Estimating Frequency Moments of Streams

In this class we will look at the two simple sketches for estimating the frequency moments of a stream. The analysis will introduce two important tricks in probability – boosting the accuracy of a random variable by consider the “median of means” of multiple independent copies of the random variable, and using k-wise independent sets of random variable.

1 Frequency Moments

Consider a stream $S = \{a_1, a_2, ..., a_m\}$ with elements from a domain $D = \{v_1, v_2, ..., v_n\}$. Let $m_i$ denote the frequency (also sometimes called multiplicity) of value $v_i \in D$; i.e., the number of times $v_i$ appears in $S$. The $k^{th}$ frequency moment of the stream is defined as:

$$F_k = \sum_{i=1}^{n} m_i^k$$

We will develop algorithms that can approximate $F_k$ by making one pass of the stream and using a small amount of memory $o(n + m)$.

Frequency moments have a number of applications. $F_0$ represents the number of distinct elements in the streams (which the FM-sketch from last class estimates using $O(\log n)$ space. $F_1$ is the number of elements in the stream $m$.

$F_2$ is used in database optimization engines to estimate self join size. Consider the query, “return all pairs of individuals that are in the same location”. Such a query has cardinality equal to $\sum_i m_i^2 / 2$, where $m_i$ is the number of individuals at a location. Depending on the estimated size of the query, the database can decide (without actually evaluating the answer) which query answering strategy is best suited. $F_2$ is also used to measure the information in a stream.

In general, $F_k$ represents the degree of skew in the data. If $F_k / F_0$ is large, then there are some values in the domain that repeat more frequently than the rest. Estimating the skew in the data also helps when deciding how to partition data in a distributed system.

2 AMS Sketch

Let’s first assume that we know $m$. Construct a random variable $X$ as follows:

- Choose a random element from the stream $x = a_i$.
- Let $r = |\{a_j | j \geq i, a_j = a_i\}|$, or the number of times the value $x$ appears in the rest of the stream (inclusive of $a_i$).
- $X = m(r^k - (r - 1)^k)$

$X$ can be constructing using $O(\log n + \log m)$ space – $\log n$ bits to store the value $x$, and $\log m$ bits to maintain $r$.

**Exercise:** We assumed that we know the number of elements in the stream. However the above can be modified to work even when $m$ is unknown. (Hint: reservoir sampling).

It is easy to see that $X$ is an unbiased estimator of $F_k$. 

1
\[
E(X) = \sum_{i=1}^{m} \frac{1}{m} E(X| {i}^{th} \text{ element in the stream was picked}) \\
= \frac{1}{m} \sum_{j=1}^{n} \sum_{k=1}^{m_i} E(X| a_i \text{ is the } k^{th} \text{ repetition of } v_j ) \\
= \frac{m}{m} \sum_{j=1}^{n} \left[ 1^k + (2^k - 1^k) + \ldots + (m_j^k - (m_j - 1)^k) \right] \\
= \sum_{j=1}^{n} m_j^k = F_k
\]

We now show how to use multiple such random variables \(X\) to estimate \(F_k\) within \(\epsilon\) relative error with high probability \((1 - \delta)\).

### 2.1 Median of Means

Suppose \(X\) is a random variable such that \(E(X) = \mu\) and \(\text{Var}(X) < c\mu^2\), for some \(c > 0\). Then, we can construct an estimator \(Z\) such that for all \(\epsilon > 0\) and \(\delta > 0\),

\[
P(\left| Z - \mu \right| > \epsilon \mu) < \delta
\]

by averaging \(s_1 = \Theta(\frac{c}{\epsilon^2})\) independent copies of \(X\), and then taking the median of \(s_2 = \Theta(\log(1/\delta))\) such averages.

**Means:** Let \(X_1, \ldots, X_{s_1}\) be \(s_1\) copies of \(X\). Let \(Y = \frac{1}{s_1} \sum_i X_i\). Clearly, \(E(Y) = E(X) = \mu\).

\[
\text{Var}(Y) = \frac{1}{s_1} \text{Var}(X) < \frac{c\mu^2}{s_1}
\]

\[
P(\left| Y - \mu \right| > \epsilon \mu) < \frac{\text{Var}(Y)}{\epsilon^2 \mu^2} \text{ by Chebyshev}
\]

Therefore, if \(s_1 = \frac{8c}{\epsilon^2}\), then \(P(\left| Y - \mu \right| > \epsilon \mu) < \frac{1}{8}\).

