**CS 395T: Sublinear Algorithms, Fall 2020** September 1st, 2020

# Lecture 2: Distinct element counting

*Prof. Eric Price* *Scribe: Devvrit, Niels Kornerup*

**NOTE:** THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

## 1 Overview

In the last lecture we looked at examples of property testing, streaming, and testing distributions.

In this lecture we cover estimation of a Bernoulli random variable and the number of distinct elements in a stream.

## 2 Bernoulli random variables

Given a weighted coin (comes up heads with probability $p \in [0, 1]$), how many flips do we need to estimate $p$? Lets assume we take $k$ samples. We will call the sample mean $\hat{p}$. As $n \to \infty, \hat{p} \to p$, meaning that this is a reasonable way to guess the value of $p$. In order to figure out how big we need to make $n$ for our answer to be reasonable, it would be useful to find the variance of $\hat{p}$.

$$\begin{aligned}
\mathrm{Var}(\hat{p}) &= \mathrm{Var}(\frac{1}{n^2}\sum x_i) \\
&= \frac{1}{n^2}\mathrm{Var}(\sum x_i) \\
&= \frac{1}{n}\mathrm{Var}(x_1) \\
&= \frac{1}{n}\,\mathbb{E}\left[(x_1 - \mathbb{E}[x_1])^2\right] \\
&= \frac{1}{n}\left(p(1-p)^2 + (1-p)p^2\right) \\
&= \frac{1}{n}\left(p(1-p)\right) \\
&\le \frac{1}{4n}
\end{aligned}$$

This tells us that the standard deviation of $\hat{p}$ is at most $\sqrt{np}$. Thus we can expect that with n samples we will get that $\hat{p} \in \left[p - \sqrt{p/n}, p + \sqrt{p/n}\right]$.

Suppose you want to estimate $p$ within an additive error of $\epsilon$ while having a failure probability of at most $\frac{1}{4}$. What $n$ value would we need? By the above reasoning, we get that setting $n = O\left(\frac{1}{\epsilon^2}\right)$ gives us that $\hat{p} \in \left[p - O\left(\frac{1}{\epsilon}\right), p + O\left(\frac{1}{\epsilon}\right)\right]$ with sufficient probability (by Chebyshev's inequality).

## 3 Basic probability inequalities

There are two simple inequalities that show up a lot when dealing with probabilities. Markov's inequality gives us that for any non-negative random variable $x$, $\mathbb{P}\left[x \geq t\right] \leq \frac{\mathbb{E}[x]}{t}$. Chebyshev's inequality gives us that for any integrable random variable with finite mean $\mu$ and standard deviation $\sigma$, $\mathbb{P}\left[|x-\mu| \geq t\sigma\right] \leq \frac{1}{t^2}$. Note that this tells us that the probability that a random variable is more than 2 standard deviations from the mean is less than $\frac{1}{4}$.

## 4 Mean estimation

Suppose I have an unknown distribution $D$ with an unknown mean $\mu$ whose standard deviation is at most $\sigma$. How many samples will I need from $D$ to estimate $\mu$ to within an additive factor of $\epsilon\sigma$ with $\frac{3}{4}$ probability?

We will take the empirical mean as the variable $\hat{\mu}$. We know that $\mathrm{Var}(\hat{\mu}) \leq \sigma^2/n$, where $n$ is the number of samples we take. Thus setting $n = \frac{4}{\epsilon^2}$ gives us that the standard deviation of $\hat{\mu}$ is at most $\frac{\epsilon\sigma}{2}$. By Chebyshev's inequality, this implies that $\hat{\mu} \in [\mu - \epsilon\sigma, \mu + \epsilon\sigma]$ with a probability of at least $\frac{3}{4}$.

## 5 Streaming distinct elements

Let's say you have a stream of items that you only get to pass through once. Your goal is to estimate the number of distinct elements ($n$) in the stream. What is the least amount of information you need to store to get a good estimate of the number of distinct elements in the stream? Lets start with a simpler problem: we are promised that either $n < T$ or $n > 2T$ and we want to tell which is true.

To solve this we will first construct a random hash function $h : U \to [T]$. We will then pick $k$ elements of $T$ and for each selected element of $T$, we will increment a counter the first time that a hash hits it. We know that each element of $T$ has a probability of $\left(1 - \frac{1}{T}\right)^n \approx e^{-n/T}$ to get an element of the stream to hash to it. When $n < T$ this gives us a probability of at most .63 and when $n > 2T$ the probability is at least .86. Thus our problem gets reduced to selecting a large enough $k$ (here, representing the number of parallel runs) such that we can distinguish between these two Bernoulli random variables. From before, we know that $O\left(\frac{1}{\epsilon^2}\right)$ space should be enough to distinguish between $(1-\epsilon)T$ and $(1+\epsilon)T$ unique elements.

