COMS E6998-9: Algorithmic Techniques for Massive Data — Sep 15, 2015

Lecture 3 – Frequency Moments, Heavy Hitters

Instructor: *Alex Andoni* — Scribes: *Daniel Alabi, Wangda Zhang*

1 Introduction

This lecture is about the second frequency moment and heavy hitters. First, we present the Tug-of-War algorithm by Alon-Matias-Szegedy to obtain a $1+\epsilon$ approximation using $O(\frac{1}{\epsilon}\log n)$ space. Second, we define heavy hitters and present the CountMin algorithm which can be used to obtain heavy hitters.

2 Second Frequency Moment

Assume – as in previous lectures – that we are given a stream of length m from which we want to obtain a frequency vector $f = (f_1, f_2, \ldots, f_n)$ (n distinct elements) where f_i is the frequency of the ith distinct element in the stream, then the kth frequency moment of the stream is

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

In Lectures 1 and 2, we estimated F_0 and F_1 corresponding to the number of distinct elements and the length of the stream respectively. Now we wish to obtain the second moment:

$$F_2 = \sum_{i=1}^{n} f_i^2 \tag{2}$$

The idea is to use i.i.d random variables $r_i (= r(i))$ where $P(r_i = 1) = P(r_i = -1) = \frac{1}{2}$ (Rademacher random variables) to get an estimator.

Thus, $\mathbb{E}[r_i] = 0$ and $\text{Var}[r_i] = 1$ (since $r_i^2 = 1$ and $\text{Var}[r_i] = E[r_i^2] - (E[r_i])^2$). We can think of r_i as a random variable defined for every distinct element so that

$$r : [n] \to \{-1, +1\} \tag{3}$$

As an example, let's consider when the entries of the frequency vector is all 1s ($\Rightarrow m = n$). Let

$$z = \sum r_i \tag{4}$$

Then $\mathbb{E}[z] = \sum \mathbb{E}[r_i] = 0$ by linearity of expectation and using the distribution of r_i. Similarly, $\text{Var}[z] = \text{Var}[\sum r_i] = m$. We can apply Chebyshev to obtain that $|z - 0| = |z| \leq O(\sqrt{m})$ with constant probability. Turns out that this bound is tight.

Now, let's consider the more general case and define

$$z = \sum_i r_i \cdot f_i \tag{5}$$

Claim 1. $\mathbb{E}[z^2] = \sum_{i=1} f_i^2 \; = F_2$

Proof.

$$\begin{aligned}
\mathbb{E}[z^2] &= \mathbb{E}[(\sum_{i=1}^{n} r_i f_i)^2] && (6)\\
&= \sum_i \mathbb{E}[r_i^2] f_i^2 + \sum_{i \neq j} \mathbb{E}[r_i]\mathbb{E}[r_j] f_i f_j && (7)\\
&= \sum_i f_i^2 \mathbb{E}[r_i^2] && (8)\\
&= \sum_i f_i^2 = F_2 && (9)
\end{aligned}$$

(7) holds because for $i \neq j$, r_i and r_j are independent, while (8) holds because $\mathbb{E}[r_i] = 0$ for all i. Finally, (9) holds because $r_i^2 = 1 (\Rightarrow \mathbb{E}[r_i^2] = 1)$. □

Claim 2. $Var[z^2] \leq O(1) \cdot F_2^2$

Proof. First, let's compute $\mathbb{E}[z^4]$

$$\begin{aligned}
\mathbb{E}[z^4] &= \mathbb{E}[(\sum_i r_i f_i)^4] && (10)\\
&= \mathbb{E}[\sum_{i,j,k,l} r_i f_i r_j f_j r_k f_k r_l f_l] && (11)\\
&= \sum_{i,j,k,l} f_i f_j f_k f_l \mathbb{E}[r_i r_j r_k r_l] && (12)\\
&= \sum_i f_i^4 + 6 \sum_{i<j} f_i^2 f_j^2 && (13)\\
&\leq O(1) \cdot (\sum_i f_i^2)^2 && (14)
\end{aligned}$$

(13) holds because:

- Of independence of the random variables
- The terms with odd powers of r_i evaluate to zero
- There are $\binom{4}{2} = 6$ terms of the form $r_i^2 r_j^2$

Finally, we obtain that $\mathrm{Var}[z^2] \leq \mathbb{E}[z^4] \leq O(1) \cdot F_2^2$ □

Having bounded $\mathbb{E}[z^2]$ and $\mathrm{Var}[z^2]$, we present the algorithm below

2.1 Tug-of-War (Alon-Matias-Szegedy 1996)

We maintain a counter z.

1. Initialize $z = 0$
2. When we see element i: $z = z + r(i)$
3. Return the estimator z^2

Earlier, we defined r_i in the form $r : [n] \to \{-1, +1\}$ where r_i acts like a hash function. The hash function for the r_is should be 4-wise independent.

Next, we apply the "average trick" using $k = O(\frac{1}{\epsilon^2})$ parallel runs of the algorithm to obtain a $1 + \epsilon$ approximation in $O(\frac{1}{\epsilon^2} \log n)$ space.

2.2 Linearity

So far we have only considered simple estimators, and we next consider something more complex. Suppose we have two parts of a stream seen by two different estimators, and we want to estimate the union of these two parts (i.e., how we can combine them).

