Algorithms for Big Data (V)

Chihao Zhang

Shanghai Jiao Tong University

Oct. 18, 2019

Review of the Last Lecture

Last time, we learnt Misra-Gries and Count Sketch for `Frequency Estimation`.

The later has the advantage of being a linear sketch.

It also generalize to turnstile model.

Count Sketch

Algorithm Count Sketch

Init:

An array $C[j]$ for $j \in [k]$ where $k = \frac{3}{\varepsilon^2}$.

A random Hash function $h : [n] \to [k]$ from a 2-universal family.

A random Hash function $g : [n] \to \{-1, 1\}$ 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 $\widehat{f}_a = g(a) \cdot C[h(a)]$.

The Performance

We can apply the median trick to obtain:

- $\mathbf{Pr}\left[\left|\widehat{f}_a - f_a\right| \geqslant \varepsilon\|\mathbf{f}\|_2\right] \leqslant \delta$;
- it costs $O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\left(\log m + \log n\right)\right)$ bits of memeory.

Today we will see another simple sketch algorithm.

Count-Min

We assume that for each entry (y, Δ), it holds that $\Delta \geqslant 0$.

Algorithm Count-Min

Init:

An array $C[i][j]$ for $i \in [t]$ and $j \in [k]$ where $t = \log(1/\delta)$ and $k = 2/\varepsilon$.

Choose t independent random Hash function $h_1, \ldots, h_t : [n] \to [k]$ 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 $\widehat{f}_a = \min_{1 \leqslant i \leqslant t} C[i][h(a)]$.

Analysis

Obviously we have $f_a \leqslant \widehat{f}_a$.

Our algorithm overestimates only if for some $b \neq a$, $h_i(b) = h_i(a)$. Let $Y_{i,b}$ be the indicator of this event.

Let X_i be $C[i][h_i(a)]$. Then

$$\mathbf{E}\left[\widehat{X}_i\right] = \sum_{b \in [n]} f_b \mathbf{E}\left[Y_{i,b}\right] = f_a + \sum_{b \in [n] : b \neq a} f_b \mathbf{E}\left[Y_{i,b}\right] = f_a + \frac{\sum_{b \in [n] : b \neq a} f_b}{k} \leqslant f_a + \frac{\|\mathbf{f}\|_1}{k}.$$

Thus,

$$\mathbf{Pr}\left[|X_i - f_a| \geqslant \varepsilon \|\mathbf{f}\|_1\right] \leqslant \frac{\|\mathbf{f}\|_1}{k \varepsilon \|\mathbf{f}\|_1} = \frac{1}{2}.$$

Since our output is the minimum out of t independent X_i's,

$$\begin{aligned}
\mathbf{Pr}\left[\widehat{f}_a - f_a \geqslant \varepsilon\|\mathbf{f}\|_1\right] &= \mathbf{Pr}\left[|\min\{X_1, \ldots, X_t\} - f_a| \geqslant \|\mathbf{f}\|_1\right] \\
&= \mathbf{Pr}\left[\bigwedge_{i=1}^{t} \left(|X_i - f_a| \geqslant \varepsilon\|\mathbf{f}\|_1\right)\right] \\
&= \prod_{i=1}^{t} \mathbf{Pr}\left[|X_i - f_a| \geqslant \varepsilon\|\mathbf{f}\|_1\right] \leqslant 2^{-t} = \delta.
\end{aligned}$$

The algorithm computes a linear sketch using

$$O\left(\frac{1}{\varepsilon} \log \frac{1}{\delta} \cdot (\log m + \log n)\right)$$

bits of memory.

It can be generalized to turnstile model (Exercise).

Frequency Moments

The k-th frequency moment of a stream is

$$F_k \triangleq \sum_{j \in [n]} f_j^k = \|\mathbf{f}\|_k^k.$$

For example, F_2 is the size of self-join of a relation r.

Many problems we met before can be viewed as estimating F_k for some special k.

AMS Estimator for F_k

Given $\langle a_1, \ldots, a_m \rangle$, then algorithm first sample a uniform index $J \in [m]$.

It then count the number of entries a_j with $a_j = a_J$ and $j \geqslant J$.

Algorithm AMS Estimator for F_k

Init: $(m, r, a) \leftarrow (0, 0, 0)$.

On Input (y, Δ)**:**

$m \leftarrow m + 1$, $\beta \sim \mathrm{Ber}(\frac{1}{m})$;

if $\beta = 1$ **then**

  $a \leftarrow y, r \leftarrow 0$;

end if

if $y = a$ **then** $r \leftarrow r + 1$

end if

Output:

$m\left(r^k - (r-1)^k\right)$.

Analysis

We first compute the expectation of the output X.

