CMPUT 675: Algorithms for Streaming and Big Data **Fall 2019**

Lecture 7 (Sep 25, 2019): Count Sketches

Lecturer: Mohammad R. Salavatipour *Scribe: Aditya Jayaprakash*

In the previous lecture, we discussed the Misra-Greis deterministic counting algorithm for finding the k most frequent elements in a stream. However, it was not a sketching algorithm. Today we see Sketching algorithms that work in the turnstile model where deletion is allowed in addition to insertion. In this lecture, we discuss two sketching algorithm to estimate f_i, the number of occurrences of $a_i \in \sigma$. In the last section, we will discuss various application of these algorithms.

7.1 CountSketch

In this section, we will study the CountSketch algorithm for estimating the frequency of an element in the stream. This was the work of Charikar, Chen and Farach-Colton[CCFC04] from 2004 . The algorithm is as follows,

CountSketch (Basic)

1. $k \leftarrow \frac{3}{\epsilon^2}$
2. $c \in \mathbb{Z}^k$ and $c[1, \ldots, k] = 0$.
3. h is a a random hash function $h : \{1, \ldots, n\} \to \{1, \ldots, k\}$ from a 2-universal hash function family.
4. g is a random hash function $g : \{1, \ldots, n\} \to \{-1, +1\}$ from a 2-universal hash function family.
5. While there is at least one token left:
6. Each token $a_j = (i_j, \Delta_j)$ where i_j is the element and $\Delta_j \in \{-1, +1\}$ denotes insert or delete.
7. $c[h(i_j)] \leftarrow c[h(i_j)] + \Delta_j g(i_j)$
8. for each $i \in [n]$ set $\hat{f}_i = g(i)c[h(i)]$.

7.1.1 Analysis

Consider a fixed $a \in \{1, \ldots, n\}$ and let X be a random variables where $X = \hat{f}_a$. Let $Y_1, \ldots, Y_n$ be independent Bernoulli random variable where

$$Y_i = \begin{cases} 1, & \text{if } h(i) = h(a) \\ 0, & \text{otherwise} \end{cases}$$

We will use the fact that any token with value i and $Y_i = 1$ changes the counter $c[h(a)]$ since both $h(i)$ and $h(a)$ hash to the same value.

$$\begin{aligned} X &= g(a) \sum_{i=1}^{n} f_i g(i) Y_i \\ &= f_a \underbrace{g(a)^2}_{1} + \sum_{i:i \neq a} f_i g(a) g(i) Y_i \\ &= f_a + \sum_{i:i \neq a} f_i g(a) g(i) Y_i \end{aligned}$$

We must also note that since hash functions g and h are independent, $\mathrm{E}[g(i)Y_i] = \underbrace{\mathrm{E}[g(i)]}_{0} \mathrm{E}[Y_i] = 0$

$$\begin{aligned} \mathrm{E}[X] &= \mathrm{E}\left[f_a + \sum_{i:i \neq a} f_i g(a) g(i) Y_i \right] \\ &= \mathrm{E}\left[f_a\right] + \mathrm{E}\left[\sum_{i:i \neq a} f_i g(a) g(i) Y_i \right] \\ &= f_a + \sum_{i:i \neq a} \mathrm{E}\left[f_i g(a) g(i) Y_i\right] \\ &= f_a + \sum_{i:i \neq a} \mathrm{E}[f_i]\mathrm{E}[g(a)] \underbrace{\mathrm{E}[g(i)]\mathrm{E}[Y_i]}_{0} \\ &= f_a \end{aligned}$$

For any Bernoulli random variable Y_i, $\mathrm{E}[Y_i^2] = \mathrm{E}[Y_i]$. Combining it with the properties of 2-universal hash functions, we get that for each $i \in \{1, \ldots, n\} \setminus \{a\}$ and a 2-universal hash function h,

$$\mathrm{E}[Y_i^2] = \mathrm{E}[Y_i] = \Pr[h(i) = h(a)] = \frac{1}{k}$$

Suppose g and h are independent 2-universal hash functions, then

$$\mathrm{E}[g(i)g(j)Y_iY_j] = \underbrace{\mathrm{E}[g(i)]}_{0} \mathrm{E}[g(j)]\mathrm{E}[Y_iY_j] = 0$$

