrm{Var}[X]=\mathrm{E}[(X-\mathrm{E}[X])^2]=\sigma_X^2$ and σ_X is called standard deviation.

Theorem 1 (Markov's inequality) *Let* X *be a non-negative random variable. Then for all* $a>0$*:* $\Pr[X \geq a] \leq \frac{\mathrm{E}[X]}{a}$. *Alternatively* $\Pr[X \geq a\mathrm{E}[X]] \leq \frac{1}{a}$.

Using Markov's inequality to bound deviation from the mean is called first moment method.

Theorem 2 (Chebyshev's inequality) *Let* X *be a random variable and* $t>0$. *Then* $\Pr[|X-\mathrm{E}[X]|>t \leq \frac{\mathrm{Var}[X]}{t^2}$. *Alternatively* $\Pr[|X-\mathrm{E}[X]|>t\sigma_X] \leq \frac{1}{t^2}$.

As an example, consider a random walk on the integers that starts at origin (zero) and at every step with probably $\frac{1}{2}$ it moves one step to left or right uniformly randomly. After n steps, how far from 0 we have traveled? Let

$$X_i = \begin{cases} 1 & \text{moved right at step } i \\ -1 & \text{moved left at step } i \end{cases}$$

Let Y_n be the position at step n. Then $Y_n=\sum_{i=1}^n X_i$, $E[Y_n]=0$ and $\mathrm{Var}[Y_n]=n$. Therefore, using Chebyshev's inequality $\Pr[|Y_n| \geq t\sqrt{n}] \leq \frac{1}{t^2}$.

Chernoff-Hoeffding: Chernoff bound is a very powerful bound giving exponentially decreasing bound on the tails of distributions. It can be applied to bound deviation from the mean for "independent" random variables. It can be derived using Markov's inequality.

Theorem 3 (Chernoff) *Let* $X_1, \ldots, X_n$ *be independent binary (Poisson) random variables where* $\Pr[X_i]=p_i$ *and let* $X=\sum_i X_i$ *and* $\mu=\mathrm{E}[X]$*. Then*

- *For any* $\delta>0$*:* $\Pr[X \geq (1+\delta)\mu] < \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu$.
- *For any* $0<\delta \leq 1$*:* $\Pr[X \geq (1+\delta)\mu] \leq e^{\mu\delta^2/3}$
- *For* $R \geq 6\mu$*:* $\Pr[X \geq R] \leq 2^{-R}$.

Although Chernoff bound gives very powerful bounds it has limited applications to independent variables. There are stronger tools that can be applied to show concentration for settings with limited dependencies.

Azuma's inequality Let X be a random variable determined by n trials $X_1, \ldots, X_n$ such that for all i and any two possible sequences of outcomes $x_1, \ldots, x_i$ and $x_1, \ldots, x_{i-1}, x_i'$:

$$|\mathrm{E}[X|X_1=x_1, \ldots, X_i=x_i]-\mathrm{E}[X|X_1=x_1, \ldots, X_{i-1}=x_{i-1}, X_i=x_i'] \leq c_i$$

for constants c_i then $\Pr[|X-\mathrm{E}[X]|>t] \leq 2e^{-t^2/(2\sum_i c_i^2)}$.

1.3 Approximate counting of events

We start with a simple problem of (approximately) counting the number of events. Suppose we want to design an algorithm that monitors a long sequence of events and the goal is to have an approximate number of the events at any given time. Clearly if we have had n events we can keep a counter using $O(\log n)$ bits. It's not difficult to show that any deterministic exact algorithm needs this much space.

We can however keep an approximate count using much less space, as little as $O(\log\log n)$ bits. To be more precise, suppose we want to have an estimate $\tilde{n}$ for n such that $\tilde{n} = (1 \pm \epsilon)n$ and $\Pr[|\tilde{n} - n| > \epsilon n] < \delta$ for a given δ. We call this an (ϵ, δ)-estimator. Here we describe Morris algorithm, that keeps a counter for $\log n$ instead of n; so only $O(\log\log n)$ bits of space are required.

Morris approximate counting

```
1. $X \leftarrow 0$
2. For each new even increment $X$ with probability $\frac{1}{2^X}$
3. return $\tilde{n} = 2^X - 1$.
```

Let X_n be the random variable representing the value of X after n steps and let $Y_n = 2^{X_n}$.

