

## Lecture 3: Item frequency estimation 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 Please do not cite this note as a reliable source.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.

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.,  $f_i = |\{j | x_j = a_i\}|$ . Our goal is to estimate  $f_i$  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. Here we look for methods to approximate the values on  $f_i$  using  $o(n)$  memory.

## 1 Sampling

The first and simplest approach is to use the uniform sampling approach above. That is, the algorithm draws samples uniformly at random from the stream with probability  $\ell/N$ . Using Chernoff along with the union bound indicates that  $\ell \in O(\log(n/\delta)/\varepsilon^2)$  is sufficient. Applying the union bound more carefully reduces the failure probability and therefore reduces  $\ell$ , the expected number of samples.

## 2 Count Min-Sketches

Note that the space dependence of random sampling on  $\varepsilon$  is inversely quadratic which might be problematic for small values of  $\varepsilon$ . Count-Min sketches were introduced in [1][2] in two similar variants. They reduce the space complexity dependence on  $\varepsilon$  to only  $1/\varepsilon$ . The creation of the sketch is given in Algorithm ???. The notation is that  $h_1, \dots, h_t$  are hash functions from the space of elements to the integers  $[2/\varepsilon]$ .

---

**Algorithm 1** Count Min Sketch: Add
 

---

**Input:**  $\varepsilon, A$   
 $t \leftarrow \lceil \log(n/\delta) \rceil, b \leftarrow \lceil 2/\varepsilon \rceil$   
 $C \leftarrow$  all zeros matrix of size  $t \times b$   
**for**  $i \in [N]$  **do**  
  **for**  $j \in [t]$  **do**  
     $C[j, h_j(A_i)] = C[j, h_j(A_i)] + 1$   
  **end for**  
**end for**  
**Return:**  $C$

---



---

**Algorithm 2** Count Min Sketch: Query
 

---

**Input:**  $C, a$   
**Return:**  $\min_{j=1, \dots, t} C[j, h_j(a)]$

---

To see why this works consider only one row of the sketch matrix. The value of  $C[1, a]$  contains the frequency of  $a$  but also the sum of frequencies of all other items  $b$  for which  $h_1(b) = h_1(a)$ . Since the event that  $h_1(b) = h_1(a)$  happens with probability  $\varepsilon/2$  we have  $\mathbb{E}[C[1, a] - f_a] \leq N\varepsilon/2$  by linearity of

expectation. By Markov's inequality we have that  $\Pr[C[1, a] - f_a \geq \varepsilon N] \leq 1/2$ . Therefore, any row in  $C$  provides a good approximate count for  $a$  with probability at least  $1/2$ . Since we return the minimal value of  $\log[n/\delta]$  such estimates our failure probability reduces to  $\delta/n$ . Using the union bound we get that all items receive a good approximation with probability at least  $1 - \delta$ . Note that we get the same guaranties as in the sampling solution but the space requirement reduced from  $O(\log(n/\delta)\varepsilon^2)$  to  $O(\log(n/\delta)/\varepsilon)$ . Alas, the update time increases from  $O(1)$  to  $O(\log(n/\delta))$ . In the next section we see how this can be improved and even derandomized.

### 3 Frequent Items

The item frequency approximation problem a brilliantly simple and deterministic algorithm in [3]. This algorithm was later rediscovered independently by both [4] and [5] who also improved its update time complexity. Their algorithm reduces the space requirement from  $O(\log(n/\delta)/\varepsilon)$  to  $O(1/\varepsilon)$ . Their algorithm is given in Algorithm box 3. To prove the algorithm's correctness, let  $n'$  denote the sum of all counters

---

**Algorithm 3** Lossy counting

---

**Input:**  $\varepsilon \in (0, 1]$ ,  $A$   
 $\ell \leftarrow \lceil 1/\varepsilon \rceil$   
 $C \leftarrow$  empty map from  $a$  to the integers with returned default value 0  
**for**  $i \in [N]$  **do**  
     $C[A_i] = C[A_i] + 1$   
    **if**  $\text{size}(C) = \ell$  **then**  
        **for**  $a \in C$  **do**  
             $C[a] = C[a] - 1$   
            **if**  $C[a] = 0$  **then**  
                 $\text{del}(C[a])$   
            **end if**  
        **end for**  
    **end if**  
**end for**  
**Return:**  $C$

