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

Lecture 2 (Sep 9, 2019): Frequency Moments and the AMS Algorithm

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

2.1 Frequency Moments

Suppose we have a stream of data $\sigma = a_1, a_2, a_3, \ldots, a_m$ where each $a_i \in \{1, \ldots, n\}$. We also define a frequency vector $f = (f_1, \ldots, f_n)$ for our stream σ. Each f_i denotes the number of occurences of a_i in σ.

Example 1 *Suppose we have a stream σ where $a_i \in \{1, 2, 3, 4, 5\}$ and let our stream be the following,*

$$\sigma = 3, 2, 1, 2, 3, 4, 5, 3, 5, 3, 2, 4, 5, 3, 2, 2$$

In this case, our frequency vector $f = (1, 5, 5, 2, 3)$.

Definition 1 *Given a stream σ, we will define the k^{th} moment σ, denoted by $F_k(\sigma)$ to be*

$$F_k(\sigma) = \sum_{i=1}^{n} f_i^k$$

We shall also assume $0^0 = 0$. Let us look at what we can say about $F_k(\sigma)$ from different values of k,

- *For $k = 0$, $F_k(\sigma) = \sum_{i=1}^{n} f_i^0 = \sum_{1 \le i \le n : f_i > 0} 1$, which is the number of distinct elements in the stream σ. In this case, we use the assumption that $0^0 = 0$.*
- *For $k = 1$, $F_k(\sigma) = \sum_{i=1}^{n} f_i = m$ is the length of the stream.*
- *As k gets bigger and $k \to \infty$, we define $F_\infty(\sigma)$ to denote the most frequency element in our stream. More formally,*
$$F_\infty(\sigma) = \max_i f_i$$

Let d denote the number of distinct elements in the stream, $d = |\{i : f_i > 0\}|$. It is possible to prove no deterministic algorithm can solve this problem in sublinear space. Hence, our goal is to estimate the value of d with high probability using a randomized approximation algorithm.

We will first present a simple algorithm by Flajolet and Martin [FM85], then study a modified version for it, popularly known as the AMS algorithm [AMS99], named after its authors, Alon, Matias, and Szegedy. [AMS99] won the Gödel Prize 2005 for laying the foundations of the analysis of data streams using limited memory. Before studying the algorithms, we will first introduce hash functions and study their properties.

2.2 Hash Functions

Suppose we have a universe U where $|U| = m$. We call $h : U \to T$ a hash function if it maps U to a table $T = \{0, \ldots, n-1\}$ where $n << m$. We want the hash function to satisfy a few properties,

- It should be easy to compute the value of $h(x)$ for any $x \in U$.
- h maps elements to tables uniformly at random (or close to it).

Definition 2 *A family $\mathcal{H}$ of hash functions $h : U \to T$ is 2-universal if $\forall x, y \in U$ where $x \neq y$ and $h \in \mathcal{H}$ selected randomly from $\mathcal{H}$ satisfies the property that*

$$\Pr[h(x) = h(y)] \leq \frac{1}{n}$$

Definition 3 *A family $\mathcal{H}$ of hash functions $h : U \to T$ is strongly 2-universal if $\forall x, y \in U$ where $x \neq y$ and all $n_1, n_2 \in \{1, \ldots, n\}$, it satisfies*

$$\Pr[h(x) = n_1 \wedge h(y) = n_2] \leq \frac{1}{n^2}$$

Definition 4 *A collection of random variables $X_1, \ldots, X_n$ are generally k-wise independent if for any subset of size at most k,*

$$\Pr[X_1 = a_1 \wedge \ldots \wedge X_t = a_t] = \prod_{j=1}^{t} Pr[X_j = Aj] \text{ where } 1 \leq t \leq k$$

Example 2 *Pairwise independence does not necessarily imply k-wise independence. Suppose X_1, X_2, X_3 are 0-1 random variables (Bernoulli) where X_1, X_2 are chosen uniformly at random and $X_3 = X_1 \bigoplus X_2$.*

In this case, all three variables are pairwise independent since it is possible to pick any two distinct variables, X_i and X_j and assign it any value (0 or 1) with probability

$$\Pr[X_i = a_i \wedge X_j = a_j] = \Pr[X_i = a_i] \cdot \Pr[X_j = a_j]$$

However, X_1, X_2, X_3 are not 3-wise independent since the assignment of any two variables dictates the value of the third variable.

