CS 261: Data Structures

Week 4: Streaming and sketching

Lecture 4b: Deterministic sampling; Heavy hitters

David Eppstein
University of California, Irvine

Spring Quarter, 2023

Deterministic sampling

Epsilon-approximation

Given a sequence S of n numbers (allowing repetitions), define the quantile $q_S(x)$ of a number x to be the fraction of sequence elements that are less than x

(So median = element whose quantile is $1/2$)

ε-approximation: subsequence $\hat{S}$ of S such that $q_{\hat{S}} \approx q_S$:

$$\forall x \quad |q_{\hat{S}}(x) - q_S(x)| \leq \varepsilon$$

It's easy to construct an ε-approximation of size $1/\varepsilon$: just choose $1/\varepsilon$ numbers equally spaced in the sorted sequence

Composition of epsilon-approximations

If we concatenate two equal-length sequences S and T to form $S\,T$, and both have equal-length ε-approximations $\hat{S}$ and $\hat{T}$

$\Rightarrow$ $\hat{S}\,\hat{T}$ is an ε-approximation (but twice as long)

To make it shorter, compute an ε-approximation of $\hat{S}\,\hat{T}$

(but now it's a 2ε-approximation)

Hierarchical sketching

Decompose stream of numbers into blocks
whose sizes are distinct powers of two

For each block, store its size 2^i, and a ε_i-approximation of its values, for a sequence of numbers

$$\varepsilon_0 < \varepsilon_1 < \varepsilon_2 < \cdots < \varepsilon$$

converging to the target ϵ

Updating a hierarchical sketch

To insert a new element:

Create a new block of one element ($i = 0$)
(even if there is already another one-element block)

While two blocks have the same size 2^i:

- Merge them into one block of size 2^{i+1}
- Concatenate their ϵ_i-approximations
- Construct a $(\varepsilon_{i+1} - \varepsilon_i)$-approximation of the concatenation

Using a hierarchical sketch

To determine the quantile of any value x:

- Use the approximations to find $q_B(x)$ for each block
- $q_A(x)$ is weighted average (weighted by block size)

To find an ϵ-approximate median:

- Find element of approximations whose quantile is near $1/2$

  (this is a weighted median problem and can be solved in time linear in the total lengths of the approximations)

How big are these sketches?

For any $\delta > 0$, the sum

$$\sum_{i=0}^{\infty} \frac{1}{(i+1)^{1+\delta}}$$

has a finite sum s_δ

Set

$$\varepsilon_{i+1} - \varepsilon = \frac{\varepsilon}{s_\delta} \cdot \frac{1}{(i+1)^{1+\delta}}$$

Summing resulting sketch sizes for all blocks gives space $O\left(\varepsilon^{-1}(\log n)^{2+\delta}\right)$

Majority and heavy hitters

Impossibility of most-frequent item

We've seen the mean and median; what about the mode (most frequent item) in a sequence?

Impossible in general!

Consider the state of a sketch after $n - 2$ items

Suppose we have already seen one item appear three times, and $(n - 3)/2$ pairs of other items

If the next two items are equal, with value x, then the result should be

- The triple, if x was not already seen
- The triple, if x has the same value
- x, if x has the same value as one of the pairs

So the sketch must know all the pairs, too much memory

Impossibility of majority

Majority element: one that occurs as more than half of the sequence elements

Naive formulation of the problem:

Either find a majority element if one exists

Or return "None" if there is no majority

Still impossible!

If first $n/2$ elements are distinct, any one could become majority

We don't have enough memory to store all of their values

Boyer–Moore majority vote algorithm

Maintain two variables, m and c, initially None and 0

For each x in the given sequence:

- If $m = x$: increment c
- Else if $c = 0$: set $m = x$ and $c = 1$
- Else decrement c

Claims:

- If sequence has a majority, it will be the final value of m
- If no majority, m may be any element of the sequence (might not be frequently occurring)
- Algorithm cannot tell us whether m is majority element or not

Boyer & Moore, "MJRTY - A Fast Majority Vote Algorithm", 1991

Streaming majority example

With the sequence elements in left-to-right order:

x:		B	A	C	A	A	A	C	B	A
m:	None	B	B	C	C	A	A	A	A	A
c:	0	1	0	1	0	1	2	1	0	1
majority?	N	Y	N	N	N	Y	Y	Y	N	Y

(We know when there is a majority but the algorithm doesn't)

After the first step, B is in the majority, and the algorithm correctly reports B as its value of m.

After the third A (out of five sequence elements), A is in the majority, and stays in the majority for most of the remaining steps; the algorithm correctly reports A as its value of m.

When there is no majority (each element has at most half of the sequence) the algorithm may choose any element as its value of m.

Heavy hitters

Generalization of the same algorithm:

Store two arrays M and C, both of length k

Initialize cells of M to empty and C to zero

For each sequence item x:

- If x equals $M[i]$ for some i: increment $C[i]$
- Else if some $C[i]$ is zero: set $M[i] = x$ and $C[i] = 1$
- Else decrement $C[i]$ for all choices of i

Claim: In a sequence of n elements, if x occurs more than $n/(k+1)$ times, then x will be one of the elements stored in M

(Special case $k = 1$ is correctness of majority algorithm)

Why does it work?

This algorithm approximately counts occurrences for all sequence elements (not just the ones it stores)!

Define $\text{count}(x) =$ actual number of occurrences of x,
$\text{estimate}(x) = C[i]$ (if $M[i] = x$ for some i) or 0 otherwise.

Then $\text{count}(x) - \dfrac{n}{k+1} \leq \text{estimate}(x) \leq \text{count}(x)$.

Proof idea: Induction on number of steps of the algorithm

If a step ends with x in M, we increase its count and estimate

Otherwise, we decrease all $k+1$ nonzero estimates

Number of increases $\leq n \Rightarrow$ number of decreases $\leq n/(k+1)$

Because heavy hitters have big estimates, they must be stored