Assuming $a = j$ at the end of algorithm, then

$$\mathbf{E}[X \mid a = j] = \mathbf{E}\left[m(r^k - (r-1)^k) \mid a = j\right] = \sum_{i=1}^{f_j} \frac{1}{f_j} \cdot m\left(i^k - (i-1)^k\right) = \frac{m}{f_j} \cdot f_j^k.$$

Therefore,

$$\mathbf{E}[X] = \sum_{j=1}^{n} \mathbf{Pr}[a = j] \cdot \mathbf{E}[X \mid a = j] = \sum_{j=1}^{n} \frac{f_j}{m} \cdot \frac{m}{f_j} \cdot f_j^k = F_k.$$

The Variance

$$
\begin{aligned}
\mathbf{Var}\left[X\right] \leqslant \mathbf{E}\left[X^2\right] &= \sum_{j=1}^{n} \frac{f_j}{m} \sum_{i=1}^{f_j} \frac{1}{f_j} \cdot m^2 \left(i^k - (i-1)^k\right)^2 \\
&\leqslant m \sum_{j=1}^{n} \sum_{i=1}^{f_j} k i^{k-1} \left(i^k - (i-1)^k\right) \\
&\leqslant m \sum_{j=1}^{n} k f_j^{k-1} \sum_{i=1}^{f_j} \left(i^k - (i-1)^k\right) \\
&= m \sum_{j=1}^{n} k f_j^{k-1} \cdot f_j^k = k \left(\sum_{j=1}^{n} f_j\right) \left(\sum_{j=1}^{n} f_j^{2k-1}\right).
\end{aligned}
$$

Assume $k \geqslant 1$ and let $f_* \triangleq \max_{j \in [n]} f_j$.

$$\begin{aligned}
\mathbf{Var}[X] &\leqslant k \sum_{j=1}^{n} f_j \cdot \left(f_*^{k-1} \sum_{j=1}^{n} f_j^k \right) \\
&\leqslant k \sum_{j=1}^{n} f_j \cdot \left(\left(f_*^k\right)^{\frac{k-1}{k}} \sum_{j=1}^{n} f_j^k \right) \\
&\leqslant k \sum_{j=1}^{n} f_j \cdot \left(\sum_{j=1}^{n} f_j^k \right)^{\frac{k-1}{k}} \sum_{j=1}^{n} f_j^k
\end{aligned}$$

Applying Jensen's inequality on $g(z) = z^{1/k}$, we can bound above by

$$k \sum_{j=1}^{n} \left(f_j^k\right)^{\frac{1}{k}} \left(\sum_{j=1}^{n} f_j^k \right)^{\frac{k-1}{k}} \sum_{j=1}^{n} f_j^k \leqslant k n^{1-1/k} \left(\sum_{j=1}^{n} f_j^k \right)^{\frac{1}{k}} \left(\sum_{j=1}^{n} f_j^k \right)^{\frac{k-1}{k}} \sum_{j=1}^{n} f_j^k = k n^{1-1/k} F_k^2.$$

Therefore,

$$\mathbf{Pr}\left[|X - F_k| \geqslant \varepsilon F_k\right] \leqslant \frac{kn^{1-1/k}}{\varepsilon^2}.$$

Now we can apply the standard averaging trick and median trick.

To kill the $n^{1-1/k}$ factor in the variance, we need to average $\Omega\left(n^{1-1/k}\right)$ estimates.

An (ε, δ) estimator requires

$$O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}kn^{1-1/k}\left(\log m + \log n\right)\right)$$

bits of memory.

The Tug-of-War Sketch

The following simple algorithm for F_2 outperforms AMS by using only $O(\log n + \log m)$ bits.

Algorithm Tug-of-War Sketch

Init:

A random Hash function $h : [n] \to \{-1, 1\}$ from a 4-universal family.

$x \leftarrow 0$.

On Input (y, Δ)**:**

$x \leftarrow x + \Delta \cdot h(y)$

Output:

Output x^2.

Analysis

Let X be the value of x at the end of our algorithm.

$$\mathbf{E}\left[X^2\right] = \mathbf{E}\left[\left(\sum_{j\in[n]} f_j h(j)\right)^2\right] = \mathbf{E}\left[\sum_{j\in[n]} f_j^2 h(j)^2 + \sum_{i,j\in[n]:i\neq j} f_i f_j h(i)h(j)\right] = F_2.$$

Using the property of 4-universal Hash family, we have

$$\begin{aligned}\mathbf{E}\left[X^4\right] &= \sum_{i,j,k,\ell\in[n]} f_i f_j f_k f_\ell \mathbf{E}\left[h(i)h(j)h(k)h(\ell)\right] \\ &= \sum_{j\in[n]} f_j^4 \mathbf{E}\left[h(j)^4\right] + 6\sum_{i,j\in[n]:j>i} f_i^2 f_j^2 \mathbf{E}\left[h(i)^2 h(j)^2\right] = F_4 + 6\sum_{i,j\in[n]:j>i} f_i^2 f_j^2.\end{aligned}$$

Therefore

$$\begin{aligned}
\mathbf{Var}\left[X^2\right] &= \mathbf{E}\left[X^4\right] - \left(\mathbf{E}\left[X^2\right]\right)^2 \\
&= F_4 - F_2^2 + 6 \sum_{i,j \in [n]: j > i} f_i^2 f_j^2 \\
&= F_4 - F_2^2 + 3(F_2^2 - F_4) \leqslant 2F_2^2.
\end{aligned}$$

Finally, we apply the median trick and it costs

$$O\left(\frac{1}{\varepsilon^2} \log \frac{1}{\delta} \left(\log n + \log m\right)\right)$$

bits of memory.