We now move to the original question of estimating number of unique elements. One idea is to repeat the above algorithm in parallel for different values of $T$. In order to get $(1 \pm \epsilon)$ error guarantee, we can repeat the above algorithm in parallel for $T = 1, (1+\epsilon), (1+\epsilon)^2, \cdots, (1+\epsilon)^{log_{1+\epsilon}(N)}$. For most of the $T$ values, the algorithm will say either *less than* or *greater than.* But a few of the runs in middle will be confused as they don't satisfy $< T$ or $> 2T$ condition.
A solution to deal with this is to repeat many times. We saw earlier in the coin flip example that to get constant success probability, $O(1/\epsilon^2)$ flips are required. We'll see later in the course it's possible to get $\geq 1 - \delta$ success probability by tossing $O\left(\frac{1}{\epsilon^2} log\left(\frac{1}{\delta}\right)\right)$ flips. Therefore, using this information,

and doing union bound over constant failure probability of all the runs, the space needed for is:

$$O\left(\underbrace{log_{1+\epsilon}(N)}_{\text{initial number of runs proposed}} \cdot \underbrace{\frac{1}{\epsilon^2}}_{\text{space complexity for each run}} \cdot \underbrace{log\left(log_{1+\epsilon}(N)\right)}_{\text{more runs for union-bounding failure prob}}\right)$$
$$= O\left(\frac{1}{\epsilon}log(N) \cdot \frac{1}{\epsilon^2} \cdot log\left(\frac{1}{\epsilon}log(N)\right)\right)$$

In order to get success probability $\geq 1 - \delta$, the space required is:

$$O\left(\frac{1}{\epsilon}log(N) \cdot \frac{1}{\epsilon^2} \cdot log\left(\frac{1}{\delta\epsilon}log(N)\right)\right)$$

We now look at another simpler algorithm. We'll hash the incoming units to a value between $[0, 1]$. That is, we consider a hash function $h : U \to [0, 1]$. Let's analyze the expected minimum value that any unit is hashed to. Let $S$ be the set comprising all elements we see. Let $y = \underset{X \in S}{min}\ h(X)$, then

$$\begin{aligned}
\mathbb{E}[y] &= \frac{1}{n+1} \\
\text{Proof: } \mathbb{P}[y \geq 1 - t] &= t^n \\
\implies \mathbb{P}[y = 1 - t] &= nt^{n-1} && \text{(From CDF to PDF by differentiating)} \\
\mathbb{E}[1 - y] &= \int_{t=0}^{1} (nt^{n-1})tdt \\
&= \frac{n}{n+1} \\
\implies \mathbb{E}[y] &= \frac{1}{n+1}
\end{aligned}$$

We analyze the variance of $y$

$$\begin{aligned}
Var(y) &= Var(1 - y) \\
&= \mathbb{E}[(1 - y)^2] - (\mathbb{E}[1 - y])^2 \\
\mathbb{E}[(1 - y^2)] &= \int_{t=0}^{1} (nt^{n-1})t^2 dt = \frac{n}{n+2} \\
\implies Var(y) &= \frac{n}{n+2} - \left(\frac{n}{n+1}\right)^2 \\
&= \frac{n}{(n+2)^2(n+1)} \\
&\approx \frac{1}{(n+1)^2}
\end{aligned}$$

*Algorithm*: Hash the incoming units to a value between $[0, 1]$. Let $y = \underset{X \in S}{min}\ h(X)$. Output $\frac{1}{y} - 1$. The above algorithm won't work well because Variance of $y$ is low. Basically, the minimum hashed value has good variation in it. Therefore, we'll use the same old technique of repeating experiment many times and taking the average

*Algorithm*: Hash the incoming units to a value between $[0,1]$ on $r$ different hash functions. let $y_i = \underset{X \in S}{min}\, h_i(X)$. Output $\frac{1}{\frac{1}{r}\sum_{i=1}^{r} y_i} - 1$.

Note that $(1 \pm \epsilon)$ factor approximation to $\frac{1}{r}\sum_{i=1}^{r} y_i$ will translate to $(1 \pm \epsilon)$ factor approximation for $\frac{1}{\frac{1}{r}\sum_{i=1}^{r} y_i}$ too. Therefore, we focus on getting

$$\frac{1}{r}\sum_{i=1}^{r} y_i \in (1 \pm \epsilon)\mathbb{E}\left[\frac{1}{r}\sum_{i=1}^{r} y_i\right]$$

$$\mathbb{E}\left[\frac{1}{r}\sum_{i=1}^{r} y_i\right] = \mathbb{E}[y] = \frac{1}{n+1}$$

$$\implies \frac{1}{r}\sum_{i=1}^{r} y_i \in \frac{1+\epsilon}{n+1}$$

$$\implies \left|\frac{1}{r}\sum_{i=1}^{r} y_i - \frac{1}{n+1}\right| \leq \frac{\epsilon}{n+1} = \epsilon\sigma$$

where $\sigma$ is the std. deviation as we calculated above. Using Chebyshev's inequality, we get that the number of samples $r$ needed is $r = O\left(\frac{1}{\epsilon^2}\right)$

Space Complexity: Need $O(1/\epsilon^2)$ hash functions, to estimate each $y_i$. The space required to store $y_i$ depends on the resolution. A resolution of $1/n^2$ is sufficient for our purposes, but we can do better. Since we only need $y_i$ to a $(1 \pm \epsilon)$ multiplicative factor, we can round our $y_i$ to the nearest $(1+\epsilon)^{-i}$ for $i \in \mathbb{Z}$. This gives $\frac{1}{\epsilon}\log(n)$ possible values for $y_i$, which only requires $\log\left(\frac{1}{\epsilon}\log(n)\right)$ bits of storage. This gives us a final space complexity of $O\left(\frac{1}{\epsilon^2}\log(\frac{1}{\epsilon}\log(n))\right)$.