DS 563, Fall 2021, Lecture 1: Count Min Sketch

Setting:

- multiset of items from some universe X
- may arrive in arbitrary order over time

Goal:

Create a data structure D that provides estimates what fraction of items is a specific item $x \in X$

Examples:

- online store: "what fraction of views is this specific product?"
- search engine: "what fraction of queries is this specific query?"

Solution 1:

explicitly store mapping $x \in X \to$ # occurrences of x

Lots of space!

Will use less space by allowing:

- small additive approximation, say, $\pm 0.01\%$
- can give wrong answer w.p. $\delta \in (0,1)$

~~FIS~~ First attempt

Suppose random hash function $h: X \to [k]$ $= \{1, \dots, k\}$

Store array $A[1..k]$ of integers

Initially: $A[i] = 0$ for all $i \in [k]$

Item x arrives: $A[h(x)] \leftarrow A[h(x)] + 1$

Estimate $g(y)$ for $y \in X$: return $\dfrac{A[h(y)]}{\sum_i A[i]}$

$\sum_i A[i] = S$ = total number of items

How good is this?

- can overestimate by a lot!
- never underestimate: $g(y) \geq \frac{f(y)}{S}$

$f(y)$ = real number of occurrences of $y \in X$

Analysis:

$$g(y) = \frac{1}{s}\left(f(y) + \sum_{\substack{x \in X \\ x \neq y}} C_{x,y} \cdot f(x)\right)$$

$$C_{x,y} = \begin{cases} 0 & h(x) \neq h(y) \\ 1 & h(x) = h(y) \end{cases}$$

$\uparrow$ random variable

h fully random $\Rightarrow$ $E[C_{x,y}] = \frac{1}{k}$ for $x \neq y$

$$g(y) = \frac{f(y)}{s} + \underbrace{\frac{\sum_{x \neq y} C_{x,y} f(x)}{s}}_{\geq 0,\ = (*)}$$

$$E[(*)] = \frac{\sum_{x \neq y} f(x)\, E[C_{x,y}]}{s} = \frac{1}{k} \cdot \frac{\sum_{x \neq y} f(x)}{s} \leq \frac{1}{k} \cdot \frac{\overbrace{\sum_{x} f(x)}^{= s}}{s} = \frac{1}{k}$$

Markov's inequality

Non-negative random variable X, $a > 0$

$$\Pr[X \geq a] \leq \frac{E[X]}{a}$$

With probability 1/2

$$(*) \leq \frac{2}{k}$$

Hence with probability 1/2,

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

Set $k = \lceil 2/\varepsilon \rceil$ to get additive ε approximation

---

To make probability of error at most $\delta \in (0, \frac{1}{2})$:

- Run $t = \lceil \log(1/\delta) \rceil$ independent copies in parallel

- On query y: return the minimum of all estimates

$$\Pr[\text{all wrong}] \leq \left(\frac{1}{2}\right)^t \leq 1/\delta$$

i.e, overestimate by more than ε

This is called Count Min Sketch

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

What is missing?

How do we store random hash functions?

We can't but pairwise independence suffices for our proof:

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

In fact,

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

is good enough, because we can slightly increase k = the size of A

(see homework 1 for examples of such hash functions)

Nice properties of Count Min sketch:

- can handle deletions

- can be computed separately for subsets and easily combined (example: different data centers having different parts of the data set)

  Warning: they all need to use the same hash function

This is example of <u>linear sketch</u>

smaller sketch $\rightarrow$ $\big[\;\big] = \big[\text{ randomized matrix }\big]\big[\;\big]$

what the algorithm maintains

what our algorithm does

frequency vector