Lecture 4: Frequency Moments Estimation

09-04-2025 Lecturer: Ali Vakilian | Scribe: Aryeh Keating | Editor: Ali Vakilian

1 Frequency Moments

While we have seen streaming methods for counting the total number of elements of streams in memory limited settings, generalizations of this streaming problem are beneficial for a broader set of applications.

In order to generalize, we first consider the concept of *k-th moments* for applications: Given a large data stream $S = e_1 \dots e_m$, where items belong to $\{1, \dots, n\}$, for every $i \in [n]$, we let f_i denote the total number of distinct indices $j \in [m]$ such that $e_j = i$ and define for $k \in \mathbb{R}_+$ the *k-th moment* of the vector $\boldsymbol{f} = (f_1, \dots, f_n) \in \mathbb{Z}_+^n$ is given by:

$$F_k = \sum_{j \in [n]} f_j^k$$

It follows by considering the case where $k \in \{0, 1, \infty\}$, F_1 denotes the total length of the stream S, F_0 denotes the total number of distinct items in the stream S, and F_∞ denotes the maximum frequency of the stream S, which is closely related to the *heavy hitters problem* (to be discussed in a future lecture). Hence the moments $\{F_k\}_{k \geq 0}$ provide a family of metrics to obtain information of interest of the stream S, that go beyond the singular measure of stream length.

2 Estimation

Now that we have defined the notion of *k-th moment* of a data stream, we look toward methods of estimating F_k for $k \geq 0$ under the constraints of limit memory and only one pass of the stream.

Relative and Additive Approximation. In order to determine the accuracy of our approximations for *k-th moments* of a stream, we denote two measures of approximation accuracy.

We say an algorithm $\mathcal{A}$ has a α-*relative approximation* to a non-negative function g over the stream S if

$$\left| \frac{\mathcal{A}(S)}{g(S)} - 1 \right| \leq \alpha$$

and a (ϵ, δ)-*relative approximation* if $\Pr\left[\left|\frac{\mathcal{A}(S)}{g(S)} - 1\right| \leq \epsilon\right] \geq 1 - \delta$.

We say an algorithm $\mathcal{A}$ has a α-*additive approximation* to a non-negative function g over the stream S if

$$\left|\mathcal{A}(S) - g(S)\right| \leq \alpha$$

and a (ϵ, δ)-*additive approximation* if $\Pr\left[\left|\mathcal{A}(S) - g(S)\right| \leq \epsilon\right] \geq 1 - \delta$.

3 Distinct Element Problem (F_0 Estimation)

Now equipped with measures of algorithm accuracy, we focus our attention to estimation of F_0 of a large stream S given limited memory space. In other words, we look to estimate the number of unique item within S.

3.1 Applications and Baseline Solutions

Applications. Knowledge of the number of distinct elements of a stream is relevant in real-time tracking of unique visitors of a website, estimation of the number of unique values in different columns for efficient execution of queries relevant in database query planning, and in online advertisement for establishing the count of unique people who view an advertisement.

Non-Streaming Solutions. While streaming is beneficial in most scenarios with memory limitations, we consider alternative *dictionary* methods for estimation of F_0 for reference. In the case we are counting d distinct items from a stream of m elements, each belonging to $\{1, \ldots, n)$, *Binary Search Trees* require $O(d)$ space and $O(n \log d)$ time, while *Hashing* requires $O(d)$ space and $O(n)$ time. Here we present a basic dictionary based algorithm for counting the number of distinct elements of a stream:

Algorithm 1 Dictionary Based Distinct Elements

```
Initialize: Dictionary D ← ∅, Counter d ← 0.
while an item e arrives in stream do
    if e ∉ D then
        Insert e into D.
        d ← d + 1.
return d.
```

While this approach is suitable for small scale data streams, we wish find an alternative capable of handling much larger streams that requires significantly less memory.

