Learning Outcomes

- Streaming framework (streaming data, streaming algorithm)
- Streaming problems
  - Frequency estimation
  - heavy hitters detection
- Streaming algorithms/sketches for above problems
  - Misra-Gries Sketch
  - Count Sketch
  - Count-Min Sketch
- Biased vs unbiased approximation
- Probabilistic analysis objectives
- Probabilistic analysis techniques
  - Markov's inequality
  - Chebyshev's inequality
  - Hoeffding's inequality

This note mainly refers to [Phillips, 2021, Chapter 11].

1 Frequency Estimation in Stream

Data, problems, constraints, and the notations.

Given:

- A universe $[u]$, which is too big to store in memory, usually $\lg u$ will be a constant.
- A stream (sequence) $X = (x_1, x_2, \ldots, x_n)$, $x_i \in [u]$ for all $i \in [n]$.

Goal: Count all the items, so you can return the frequency of any query $q \in [u]$:

$$c(q) := \sum_{x \in X} \mathbb{1}(x, q), \quad f(q) := \frac{c(q)}{n}.$$

Where $\mathbb{1}$ is the indicator function, $\mathbb{1}(x, q) = 1$ if $x = q$, $\mathbb{1}(x, q) = 0$ otherwise. We know the space lower bound: $\Omega(n)$.

Streaming implies:

- we cannot store the whole data in memory, space have to be independent of n ($\log n$ is fine, some paper allows $\log^2 n$).
- we can only process X once, from x_1 to x_n (some paper allows multiple passes, but we will stay with 1 pass).

Motivation: DDoS attack detection at router, i.e., detect high frequency IP addresses with limited memory.

The compromised goal is: $\hat{f}_\varepsilon$, the ε-approximation of f, s.t. for all $q \in [u]$:

$$f(q) - \varepsilon \leq \hat{f}_\varepsilon(q) \leq f(q) + \varepsilon$$

The key problem of this note is to find a streaming data structure for $\hat{f}_\varepsilon$. We will see 3 of such data structures in this note.

1.1 Heavy Hitters in Stream

A highly related problem is to find the ϕ-heavy hitters. $y \in [u]$ is a ϕ-heavy hitter iff $f(y) > \phi$.

The ε-approximation of the ϕ-heavy hitters, $\hat{H}^\phi_\varepsilon$, is a set such that:

- $y \in \hat{H}_\varepsilon^\phi$ if $f(y) > \phi$.
- $y \notin \hat{H}_\varepsilon^\phi$ if $f(y) < \phi - \varepsilon$.

If $\phi - \varepsilon \le f(y) \le \phi$, then y may or may not be in $\hat{H}_\varepsilon^\phi$.

If you have $\hat{f}_\varepsilon$, then you have $\hat{H}_\phi^\phi$ for any $\phi \ge \varepsilon$.

Lemma 1. *Given $\hat{f}_\varepsilon$, if $\phi \ge \varepsilon$, then $\hat{H} := \{y \in [u] \mid \hat{f}_\varepsilon(y) > \phi - \varepsilon\}$ is a $\hat{H}_\phi^\phi$.*

Proof. If $f(y) > \phi$, then $\hat{f}_\varepsilon(y) \ge f(y) - \varepsilon > \phi - \varepsilon$, so $y \in \hat{Y}$. The second requirement of the $\hat{H}_\phi^\phi$ is trivial since there is no y s.t. $f(y) < 0s$. □

2 Misra-Gries Sketch

2.1 Majority

Majority problem: Find y if $f(y) > \frac{1}{2}$. Equivalent to the 1/2-approximation of the 1/2-heavy hitter, $\hat{H}_{1/2}^{1/2}$, $y \in \hat{H}_{1/2}^{1/2}$ if $f(y) > \frac{1}{2}$.

Algorithm 1, a beautiful algorithm by Boyer and Moore [1981], Moore [1981], solve the majority problem in with $O(\log u + \log n)$ space, actually just a key and a counter, if a single machine word fits each, then the space is $O(1)$, 2 words + code, cannot be less.

