(CS602) Applied Algorithms 23 Jan, 2014

Lecture 6: Count-Min and Count sketches

Lecturer: Nutan Limaye *Scribe: Nutan Limaye*

In the last class we defined the notion of frequency moments and gave (ε, δ) approximation algorithm for approximating the second frequency moment using space $O(\frac{1}{\varepsilon^2} \cdot \log\left(\frac{1}{\delta}\right) \cdot \log m)$.

Today we will present two different randomized approximation algorithm which can compute all frequency moments using $O(\frac{1}{\varepsilon} \cdot \log\left(\frac{1}{\delta}\right) \cdot \log m)$ space, albeit their approximation guarantee will be weaker: i.e. approximation error will be *absolute* instead of *relative* (as in the previous classes).

6.1 Algorithm 1: Count-Min sketch algorithm

Recall some notation from the last time. Let $x_1, x_2, \ldots, x_n$ be the input stream and for each $i \in [n]$ let $x_i \in [m]$. Let f_j denote the number of times the element $j \in [m]$ appears in the stream. After the algorithm processes the entire stream, the task is to answer queries of the following form: *how many times has element "a" appeared in the stream?* or *what is the most frequently occuring element in the stream?* etc.

We now give the first algorithm.

Pick $h_1, h_2, \ldots, h_t$ uniformly randomly from pairwise independent family of functions $\mathcal{F} = \{h : [m] \to [k]\}$;
for $i = 1$ *to* t **do**
  for $j = 1$ *to* k **do**
    $C[i][h_i(j)] \leftarrow 0$;
  end
end
while *there exists* x*, an input element* **do**
  for $i = 1$ *to* t **do**
    $C[i][h_i(x)] \leftarrow C[i][h_i(x)] + 1$;
  end
end
On query a, output $\hat{f}_a \leftarrow min_i\{C[i][h_i(a)]\}$;

This algorithm is called the Count-Min sketch algorithm for obvious reasons and has been designed by Cormode and Muthukrishnan in 2003 [2].

In the algorithm, k, t are parameters which will be fixed later. In terms of these parameters, the space used by the algorithm can analysed easily. The number of bits needed to pick pairwise independent hash functions is $O(t \log m \log k)$ and the number of bits needed to store the table C is $O(tk \log n)$. Therefore, the total space requirement is $O(kt(\log m + \log n))$. This indicates that to minimize the space usage, we need to keep both parameters as low as possible.

Suppose $k \ll m$, then a random function from $[m]$ to $[k]$, maps around m/k elements of the domain to the same element in the range. Suppose the stream has a lot of occurences of one element, say a_1, while very few of the other, say a_2, and say under a random function they get mapped to the same bucket (which can happen with probability $1/k$), then the algorithm's estimate for the number of a_2 will be very erroneous. To counter this effect, the algorithm chooses t (pairwise) random functions. Now the probability that a_1 and a_2 collide under all t functions is small.

One thing we can quickly observe is that f_a, which is the number of times the element 'a' appears in the stream, is cerianly upper bounded by $\hat{f}_a$. This is because if under some h_i, no other element gets mapped to $h_i(a)$ then the count $C[i][h_i(a)]$ will be equal to f_a and all other $C[j][h_j(a)] \geq C[i][h_i(a)]$.

We will now prove the following lemma:

Lemma 6.1.1. *For every constant $\varepsilon, \delta > 0$, $\Pr\left[\left(\hat{f}_a - f_a\right) \geq \varepsilon n\right] \leq \delta$.*

Remark 6.1.2. *Ideally, we would have liked to prove that for every constant $\varepsilon, \delta > 0$, $\Pr\left[\left(\hat{f}_a - f_a\right) \geq \varepsilon f_a\right] \leq \delta$. That is, we would have liked to have a multiplicative approximation error, but we get an additive error. In fact, it is known that any randomized approximation algorithm (with multiplicative approximation error) for $\max_i \{f_i\}$ (which trivially is a restriction of the function we are computing here) requires space $\Omega(n)$ [1].*

Proof of Lemma 6.1.1: For $j \in [n] \setminus \{a\}$ let $Y_{i,j}$ denote the excess in the counter $C[i][h_i(a)]$. Then we have, $Y_{i,j} = \begin{cases} f_j & \text{if } h_i(a) = h_i(j) \quad \text{(with probability } 1/k\text{)} \\ 0 & \text{otherwise} \qquad\quad \text{(with probability } 1 - 1/k\text{)} \end{cases}$

Let $Y_i := \sum_j Y_{i,j}$. Then we have that $\hat{f}_a = min_i \{Y_i\} + f_a$. Therefore, $\hat{f}_a - f_a = min_i \{Y_i\}$. Therefore, we need to prove that for every constant $\varepsilon, \delta > 0$, $\Pr[min_i \{Y_i\} \geq \varepsilon n] \leq \delta$. That is, we need to prove that for every constant $\varepsilon, \delta > 0$, $\Pr[\forall i : Y_i \geq \varepsilon n] \leq \delta$. Now, note that all Y_is are independent, therefore, it suffices to prove that for every constant $\varepsilon, \delta > 0$, $\Pr[Y_i \geq \varepsilon n] \leq \delta^{1/t}$. We will analyze the expected value of Y_i in order to prove this. From the definition of $Y_{i,j}$ and linearity of expectation, we get that $\mathbb{E}(Y_i) = \sum_{j \in [m] \setminus \{a\}} \mathbb{E}(Y_{i,j}) = \frac{\sum_{j \in [m] \setminus \{a\}} f_j}{k} = \frac{n - f_a}{k}$. Therefore, using Markov's inequality we get that $\Pr[Y_i \geq \varepsilon n] \leq \frac{n - f_a}{k \varepsilon n}$. If k is chosen such that $k = \lceil \frac{2}{\varepsilon} \rceil$ then we get that $\Pr[Y_i \geq \varepsilon n] < \frac{1}{2}$. And if t is chosen to be $\lceil 2 \log \frac{1}{\delta} \rceil$, we get the lemma.