Claim 3. *Linearity: given estimates z' for f' and z'' for f'', we can combine them as $z = z' + z''$ for $f = f' + f''$.*

Proof. Since $z' = \sum r_i f'_i$ and $z'' = \sum r_i f''_i$, we have $z' + z'' = \sum r_i(f'_i + f''_i)$. Note that we need to use the same randomness for two estimates. $\square$

Similarly, we can use $(z' - z'')^2$ to estimate $\sum (f'_i - f''_i)^2$. However, we cannot use linearity in a similar way for $\sum |f'_i - f''_i|$, and will discuss this pointer later in class.

2.3 General Streaming Model

We now consider a more generalized model: at each moment, we have an update (i, δ_i) to increase the i-th entry by δ_i. (δ_i may be negative)

A linear algorithm S handles this easily, $S(f + e_i\delta_i) = S(f) + S(e_i\delta_i)$. We call S a sketch. According to [Nguyen-Li-Woodruff'14], any algorithm for general streaming might as well be linear.

3 Heavy Hitters

Now that we are able to compute many types of frequency counts, we wonder if we can also compute the max frequency in a stream. It turns out that we cannot: it is impossible to approximate the max frequency in sublinear space. Therefore, we will solve a more modest problem where we want to detect the max-frequency element if it is very heavy. We will show that we can find these heavy hitters in space $O(1/\phi)$.

Definition 4. *i is ϕ-heavy if $f_i \geq \phi \sum_j f_j$.*

The basic idea is still to use hash functions. A first-attempt method uses a single hash function h mapping from $[n]$ to $[w]$ randomly, where $w = O(1/\phi)$. Then each element i goes to bucket i, and we sum up the frequencies in each of the w buckets. We denote the sum of each bucket as S. So the estimator for f_i is $\hat{f} = S(h(i))$.

For example, consider a stream of 2, 5, 7, 5, 5. If the hash function h_1 works as $h_1(2) = 2, h_1(5) = 1, h_1(7) = 2$, then we will obtain the following estimates: $\hat{f_2} = 2, \hat{f_5} = 3, \hat{f_7} = 2$. However, for an element that never appears in the stream, e.g. 11, this method also estimates its frequency as $\hat{f_{11}} = 2$, assuming $h_1(11) = 2$.

Claim 5. *$S(h(i))$ is a biased estimator.*

Proof. Analyzing this estimator, we have $\hat{f_i} = S(h(i)) = f_i + \sum_{\{j:h(j)=h(i)\}} f_j$. Let $C = \sum_{\{j:h(j)=h(i)\}} f_j$. Thus,

$$\mathbb{E}[C] = \sum_j Pr[h(j) = h(i)] \cdot f_j = \sum_{j \neq i} \frac{f_j}{w} \neq 0$$

□

However, it is easy to see that $\mathbb{E}[C] \leq \frac{\sum_j f_j}{w}$. So the bias is at most $\sum_j f_j/w$, which is small for $f_i \gg \sum_j f_j/w$. By Markov Inequality, we have $Cle\frac{10\sum_j f_j}{w}$ with at least 90% probability, i.e.

$$Pr[\hat{f_i} - f_i < O\left(\frac{\sum_j f_j}{w}\right)] \geq 0.9$$

For $w = O\left(\frac{1}{\epsilon\phi}\right)$ and $f_i \geq \phi \sum_j f_j$, we have $C \leq \epsilon f_i$. That is, $\hat{f_i}$ is a $(1 + \epsilon)$ approximation (with 90% probability).

Still, there are two issues with this estimator: (1) only constant probability; (2) overestimate for many indices (10%). Fundamentally, there is a conflict between avoiding many collisions and reducing space used by the hash table.

3.1 CountMin

This motivates us to use the "median trick". We can use $L = O(\log n)$ hash tables with hash functions h_j. The CountMin algorithm works as follows:

```
Initialize(r, L):
    array S[L][w]
    L hash functions h_1,...,h_L, into {1,...,w}

Process(int i):
    for (j = 0; j < L; ++j)
        S[j][h_j(i)] += 1

Estimator:
    foreach i in PossibleIP:
        f̂_i = median_j(S[j][h_j(i)]
```

Claim 6. *The median is a* $\pm\epsilon\phi$ *estimator with* $(1-1/n^2)$ *probability.*

Proof. For an index i, each row (out of the L rows) gives $\hat{f}_i = f_i \pm \epsilon\phi$ with 90% probability. By Median Trick (see Lecture 2), the median gives an estimator of $\pm\epsilon\phi$ with $(1-1/n^2)$ probability. $\square$

Alternatively, we can take the min, instead of median, since all counts are overestimated.

3.2 Output Heavy Hitters

We can now identify the heavy hitters by iterating over all i's and outputing those with $\frac{\hat{f}_i}{\sum_j f_j} \geq \phi$. In particular, for true frequencies f_is,

- if $\frac{f_i}{\sum_j f_j} \leq \phi(1-\epsilon)$, then i is not in output;
- if $\frac{f_i}{\sum_j f_j} \geq \phi(1+\epsilon)$, then i is reported as a heavy hitter;
- if $\phi(1-\epsilon) \leq \frac{f_i}{\sum_j f_j} \leq \phi(1+\epsilon)$, then i may or may not be reported as a heavy hitter.

If we really care about those elements in between, then we could take more space to further narrow the gap.

The space used is $O\left(\frac{\log^2 n}{\epsilon\phi}\right)$ bits. Since we iterate over all i's, the time complexity is $\Omega(n)$.

We can improve this time complexity, at the cost of increasing space to $O\left(\frac{\log^3 n}{\epsilon\phi}\right)$ bits. The idea is to use dyadic intervals. For each level, we maintain its own sketch, and find the heavy hitters by following down the subtrees of heavy hitters in intermediary.