Lemma 1 *If $X_1, \ldots, X_n$ are pairwise independent and $Y = \sum_{i=1}^{n} X_i$, then*

1. $\mathrm{Var}[Y] = \sum_{i=1}^{n} \mathrm{Var}[X_i]$
2. *Suppose $X_1, \ldots, X_n$ are 0-1 random variables (Bernoulli), then* $\mathrm{Var}[Y] \leq \sum_{i=1}^{n} \mathrm{E}[X_i^2] = \sum_{i=1}^{n} \mathrm{E}[X_i] = \mathrm{E}[Y]$

Proof.

$$\mathrm{Var}[Y] = \mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathrm{Var}[X_i] + 2\sum_{i=1}^{n}\sum_{j>i} \mathrm{Cov}(X_i, X_j)$$

X_i and X_j are pairwise independent for any $i, j \in \{1, \ldots, n\}$ and $i \neq j$ which implies $\mathrm{Cov}(X_i, X_j) = 0$.

$$\begin{aligned}\mathrm{Var}[Y] &= \mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] \\ &= \sum_{i=1}^{n} \mathrm{Var}\left[X_i\right] + \underbrace{2\sum_{i=1}^{n}\sum_{j>i} \mathrm{Cov}(X_i, X_j)}_{0} \\ &= \sum_{i=1}^{n} \mathrm{Var}\left[X_i\right]\end{aligned}$$

This proves the first part. To prove the second part, we will first show that $\mathrm{E}[X_i^2] = \mathrm{E}[X_i]$ when X_i is a Bernoulli random variable.

$$\begin{aligned}
\mathrm{E}[X_i^2] &= 1^2 \cdot \Pr[X_i = 1] + 0^2 \cdot \Pr[X_i = 0] \\
&= 1 \cdot \Pr[X_i = 1] + 0 \cdot \Pr[X_i = 0] \\
&= \mathrm{E}[X_i]
\end{aligned}$$

We will start to second prove part by using first part,

$$\begin{aligned}
\mathrm{Var}[Y] &= \sum_{i=1}^{n} \mathrm{Var}[X_i] \\
&= \sum_{i=1}^{n} \mathrm{E}[X_i^2] - \mathrm{E}[X_i]^2 \\
&\leq \sum_{i=1}^{n} \mathrm{E}[X_i^2] \\
&= \sum_{i=1}^{n} \mathrm{E}[X_i] \text{ using the fact that } \mathrm{E}[X_i^2] = \mathrm{E}[X_i] \\
&= \mathrm{E}[Y] \text{ using linearity of expectations}
\end{aligned}$$

■

Suppose we have a strongly 2-universal hash family, then $\forall x, y \in U$, $h(x)$ and $h(y)$ are pairwise independent. A k-universal hash function can be defined similarly where each $h(x)$ is uniformly distributed in the range and they are k-wise independent. There are constructions of k-universal hash families that can be stored in $O(k \log |U|)$ space.

2.3 The Flajolet-Martin Counter

We will first discuss a simple algorithm to estimate the number of distinct elements. The big picture idea of the algorithm is to put each distinct element between 0 and 1 uniformly at random. On expectation, if there are t distinct elements, then the interval $[0, 1]$ is divided into parts of size approximately $\frac{1}{t+1}$. We use a 2-universal hash function in order to place it uniformly at random between 0 and 1. The algorithm is the following,

1. Pick a random hash function $h \in \mathcal{H}$ where $h : \{1, \ldots, n\} \to [0, 1]$ and $\mathcal{H}$ is a family of 2-universal hash functions.
2. Maintain a counter X where $X = \min_i h(x_i)$

The algorithm works on the fact that $\mathrm{E}[X] = \frac{1}{t+1}$.

2.4 The AMS Algorithm

Before we state and analyze the algorithm, we will first introduce the ZEROES function. ZEROES(t) for any $t \in \{1, \ldots, n\}$ represents the largest power of 2 it is divisible by and it is also the number of leading 0's (counting from the right) of the binary representation of t. Let us look at a few examples,

- Suppose $t = 1024$. We can write $1024 = 2^{10}$ and $1024 = (10000000000)_2$ showing that $\textsc{zeroes}(t) = 10$.
- Suppose $t = 23$. The largest power of 2 which divides t is zero and $t = (10111)_2$ showing that $\textsc{zeroes}(t) = 0$. In general, for any prime p except 2, $\textsc{zeroes}(p) = 0$.
- Suppose $t = 48$. The largest power of 2 which divides t is 2^4 and $t = (110000)_2$ showing that $\textsc{zeroes}(t) = 4$.

We will first explain the big picture idea of the algorithm. We will assume we have seen sufficiently many numbers and these numbers are distributed uniformly. We will look at $\textsc{zeroes}(x)$ for each number in the stream and we expect one of the element, we will call it x (out of d distinct elements) to have value $\textsc{zeroes}(x) \geq \log d$ i.e., have $\log d$ zeroes from the right in its binary representation.

However, this relies on the idea that our stream has elements that are uniformly distributed, but this might not be the case. To ensure our elements are uniformly distributed, we first use a 2-universal hash function picked at random, $h \in \mathcal{H}$ and we check $\textsc{zeroes}(h(x))$ for $x \in \sigma$. We expect that on average, one out of the d distinct elements have $\textsc{zeroes}(h(x)) \geq \log d$. Using this idea, $\max\limits_{x \in \sigma} \textsc{zeroes}(h(x))$ would give us a good estimate of the number of distinct elements in the stream. The AMS algorithm is as follows,

```
1. Let $h \in \mathcal{H}$ be chosen uniformly at random from a 2-universal hash family $\mathcal{H}$.
2. Initialize $z$ to 0.
3. While there are elements in the stream, do:
4.     Pick the next element $a_i$
5.     $z \leftarrow \max\{z, \textsc{zeroes}(h(a_i))\}$
6. return $2^{z+\frac{1}{2}}$
```