3.2 Idealized Flajolet-Martin Algorithm

One efficient estimation approach to the distinct element problem is given via the *Flajolet-Martin Algorithm*. This algorithm utilizes a hash function $h : [n] \to [m]$, where m is typically polynomial in n. It only stores the minimum hash value seen so far, $\min_{i \in [n]} h(e_i)$, and uses space complexity $O(\log m) = O(\log(\text{poly}(n))) = O(\log n)$, disregarding hash function storage.

The idealized version, used for analysis, assumes the hash function maps elements to continuous values in $[0, 1]$.

Algorithm 2 Idealized Flajolet-Martin Algorithm

```
Initialize: Ideal hash function h : U → [0, 1], y ← 1.
while an item e enters stream do
    y = min(y, h(e))
return d̂ = 1/y − 1.
```

3.3 Analysis of Idealized Estimator

We analyze the accuracy of estimations in the ideal case where our hash function maps the d distinct elements to values chosen independently and uniformly at random from $[0, 1]$.

Theorem 3.1. *Suppose $X_1, \ldots, X_d$ are random variables that are independent and uniformly distributed in $[0, 1]$, and let $Y = \min_{i \in [d]} X_i$. Then:*

$$\mathbb{E}[Y] = \frac{1}{d+1} \quad \textit{and} \quad \text{Var}[Y] \leq \frac{1}{(d+1)^2}$$

Proof. First, calculate the Cumulative Distribution Function (CDF) for Y. For any $t \in [0,1]$:

$$\Pr[Y > t] = \Pr[\{X_1 > t\} \wedge \cdots \wedge \{X_d > t\}] = \prod_{i \in [d]} \Pr[X_i > t] = (1-t)^d$$

The CDF is $\Pr[Y \leq t] = 1 - (1-t)^d$. The Probability Density Function (PDF) is found by differentiation: $f_Y(t) = \frac{d}{dt}(1-(1-t)^d) = d(1-t)^{d-1}$.

Calculating $\mathbb{E}[Y]$: We calculate the expectation by integrating $t \cdot f_Y(t)$. The integral can be solved using a change of variables.

$$\mathbb{E}[Y] = \int_0^1 t \cdot f_Y(t) dt = \int_0^1 t d(1-t)^{d-1} dt$$

Let $z = 1-t$. This implies $t = 1-z$ and $dt = -dz$. We update the integration bounds: when $t = 0$, $z = 1$; when $t = 1$, $z = 0$.

$$\begin{aligned}
\mathbb{E}[Y] &= d \int_{z=1}^{z=0} (1-z) z^{d-1} (-dz) \\
&= d \int_0^1 (1-z) z^{d-1} dz && \text{(Flipping bounds changes sign)} \\
&= d \int_0^1 (z^{d-1} - z^d) dz \\
&= d \left[\frac{z^d}{d} - \frac{z^{d+1}}{d+1} \right]_0^1 \\
&= d \left(\left(\frac{1}{d} - \frac{1}{d+1} \right) - (0-0) \right) = d \left(\frac{d+1-d}{d(d+1)} \right) = \frac{1}{d+1}
\end{aligned}$$

Calculating Variance Bound: A similar calculation yields the second moment: $\mathbb{E}[Y^2] = \int_0^1 t^2 d(1-t)^{d-1} dt = \frac{2}{(d+1)(d+2)}$. Now, we calculate the variance:

$$\mathrm{Var}[Y] = \mathbb{E}[Y^2] - \mathbb{E}[Y]^2 = \frac{2}{(d+1)(d+2)} - \frac{1}{(d+1)^2} = \frac{2(d+1) - (d+2)}{(d+1)^2(d+2)} = \frac{d}{(d+1)^2(d+2)}$$

Since $d < d+2$, we have $\frac{d}{d+2} < 1$. Therefore, $\mathrm{Var}[Y] < \frac{1}{(d+1)^2}$. □

