CSCE 689: Special Topics in Modern Algorithms for Data Science Fall 2023

Lecture 11 — September 15, 2023

Prof. Samson Zhou *Scribe: Matthew Chang*

1 Insertion-Deletion Streams

Definition (Insertion-deletion Stream)**.** An insertion-deletion stream consists updates to a vector, $f \in \mathbb{R}^n$, which tracks incoming changes by either increasing or decreasing a coordinate in f.

When tracking changes, suppose we have a stream length $m = \Theta(n)$, a universe of size $[n]$, and an underlying vector $f \in \mathbb{R}^n$.

For example, we have a vector f.

f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	0	0	0	0

If "Decrease f_6" were to appear in our stream, we would get:

f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	0	0	-1	0

1.1 Applications

To name a few applications of insertion-deletion streams:

- Database Management - track database changes to ensure data integrity and support rollbacks and recovery
- Version Control Systems - track changes to files for collaboration and software development
- Traffic Flow and Transportation Systems - analyze traffic patterns and changes in transportation systems to optimize traffic flow, manage congestion, and improve transportation infrastructure

2 CountMin

In the previous lecture, we used Misa Gries algorithm for (ϵ, k)-frequent items problem. However, Misa Gries only works on insertion-only streams and never overestimates the true frequencies.

The CountMin algorithm can be used on insertion-deletion streams and easily parallelized across multiple servers. A drawback is that is does require randomness when it comes to hashing. It differs from Misa Gries in that it is not deterministic and it never underestimates the true frequency on insertion-only streams.

2.1 Algorithm

The CountMin algorithm is as follows:

1. Initially, we create b buckets of counters and use a random hash function $h : [n] \to [b]$
2. As we receive each update x_i, increment the counter $h(x_i)$
3. At the end of the stream, output the counter $h(x_i)$ as the estimate for x_i
4. Repeat steps 1-3 with different random hash functions $l := O(\log(n))$ times to get estimates $e_1, ..., e_l$ for each $i \in [n]$ and set $\hat{f_i} = min(e_1, ..., e_l)$

2.2 Analysis

2.2.1 Estimation

Suppose $h(i) = a$ so that $c_a = \hat{f_i}$ where c_a counts the number f_j occurences of any j with $h(j) = a = h(i)$, including f_i itself.

Note that for insertion-only streams, the algorithm will never underestimate the true value for any f_i. So we always have $\hat{f_i} \geq f_i$ where $\hat{f_i}$ is the estimated frequency for f_i

2.2.2 Expected Error

The expected value for c_a, the counter for bin a, is the sum of the updates for f_i plus the sum of the updates for all of the other f_j that collide with the same hash function:

$$c_a = f_i + \mathbb{E}[\sum_{j \neq i, j : h(j) = a} f_j]$$

Observe that the expected error is the second term. Thus we have:

$$\mathbb{E}[\sum_{j \neq i, j : h(j) = a} f_j] = \sum_{j \neq i} E[f_j \cdot I_{h(j)=h(i)}],$$

where I is an indicator variable that equals 1 if $h(j) = h(i)$ and 0 otherwise. Then

$$\begin{aligned}
\mathbb{E}[\sum_{j \neq i, j : h(j) = a} f_j] = \sum_{j \neq i} E[I_{h(j)=h(i)}] \cdot f_j \\
&\leq \sum_{j \neq i} Pr[h(j) = h(i)] \cdot |f_j| \\
&= \sum_{j \neq i} \frac{1}{b} \cdot |f_j| \\
&= \frac{\|f\|_1}{b}.
\end{aligned}$$

Note that we always have $\|f\|_1 \leq m$. Thus the expected error of f_i is dependent on the number of bins in our hash function. We can decrease or increase the number of bins b in f to increase or decrease the error in our CountMin estimation, respectively. By setting $b = O\left(\frac{k}{\varepsilon}\right)$, we can solve the (ε, k)-Frequent Items Problem, in which case CountMin would use $O\left(\frac{k}{\varepsilon}\log n\right)$ bits of space.