Lemma 3 $\mathrm{E}[Y_n] = n + 1$

Proof. We use induction on n. The base case of $n = 0$ is easy. For induction step, assume it is true for Y_n. Then:

$$\begin{aligned}
\mathrm{E}[Y_{n+1} &= \sum_{i=0} \Pr[X_n = i]\mathrm{E}[2^{X_{n+1}}|X_n = i] \\
&= \sum_{i=0} \Pr[X_n = i] \left(2^i(1 - \frac{1}{2^i}) + \frac{1}{2^i} \cdot 2^{i+1}\right) \\
&= \sum_{i=0} \Pr[X_n = i]2^i + \sum_{i=0} \Pr[X_n = i] \\
&= \mathrm{E}[Y_n] + 1 \\
&= (n + 1) + 1.
\end{aligned}$$

■

Thus, the output $\tilde{n} = 2^X - 1$ is an estimate for n (in expectation). Also, since $\mathrm{E}[Y_n] = n + 1$, it implies that $\mathrm{E}[X_n] = \log_2(n + 1)$ and so the expected number of bits used after n steps is $O(\log\log n)$.

Lemma 4 $\mathrm{E}[Y_n^2] = \frac{3}{2}n^2 + \frac{3}{2}n + 1$ *and* $\mathrm{Var}[Y_n] = \frac{n(n-1)}{2}$.

Proof. Again we use induction on n. Base case of $n = 0$ is easy to check. For induction step, assuming that the statement is true for Y_n:

$$\begin{aligned}
\mathrm{E}[Y_{n+1}^2] &= \sum_{i=0} 2^{2i}\Pr[X_{n+1}=i] \\
&= \sum_{i=0} 2^{2i}\left(\Pr[X_n=i](1-\frac{1}{2^i})+\Pr[X_n=i-1]\frac{1}{2^{i-1}}\right) \\
&= \sum_{i=0} 2^{2i}\Pr[X_n=i]+\sum_{i=0}(-2^i\Pr[X_n=i-1]+4\times 2^{i-1}\Pr[X_n=i-1]) \\
&= \mathrm{E}[Y_n^2]+3\mathrm{E}[Y_n] \\
&= \frac{3}{2}n^2+\frac{3}{2}n+1+3(n+1) \\
&= \frac{3}{2}(n+1)^2+\frac{3}{2}(n+1)+1.
\end{aligned}$$

Also $\mathrm{Var}[Y_n]=\mathrm{E}[Y_n^2]-\mathrm{E}[Y_n]^2=\frac{n(n-1)}{2}$. ■

Thus, using this lemma and Chebyshev's inequality:

$$\Pr[|\tilde{n}-n|>\epsilon n]<\frac{1}{(\epsilon n)^2}\cdot\frac{n(n-1)}{2}\simeq\frac{1}{2\epsilon^2}$$

But this is useless for small values of ϵ. So we need to boost the success probability. We would like, given $\epsilon,\delta>0$, have a bound of the form $\Pr[|\tilde{n}-n|>\epsilon n]\le\delta$ using $O(\log\log n)$ bits.

1.3.1 Morris+: Using average to boost probability

Suppose we run r parallel copies of Morris algorithm and find values $\tilde{n}_i$ for $1\le i\le r$ and then let $\tilde{n}=\frac{1}{r}\sum_{i=1}^{r}\tilde{n}_i$. Since each $\tilde{n}_i$ is an estimator for n then

$$\Pr[|\tilde{n}-n|>\epsilon n]\le\frac{1}{2r\epsilon^2}<\delta$$

if we choose $r>\frac{1}{2\epsilon^2\delta}$. The amount of space used will be $O(\log\log n/(\epsilon^2\delta)$. In particular, if we choose $r>\frac{2}{\epsilon^2}$ then we get $\Pr[|tilden-n|>\epsilon n]<\frac{1}{4}$.

1.3.2 Morris++: Using Median to boost probability

We can do even better to boost success probability: instead of using average we use the median. More specifically, run $\ell=c\log\frac{1}{\delta}$ parallel copies of Morris+, for some large constant c. Suppose we get estimators $Z_1,\dots,Z_\ell$ and let $\tilde{n}$ be the *median* of them. Note that by the arguments for Morris+ $\Pr[|Z_i-n|>\epsilon n]<\frac{1}{4}$ for each i. Thus, if we define a 0/1 random variable $Y_i=1$ if $|Z_i-n|>\epsilon n$ then Y_i's are independent and $\Pr[Y_i=1]<\frac{1}{4}$ and $\mathrm{E}[\sum_i Y_i]<\ell/4$. We will have $|\tilde{n}-n|>\epsilon n$ only if at least $\frac{\ell}{2}$ of the Z_i's are larger than n by ϵn. Using Chernoff bound:

$$\Pr[|\tilde{n}-n|>\epsilon n]\le\Pr[|\sum_i Y_i-\mathrm{E}[\sum_i Y_i]|>\frac{\ell}{4}]<(\frac{e}{4})^{\ell/4}<\delta$$

for $\ell=c\log\frac{1}{\delta}$ for large constant c. Also, the space complexity will be $O(\epsilon^{-2}\log\frac{1}{\delta}\log\log(\frac{n}{\epsilon\delta}))$.

References

Mor787 R. Morris, Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840-842, 1978. Matrix multiplication via arithmetic progressions,