We will bound the probability of failure using Chebyshev's inequality, but we would need to compute the variance first.

$$
\begin{aligned}
\mathrm{Var}[X] &= \mathrm{Var}\left[f_a + \sum_{i:i\neq a} f_i g(a) g(i) Y_i\right] \\
&= 0 + \underbrace{g(a)^2}_{1} \mathrm{Var}\left[\sum_{i:i\neq a} f_i g(i) Y_i\right] \\
&= \mathrm{E}\left[\left(\sum_{i:i\neq a} f_i g(i) Y_i\right)^2\right] - \mathrm{E}\left[\sum_{i\neq a} f_i g(i) Y_i\right]^2 \\
&= \mathrm{E}\left[\sum_{i\neq a} f_i^2 Y_i^2 + \sum_{i,j\in[n]\setminus\{a\}} f_i f_j g(i) g(j) Y_i Y_j\right] - \mathrm{E}\left[\sum_{i\neq a} f_i g(i) Y_i\right]^2 \\
&= \mathrm{E}\left[\sum_{i\neq a} f_i^2 Y_i^2\right] + \underbrace{\mathrm{E}\left[\sum_{i,j\in[n]\setminus\{a\}} f_i f_j g(i) g(j) Y_i Y_j\right]}_{0 \text{ since } \mathrm{E}[g(i)g(j)Y_iY_j]=0} - \underbrace{\left(\sum_{i\neq a} f_i \mathrm{E}\left[g(i) Y_i\right]\right)^2}_{0 \text{ since } E[g(i)=0]} \\
&= \sum_{i\neq a} \mathrm{E}\left[f_i^2 Y_i^2\right] = \sum_{i\neq a} f_i^2 \mathrm{E}\left[Y_i^2\right] = \sum_{i\neq a} \frac{f_i^2}{k} \\
&= \frac{||f||_2^2 - f_a^2}{k} \leq \frac{||f||_2^2}{k}
\end{aligned}
$$

We will now use Chebyshev's inequality to bound the probability,

Theorem 1 (Chebyshev's inequality) *Let X be a random variable and $t > 0$. Then* $\Pr[|X - \mathrm{E}[X]| > t \leq \frac{\mathrm{Var}[X]}{t^2}$. *Alternatively* $\Pr[|X - \mathrm{E}[X]| > t\sigma_X] \leq \frac{1}{t^2}$.

$$
\begin{aligned}
\Pr[|\hat{f}_a - f_a| \geq \epsilon ||f||_2] &= \Pr[|X - \mathrm{E}[X]| \geq \epsilon ||f||_2] \\
&\leq \frac{\mathrm{Var}[X]}{\epsilon^2 ||f||^2} \leq \frac{||f||_2^2}{\epsilon^2 k ||f||^2} \\
&= \frac{1}{k\epsilon^2} = \frac{1}{3}
\end{aligned}
$$

Using the median of means trick, we can run t independent copies of it while setting $k = \frac{3}{\epsilon^2}$ and $t = O\left(\log \frac{1}{\delta}\right)$ and running the more improved algorithm,

CountSketch

1. $k \leftarrow \frac{3}{\epsilon^2}$
2. $t = O\left(\log \frac{1}{\delta}\right)$
3. $c \in \mathbb{Z}^t \times \mathbb{Z}^k$ and $c[1,\ldots,t][1,\ldots,k] = 0$.
4. Choose t 2-universal hash functions $h_1,\ldots,h_t$ such that h_i is random hash function $h_i : \{1,\ldots,n\} \to \{1,\ldots,k\}$ from a 2-universal hash function family.
5. Choose t 2-universal hash functions $g_1,\ldots,g_t$ such that g_i is a random hash function $g_i : \{1,\ldots,n\} \to \{-1,+1\}$ from a 2-universal hash function family.
6. While there is at least one token left:
7. Each token $a_j = (i_j, \Delta_j)$ where i_j is the element and $\Delta_j \in \{-1,+1\}$ denotes insert or delete.
8. for $i \leftarrow 1$ to t do,
9. $c\,[i, h_i(i_j)] \leftarrow c\,[i, h_i(i_j)] + \Delta_j g_i(i_j)$
10. For each $a \in [n]$, let $\hat{f}_a = \underset{1\le i\le t}{\text{median}}\; g_i(a) \cdot c[i, h_i(a)]$.