Concentration Analysis and Bias Remark The theorem shows that Y is a good estimator for $1/(d+1)$ on average. However, we must check if the estimator for d, which is $\hat{d} = 1/Y - 1$, concentrates near d. In particular, note that while $\mathbb{E}[Y] = 1/(d+1)$, the final estimator $\hat{d} = 1/Y - 1$ is *not* an unbiased estimator for d. By Jensen's Inequality, for a convex function $g(x) = 1/x$, we have $\mathbb{E}[g(Y)] > g(\mathbb{E}[Y])$. Therefore:

$$\mathbb{E}[\hat{d}] = \mathbb{E}\left[\frac{1}{Y} - 1\right] = \mathbb{E}\left[\frac{1}{Y}\right] - 1 > \frac{1}{\mathbb{E}[Y]} - 1 = \frac{1}{1/(d+1)} - 1 = (d+1) - 1 = d$$

The estimator systematically overestimates the true value d.

Back to the estimator Y, let's apply Chebyshev's Inequality to bound the concentration of Y around its mean:

$$\Pr[|Y - \mathbb{E}[Y]| \geq \epsilon \mathbb{E}[Y]] \leq \frac{\mathrm{Var}[Y]}{(\epsilon \mathbb{E}[Y])^2} \leq \frac{1/(d+1)^2}{\epsilon^2 (1/(d+1))^2} = \frac{1}{\epsilon^2}$$

This concentration bound is too weak for our purpose of finding a good relative approximation for d. To see why, consider what a good approximation requires. We want our final estimate $\hat{d}$ to be within $(1 \pm \epsilon')$ of d. Let's analyze the failure condition. If Y deviates significantly from $\mathbb{E}[Y]$, say $Y = (1+\epsilon)\mathbb{E}[Y]$, then the estimate $\hat{d}$ becomes:

$$\hat{d} \in \left[\frac{1}{(1+\epsilon)\mathbb{E}[Y]} - 1, \frac{1}{(1-\epsilon)\mathbb{E}[Y]} - 1\right] = \left[\frac{d+1}{1+\epsilon} - 1, \frac{d+1}{1-\epsilon} - 1\right] = \left[\frac{d-\epsilon}{1+\epsilon}, \frac{d+\epsilon}{1-\epsilon}\right]$$
$$\approx [d(1-\epsilon), d(1+\epsilon)]$$

The relative error in $\hat{d}$ is roughly proportional to the relative error ϵ in Y. The Chebyshev bound $\Pr[\text{failure}] \leq 1/\epsilon^2$ only provides a non-trivial probability guarantee (i.e., less than 1) when $\epsilon > 1$. An error factor $\epsilon > 1$ (i.e., $> 100\%$ relative error) is far too large to be useful for approximation purposes. To achieve a small relative error (e.g., $\epsilon = 0.1$), the bound $1/\epsilon^2 = 100$ provides no information. This high variance necessitates techniques like averaging to improve concentration.

3.4 Improving Concentration via Averaging and Median Trick

As we have seen, the variance of a single estimator is too high for reliable approximation. We apply a standard two-step technique to improve accuracy and confidence.

Step 1: Variance Reduction via Averaging To reduce variance, we run k independent copies of the estimator in parallel. Let $Y^{(1)}, \ldots, Y^{(k)}$ be the results from k independent hash functions. Define the average estimator $Y_{\text{avg}} := \frac{1}{k}\sum_{i\in[k]} Y^{(i)}$. The final estimate is given by $\frac{1}{Y_{\text{avg}}} - 1$.

The expectation remains $\mathbb{E}[Y_{\text{avg}}] = \frac{1}{d+1}$, but the variance improves significantly: $\text{Var}[Y_{\text{avg}}] = \frac{\text{Var}[Y]}{k} \leq \frac{1}{k(d+1)^2}$. Applying *Chebyshev's Inequality* to the average:

$$\Pr[|Y_{\text{avg}} - \mathbb{E}[Y_{\text{avg}}]| \geq \epsilon\mathbb{E}[Y_{\text{avg}}]] \leq \frac{\text{Var}[Y_{\text{avg}}]}{(\epsilon\mathbb{E}[Y_{\text{avg}}])^2} \leq \frac{1/(k(d+1)^2)}{\epsilon^2/(d+1)^2} = \frac{1}{k\epsilon^2}$$

By choosing $k = O(1/\epsilon^2)$, we can achieve a constant success probability. For example, setting $k = 1/\epsilon^2$ gives a failure probability of at most ϵ^2, but to guarantee failure probability $\leq 1/4$, we set $k = 4/\epsilon^2$. This provides an $(\epsilon, O(1))$-relative estimator.