□

6.2 Count Sketch

In this section, we present a small modification of the algorithm presented in the previous section. We will analyze the algorithm in this lecture and compare and contrast the performance of the two algorithms in the next lecture.

Like in the previous section, let for $j \in [n] \setminus \{a\}$ let $Y_{i,j}$ denote the excess in the counter $C[i][h_i(a)]$.

Then we have, $Y_{i,j} = \begin{cases} f_j & \text{if } h_i(a) = h_i(j) \text{ and } g_i(j) = +1 \quad \text{(with probability } 1/2k\text{)} \\ -f_j & \text{if } h_i(a) = h_i(j) \text{ and } g_i(j) = -1 \quad \text{(with probability } 1/2k\text{)} \\ 0 & \text{otherwise} \qquad\qquad\qquad\quad \text{(with probability } 1 - 1/k\text{)} \end{cases}$

Pick $h_1, h_2, \ldots, h_t$ uniformly randomly from pairwise independent family of functions $\mathcal{F} = \{f : [m] \to [k]\}$ and pick $g_1, g_2, \ldots, g_t$ uniformly randomly from pairwise independent family of functions $\mathcal{G} = \{g : [m] \to \{\pm\}\}$;
for $i = 1$ *to* t **do**
  for $j = 1$ *to* k **do**
    $C[i][h_i(j)] \leftarrow 0$;
  end
end
while *there exists* x*, an input element* **do**
  for $i = 1$ *to* t **do**
    $C[i][h_i(x)] \leftarrow C[i][h_i(x)] + g_i(x)$;
  end
end
On query a, output $\text{Median}_{1 \le i \le t} \{g_i(a) C[i][h_i(a)]\}$;

Let $Y_i := \sum_j Y_{i,j}$. The output of the algorithm and the Y_is are related in the following way: Y_i counts the total error terms in $C[i][h_i(a)]$. Therefore, $C[i][h_i(a)] = Y_i + g_i(a) \cdot f_a$. In order to bound the error in the output, let us first compute the expected value of Y_i and variance of Y_i.

Lemma 6.2.1. *For every* $1 \le i \le t$ $\mathbb{E}(Y_i) = 0$ *and* $\mathbb{V}ar(Y_i) = \frac{\sum_{j \in [m] \setminus \{a\}} f_j^2}{k}$.

Proof.

$$\begin{aligned}
\mathbb{E}(Y_i) &= \mathbb{E}\left(\sum_{j \in [m] \setminus \{a\}} Y_{i,j}\right) && \text{(By the definition of } Y_i\text{)} \\
&= \sum_{j \in [m] \setminus \{a\}} \mathbb{E}(Y_{i,j}) && \text{(By linearity of expectation)} \\
&= \sum_{j \in [m] \setminus \{a\}} \frac{f_j}{2k} + \frac{-f_j}{2k} && \text{(By the definition of } Y_{i,j}\text{)} \\
&= 0
\end{aligned}$$

$$
\begin{aligned}
\mathbb{E}(Y_i^2) &= \mathbb{E}\left(\sum_{j\in[m]\setminus\{a\}} Y_{i,j}^2 + \sum_{j\neq j'\in[m]\setminus\{a\}} Y_{i,j}Y_{i,j'}\right) && \text{(By the definition of } Y_i\text{)}\\
&= \sum_{j\in[m]\setminus\{a\}} \mathbb{E}(Y_{i,j}^2) + \sum_{j\neq j'\in[m]\setminus\{a\}} \mathbb{E}(Y_{i,j}Y_{i,j'}) && \text{(By linearity of expectation)}\\
&= \sum_{j\in[m]\setminus\{a\}} \frac{f_j^2}{2k} + \frac{(-f_j)^2}{2k} + \sum_{j\neq j'\in[m]\setminus\{a\}} \mathbb{E}(Y_{i,j})\mathbb{E}(Y_{i,j'}) && \text{(By the pairwise independence of } Y_{i,j}\text{)}\\
&= \frac{\sum_{j\in[m]\setminus\{a\}} f_j^2}{k}
\end{aligned}
$$

□

In the next class we will bound the expectation and variance of the output of the algorithm using Lemma 6.2.1.

References

[1] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 20–29, New York, NY, USA, 1996. ACM.

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

6.3 Exercises

Exercise 1. *Make approrpiate modifications to the algorithms presented here so that they work in the turnstile model. Work out all the calculations for the modified algorithms. Read about the turnstile model at:*
http://en.wikipedia.org/wiki/Streaming_algorithm#Models