**Median of means:** Now let \(Z\) be the median of \(s_2\) copies of \(Y\). Let \(W_i\) be defined as follows:

\[
W_i = \begin{cases} 
1 & \text{if } |Y - \mu| > \epsilon \mu \\
0 & \text{else}
\end{cases}
\]

From the previous result about \(Y\), \(E(W_i) = \rho < \frac{1}{8}\). Therefore, \(E(\sum_i W_i) < s_2/8\). Moreover,
whenever the median \( Z \) is outside the interval \( \mu \pm \epsilon \), \( \sum_i W_i > s_2/2 \). Therefore,

\[
P(|Z - \mu| > \epsilon \mu) < P(\sum_i W_i > s_2/2)
\]

\[
\leq P(\left| \sum_i W_i - E(\sum_i W_i) \right| > s_2/2 - s_2\rho)
\]

\[
= P(\left| \sum_i W_i - E(\sum_i W_i) \right| > (\frac{1}{2\rho} - 1)s_2\rho)
\]

\[
\leq 2e^{-\frac{1}{4} \left( \frac{1}{2\rho} - 1 \right)^2 s_2\rho} \text{ by Chernoff bounds}
\]

\[
< 2e^{-\frac{\rho}{3}} \text{ when } \rho < \frac{1}{8}, \rho \left( \frac{1}{2\rho} - 1 \right)^2 > 1
\]

Therefore, taking the median of \( s_2 = 3 \log \left( \frac{3}{\delta} \right) \) ensures that \( P(|Z - \mu| > \epsilon \mu) < \delta \).

### 2.2 Back to AMS

We use the medians of means approach to boost the accuracy of the AMS random variable \( X \). For that, we need to bound the variance of \( X \) by \( c \cdot F_k^2 \).

\[
Var(X) = E(X^2) - E(X)^2
\]

\[
E(X^2) = \frac{m^2}{m} \sum_{i=1}^n \left[ 1^{2k} + (2^{2k} - 1^k) + \ldots + (m_i^{2k} - (m_i - 1)^{2k}) \right]
\]

When \( a > b > 0 \), we have

\[
a^k - b^k = (a - b)(\sum_{j=0}^{k-1} a^j b^{k-1-j}) \leq (a - b)(ka^{k-1})
\]

Therefore,

\[
E(X^2) \leq m \left[ k1^{2k-1} + (k2^{k-1})(2^k - 1^k) + \ldots + km^{k-1}(m_i^{k} - (m_i - 1)^{k}) \right]
\]

\[
\leq m \left[ km_1^{2k-1} + km_2^{2k-1} + \ldots + km_n^{2k-1} \right]
\]

\[
= kF_{1} F_{2k-1}
\]

**Exercise:** We can show that for all positive integers \( m_1, m_2, \ldots, m_n \),

\[
\left( \sum_i m_i \right) \left( \sum_i m_i^{2k-1} \right) \leq n^{1 - \frac{1}{k}} \left( \sum_i m_i^{k} \right)^2
\]

Therefore, we get that \( Var(X) \leq kn^{1 - \frac{1}{k}} F_k^2 \). Hence, by using the median of means aggregation technique, we can estimate \( F_k \) within a relative error of \( \epsilon \) with probability at least \( (1 - \delta) \) using \( O(kn^{1 - \frac{1}{k}} \frac{1}{\epsilon} \log \left( \frac{1}{\delta} \right)) \) independent estimators (each of which take \( O(\log n + \log m) \) space.

3
### 3 A simpler sketch for $F_2$

Using the above analysis we can estimate $F_2$ using $O(\sqrt{\pi}(\log n + \log m) \log (\frac{1}{\delta}))$ bits. However, we can estimate $F_2$ using much smaller number of bits as follows.

Suppose we have $n$ independent uniform random variables $x_1, x_2, \ldots, x_n$ each taking values in $\{-1, 1\}$. (This requires $n$ bits of memory, but we will show how to do this in $O(\log n)$ bits in the next section). We compute a sketch as follows:

- Compute $r = \sum_{i=1}^{n} x_i \cdot m_i$
- Return $r^2$ as an estimate for $F_2$.