Algorithm 1: Majority(X)

```
1 $y \leftarrow \text{NaN}, c \leftarrow 0$
2 **forall** $x \in X$ **do**
3     **if** $y = x$ **then** $c \leftarrow c + 1$
4     **else if** $c = 0$ **then** $y \leftarrow x, c \leftarrow 1$
5     **else** $c \leftarrow c - 1$
6 **return** $y$
```

Theorem 2. *If there is a majority z in X, i.e. $nf(z) > n/2$, then the Algorithm 1 outputs z after processed X.*

Here is one way to get an insight into the correctness of the algorithm. When Line 5 is executed, the algorithm must have seen a pair of distinct items $x_i \neq x_j, i < j$, $x_i = y, x_j = x$. And the result state of the algorithm will be the same if x_i, x_j was not in the stream. If there is a majority z in X, i.e. $nf(z) > n/2$, then there are at most $n - nf(z) < n/2$ pairs of distinct items. If you delete all these pairs, the rest will be a sequence of all z's, so the algorithm will output z.

Another way to see the correctness, is to understand two factors when c reach 0:

- The algorithm goes back to inital state and starts to process the rest of the sequence. This mean the output of the whole sequence will be the same with the output of the rest sequence.
- The majority of the whole sequence must be the majority of the rest sequence as well.

Both of these two ways can be formalized into a proof by induction.

2.2 Misra-Gries Sketch

Misra and Gries [1982] extends the majority algorithm by increasing the number of keys and counters from 1 to k, so that the algorithm can solve the generalize majority problem: find the ε-approximation of the ε-heavy hitters, $\hat{H}_\varepsilon^\varepsilon$, i.e. $y \in \hat{H}_\varepsilon^\varepsilon$ if $f(y) > \varepsilon$.

Algorithm 2: Misra-Gries(X, k)

```
1 $Y \leftarrow [\text{NaN}] * k, C \leftarrow [0] * k$
2 forall $x \in X$ do
3     if $\exists i(Y[i] = x)$ then $C[i] \leftarrow C[i] + 1$
4     else if $\exists i(C[i] = 0)$ then $Y[i] \leftarrow x, C[i] \leftarrow C[i] + 1$
5     else
6         forall $i$ do $C[i] \leftarrow C[i] - 1$
7 return $Y, C$
```

To approximate the frequency of any $q \in [u]$, return

$$\hat{f}_{MG}(q) := \begin{cases} \frac{C[i]}{n} & \exists i(Y[i] = q) \\ 0 & \text{otherwise} \end{cases}$$

Lemma 3. *Above approximation of the frequency satisfies*

$$f(q) - \frac{1}{k+1} \leq \hat{f}_{MG}(q) \leq f(q)$$

for all $q \in [u]$.

The upper bound is obvious. The lower bound depens by the number of times Line 6 executes.

When Line 6 executes, there must be $k+1$ distinct item are decremented. It can hapen at most $n/(k+1)$ times.

Theorem 4. *If $k \geq \frac{1}{\varepsilon} - 1$, then the output of Algorithm 2, $\hat{f}_{MG}$ is an $\hat{f}_\varepsilon$.*

Proof. Use Lemma 3, set $\frac{1}{k+1} = \varepsilon$, we have

$$f(q) - \varepsilon \leq \hat{f}_{MG}(q) \leq f(q) < f(q) + \varepsilon$$

and $k = \frac{1}{\varepsilon} - 1$. □

Theorem 5. *If $k \geq \frac{1}{\varepsilon} - 1$, then the output of Algorithm 2, Y is a $\hat{H}^\varepsilon_\varepsilon$.*

Proof. If $f(y) > \varepsilon$, then $f(y) - \varepsilon > 0$. By Lemma 3, we have

$$\hat{f}_{MG}(y) \geq f(y) - \frac{1}{k+1} = f(y) - \varepsilon > 0,$$

so $y \in Y$. □

3 Count Sketch

Count sketch is a hashing based streaming data structure presented by Charikar et al. [2002].

Before processing the stream, Algorithm 3 initialize counters C with $t \times k$ 0s, randomly draw a set of t hash functions $H_i : [u] \to [k]$, and another set of t has functions $S_i : [u] \to [\pm 1]$. S_i must be drawn from a pairwise independent family.