Just like the previous analysis, suppose we fix l and $X_l = g_l(a) \cdot c[l, h_l(a)]$ is a random variable, then

$$\mathrm{E}[X_l] = f_a$$

Using the same analysis as before and Chebyshev's inequality,

$$\Pr[|X_l - f_a| \ge \epsilon||f||_2] \le \frac{1}{3}$$

Using the median of means trick, we can show

$$\Pr[|\text{median }\{X_1,\ldots,X_t\} - f_a| > \epsilon||f||_2] \le e^{-O\left(\log \frac{1}{\delta}\right)} \le \delta$$

The space required to store the hash functions is $O(t \log n)$ and each counter might need to store a value up to m and there are tk counters, taking up $O(tk \log m)$ space. Hence, the total space complexity is $O(kt(\log m + \log n)) = O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta} (\log m + \log n)\right)$. We will now look at an alternate algorithm called CountMinSketch which solves the same problem but gives different guarantees.

7.2 CountMinSketch

This algorithm was introduced by Cormode and Muthukrishnan [CM05] in 2005. The ideas of the algorithm are very close to the algorithm we saw before. We maintain an array of size $t \times k$ consisting of counters which we update using hash functions. We can visualize it as a matrix where for each row, i, there is a 2-universal hash function $h_i : \{1,\ldots,n\} \to \{1,\ldots,k\}$ which, like the previous algorithm, we will use to map elements to counters. The algorithm works as follows,

CountMinSketch

1. Choose t 2-universal hash functions $h_1, \ldots, h_t$ such that h_i is random hash function $h_i : \{1, \ldots, n\} \to \{1, \ldots, k\}$ from a 2-universal hash function family.
2. $c \in \mathbb{Z}^t \times \mathbb{Z}^k$ and $c[1, \ldots, t][1, \ldots, k] = 0$.
3. While there is at least one token left:
4. for $i \leftarrow 1$ to t do,
5. $c\left[i, h_i(i_j)\right] \leftarrow c\left[i, h_i(i_j)\right] + \Delta_j$
6. For each $a \in [n]$, let $\hat{f}_a = \min_{1 \le i \le t} c[i, h_i(a)]$.

Suppose for the purpose of this analysis, we assume that $\Delta_j \ge 0$ for all $1 \le j \le n$, then

$$\sum_{l: h_i(i_l) = j} \Delta_l = c[i, j]$$

In any case, $\hat{f}_a$ will be an over estimation of f_a (for $\Delta_j \ge 1$). Let X_i be the random variable denoting the excess of $c[i, h_i(a)]$ compared to f_a. We can write $X_i = c[i, h_i(a)] - f_a$. Just like before, we will also have a random variable Y_{ij} for $j \in \{1, \ldots, n\} \setminus \{a\}$ where

$$Y_{ij} = \begin{cases} 1, & \text{if } h_i(j) = h_i(a) \\ 0, & \text{otherwise} \end{cases}$$

According to our definition, if $Y_{ij} = 1$, then f_j is added to the i^{th} counter for f_a.

$$X_i = \sum_{j \in \{1, \ldots, n\} \setminus \{a\}} f_i Y_{ij}$$

Since Y_{ij} is a Bernoulli random variable and by the properties of 2-universality, we have

$$\begin{aligned}
\Pr[Y_i = 1] &= \frac{1}{k} = \mathrm{E}[Y_{ij}] \\
\mathrm{E}[X_i] &= \mathrm{E}\left[\sum_{j \in \{1, \ldots, n\} \setminus \{a\}} f_j Y_{ij}\right] \\
&= \sum_{j \in \{1, \ldots, n\} \setminus \{a\}} f_j \mathrm{E}[Y_{ij}] \\
&= \sum_{j \in \{1, \ldots, n\} \setminus \{a\}} \frac{f_j}{k} \\
&= \frac{||f||_1 - f_a}{k} \text{ which we will denote by } ||f_{-a}||
\end{aligned}$$

We will now use Markov's inequality to bound the probability of failure.