Note that $r$ can be maintained as the new elements are seen in the stream by increasing/decreasing $r$ by 1 depending on the sign of $x_i$. Why does this work?

\[
E(r^2) = E[(\sum_i x_i m_i)^2] = \sum_i m_i^2 E[x_i^2] + 2 \sum_{i<j} E[x_i x_j m_i m_j]
\]

\[
= \sum_i m_i^2 = F_2 \text{ since } x_i, x_j \text{ are independent, } E(x_i x_j) = 0
\]

\[
Var(r^2) = E(r^4) - F_2^2
\]

\[
E(r^4) = E\left[ (\sum_i x_i m_i)^2 (\sum_i x_i m_i)^2 \right]
\]

\[
= E\left[ (\sum_i x_i^2 m_i^2) + 2 \sum_{i<j} x_i x_j m_i m_j \right]^2
\]

\[
= E\left[ (\sum_i x_i^2 m_i^2)^2 \right] + 4E\left[ (\sum_i x_i x_j m_i m_j)^2 \right] + 4E\left[ (\sum_i x_i^2 m_i^2) (\sum_{i<j} x_i x_j m_i m_j) \right]
\]

The last term is 0 since every pair of variables $x_i$ and $x_j$ are independent. Since $x_i^2 = 1$, the first term is $F_2^2$.

\[
Var(r^2) = E(r^4) - F_2^2 = 4E\left[ (\sum_{i<j} x_i x_j m_i m_j)^2 \right]
\]

\[
= 4E\left[ \sum_{i<j} x_i^2 x_j^2 m_i^2 m_j^2 \right] + 4E\left[ \sum_{i<j<k<\ell} x_i x_j x_k x_l m_i m_j m_k m_l \right]
\]

Again, the last term is 0 since every set of 4 random variables is independent of each other. Therefore,

\[
Var(r^2) = 4 \sum_{i<j} m_i^2 m_j^2 \leq 2F_2^2
\]

Therefore, by using the median of means method, we can estimate $F_2$ using $\Theta(\frac{1}{\sqrt{\pi}} \log (\frac{1}{\delta}))$ independent estimates. However, the technique we presented needs $O(n)$ random bits. We will reduce this to $O(\log n)$ bits in the next section by using 4-wise independent random variables rather than fully independent random variables.
3.1 $k$-wise Independent Random Variables

In the previous analysis, note that we only needed to use the fact that every set of 4 distinct random variables $x_i, x_j, x_k, x_l$ are independent of each. We call a set of random variables $X = \{x_1, \ldots, x_n\}$ to be $k$-wise independent random variables if every subset of $k$ random variables are independent. That is:

\[ \forall 1 \leq i_1 < i_2 < \ldots < i_k \leq n, P(\wedge_{j=1}^{k} x_{i_j} = a_j) = \prod_{j=1}^{k} P(x_{i_j} = a_j) \]

**Example:** Consider two fair coins $x$ and $y$. Let $z$ be a random variable that returns “heads” if $x$ and $y$ both land heads or both land tails (think XOR), and “tails” otherwise. We can easily check that any pair of $x$, $y$ and $z$ are independent, but all $x$, $y$ and $z$ are not independent.

In the above $F_2$ sketch, we only need the set of random variables $X$ to be 4-wise independent. We can generate $2^n$ $k$-wise independent variables using $O(n)$ bits using the random polynomial construction (and thus generate each $F_2$ estimate using $O(\log n)$ bits). The construction of 2-wise (or pairwise) independent random variables is shown below.

Consider a family of hash functions $\mathcal{H} = \{h_{a,b}|a, b \in \{0, 1\}^n\}$, where each $h_{a,b} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is defined as follows:

\[ h_{a,b}(x) = ax + b \]

That is a hash function is constructed by choosing $a$ and $b$ uniformly at random from $\{0, 1\}^n$. All elements are hashed using $h_{a,b}$. The values resulting from applying $\mathcal{H}$ to values in $\{0, 1\}^n$ are $2^n$ pairwise independent random variables.

**Lemma 1.**

\[ \forall x, y, P(\mathcal{H}(x) = y) = 2^{-n} \]

**Proof:** Exercise

**Lemma 2.**

\[ \forall x, y, z, w P(\mathcal{H}(x) = y \land \mathcal{H}(z) = w) = 2^{-2n} \]

**Proof sketch:** Consider any hash function $h_{a,b}$. Given hash values $y, w$ for $x, z$ respectively, we can find a unique solution for the linear system of equations involving $a, b$. Therefore, only one pair out of the $2^{-2n}$ pairs will result in $x, z$ hashing to $y, w$ respectively. Therefore, the probability is $2^{-2n}$.

We can also easily see that the resulting variables $\mathcal{H}(x)$ are not 3-wise independent. For instance,

\[ P(\mathcal{H}(1) = 2 \land \mathcal{H}(2) = 3 \land \mathcal{H}(3) = 100) = 0 \]

This is because the first two has $h$ value force $a = 1, b = 1$, and the third hash value is not possible using $h_{1,1}$.

The above construction can be extended to generate $k$-wise independent random variables by using random polynomials of the form $\sum_{i=0}^{k-1} a_i x^i$. 