COMS 4995-3: Advanced Algorithms Feb 1, 2017

Lecture 5 – Heavy Hitter Problem

Instructor: *Alex Andoni* Scribes: *Shankara Pailoor and Piyushi Bishnoi*

1 Introduction

In lecture 4, we discussed the FM, and FM+ sketching algorithm to count number of distinct elements in a stream. In this lecture we begin by proving some impossibility results on the distinct element counting problem described in lecture 4. We then discuss a different but related sketching problem called **Heavy Hitters**. The motivation behind the impossibility results is to use an algorithm that will require $\Omega(n)$ space while the motivation for the Heavy Hitters problem is to find the value in the stream that is the most frequent, as well in $\Omega(n)$ time.

2 Impossibility Results

Theorem 1. *A deterministic Exact Count algorithm requires $\Omega(n)$ space*

Proof. Any deterministic algorithm A will take as input a stream of values $x_1, x_2, \cdots x_m$ and store the count with some s bits. (Just as a quick note, $s = \#$ bits used by an algorithm solving the problem, which is this case is the determining exact problem).

You can fix this algorithm using s bits where $m \leq n + 1$. Since the input values are represented by n bits, if we can construct an injective function from $\{0,1\}^n$ to $\{0,1\}^s$ then $\{0,1\}^n \subset \{0,1\}^s$ giving us $s = \Omega(n)$. Take $z \in \{0,1\}^n$ and let z_i denote the value of the i^{th} bit. We construct a stream from z by putting i into the stream if $z_i = 1$ and denote it by stream_z. We claim the following function is injective $f(z) \triangleq A(\text{stream}_z)$ and prove it below.

□

Claim 2. *$f(z)$ is injective*

Proof. We will prove this by reconstructing z from $f(z)$. Let $\alpha = A(\text{stream}_z)$ and let $\beta_i = A(\text{stream}_z \cup \{i\})$ where z_i is the i^{th} bit of z. To reconstruct z from $f(z)$ we do the following:

- Compute α
- For each $i = 1, \cdots, n$ compute β_i.
- If $\beta_i = \alpha$ then $z_i = 1$

  If $\beta_i - 1 = \alpha$ then $z_i = 0$.

Since we only send in the positions of the bits of z which are 1 and the algorithm is deterministic and exact, we recover z.

Also, if f is invertible, then $s \geq n$. □

Outline: Proving impossibility for determining the approximate count

Theorem 3. *A deterministic $1+\epsilon$ approximation algorithm requires $\Omega(n)$ space.*

Proof. The proof assumes that there exists a set $T \subset \{0,1\}^n$ of size $2^{\Omega(n)}$ such that $z = (z_1, z_2 \cdots z_n) \in T \implies$ (1) $\|z\|_1 = \frac{n}{2}$, (2) For $x, y \in T, \|x-y\|_1 > \frac{n}{6}$. Like in our proof above, we construct the same function f but restricted to the domain of T. Our procedure to recover z is as follows:

- Compute $\alpha = f(z)$
- For each $y \in T$, run $f(\text{stream}_z \cup \text{stream}_y\})$
- If $\beta_y - \alpha \leq \frac{n}{2} * \epsilon$ then $z = y$

We also notice that if $z = y$ then # distinct elements doesn't change much and if $z \neq y$ then the # of distinct elements changes a lot.

The last claim can be proven by considering two cases. If $z = y$ then there will be a total of $\frac{n}{2}$ distinct elements seen. Since the difference when running the algorithm twice on the same input is less than $1+\epsilon$ we have $\beta_y - \alpha \leq \frac{n}{2} * \epsilon$. If $z \neq y$ then since $\|z-y\| \geq n/6$ and $\|z\| = \|y\| = n/2$ there are at least $n/12$ positions where $z_i = 0$ and $y_i = 1$. Therefore $\beta_y \geq n/2 + n/12$ and so

$$\beta_y - \alpha > n/2 + n/12 - n/2(1+\epsilon) > n/2 * \epsilon$$

Therefore, $y = x \iff \beta_y - \alpha \leq \frac{n}{2} * \epsilon$. This tells us that $T \subset \{0,1\}^s$ and since $|T| = \Omega(2^n)$ we have $s = \Omega(n)$ □

3 Heavy Hitters

We will now describe a new sketching/streaming problem. Consider a sequence of values $x_1, x_2, \cdots, x_m$ where $x_c \in [n]$. What is the most frequent value? This is known as the Heavy Hitters problem and unfortunately, all randomized and 2-approximation solutions take $\Omega(n)$ space. However, we can tackle a more modest goal of finding values that are *sufficiently* heavy.

Example 4. *You are running a web application and your sequence of values is ips. In order to detect possible DOS attacks you want to keep, track of who is hitting you the hardest.*

Definition 5. *Assume $f_i \triangleq$ frequency of i. Take $i \in [n]$, $\phi \in (0,1)$ and let f_i denote the frequence of i in the stream. We say that i is* **heavy** *if $f(i) \geq \phi \sum_{j=1}^{n} f_j = m\phi$.*

We can now ask the more modest question: *Given a sequence of values $x_1, x_2, \cdots, x_m$ where $x_c \in [n]$, what are the heavy elements?* Our first attempt is to look at the general idea of hashing. Our second attempt at a solution is called the Bucketing Algorithm. The idea behind it is to use a universal hash

function h which points to w buckets. The value we store in the array A is the sum of collisions we have in the bucket. While this is an overestimate, we can say with a high probability e.g. 90% that it is heavy. The Bucketing Algorithm is outlined below. It takes as input a universal hash function $h : [n] \to [w]$

```
1: procedure BucketingAlgorithm(h)
2:     A[1 ··· w] = 0
3:     while next input do
4:         x_c ← next input
5:         A[h(x_c)] += 1
```

Estimator: Once all the values have been bucketed we return all frequencies $\hat{f}_i = A[h(i)]$ if $\hat{f}_i > m\phi$. Note: If there are zeroes in this stream, we can map them anywhere because they don't make a difference in the sum of collisions.