Algorithm 3: Count-Sketch(X, t, k)

```
1 $C \leftarrow 0^{t\times k}, H \leftarrow (H_i : [u] \to [k])_{i=1}^t, S \leftarrow (S_i : [u] \to [\pm 1])_{i=1}^t$
2 forall $x \in X$ do
3     forall $i$ in $[t]$ do
4         $j \leftarrow H_i(x)$
5         $C_{i,j} \leftarrow C_{i,j} + S_i(x)$
6 return $C, H, S$
```

$$H = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_t \end{bmatrix} \quad S = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_t \end{bmatrix} \quad C = \begin{bmatrix} C_{1,1} & C_{1,2} & \dots & C_{1,k} \\ C_{2,1} & C_{2,2} & \dots & C_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ C_{t,1} & C_{t,2} & \dots & C_{t,k} \end{bmatrix}$$

During a query for $q \in [u]$, we will first calculate t different frequency approximations $\hat{f}_i(q) := \frac{1}{n} S_i(q) C_{i,H_i(q)}$ for all $i \in [t]$. Then we return the median of these as the final frequency approximation of q.

$$\hat{f}_{CS}(q) := \operatorname*{median}_{i\in[t]} \hat{f}_i(q).$$

Since the algorithm is not deterministic, it is randomized. We will analize it in a probabilistic way. Before that, we must be clear about the following question.

Question 1. Where is the randomness come from? Or what are the random variables?

Answer. The randomness comes from the choice of hash functions in H and S. So the random variables are H_is and S_is for all $i \in [t]$. Yes, these are functions, not common variables. If you have dificulty to think functions as variables, you can go one step further, $H_i(q)$ and $S_i(q)$ are the real random variables behind these functions (for all $q \in [u]$). □

So in the following probabilistic analysis, all the probabilities are with respect to the random choice of hash functions in H and S.

To simply the notation, let $C^x_{i,j} := nf(x)S_i(x)\mathbb{1}(H_i(x), j)$ be the random variable for the part of $C_{i,j}$ caused by x, s.t. we can simply write the random variable $C_{i,j}$ as

$$C_{i,j} := \sum_{x\in X} S_i(x)\mathbb{1}(H_i(x), j) = \sum_{x\in[u]} nf(x)S_i(x)\mathbb{1}(H_i(x), j) = \sum_{x\in[u]} C^x_{i,j}$$

where $\mathbb{1}(H_i(x), j)$ is a random variable, which is 1 if $H_i(x) = j$ and 0 otherwise. Then for any $q \in [u]$ and any $i \in [t]$, we can write each $\hat{f}_i(q)$ as

$$\hat{f}_i(q) := \frac{1}{n} S_i(q) C_{i,H_i(q)} = \frac{1}{n} \sum_{x\in[u]} S_i(q) C^x_{i,H_i(q)} = f(q) + \frac{1}{n} \sum_{x\in[u],x\neq q} S_i(q) C^x_{i,H_i(q)} \qquad (1)$$

because

$$\begin{aligned}
S_i(q)C_{i,H_i(q)} &= S_i(q)\sum_{x\in[u]} C^x_{i,H_i(q)} = S_i(q)C^q_{i,H_i(q)} + \sum_{x\in[u],x\neq q} S_i(q)C^x_{i,H_i(q)} \\
&= S_i(q)nf(q)S_i(q)\mathbb{1}(H_i(q),H_i(q)) + \sum_{x\in[u],x\neq q} S_i(q)C^x_{i,H_i(q)} \\
&= nf(q) + \sum_{x\in[u],x\neq q} S_i(q)C^x_{i,H_i(q)}
\end{aligned}$$

Lemma 6. *For any $q \in [u]$ and any $i \in [t]$, if S_i is randomly choosen from a pairwise independent hash function family, and H_i is choosen independently of S_i, then the mean of $\hat{f}_i(q)$ is*

$$\mathrm{E}[\hat{f}_i(q)] = f(q)$$

Proof. We first show that

$$\mathrm{E}\left[S_i(q)C^x_{i,H_i(q)}\right] = 0 \quad \text{For all } x \in [u], x \neq q. \tag{2}$$

$$\begin{aligned}
&\mathrm{E}\left[S_i(q)C^x_{i,H_i(q)}\right] = \mathrm{E}\left[S_i(q)nf(x)S_i(x)\mathbb{1}(H_i(x),H_i(q))\right] \\
&= nf(x)\,\mathrm{E}\left[S_i(q)S_i(x)\mathbb{1}(H_i(x),H_i(q))\right] \\
&= nf(x)\,\mathrm{E}\left[S_i(q)S_i(x)\right]\mathrm{E}\left[\mathbb{1}(H_i(x),H_i(q))\right] && S_i \text{ and } H_i \text{ are independent} \\
&= nf(x)\,\mathrm{E}\left[S_i(q)\right]\mathrm{E}\left[S_i(x)\right]\mathrm{E}\left[\mathbb{1}(H_i(x),H_i(q))\right] && S_i \text{ is pairwise independent} \\
&= 0 && \mathrm{E}\left[S_i(x)\right] = 0
\end{aligned}$$

