COMS E6998-9: Algorithms for Massive Data (Spring'19)	Jan 31, 2019

Lecture 4: Heavy Hitters, CountMin, Linearity

Instructor: *Alex Andoni* Scribes: *Jia Wan and Weston Jackson*

1 Heavy Hitters

Theorem 1. *Any algorithm for 2-approximation for F_∞ (randomized) requires $\Omega(n)$ space.*

Definition 2. *For $\phi \in (0,1)$, $i \in [n]$ is a ϕ heavy-hitter (HH) if $f_i > \phi \sum_j f_j = \phi \|f\|_1$.*

Our goal is to show that we can find all $\phi-$heavy hitters using space $O(\frac{\log n}{\phi})$.

Idea: Use a hash function $\mathcal{H} : [n] \to [w]$, where $w = O(\frac{1}{\phi})$ and hash table $H[w] \in \mathbb{R}^w$. Then for each element i that goes to bucket k, we sum up the frequencies for each of the w buckets, denote:

$$H[k] = \sum_{i:h(i)=k} f_i, \forall k \in [w]$$

In the form of algorithm:

- Update i: H[h(i)]++
- Estimator: $\forall i \in [n], \hat{f}_i = H[h(i)]$.

Note that $\hat{f}_i$ is a biased estimator, but we can show the bias is small.
Fix a heavy hitter i, then we have $f_i > \phi \|f\|_i$ and:

$$\hat{f}_i = f_i + \sum_{j \neq i: h(j)=h(i)} f_j \text{ (extra)}$$

where:

$$\begin{aligned}
\mathbb{E}_h[\text{extra}] &= \mathbb{E}[\sum_{j \neq i} f_j \chi[h(j) = h(i)]] \\
&= \sum_{j \neq i} f_j \mathbb{E}[\chi[h(j) = h(i)]] \\
&\leq \frac{\|f\|_1}{w}
\end{aligned}$$

Suppose $w = \frac{100}{\phi}$, then $\mathbb{E}[\text{extra}] \leq \frac{\phi}{100}\|f\|_1$. By Markov Inequality:

$$Pr[\text{extra} > \frac{\phi}{10}\|f\|_1] \leq 0.1$$

This means that with probability 90%, $\hat{f}_i \in [f_i, f_i + \frac{\phi}{10}\|f\|_1]$, therefore:

- If i is heavy hitter, $\hat{f}_i > \phi\|f\|_1$
- If i is not $\frac{\phi}{2}$-heavy hitter, $\hat{f}_i < \phi\|f\|_1$

However, there are issues with this estimator. Any j colliding with heavy hitter i will be indistinguishable. Fix: use the median trick, with $L = O(\log n)$ hash tables.

1.1 CountMin

CountMin: $L = O(\log n), h_1, h_2, \ldots, h_L : [n] \to [w], H_1, H_2, \ldots, H_L \in \mathbb{R}^w$
Algorithm:

- $\forall k \in [n], \forall j \in [L], H_j[k] = \sum_{i:h_j(i)=k} f_i$
- $\hat{f}_i = med_{j\in[L]} H_j[h_j(i)]$

Claim 3. *For $w = O(\frac{1}{\epsilon\phi})$, with probability $1 - \frac{1}{n}, \forall i \in [n]$, $f_i \leq \hat{f}_i \leq f_i + \epsilon\phi\|f\|_1$.*

Proof. By median trick w/$\delta = \frac{1}{n^2}$:

$$Pr[\hat{f}_i \leq f_i + \epsilon\phi\|f\|_1] \geq 1 - \frac{1}{n^2}$$

union bound over all $i \in [n]$:

$$Pr[\hat{f}_i \leq f_i + \epsilon\phi\|f\|_1] \geq 1 - \frac{1}{n}$$

□

Observation 4. *We can take minimum instead of median since each estimator can only overestimate, so $\hat{f}_i \leq f_i + \epsilon\phi\|f\|_1$ still holds and everything follows.*

Theorem 5. *CountMin algorithm uses space $O(\frac{1}{\epsilon\phi}\log n)$ words of $O(\log n)$ bits, and outputs a set S such that with probability $\geq 1 - \frac{1}{n}$:*

- *if i is $\phi(1+\epsilon) - HH$ then $i \in S$*
- *if i is not $\phi(1-\epsilon) - HH$, then $i \notin S$*