Claim 6. *For a given $\phi < 1$, there exists $\epsilon > 0$ such that $f_i \leq \hat{f}_i \leq f_i + m\epsilon\phi$ with probability $\geq 90\%$*

Proof. Fix $i \in [n]$. We observe $A[h(i)] = f_i + \sum_{j \neq i : h(j) = h(i)} f_j$ because all occurrences of i and those elements which collide with i will land in $h(i)$. Let $C = \sum_{j \neq i : h(j) = h(i)} f_j$ (which is what we will notice is the "bad" part as it is an overestimate). We now want to compute $\mathbb{E}[C]$

$$\mathbb{E}[C] = \mathbb{E}_h \left[\sum_{j \neq i : h(j) = h(i)} f_j \right] \tag{1}$$

$$= \mathbb{E}_h \left[\sum_{j \neq i} \mathbb{1}_{h(j) = h(i)} f_j \right] \tag{2}$$

$$= \sum_{i \neq j} \frac{1}{w} f_j \leq \frac{m}{w} \tag{3}$$

By Markov's, $Pr\left[C \geq \frac{10m}{w}\right] = \frac{\mathbb{E}[C]w}{10m} \leq \frac{mw}{10mw} = 0.1$. For sufficiently small ϵ we have $\frac{10m}{w} = m\epsilon\phi$. Rearranging terms, we get $w = \frac{10}{\phi\epsilon} \implies Pr[f_i \leq \hat{f}_i \leq f_i + m\epsilon\phi] \geq .9$

If $f_i < (1 - \epsilon)\phi m \to$ then not reported as heavy.

If $f_i \geq \phi m$ then surely reported as heavy.

This gives us a $(1 + \epsilon)$ approximation to the Heavy Hitters Problem. □

The drawback of this approach is that approximately 10% of input will be flagged as heavy and for sufficiently large input this could be very noisy. However, we can reduce this probability of false positive in a similar manner to FM+ by running L copies of bucketing algorithm. We call this the Count Min Algorithm.

1: **procedure** CountMinAlgorithm()
2: Select independent hash functions $h_1, h_2, \cdots, h_L \in H$
3: Initialize $A[L][w]$ each with $h_i[n] \to [w]$
4: Run L copies of Bucketing Algorithm
5: **return** Estimators $\hat{f}_i = \min_j A[h_j(i)]$

Claim 7. $Pr[\hat{f}_i > f_i + m\epsilon\phi] \leq 0.1^L$

Proof. Since our hash functions were selected independently, we have

$$Pr[\hat{f}_i \geq f_i + \phi\epsilon m] = \prod_{j \in [L]} Pr[A[h_j(i)] > f_i + \phi\epsilon m] \leq 0.1^L$$

□

To achieve an error rate $\delta < \frac{1}{n^2}$ for a given $\hat{f}_i$ we need $0.1^L \leq \frac{1}{n^2} \implies L > \frac{2\log(n)}{\log(10)}$. Taking $L = C\log(n)$ for $C > \frac{2}{\log(10)}$ is sufficient.

Claim 8. *The Counting Min Sketch Algorithm for the Heavy Hitters Problem with fixed n, $\phi < 1$ and $\epsilon \in (0,1)$ satisfies the following (1) $Pr[(\exists i)(\hat{f}_i > f_i + \epsilon\phi m)] < \frac{1}{n}$ (2) Uses $O(L\omega) = O(\frac{\log(n)}{\phi\epsilon})$ space.*

Proof. $Pr[(\exists i)$ s.t. $(\hat{f}_i$ is an overestimate by $> f_i + \epsilon\phi m)] < \sum_{i=1}^{n} Pr[\hat{f}_i > f_i + \epsilon\phi m] \leq n\frac{1}{n^2} = \frac{1}{n}$. The first inequality follows from union bound and so proves the first property. The algorithm uses $O(wL)$ space. As we showed in Claim 7, we can take $L = C\log(n)$ for a sufficiently large n. Moreover, as showed in Claim 6, we can take $w = \frac{10}{\phi\epsilon}$ and maintain these bounds, which gives us $Lw = \frac{C\log(n)}{10\phi\epsilon} \leq C'\frac{\log(n)}{\phi\epsilon}$. Thus, the algorithm takes $O(\frac{\log(n)}{\epsilon\phi})$ space. □

4 Heavy Hitters Summary

Our initial solution to the more modest goal of the H.H.P was the Bucketing Algorithm. The downside was its high error rate of 10%. As a result we leveraged a technique we used in FM of lecture 4 to run L copies of the bucketing algorithm and return the minimum value across all buckets. We called this algorithm the Count Min Sketch. The reason we return the minimum is that all the estimators are biased because since they returned frequencies larger than the actual frequency. Thus, the minimum estimator for $i \in [n]$ is closest to the true value. By choosing $L = O(\log(n))$ along with sufficiently small ϵ and ϕ, the fraction of incorrect heavy hitters dropped to $\frac{1}{n}$ and our algorithm uses $O(\frac{\log(n)}{\phi\epsilon})$ space.