Then

$$\begin{aligned}
\mathrm{E}[\hat{f}_i(q)] &= \mathrm{E}\left[f(q) + \frac{1}{n}\sum_{x\in[u],x\neq q} S_i(q)C^x_{i,H_i(q)}\right] && \text{by Equation 1} \\
&= f(q) + \frac{1}{n}\sum_{x\in[u],x\neq q} \mathrm{E}\left[S_i(q)C^x_{i,H_i(q)}\right] \\
&= f(q) && \text{by Equation 2}
\end{aligned}$$

□

Lemma 6 shows each $\hat{f}_i(q)$ is an unbiased approximations of $f(q)$.

Lemma 7. *For any $q \in [u]$ and any $i \in [t]$, if S and H are choosen independently, then the variance of $\hat{f}_i(q)$ is*

$$\mathrm{V}[\hat{f}_i(q)] \leq \frac{1}{k}F_2^2$$

where $F_2^2 = \sum_{x\in[u]} f(x)^2$.

Proof. We first show that, for all $x, y \in [u], x, y \neq q, x \neq y$,

$$\begin{aligned}
&\operatorname{cov}\left[S_i(q)C^x_{i,H_i(q)}, S_i(q)C^y_{i,H_i(q)}\right] \\
=\ & \mathrm{E}\left[\left(S_i(q)C^x_{i,H_i(q)} - \mathrm{E}[S_i(q)C^x_{i,H_i(q)}]\right)\left(S_i(q)C^y_{i,H_i(q)} - \mathrm{E}[S_i(q)C^y_{i,H_i(q)}]\right)\right] \\
=\ & \mathrm{E}\left[\left(S_i(q)C^x_{i,H_i(q)}\right)\left(S_i(q)C^y_{i,H_i(q)}\right)\right] && \text{by Equation 2} \\
=\ & \mathrm{E}\left[\left(nf(x)S_i(x)\mathbb{1}(H_i(x), H_i(q))\right)\left(nf(y)S_i(y)\mathbb{1}(H_i(y), H_i(q))\right)\right] \\
=\ & n^2 f(x)f(y)\,\mathrm{E}\left[S_i(x)S_i(y)\right]\mathrm{E}\left[\mathbb{1}(H_i(x), H_i(q))\mathbb{1}(H_i(y), H_i(q))\right] && S_i \text{ and } H_i \text{ are indep.} \\
=\ & n^2 f(x)f(y)\,\mathrm{E}\left[S_i(x)\right]\mathrm{E}\left[S_i(y)\right]\mathrm{E}\left[\mathbb{1}(H_i(x), H_i(q))\mathbb{1}(H_i(y), H_i(q))\right] && S_i \text{ is pairwise indep.} \\
=\ & 0
\end{aligned}$$