Theorem 2 (Markov's inequality) *Let X be a non-negative random variable. Then for all $a > 0$:* $\Pr[X \ge a] \le \frac{\mathrm{E}[X]}{a}$. *Alternatively* $\Pr[X \ge a\mathrm{E}[X]] \le \frac{1}{a}$.

$$\Pr[X_i \geq \epsilon||f_{-a}||_1] \leq \frac{||f_{-a}||_1}{k\epsilon||f_{-a}||_1} = \frac{1}{2}$$

By our choice of k and using the fact that we have t independent counters, we will show $\hat{f}_a - f_a$ is the minimum over all with high probability.

$$\begin{aligned}
\Pr[\hat{f}_a - f_a \geq \epsilon||f_{-a}||_1] &= \Pr[\ \min \{X_1, \ldots, X_t\} \geq \epsilon||f_{-a}||_1] \\
&= \Pr\left[\bigwedge_{i=1}^{t} (X_i \geq \epsilon||f_{-a}||_1)\right] \\
&= \prod_{i=1}^{t} \Pr[X_i \geq \epsilon||f_{-a}||_1] \\
&\leq 2^{-t} = \delta
\end{aligned}$$

To make our error as small as delta, we can find a choice of $t = O\left(\log \frac{1}{\delta}\right)$. We have shown that that we can guarantee the value of f_a with additive error with high probability,

$$f_a \leq \hat{f}_a \leq f_a + \epsilon||f_{-a}||_1 \leq f_a + \epsilon||f||_1$$

While both CountSketch and CountMinSketch have the same approach and same goal of estimating the frequency f_a of some element a, they both provide different guarantees. For instance, suppose we consider the problem of estimating frequency moments, CountSketch outputs an estimate $\hat{f}_a$ of f_a with an additive error of $\epsilon||f||_2$ while CountMinSketch provides an error of $\epsilon||f||_1$ and $||f||_2 \leq ||f||_1$, meaning the additive error of CountMinSketch is much worse in comparison. One could always pick an algorithm depending on what metric (L_2 or L_1) they wish to optimize for. However, CountMinSketch has a one-sided error guarantee when $\Delta \geq 0$ which could be useful depending on the application. CountMinSketch is also more preferable when it comes to space complexity since one would only need $O\left(\frac{1}{\epsilon} \log \frac{1}{\delta}\right)$ counters. We can summarize what we described in the following table,

Algorithm	Error Bound	Space Complexity
CountSketch	$\lvert\hat{f}_a - f_a\rvert \leq \epsilon\lvert\lvert f_{-a}\rvert\rvert_2$	$O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta} (\log m + \log n)\right)$
CountMinSketch	$\lvert\hat{f}_a - f_a\rvert \leq \epsilon\lvert\lvert f\rvert\rvert_1$	$O\left(\frac{1}{\epsilon} \log \frac{1}{\delta} (\log m + \log n)\right)$

7.3 Applications

In this section, we will discuss applications of the algorithms we previously discussed.

1. Point Queries: Given an item i, the point query $Q(i)$ would return an estimate of f_i.This query would require returning the value of a counter in the sketch.
2. Range Queries : Given $Q(l, r)$, we would want to estimate the number of elements of each type from l to r, so we would like to return $\sum_{l \leq i \leq r} f_i$.
3. Inner Product : Given $Q(\vec{f}, \vec{g})$, we would to approximate $\langle \vec{f}, \vec{g} \rangle = \sum_{i=1}^{n} f_i g_i$.
4. Heavy Hitter: We call an index i an α-HH (for heavy hitter) where $\alpha \in (0, 1]$ if $f_i \geq \alpha||f||_1$. We wish to identify elements that are α-HH.

We can use CountSketch and CountMinSketch to solve all the problems mentioned above. Next lecture, we will look at how to solve range queries using dyadic intervals.

References

CCFC04 M. Charikar, K.C. Chen, and M. Farach-Colton, Finding frequent items in data streams. *Theoretical Computer Science*, 312:03–15, 2004.

CM05 G. Cormode and S. Muthukrishnan, An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.