CS 388R: Randomized Algorithms, Fall 2019 Oct. 31, 2019

Lecture 19: Streaming (continued)

Prof. Eric Price *Scribe: Haoran Zhang, Sung Yeon Choi*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Introduction

Given a universe $\mathbb{U} = [n]$ and incoming stream $\{x_1...x_m\}$, we defined the frequency of each element by $f_j = |\{x_i | x_i = j\}|$. Last class, we learned how to estimate $||f||_0$, which shows a number of distinct elements in the stream. $||f||_1$ is number of elements in the stream (and thus can be calculated easily using O(log m) space). In this lecture, we will see how to estimate $||f||_2$.

2 Estimating 2-norm - AMS algorithm

Suppose we are interested in finding the second moment of frequencies. The easiest solution would be to keep count of all possible elements that come in during the stream then calculate the moment. This will take $n \log m$ space. Suppose that we don't have that much space and would like to use only $O(\log n)$ space to estimate $||f||_2$, or more formally, find the value of an estimate $\hat{F}_2$ such that,

$$(1-\epsilon)||f||_2 \leq \hat{F}_2 \leq (1+\epsilon)||f||_2 \qquad \text{w.p. } 1-\delta \tag{1}$$

We will achieve this by using AMS algorithm, which uses the idea of Johnson-Lindenstrauss lemma we saw in Lecture 17.

2.1 Johnson-Lindenstrauss

Recall that Johnson-Lindenstrauss states $\exists A \in \mathbb{R}^{kxn}$ where $A_{ij} \sim N(0, \frac{1}{k})$ such that, for sequence $\{x_1...x_n\}$,

$$(1-\epsilon)||x||_2 \leq ||Ax||_2 \leq (1+\epsilon)||x||_2 \qquad \text{w.p. } 1-\delta \tag{2}$$

where $k = O(\frac{1}{\epsilon^2} \log \frac{2}{\delta})$.

It seems that we can use directly to estimate $||f||_2$, but there is a problem: matrix A can be large. We use the intuition from JL lemma to achieve the next algorithm with a good bound.

2.2 AMS algorithm

AMS algorithm let us estimate $||f||_2$ by maintaining a linear sketch of f. Idea of algorithm is,

- Pick $\{s_1...s_n\} \sim \{-1, 1\}$ using 4-wise independent hash functions

- Calculate $Y = \sum_{i \in [n]} s_i f_i$
- Return Y^2

Notice that since frequency is updated linearly, $Y = \sum_{i \in [n]} s_i f_i$ is equivalent to $Y = \sum_{j \in [m]} h(x_j)$ where h is 4-wise independent hash function, and therefore does not require any space to compute other than the space required for hash family.

Let us now look at the expectation of Y to see if the algorithm actually outputs the values we want.

$$\begin{aligned}
\mathbb{E}[Y] &= \sum_{i \in [n]} \mathbb{E}[s_i] f_i \\
&= 0 \\
\mathbb{E}[Y^2] &= \mathbb{E}[\sum_{i \in [n]} s_i^2 f_i^2 + \sum_{i \neq j} s_i s_j f_i f_j] \\
&= ||f||_2^2 \\
\mathbb{E}[Y^4] &\leq 2||f||_2^4 \\
Var(Y^2) &= \mathbb{E}[Y^4] - \mathbb{E}[Y^2]^2 \\
&\leq \mathbb{E}[Y^2]^2
\end{aligned}$$

From above, we see that the expected value of Y^2 is $||f||_2^2$, which gives us the 2-norm estimation of frequency. Applying Chebyshev's inequality and median of means technique, we get $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ space used like in JL lemma. Since constructing the hash family takes $O(\log n)$ space, we need $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log n)$ space in total, which is what we aimed for.

3 Heavy Hitter

Suppose we are interested in finding the set of elements in the stream that appear frequently (i.e. the heavy hitters). More formally, we can define the problem that given a stream $x_1, \ldots, x_m$, let frequency $f_j = |\{x_i | x_i = j\}|$, we want to find the set H_α of elements that appears more than αm times,

$$H_\alpha = \{i \in [n] | f_i \geq \alpha m\} \tag{3}$$

where $\alpha \in (0, 1)$.

3.1 Insertion-only

We start with the insertion only model. For simplicity, let $\alpha = \frac{1}{k}$, the goal is to find $H_{\frac{1}{k}}$. With **Misra-Gries** algorithm, we can solve this problem deterministically. More specifically, we keep at most $k - 1$ counters of $(j, \hat{f}_j)$. As scanning the stream, we modify the counters according to each incoming x_i

Claim 1. $H_{\frac{1}{k}} \subseteq M$

Algorithm 1 Misra-Gries Algorithm

```
procedure MISRA-GRIES(k)
    M = ∅
    for i in [m] do
        if x_i = j ∈ M then
            f̂_j += 1
        else
            if |M| < k − 1 then
                Insert new counter (j, 1)
            else
                ∀j ∈ M, (j, f̂_j) ← (j, f̂_j − 1)
                if f̂_j == 0 then
                    Remove (j, f̂_j)
    return M
```

Proof. In another word, with Misra-Gries algorithm, we have no false negatives, but it is possible to have false positive.

As shown in algorithm 1, the counters are only decremented when M is full. For each time, k items are decremented together ($k-1$ items in M and the incoming item). With m counts in total, we can only have at most $\frac{m}{k}$ decrements. Thus, if $f_j > \frac{m}{k}$, $\hat{f}_j > 0$. □

Claim 2. $f_i - \frac{m}{k} \leq \hat{f}_i \leq f_i$

Proof. Similar to the proof of claim 1, with at most $\frac{m}{k}$ decrements, it can be easily shown that such inequality holds. □

3.2 Strict turnstile - Count Min Sketch

However, if deletion is allowed, Misra-Gries algorithm can be easily fooled. How should we solve the problem in this case?

Consider a hash fucntion $h : [n] \to [B]$. Then $Y_{h(j)} = \sum_{j,h(j)=u} f_j$ is the number of items in bucket $u \in B$. Let $\hat{f}_j = Y_{h(j)}$ be the estimator of f_j, we can have

$$\begin{aligned} \mathbb{E}[|f_j - \hat{f}_j|] &= \mathbb{E}[\sum_{k \neq j} f_k \cdot \mathbb{1}_{h(k)=h(j)}] \\ &\leq \frac{||f||_1}{B} \end{aligned} \tag{4}$$

Let $B = \frac{\alpha}{4}$, by Markov's inequality we can have

$$Pr(|f_j - \hat{f}_j| \geq \alpha ||f||_1) \leq \frac{1}{4} \tag{5}$$

We can augment such result by running with $r = O(\log n)$ pairwise independent hash functions and let $\hat{f}_j = \min_r Y_{h(j),r}$, which gives us $f_j \leq \hat{f}_j \leq f_j + \alpha ||f||_1$ w.h.p.