---

in the returned map  $C$ . Let  $\delta_i = 1$  if the inner loop of the algorithm is executed in the  $i$ 'th iteration and zero else. Note that in each iteration the sum of counters is increased by 1 and reduced by  $\ell\delta_i$ . Therefore  $N' = \sum_{i=1}^N 1 - \ell\delta_i = N - \ell \sum_{i=1}^N \delta_i$ . This gives that  $\sum_{i=1}^N \delta_i \leq (N - N')/\ell \leq \varepsilon(N - N')$ . Since  $N' \geq 0$  and any single item counter is decreased at most  $\sum_{i=1}^N \delta_i$  times we get that  $f_a \geq C[a] \geq f_a - \varepsilon N$ .

This reduces the amount of memory from  $O(\log(n/\delta)/\varepsilon)$  required by Count-Min sketches to  $O(1/\varepsilon)$ . Moreover, some modifications to the data structure in the algorithm [5] allow updates to require only  $O(1)$  operations. This significantly improves on the  $O(\log(n/\delta))$  operations required by Count-Min sketches. As a last remark, note that this algorithm is deterministic which eliminates the failure probability altogether.

### 4 Count Sketches

In many cases where frequent items are sought the guaranty that  $|f_i - g_i| \leq \varepsilon N = \varepsilon \sum_{j=1}^n f_j$  is insufficient. For example, if the item distribution is very skewed, a few most frequent items can correspond to most of the appearances in the stream. Thus, a more desirable guaranty would be of the form  $|f_i - g_i| \leq \varepsilon N = \varepsilon \sum_{j=k+1}^n f_j$  for some prespecified  $k$ . Here we assume without loss of generality that the items are indexed in decreasing frequency order.

One idea from [6] suggests that this is possible by using  $O(k \log(n/\delta))$  approximate counters. First, we create  $3k$  different approximate counters and distribute elements between them using a hash function.

So, only with probability  $1/3$  element  $a$  falls into the same sketch as one of the top  $k$  element. Therefore, with probability  $2/3$ , the sketch containing  $a$  will give a frequency approximation guaranty proportional to  $\varepsilon \sum_{j=k+1}^n f_j$ . Since this only happens with probability  $2/3$  we must repeat the construction  $O(\log(n/\delta))$  times and return the median of the results returned by the counters.

The second idea is to alter the approximate counters themselves by incorporating a random sign into the summation. That is, when element  $a$  is encountered, its counter is incremented by  $s(a)$  where  $s$  is a hash function mapping items from the universe uniformly into  $\{-1, 1\}$ . This reduces the approximation error to be relative to  $O(\sqrt{\sum_{j=k+1}^n f_j^2})$ .

---

**Algorithm 4** Count Sketch: Add

---

**Input:**  $\varepsilon, A$   
 $C \leftarrow$  all zeros matrix of size  $t \times b$   
**for**  $i \in [N]$  **do**  
    **for**  $j \in [t]$  **do**  
         $C[j, h_j(A_i)] = C[j, h_j(A_i)] + s_j(A_i)$   
    **end for**  
**end for**  
**Return:**  $C$

---



---

**Algorithm 5** Count Sketch: Query

---

**Input:**  $C, a$   
**Return:**  $\text{median}_{j=1, \dots, t} C[j, h_j(a)] s(a)$

---

## References

- [1] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.
- [2] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *VLDB*, pages 346–357, 2002.
- [3] Jayadev Misra and David Gries. Finding repeated elements. Technical report, Ithaca, NY, USA, 1982.
- [4] Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Frequency estimation of internet packet streams with limited space. In *Proceedings of the 10th Annual European Symposium on Algorithms, ESA '02*, pages 348–360, London, UK, UK, 2002. Springer-Verlag.
- [5] 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.
- [6] Moses Charikar, Kevin Chen, and Martin Farach-colton. Finding frequent items in data streams. pages 693–703, 2002.