Algorithms for Big Data (IV) (Fall 2020)

Instructor: Chihao Zhang
Scribed by: Guoliang Qiu

Last modified on Oct 28, 2020

In today's lecture, we will introduce some algorithms for the frequency estimation problem in the streaming model.

1 The Problem

In the streaming model, one may need to know the number of occurrences of a specific element in the data stream. For example, a network router may query the click numbers of a particular visitor. Formally, consider a stream of numbers of $\langle a_1, \cdots, a_m \rangle$ where $a_i \in [n]$ and its frequency vector $\mathbf{f} = (f_1, \cdots, f_n)$. We want to estimate f_a for each $a \in [n]$.

In the following sections, we will introduce three different algorithms for the frequency estimation problem. The first one, the Misra-Gries algorithm, is a deterministic algorithm but has the drawback that it cannot apply to the so-called *turnstile model* which will be defined later. After that, we introduce the Count Sketch and Count Min algorithms. Both of them work well in the turnstile model but have different performance guarantees.

2 The Misra-Gries Algorithm

If we do not limit the use of the memory, then we can use a Hash table A to faithfully record the frequency vector $\mathbf{f}$. This require us to store n numbers, and each of which costs $\log m$ bits. On the other hand, if we only allow A to store say k numbers, how can we maintain the table so that we can recover the count of any number in the stream with reasonable accuracy? The idea of Misra-Gries is simple: The table A keeps at most the count of k numbers seen so far. Once there comes a new number whose count is not recorded and the table A is full, the algorithm decreases the count of each number in A. The algorithm is described in Algorithm 1, The memory cost is clearly $O(k \log m)$. For the accuracy of the estimation, we claim that for each $j \in [n]$, the output $\hat{f}_j$ satisfies

$$f_j - \lfloor \frac{m}{k+1} \rfloor \leq \hat{f}_j \leq f_j.$$

The bound $\hat{f}_j \leq f_j$ is obvious, so we should only explain why $f_j - \lfloor \frac{m}{k+1} \rfloor \leq \hat{f}_j$. Consider those rounds of the algorithm where the gap between $\hat{f}_j$ and f_j enlarges. It must be the case that in these rounds, we were decreasing the counts for each of the k elements in A. Moreover, the number y we immediately received in this round is not recorded in A. Therefore each increment of the gap between $\hat{f}$ and f is witnessed by $k+1$ distinct positions in the input stream. There are m numbers in the stream in total, so we have $f_j - \lfloor \frac{m}{k+1} \rfloor \leq \hat{f}_j$.

Algorithm 1 Misra-Gries Algorithms for Frequency-Estimation

Init:
A Table A (A set of pairs ⟨key, value⟩ and the size of the table is k)
On Input y:
if $y \in keys(A)$ **then** $A[y] \leftarrow A[y] + 1$
else
  if $|keys(A)| \leq k - 1$ **then** $A[y] \leftarrow 1$
  else
    for all $l \in keys(A)$ **do**
      $A[l] \leftarrow A[l] - 1$
      if $A[l] = 0$ **then**
        Remove l from A
      end if
    end for
  end if
end if
Output: On query j,
if $j \in keys(A)$ **then**
  $\hat{f}_j = A[j]$
else
  $\hat{f}_j = 0$
end if

One main drawback of the Misra-Gries algorithm is that it does not apply to the turnstile model. In the turnstile model, instead of receiving a simple a_j in each round, the algorithm receives a pair (a_j, Δ_j) meaning Δ_j copies of a_j arrive at the same time. Therefore we would equivalently have the frequency $f_{a_j} \leftarrow f_{a_j} + \Delta_j$. Note that the number Δ_j can even be negative.

3 The Count-Sketch Algorithm

The Count-Sketch algorithm is described in Algorithm 2.

Algorithm 2 Count Sketch

Init:
An array $C[j]$ for $j \in [k]$ where $k = \frac{3}{\epsilon^2}$.
A random Hash function $h : [n] \rightarrow [k]$ from a 2-universal family.
A random Hash function $g : [n] \rightarrow \{-1, 1\}$ from a 2-universal family.
On Input (y, Δ):
$C[h(y)] \leftarrow C[h(y)] + \Delta \cdot g(y)$
Output: On query a:
Output $\hat{f}_a = g(a) \cdot C[h(a)]$.

Now we shall prove that the estimate $\hat{f}_a$ is close to f_a with high probability. Let $X = \hat{f}_a$ be the output on the query a and for every $j \in [n]$, let Y_j be the indicator of $h(j) = h(a)$. Then we can represent the

random variable X as:

$$X = g(a) \cdot \left(\sum_{j=1}^{n} f_j \cdot g(j) \cdot Y_j \right) = f_a + \sum_{j \in [n] \setminus \{a\}} f_j \cdot g(a) \cdot g(j) \cdot Y_j = f_a.$$

To prove concentration, we compute the variance of X. Let $Z := \sum_{j \in [n] \setminus \{a\}} f_j \cdot g(a) \cdot g(j) \cdot Y_j$, then $X = f_a + Z$ and $\mathbf{Var}[X] = \mathbf{Var}[Z]$. We first analyze $\mathbf{E}\left[Z^2\right]$.