Step 2: Probability Amplification via Median Trick To amplify the success probability from constant to $1-\delta$, we apply the *median trick*. We repeat the entire averaging process (from Step 1) $\ell = O(\log(1/\delta))$ times and compute the median of the resulting ℓ estimates. This value is an (ϵ, δ)-relative estimator.

3.5 Practical Implementation Details

Implementing Hash Function. The assumption of an ideal hash function $h : U \to [0,1]$ is not practical. Instead, we use hash functions from a family $\mathcal{H}$ where $h : [n] \to [N]$ for some sufficiently large integer N. For the analysis to hold, we do not need full independence, but rather a weaker condition called pairwise independence.

A family $\mathcal{H} = \{h : [n] \to [m]\}$ is *pairwise-independent* (or *strongly 2-universal*) if, for all distinct $x \neq y \in [n]$ and all $i, j \in [m]$:

$$\Pr_{h\sim\mathcal{H}}[\{h(x) = i\} \wedge \{h(y) = j\}] = \frac{1}{m^2}$$

An implication is that the hash function distributes elements uniformly: $\Pr_{h\sim\mathcal{H}}[h(x) = i] = \frac{1}{m}$. A common construction uses modular arithmetic: for a prime $p \in [n, 2n]$,[1] the family $\mathcal{H} = \{h_{a,b}(x) = (ax + b) \mod p \mid a, b \in \{1, \ldots, p-1\}\}$ is pairwise independent.

Flajolet-Martin Algorithm A practical variant of the algorithm estimates d using the number of trailing zeros in the binary representation of hash values. Let $h : [n] \to [2^L]$ where $L = \lceil \log n \rceil$. Let zeros(s) be the number of trailing zeros of binary string s. The algorithm tracks $Z = \max_{e\in\text{stream}}\{\text{zeros}(h(e))\}$.

The intuition is that for a random value, $\Pr[\text{zeros}(h(e)) \geq r] = 1/2^r$. If we observe d distinct elements, we expect $d \cdot (1/2^r) \approx 1$ element to have roughly $r \approx \log_2 d$ trailing zeros. Thus, d can be estimated as 2^Z.

Analysis with Pairwise Independence Alon, Matias, and Szegedy (1999) proved that pairwise independence suffices for this analysis. Let Y_r be the number of distinct elements e_i such that zeros$(h(e_i)) \geq r$. Under pairwise independence, one can show $\mathbb{E}[Y_r] = d/2^r$ and bound Var$[Y_r]$. The concentration arguments (averaging and median trick) can then be applied to achieve accurate estimation.

Takeaway 3.1

Limited Independence as a Key Design Tool: While many analyses for randomized algorithms initially assume full independence (i.e., truly random hash functions), this assumption is impractical, in particular in the streaming setting, as storing a fully random function requires enormous space. A powerful and practical alternative is to use hash functions drawn from a family with **limited independence**, such as **pairwise independence**. The primary advantages are twofold:

1. **Space Efficiency:** A pairwise independent hash function can be represented very compactly, typically requiring only $O(\log n)$ bits of space. More generally, a k-wise independent hash function requires only $O(k \log n)$ bits.
2. **Analytical Sufficiency:** For many algorithms, including Flajolet-Martin and its improvements, pairwise independence provides enough structure to prove the necessary bounds on expectation and variance. This allows us to apply standard concentration techniques like averaging and the median trick, achieving strong (ϵ, δ) guarantees without the cost of full independence.

[1] Bertrand-Chebyshev theorem proves that for every integer n, there exists a prime number in $[n, 2n]$. See this Wiki page.