Thus

$$\begin{aligned}
\mathrm{V}[\hat{f}_i(q)] &= \mathrm{V}\left[f(q) + \frac{1}{n}\sum_{x\in[u],x\neq q} S_i(q)C^x_{i,H_i(q)}\right] \\
&= \frac{1}{n^2}\mathrm{V}\left[\sum_{x\in[u],x\neq q} S_i(q)C^x_{i,H_i(q)}\right] \\
&= \frac{1}{n^2}\sum_{x\in[u],x\neq q}\mathrm{V}\left[S_i(q)C^x_{i,H_i(q)}\right] \\
&\quad + \frac{1}{n^2}\sum_{x\in[u],x\neq q}\ \sum_{y\in[u],y\neq q,y\neq x}\operatorname{cov}\left[S_i(q)C^x_{i,H_i(q)}, S_i(q)C^y_{i,H_i(q)}\right] \\
&= \frac{1}{n^2}\sum_{x\in[u],x\neq q}\mathrm{V}\left[S_i(q)C^x_{i,H_i(q)}\right] \\
&\leq \frac{1}{n^2}\sum_{x\in[u],x\neq q}\mathrm{E}\left[(S_i(q)C^x_{i,H_i(q)})^2\right] \\
&= \frac{1}{n^2}\sum_{x\in[u],x\neq q}\mathrm{E}\left[\left(nf(x)S_i(x)\mathbb{1}(H_i(x), H_i(q))\right)^2\right] \\
&= \sum_{x\in[u],x\neq q} f^2(x)\,\mathrm{E}\left[\left(\mathbb{1}(H_i(x), H_i(q))\right)^2\right] \\
&= \sum_{x\in[u],x\neq q} f^2(x)\frac{1}{k} \\
&= \frac{1}{k}(F_2^2 - f^2(q)) \leq \frac{1}{k}F_2^2
\end{aligned}$$

□

Lemma 8. *For any $q \in [u], i \in [t]$, if $k = \frac{F_2^2}{\delta\varepsilon^2}$, then $|\hat{f}_i(q) - f(q)| \leq \varepsilon$ with probability at least $1 - \delta$.*

Proof.

$$\begin{aligned}
\Pr\left[\left|\hat{f}_i(q) - f(q)\right| \geq \varepsilon\right] &= \Pr\left[\left|\hat{f}_i(q) - \mathrm{E}[\hat{f}_i(q)]\right| \geq \varepsilon\right] \\
&\leq \frac{\mathrm{V}[\hat{f}_i(q)]}{\varepsilon^2} \leq \frac{F_2^2}{k\varepsilon^2} && \text{by the Chebyshev's inequality}
\end{aligned}$$

Setting $\frac{F_2^2}{k\varepsilon^2} = \delta$, or $k = \frac{F_2^2}{\delta\varepsilon^2}$, we have

$$\Pr\left[\left|\hat{f}_i(q) - f(q)\right| \geq \varepsilon\right] \leq \delta$$

□

Confidence boosting trick: if you have a randomized algorithm with constant failure probability, then you can achieve an arbitrarily small failure probability δ by repeating $O(\log \frac{1}{\delta})$ versions of the algorithm in parallel, and choosing the best result.

Lemma 9 (Hoeffding's Inequality for Binomial Tail Bound)**.** *For the binomial distribution $B(n, p)$, its CDF is bounded by*

$$F(k; n, p) \leq \exp\left(-2n\left(p - \frac{k}{n}\right)^2\right)$$

Theorem 10. *If $k = \frac{F_2^2}{4\varepsilon^2}$ and $t = 8\log\frac{1}{\delta}$, then the output of Algorithm 3, $\hat{f}_{CS}$ is an $\hat{f}_\varepsilon$ with probability at least $1 - \delta$.*

Proof. Using Lemma 8, setting $k = \frac{F_2^2}{4\varepsilon^2}$, give us

$$\Pr\left[\left|\hat{f}_i(q) - f(q)\right| \geq \varepsilon\right] \leq \frac{1}{4}$$

If the median of $\hat{f}_i(q)$ has error more than ε, then there must be at least $t/2$ of $\hat{f}_i(q)$ with error more than ε. Let D_t be the random variable equal to the number of failed $\hat{f}_i(q)$ for $i \in [t]$, i.e. $|\hat{f}_i(q) - f(q)| \geq \varepsilon$.

$$\Pr\left[\left|\hat{f}_{CS}(q) - f(q)\right| \geq \varepsilon\right] = \Pr\left[\left|\text{median}(\hat{f}_i(q)) - f(q)\right| \geq eps\right] \leq \Pr[D_t \geq t/2]$$

Since S_i and H_i are choosen independently, so $\hat{f}_i(q)$ are also independent. Thus D_t is a binomial random variable with t trials and probability of success $p \leq \frac{1}{4}$. By the Hoeffding's inequality, we have

$$\Pr[D_t \geq t/2] \leq \exp\left(-2t\left(\frac{1}{2} - p\right)^2\right) \leq \exp\left(-\frac{t}{8}\right)$$

Setting $\exp(-t/8) = \delta$, or $t = 8\log\frac{1}{\delta}$, we have

$$\Pr\left[\left|\hat{f}_{CS}(q) - f(q)\right| \geq \varepsilon\right] \leq \Pr[D_t \geq t/2] \leq \exp(-t/8) = \delta$$

□

Question 2. Why median not mean?

