

Lecture 2: Frequent Items in Streams

Lecturer: Edo Liberty

Warning: This note may contain typos and other inaccuracies which are usually discussed during class. Please do not cite this note as a reliable source. If you find mistakes, please inform me.

1 Approximated histograms

In this section we will describe a simple modification of the algorithm described in [1]. Say we are given a stream of elements $X = [x_1, \dots, x_N]$ where $x_i \in \{a_1, \dots, a_n\}$. Let n_i denote the number of times element a_i appeared in the stream, i.e., $n_i = |\{j | x_j = a_i\}|$. Our goal is to estimate n_i for all frequent elements. This can be solved exactly by keeping a counter for each element $\{a_1, \dots, a_n\}$. Alas, this might require, $\Theta(n)$ memory. Another approach is to sample a large enough fraction of the stream and compute count the frequencies in the sample (see homework question). Here we suggest a deterministic algorithm.

Algorithm 1 Frequent items counter

```

input:  $\varepsilon, \theta \in (0, 1], X = [x_1, \dots, x_N]$ 
 $C \leftarrow \{\}$ 
for  $x \in X$  do
  if  $x \in C$  then
     $C[x]++$ 
  else if  $\text{size}(C) < 1/\varepsilon\theta$  then
     $C[x] = 1$ 
  else
    for  $a \in C$  do
       $C[a]--$ 
      if  $C[a] == 0$  then
         $\text{del}(C[a])$ 
      end if
    end for
  end if
end for
for  $a \in C$  do
  if  $C[a] \leq N\theta(1 - \varepsilon)$  then
     $\text{del}(C[a])$ 
  end if
end for

```

Claim 1.1. For elements a_i for which $n_i \leq N\theta(1 - \varepsilon)$ we have $n_i \notin C$.

This is easy to see since we add 1 to the counter of $C[a]$ every time we encounter a . So, clearly $C[a_i] \leq n_i \leq N\theta(1 - \varepsilon)$. Therefore, in the last loop of the algorithm it will be deleted.

Claim 1.2. For elements a_i for which $n_i \geq N\theta$ we have $n_i \geq C[a_i] \geq n_i(1 - \varepsilon)$.

This is slightly less obvious. Notice that every time we decrease the counters in the map C we have that $\text{size}(C) \geq 1/\varepsilon\theta$. That means that we decrement at least $1/\varepsilon\theta$ different counters simultaneously. If we let t denote the the number of times this step is performed, we have $t/\varepsilon\theta \leq N$ because we could not have deleted more items than the entire stream. Using the observation that $C[a_i] \geq n_i - t$ we have $C[a_i] \geq n_i - N\varepsilon\theta \geq n_i(1 - \varepsilon)$.

Remarks: note that this algorithm uses $O(1)$ memory (assuming ε and θ are constants).

Count Sketches

Here we learn about a structure names CountSketch which was suggested in [2]. It will allow us to estimate the frequency of the k most frequent items in a stream even if it is less than a constant fraction of the stream. There will, however, be other limitations.

We denote the elements by o_1, \dots, o_m having each appeared $n_1 \geq \dots \geq n_m$ (the names of the elements are ordered according to their frequency). Before describing the CountSketch structure, let us first analyze one of its building blocks. For lack of a more creative name, we will call it B . B is an array of length b which is associated with two hash functions: $h : o \rightarrow [1, \dots, b]$ and $s : o \rightarrow [-1, 1]$.

We define two function for B one for adding elements into it.

1. define $Add(o)$:
2. $B[h(o)] = B[h(o)] + s(o)$.

and one for returning an estimate for n_i given o_i

1. define $Query(o)$:
2. return $B[h(o)]s(o)$.

In order to compute the expectation of $B[h(o)]s(o)$ we need to define the “inverse” of h . Let $h^{-1}(o_i) = \{o_j | h(o_j) = h(o_i)\}$. In words, $h^{-1}(o_i)$ is the set of all elements for $h(o_i) = h(o_j)$. Since each element in $o_j \in h^{-1}(o_i)$ is encountered exactly n_j times and for each of those $s(o_j)$ is added to $B[h(o)]$ we have that $B[h(o_i)] = \sum_{o_j \in h^{-1}(o_i)} n_j s(o_j)$. Let us compute the expected result of a query.

$$\begin{aligned} \mathbb{E}[B[h(o_i)]s(o_i)] &= \mathbb{E}\left[\sum_{o_j \in h^{-1}(o_i)} n_j s(o_j) s(o_i)\right] \\ &= n_i + \mathbb{E}\left[\sum_{o_j \in h^{-1}(o_i), o_i \neq o_j} n_j s(o_j) s(o_i)\right] = n_i \end{aligned}$$

