DS-563/CS-543 Lecture 3 2024-01-25

Notes/Reminders
- webpage: https://onak.pl/ds563
  cs543
- HW 0 / HW 1 (deadline extended to 2/1)
- electronic device policy

Last time:
- Problem: provide estimates of what fraction of items in an evolving collection is any given element
- Started: CountMin Sketch

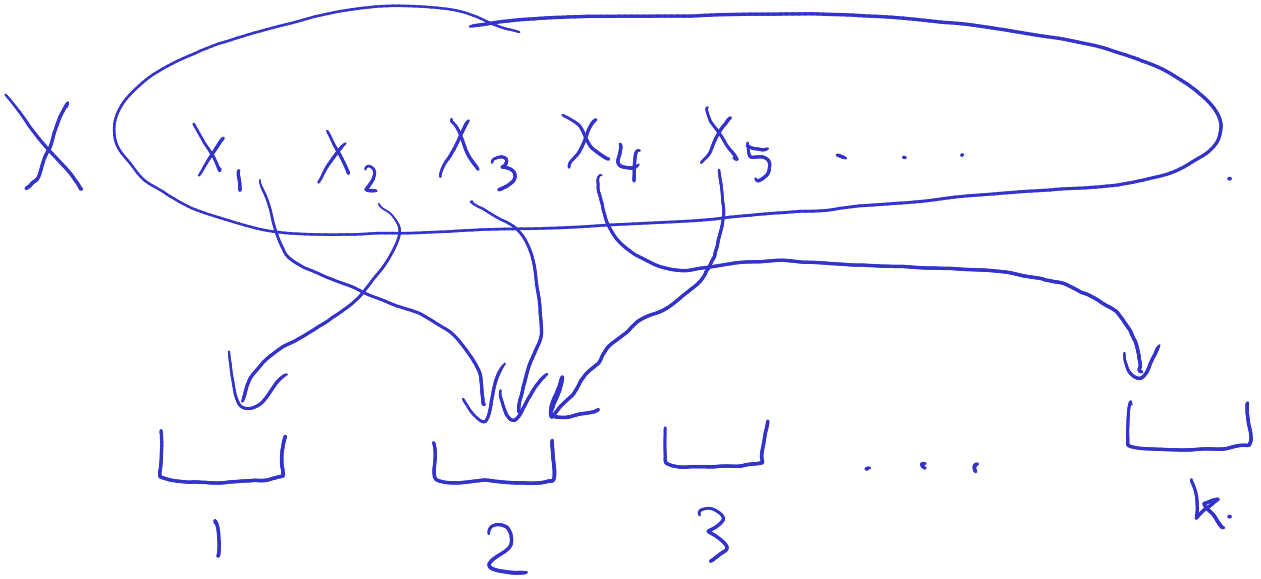
Today:
- Wrap up CountMin Sketch (CMS)
- Important concepts: Linear Sketches and Streaming Algorithms

~~- Deterministic estimates (Misra-Gries)~~
Next time

CMS Last Time

Randomly map elements of universe X to buckets via $h: X \to [k]$

$[k] = \{1, 2, \ldots, k\}$



- keep count $A[i]$ of number of elements in each bucket i
- fraction estimate for $y \in X$:

$$\frac{A[h(y)]}{\sum_{i=1}^{k} A[i]}$$

$\sum_{i=1}^{k} A[i] \overset{\text{def}}{=} S \leftarrow$ total number of items

For a fixed y:

$$\frac{f(y)}{s} \leq \text{estimate} \leq \frac{f(y)}{s} + \frac{2}{k}$$

- $\frac{f(y)}{s} \leq$ estimate: always true
- estimate $\leq \frac{f(y)}{s} + \frac{2}{k}$: w.p. $\geq 1/2$

$f(x)$ = # occurrences of x

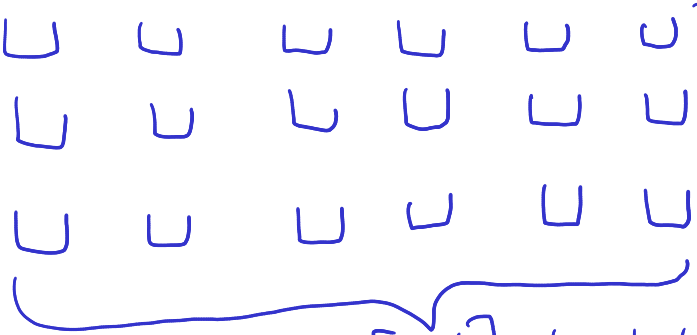
Set $k = \lceil 2/\varepsilon \rceil$ to overestimate by at most ε w.p. $\geq 1/2$

How to make probability of overestimating by more than ε at most $\delta \in (0, 1/2)$?

- Run $t = \lceil \log(1/\delta) \rceil$ independent copies
- Query y: return the <u>minimum</u> of estimates from all copies

$$\Pr\left[\begin{array}{c}\text{all overestimate} \\ \text{by more than } \varepsilon\end{array}\right] \leq \left(\frac{1}{2}\right)^t \leq \delta$$

Visually:



Total space: $O\left(\frac{1}{\varepsilon}\log(1/\delta)\right)$

What is missing?

How do we store fully random hash functions?

- We can't! Lots of space!
- Pairwise independence suffices

  For $x \neq y$: $\mathbb{E}[C_{x,y}] = \Pr[h(x)=h(y)] \leq \frac{1}{k}$

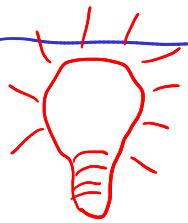
  Or even

  $$\mathbb{E}[C_{x,y}] \leq \frac{O(1)}{k} \leftarrow \text{fixed constant}$$

  (just increase k by a constant factor)

- discussion section:

  universal hashing / sample construction for strings



Linear sketches

(CountMin sketch is an example)

maintained sketch → $\begin{bmatrix} \\ \\ \end{bmatrix} = \begin{bmatrix} \text{randomized} \\ \text{matrix} \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}$ ← input: frequency vector $f(\ldots)$

For CMS, all the kt buckets

↑ what our algorithm does

$q \times 1$ $\quad$ $q \times N$ $\quad$ $N \times 1$ dimensions

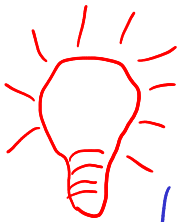
Linear sketching algorithm projects high-dimensional input onto low dimensional sketch

Nice properties of linear sketches:
- Can handle both insertions & deletions (insertion & deletion are the "same" change to the sketch but with the opposite sign)
- Can be computed separately for subsets and easily combined via addition

$$[\quad](f_1+f_2+\ldots+f_n)=[\quad]f_1+\ldots+[\quad]f_n$$

Application: different centers handling different parts of the data set

---



Streaming Algorithms

huge stream of data

Streaming algorithm A $\leftarrow 7, 3, 8, 11, 563, 47, \ldots$

- A reads and processes items one by one
- A should use much less space than the input size

A few comments:

- Items in the stream can be anything

  Common examples: integers, points from some metric space, database records, graph edges, ...

- CMS is a streaming algorithm
- Here the algorithm gets to read the stream once ("one pass"), but sometimes a small number of passes over the stream makes sense in some scenarios, e.g., data living on a storage device in which case sequential reading can maximize throughput