4 Count-Min Sketch

By Cormode and Muthukrishnan [2005], a hashing based sketching, an extension to Bloom Filter.
Recall Algorithm 4, Bloom Filter.
Where $H := \{H_1, \ldots, H_t\}$ is a set of hash functions randomly choosen independent of the data.
For a query q, first calculate all $\hat{f}_i(q) := \frac{1}{n}C_{i,H_i(q)}$, then return the minimum of them

$$\hat{f}_{CMS}(q) = \frac{1}{n}\min_{i\in[t]}\hat{f}_i(q)$$

Algorithm 4: Bloom-Filter(X, H)

```
1 $C \leftarrow 0^{t\times k}, H \leftarrow (H_i : [u] \to [k])_{i=1}^t$
2 **forall** $x \in X$ **do**
3     **forall** $i$ *in* $[t]$ **do**
4         $C_{i,H_i(x)} \leftarrow 1$
5 **return** $C, H$
```

Algorithm 5: Count-Min(X, t, k)

```
1 $C \leftarrow 0^{t\times k}, H \leftarrow (H_i : [u] \to [k])_{i=1}^t$
2 **forall** $x \in X$ **do**
3     **forall** $i$ *in* $[t]$ **do**
4         $C_{i,H_i(x)} \leftarrow C_{i,H_i(x)} + 1$
5 **return** $C, H$
```

Lemma 11. *For any $q \in [u], i \in [t]$,*

$$f(q) \le \hat{f}_i(q).$$

If H_i is drawn from a pairwise independent hash family, then

$$\mathrm{E}[\hat{f}_i(q) - f(q)] \le \frac{1}{k}.$$

If $k = \frac{1}{\delta\varepsilon}$, then

$$\Pr[\hat{f}_i(q) - f(q) \ge \varepsilon] \le \delta$$

Proof. The lower bound is obvious. For the upper bound, we first define random functions $\mathbb{1}(a, b)$ be 1 if $a = b$, and 0 otherwise. Then we have

$$\hat{f}_i(q) = \frac{1}{n} C_{i,H_i(q)} = \frac{1}{n} \sum_{x\in[u]} n f(x) \mathbb{1}(H_i(x), H_i(q)) = f(q) + \sum_{x\in[u], x\neq q} f(x) \mathbb{1}(H_i(x), H_i(q))$$

And then

$$\begin{aligned}
\mathrm{E}[\hat{f}_i(q)] =& f(q) + \sum_{x\in[u], x\neq q} f(x)\, \mathrm{E}[\mathbb{1}(H_i(x), H_i(q))] \\
=& f(q) + \sum_{x\in[u], x\neq q} f(x) \frac{1}{k} \le f(q) + \frac{1}{k} && H_i \text{ is pairwise indep.}
\end{aligned}$$

Use the Markov's inequality, we get

$$\Pr[\hat{f}_i(q) - f(q) \ge \varepsilon] \le \mathrm{E}[\hat{f}_i(q) - f(q)]/\varepsilon \le \frac{1}{k\varepsilon}$$

Set $\frac{1}{k\varepsilon} = \delta$, or $k = \frac{1}{\delta\varepsilon}$, we get

$$\Pr[\hat{f}_i(q) - f(q) \ge \varepsilon] \le \delta$$

□

We can again use the confidence boosting technique to get a better space bound. By using more hash fuanctions and counters, we can reduce the dependence of the space bound on the inverse failure probability from linear to logarithm.

Theorem 12. *Consider Algorithm 5, if* $t = \lg \frac{1}{\delta}, k = \frac{2}{\varepsilon}$, *then for any* $q \in [u]$, $\hat{f}_{CMS}(q) := \min_{i \in [t]} \hat{f}_i(q)$ *is a* $\hat{f}_\varepsilon(q)$ *with probability at least* $1 - \delta$.