As a reminder, we are interested in the frequencies n_1, \dots, n_k , for the top k most items. We see that if $b > 8k$ we have that $|h^{-1}(o_i) \cap \{o_1, \dots, o_k\}| = 0$ with probability at least $7/8$. In other words, the element o_i does not map under h to the same cell in B with any of the top k frequency items. We will define $h_{>k}^{-1} = h^{-1}(o_i) \cap \{o_{k+1}, \dots, o_m\}$. We will assume from this point on that $h^{-1}(o_i) \subset \{o_{k+1}, \dots, o_m\}$ or in other words that $h_{>k}^{-1} = h^{-1}(o_i)$.

Now, let us bound the variance of $B[h(o_i)]s(o_i)$.

$$\begin{aligned}
\text{Var}(B[h(o_i)]s(o_i)) &\leq E[B[h(o_i)]^2 s(o_i)^2] \\
&= E[(\sum_{o_j \in h_{>k}^{-1}(o_i)} n_j s(o_j))(\sum_{o_{j'} \in h_{>k}^{-1}(o_i)} n_{j'} s(o_{j'}))] \\
&= E_h \sum_{o_j \in h_{>k}^{-1}(o_i)} \sum_{o_{j'} \in h_{>k}^{-1}(o_i)} E_s[n_j n_{j'} s(o_j) s(o_{j'})] \\
&= E_h \sum_{o_j \in h_{>k}^{-1}(o_i)} n_j^2 \\
&= \sum_{j=k+1}^m n_j^2 / b
\end{aligned}$$

Note that we have both an expectation over the choice of the hash function s and over the hash function h .

Using this bound on the variance of $B[h(o_i)]s(o_i)$ and Chebyshev's inequality we attain that:

$$\Pr \left[|B[h(o_i)]s(o_i) - n_i| > \sqrt{8 \sum_{j=k+1}^m n_j^2 / b} \right] \leq 1/8$$

However, note that we also demanded that none of the top k elements map to the same cell as o_i which only happened with probability $7/8$. Using the union bound on these two events we get:

$$\Pr[|\hat{n}_i - n_i| \leq \gamma] \geq 3/4$$

where we denote $\hat{n}_i = B[h(o_i)]s(o_i)$ and $\gamma = \sqrt{8 \sum_{j=k+1}^m n_j^2 / b}$.

Note that this happens for every elements individually only with constant probability. We would like to get that this holds with probability $1 - \delta$ for all elements simultaneously. We do that by repeating this entire structure t times creating the CountSketch B_1, \dots, B_t . When inserting an element we insert it into all t arrays B_i and above. When querying the CountSketch we return $query(o_i) = \text{Median}(\hat{n}_i^1, \dots, \hat{n}_i^t)$ where \hat{n}_i^ℓ is the estimator \hat{n}_i from B_ℓ .

Because $\Pr[|\hat{n}_i - n_i| \leq \gamma] \geq 3/4$ we get from Chernoff's inequality that at least half the values \hat{n}_i^ℓ will be such that $|\hat{n}_i^\ell - n_i| \leq \gamma$ (including the median) for all m elements with probability at least $1 - \delta$ for $t \in O(\log(m/\delta))$.

The only thing left to do is set the correct value for b (the length of B). We will demand that $\gamma \leq \epsilon n_k$. This gives $b \geq 8 \sum_{i=k+1}^m n_i^2 / \epsilon^2 n_k^2$. Therefore, for $t = O(\log(m/\delta))$ and $b \geq 8 \max(k, \frac{\sum_{i=k+1}^m n_i^2}{\epsilon^2 n_k^2})$ with probability at least $1 - \delta$ for each element in the stream $|\hat{n}_i - n_i| \leq \epsilon n_k$.

The algorithm for finding the most frequent items is therefore to go over the stream and keep a CountSketch of all the elements seen this far. When we process an element, we also estimate it's frequency \hat{n} and keep the top k most frequent items in estimated frequencies. These are guaranteed to contain all elements o_i for which $n_i > (1 + 2\epsilon)n_k$ and not to contain any element o_i for which $n_i < (1 - 2\epsilon)n_k$.

References

- [1] Richard M. Karp, Christos H. Papadimitriou, and Scott Shenker. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems*, 28:2003, 2003.
- [2] Moses Charikar, Kevin Chen, and Martin Farach-colton. Finding frequent items in data streams. pages 693–703, 2002.