Proof. Let S be the set of all items where $\hat{f}_i > \phi\|f\|_1$

- If i is $\phi - HH$, then $\hat{f}_i > f_i > \phi\|f\|_1$, therefore $i \in S$.
- If i is not $\phi(1-\epsilon) - HH$, then $\hat{f}_i \leq f_i + \epsilon\phi\|f\|_1 < (\phi(1-\epsilon) + \epsilon\phi)\|f\|_1 \leq \phi\|f\|_1$, therefore $i \notin S$

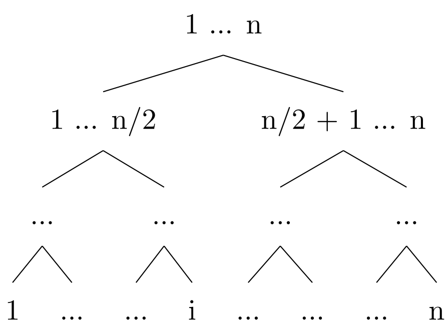
□

Problem: to compute S, need $\hat{f}_i$ for all $i \in [n]$, which takes $\Theta(n \log n)$ time. We can improve the time complexity at the cost of increasing space.

1.2 Faster Runtime?

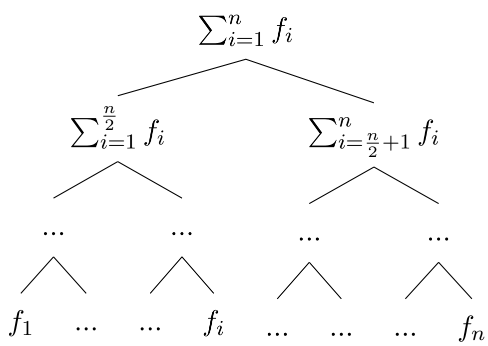
Theorem 6. *CM can achieve $O(\frac{1}{\phi}\log^2 n)$ time with $O(\frac{1}{\phi\epsilon}\log^2 n)$ space.*

Proof. Use dyadic intervals:



Construction:

- Leaves of the tree are the elements in the frequency vector.
- Each node in the tree calculates the total frequency from elements in its subtree.
- It follows the total number of levels is $\log n$.



We create a stream for each level of the tree:

- $Stream_1$ is a stream with single frequency: $\sum f_i$
- $Stream_2$ is a stream with frequencies: $\sum_{i=1}^{n/2} f_i$, $\sum_{i=n/2+1}^{n} f_i$
- ...
- $Stream_{\log n}$ is a stream with frequencies: $f_1, f_2, ...f_n$

Keep a CM-sketch for each of these streams.

Observation 7. *If i is a ϕ-HH then all ancestors of i in the tree are also ϕ-HH.*

This holds because each ancestor I s.t. $\{i\} \subset I$ has $f_I \geq f_i \geq \phi||f||_1$. Thus, our HH algorithm is simply to do a binary search for the heavy hitters on the tree!

Runtime analysis:

- Number of active nodes per-level is $\leq \frac{1}{\phi}$
- Total number of nodes of the tree to check is $\leq O(\frac{1}{\phi} \log n)$
- Total runtime is $O(\frac{1}{\phi} \log^2 n)$

$\square$

2 Linearity

Assume we have two traffic streams f and f', and we want to take of sketch of the entire traffic. The entire traffic is:

$$f + f' \in \mathbb{N}^n$$

We can assume some random seed for $CM(f)$ such that:

$$CM(f) = A \cdot f$$

where A is an $(Lw) \times n$ matrix and $f \in \mathbb{N}^n$. Thus we have:

$$CM(f + f') = A(f + f') = Af + Af' = CM(f) + CM(f')$$

3 General Turnstyle Streaming

We want to calculate $||f - f'||_p$ (for example, the number of packets that got lost/generated in a cyber network).

Definition 8. *G.T.S.M: Sequence of updates* (x, δ_x) *s.t.* $f_x := f_x + \delta_x$ *where* $x \in [n]$ *and* $\delta_x \in \mathbb{R}$.

- So far in CM-sketch, $\delta_x = 1$
- CM is not good enough to solve GTSM for $p = 1$. Consider $p = 1$, $||f - f'|| = 2m$ if f and f' are disjoint items.
- Neither is FM for $p = 0$ good enough
- $p = 2$, ToW works in GTSM model:

$$ToW(f) - ToW(f') = ToW(f - f') \approx ||f - f'||_2$$