2.4.1 Analysis of the AMS algorithm

For each $j \in \{1, \dots, n\}$ and $r \geq 0$, we will define a Bernoulli random variable $X_{r,j}$ as follows,

$$X_{r,j} = \begin{cases} 1, & \text{if } \textsc{zeroes}(h(j)) \geq r \\ 0, & \text{otherwise} \end{cases}$$

Let S be the set of all distinct elements in the stream.

$$Y_r = \sum_{j \in S} X_{r,j} = |\{x \in \sigma : \textsc{zeroes}(h(x)) \geq r\}|$$

Y_r is the random variable which represents the number of distinct elements with at least r zeroes from the right in the binary representation of their hashed value using $h \in \mathcal{H}$. We must note that $h(x)$ is uniformly random in $\{1, \dots, n\}$.

$$\mathrm{E}[X_{r,j}] = \Pr[\textsc{zeroes}(h(j)) \geq r] = \Pr[2^r \text{ divides } h(j)] = \frac{\frac{n}{2^r}}{n} = \frac{1}{2^r}$$

$$\mathrm{E}[Y_r] = \sum_{j \in S} \mathrm{E}[X_{r,j}] = \frac{d}{2^r} \text{ using linearity of expectations}$$

We must note that if d is a power of 2, then $\mathrm{E}[Y_{\log d}] = \frac{d}{2^{\log d}} = 1$. Since our family of hash functions $\mathcal{H}$ are 2-universal, we get that $X_{r,j}$ and $X_{r,j'}$ are pairwise independent. Since they are also 0-1 random variables, using the lemma we showed in this lecture, we obtain

$$\mathrm{Var}[Y_r] = \sum_{j\in S} \mathrm{Var}[X_{r,j}] \le \sum_{j\in S} \mathrm{E}\left[X_{r,j}\right] = \frac{d}{2^r}$$

We will bound the probability that $Y_r = 0$ and $Y_r > 0$ using concentration inequalities we defined in the previous lecture.

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

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

$$\begin{aligned}
\Pr[Y_r > 0] &= \Pr[Y_r \ge 1] \\
&\le \frac{\mathrm{E}[Y_r]}{1} \text{ using Markov's Inequality} \\
&\le \frac{d}{2^r} \\
\Pr\left[Y_r = 0\right] &\le \Pr\left[|Y_r - \mathrm{E}[Y_r]| \ge \frac{d}{2^r}\right] \\
&\le \frac{\mathrm{Var}[Y_r]}{\left(\frac{d}{2^r}\right)^2} \text{ using Chebyshev's inequality} \\
&\le \frac{2^r}{d}
\end{aligned}$$

Let t be the final value of z in the algorithm, so we return $\tilde{d} = 2^{t+\frac{1}{2}}$. Let a be the smallest integer such that $2^{a+\frac{1}{2}} \ge 3d$ where 3 is some constant factor. We want to know the probability that our output is more than thrice the the number of distinct elements,

$$\Pr[\tilde{d} \ge 3d] \le \Pr[t \ge a] = \Pr\left[Y_a > 0\right] \le \frac{d}{2^a} \le \frac{\sqrt{2}}{3}$$

Similarly, let b be the largest integer such that $2^{b+\frac{1}{2}} \le \frac{d}{3}$. Then

$$\Pr\left[\tilde{d} \le \frac{d}{3}\right] = \Pr[t \le b] = \Pr\left[Y_{b+1} = 0\right] \le \frac{2^{b+1}}{d} \le \frac{\sqrt{2}}{3}$$

We will now show the probability of $\tilde{d}$ being in between $\frac{d}{3}$ and d is at least $1 - \frac{2\sqrt{2}}{3}$.

$$\begin{aligned}
\Pr\left[\frac{d}{3} < \tilde{d} < 3d\right] &\ge 1 - \Pr\left[\tilde{d} \ge 3d \vee \tilde{d} \le \frac{d}{3}\right] \\
&\ge 1 - \left(\Pr\left[\tilde{d} \ge 3d\right] + \Pr\left[\tilde{d} \le \frac{d}{3}\right]\right) \text{ using union bound} \\
&\ge 1 - \left(\frac{\sqrt{2}}{3} + \frac{\sqrt{2}}{3}\right) \\
&= 1 - \frac{2\sqrt{2}}{3} \approx 0.057
\end{aligned}$$

We now have a $(3, 0.057)$ approximator which we can boost to $(3, 1-\delta)$ by repeating the above algorithm $O\left(\log\left(\frac{1}{\delta}\right)\right)$ times and picking the median of averages. Each iteration of this algorithm requires $O(\log n)$ space. If we run it $O\left(\log\left(\frac{1}{\delta}\right)\right)$ times, the total space complexity is $O\left(\log n \log\left(\frac{1}{\delta}\right)\right)$. The constant 3 can be decreased close to 1, which will be the topic of the next lecture.

References

FM85 P. Flajolet and G.N. Martin, Probabilistic Counting Algorithms for Data Base Applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

AMS99 N. Alon, Y. Matias, and M. Szegedy, The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.