Proof. By setting $k = \frac{2}{\varepsilon}$ in Lemma 11, we have the failure probability of one hash function

$$\Pr[\hat{f}_i(q) - f(q) \geq \varepsilon] \leq \frac{1}{2}$$

Since t hash functions are choosen independently, the failure probability of all hash functions is

$$\Pr[\min_{i \in [t]} \hat{f}_i(q) - f(q) \geq \varepsilon] = \Pr[\forall i \in [t](\hat{f}_i(q) - f(q) \geq \varepsilon)] = \prod_{i \in [t]} \Pr[\hat{f}_i(q) - f(q) \geq \varepsilon] \leq 1/2^t$$

Setting the failure probability $1/2^t = \delta$, or $t = \lg \frac{1}{\delta}$, we finish the proof. □

5 Summary

Sketch	Space	Technique	Deterministic
Misra-Gries	$O(1/\varepsilon)$	Counter	Yes
Count Sketch	$O\left(\frac{F_2^2}{4\varepsilon^2} \log \frac{1}{\delta}\right)$	Hashing	No
Count-Min Sketch	$O\left(\frac{1}{\varepsilon} \log \frac{1}{\delta}\right)$	Hashing	No

Table 1: Studied Sketches to obtain $\hat{f}_\varepsilon$ (with probability at least $1 - \delta$ if aplicable).

Compare the studied sketches Note: It seems like the Count-Min sketch is better than the Count sketch in the error-space tradeoff, but the bound is based on F_2^2 which is usually much smaller than 1. The Count sketch is also more versatile than Count-Min sketch and works very well in practice.

Definition 1 (biased approximation, unbiased approximation, under-estimated approximation, over-estimated approximation)**.** For a ground truth value f, a random variable $\hat{f}$ (the output of any estimation method) is

- unbiased approximation if $\mathrm{E}[\hat{f}] = f$;
- biased approximation if $\mathrm{E}[\hat{f}] \neq f$;
- under-estimated approximation if $\mathrm{E}[\hat{f}] < f$;
- over-estimated approximation if $\mathrm{E}[\hat{f}] > f$;

Question 3. Which approximations are unbiased?

Question 4. Does unbiased approximation always better than biased approximation?

Answer. It depends. Most of the time you will see "unbiased" as a requirement of a problem or a character of a solution. If it is required, you are ristricted in the unbiased solution space. A biased solution in this case is unvalid, not just not good enough. To compare which one is better, you need some numerical measurments, in terms of error and failure rate. □

Question 5. Can Count/Count-Min sketch solve heavy hitters? What is the query time? Invertable hasing? Min heap?

Answer. We can use a min heap. Whenever we see a item x, we update the sketch, estimate the frequency and check if the estimate is greater enough. If yes, we update the heap to include s. The computational cost to update the min heap is $O(\log k)$, therefore the total update cost is $O(n \log k)$ in the worst case (when we always update the heap). □

Question 6. What is the failure probability of Count/Count-Min sketch actually is? For one q or for all $q \in [u]$?

Question 7. What about weighted data? Real value weights? Negative weights?

5.1 Vector streaming data

All three sketches above have extensions for vector data.

6 Conclusion

References

Robert S. Boyer and J. Strother Moore. MJRTY - A Fast Majority Vote Algorithm. Technical Rept. ADA131702, TEXAS UNIV AT AUSTIN INST FOR COMPUTING SCIENCE AND COMPUTER APPLICATIONS, February 1981.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding Frequent Items in Data Streams. In Peter Widmayer, Stephan Eidenbenz, Francisco Triguero, Rafael Morales, Ricardo Conejo, and Matthew Hennessy, editors, *Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 693–703, Berlin, Heidelberg, 2002. Springer. ISBN 978-3-540-45465-6. doi: 10.1007/3-540-45465-9_59.

Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, April 2005. ISSN 0196-6774. doi: 10.1016/j.jalgor.2003.12.001.

J. Misra and David Gries. Finding repeated elements. *Science of Computer Programming*, 2(2): 143–152, November 1982. ISSN 0167-6423. doi: 10.1016/0167-6423(82)90012-0.

J Strother Moore. A fast majority vote algorithm. Technical report, Institute for Computing Science and Computer Applications, The University of Texas at Austin, 1981.

Jeff M. Phillips. *Mathematical Foundations for Data Analysis.* Springer Series in the Data Sciences. Springer, Cham, Switzerland, 2021. ISBN 978-3-030-62341-8 978-3-030-62340-1. doi: 10.1007/978-3-030-62341-8.