$$\begin{aligned}
\mathbf{E}\left[Z^2\right] &= \mathbf{E}\left[\left(\sum_{j \in [n] \setminus \{a\}} f_j \cdot g(a) \cdot g(j) \cdot Y_j \right)^2 \right] \\
&= \mathbf{E}\left[\sum_{j \in [n] \setminus \{a\}} f_j^2 \cdot Y_j^2 + \sum_{j, j' \in [n] \setminus \{a\} : j \neq j'} f_j \cdot f_{j'} \cdot g(j) \cdot g(j') \cdot Y_j \cdot Y_{j'} \right] \\
&= \mathbf{E}\left[\sum_{j \in [n] \setminus \{a\}} f_j^2 \cdot Y_j^2 \right] = \sum_{j \in [n] \setminus \{a\}} f_j^2 \cdot \mathbf{E}\left[Y_j^2\right].
\end{aligned}$$

Note that for every $j \neq a$,

$$\mathbf{E}\left[Y_j^2\right] = \mathbf{E}\left[Y_j\right] = \mathbf{Pr}\left[h(j) = h(a)\right] = \frac{1}{k}.$$

Therefore

$$\mathbf{Var}[X] = \mathbf{E}\left[Z^2\right] - (\mathbf{E}[Z])^2 = \frac{\sum_{j \in [n] \setminus \{a\}} f_j^2}{k} = \frac{\|\mathbf{f}\|_2^2 - f_a^2}{k}.$$

Using the Chebyshev inequality,

$$\mathbf{Pr}\left[\left| \hat{f}_a - f_a \right| \geq \epsilon \left\| \mathbf{f}_{-a} \right\|_2 \right] \leq \frac{1}{k\epsilon^2} = \frac{1}{3}.$$

where $\|\mathbf{f}_{-a}\|_2 := \sqrt{\|\mathbf{f}\|_2^2 - f_a^2}$.

The memory cost of the above algorithm is $O(\frac{1}{\epsilon^2} \log m + \log n)$. We can use the median trick to boost the success probability and obtain:

$$\mathbf{Pr}\left[\left| \hat{f}_a - f_a \right| \geq \epsilon \left\| \boldsymbol{f}_{-\boldsymbol{a}} \right\|_2 \right] \leq \delta,$$

with memory cost $O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log m + \log \frac{1}{\delta} \log n\right)$.

The word "sketch" in the name of the algorithm is an important notion in streaming algorithms. Given a stream σ, the algorithm in fact computes a data structure C so that one can retrieve information on frequencies from C. The object C can be viewed as a "summary" of the information in σ. Moreover, one can design an algorithm $\mathcal{A}$ so that given two summaries C_1 and C_2 with respect to two streams σ_1 and σ_2 respectively, $\mathcal{A}(C_1, C_2)$ is the summary of $\sigma_1 \circ \sigma_2$, the concatenation of σ_1 and σ_2.

In fact, in the Count-Sketch algorithm, the table C can be viewed as a vector in $\mathbb{R}^k$, and the mapping from $\mathbf{f} \in \mathbb{R}^n$ to $C \in \mathbb{R}^k$ is linear. Namely there exists a matrix A so that $A\mathbf{f} = C$. So for two streams σ_1 and σ_2 with corresponding frequency vectors $\mathbf{f}_1$ and $\mathbf{f}_2$, we have $A(\mathbf{f}_1 + \mathbf{f}_2) = A\mathbf{f}_1 + A\mathbf{f}_2$. Therefore the "combing algorithm $\mathcal{A}$" for the Count-Sketch algorithm is a linear operator. A sketching algorithm like this is sometimes called "linear sketching".

4 Count Min

Another simple algorithm that can be applied to the turnstile model for estimating frequencies is Count-Min. However, in the lecture, we assume each input $(a_j, \Delta_j$ satisfies $\Delta_j > 0$. How to drop the requirement has been left as an exercise.

Algorithm 3 Count Min

Init:
An array $C[1, \cdots, t][1, \cdots, k]$ where $t = \lceil \log \frac{1}{\delta} \rceil$ and $k = \frac{2}{\epsilon}$.
Choose t independent random Hash functions $h_1, \cdots, h_t : [n] \to [k]$ from a 2-universal family.
On Input (y, Δ):
For each $i \in [t]$, $C[i][h_i(y)] \leftarrow C[i][h_i(y)] + \Delta$
Output: On query a:
Output $\hat{f}_a = \min_{1 \le i \le t} C[i][h_i(a)]$.

Given a, i, we define $Y_{i,j}$ as the indicator random variable of $h_i(a) = h_i(j)$. Let $X_i := C[i][h_i(a)] - f_a \ge 0$, then $X_i = \sum_{j \in [n] \setminus \{a\}} f_j \cdot Y_{i,j}$. Therefore,

$$\mathbf{E}[X_i] := \sum_{j \in [n] \setminus \{a\}} f_j \mathbf{E}\left[Y_{i,j}\right] = \frac{\|f_{-a}\|_1}{k}.$$

where $\|f_{-a}\|_1 := \sum_{j \in [n] \setminus \{a\}} f_j$.

According to the Markov inequality, we can show that

$$\mathbf{Pr}\left[X_i \ge \epsilon \|f_{-a}\|_1\right] \le \frac{1}{k\epsilon} = \frac{1}{2},$$

Due to the fact that the hash function $h_1, \cdots, h_t$ are independent, we have

$$\mathbf{Pr}\left[\left|\hat{f}_a - f_a\right| \ge \epsilon \|f_{-a}\|_1\right] \le 2^{-t} = \delta.$$

The algorithm computes a linear sketch using $O\left(\frac{1}{\epsilon} \log \frac{1}{\delta} \cdot \log m + \log \frac{1}{\delta} \log n\right)$ bits of memory.