CS 261: Data Structures

Week 4: Streaming and sketching

Lecture 4c: Count-min sketch and invertible Bloom filter

David Eppstein
University of California, Irvine

Spring Quarter, 2023

Count-min sketch

Main idea

We want to estimate frequencies of individual items (like heavy hitter data structure) in the turnstile model, allowing removals

Use hashing to map each item to a multiple cells in a small table
(like Bloom filter, but smaller)

Each cell counts how many elements map to it

- Include another copy of x: increment all cells for x
- Remove a copy of x: decrement all cells for x

Estimate of frequency of x: minimum of counts in cells for x

Cormode & Muthukrishnan, "An Improved Data Stream Summary: The Count-Min Sketch and its Applications", 2005

Details

Set up according to two parameters ε and δ

- ε: how accurate the estimates should be
- δ: how likely estimates are to be accurate

2d table $C[i, j]$, initially all zero

- Each element maps to exactly one cell in each row
- Horizontal dimension (i): $\lceil e/\varepsilon \rceil$ cells
- Vertical dimension (j): $\lceil \ln 1/\delta \rceil$ cells

Hash functions h_j (for $j = 0, \ldots \lceil \ln 1/\delta \rceil - 1$)

- Include x: increment $C[h_j(x), j]$ for all j
- Remove x: decrement $C[h_j(x), j]$ for all j

Estimate of count(x): $\min_j C[h_j(x), j]$

(the smallest count among the cells for x in each row)

Accuracy

Clearly, estimate $\geq$ count; let $N =$ total number of elements

Claim: With probability $\geq 1 - \delta$, estimate $\leq$ count $+ \varepsilon N$

Follows from:

With probability $\geq 1 - \frac{1}{e}$, for any j, $C[h_j(x), j] \leq$ count $+ \varepsilon N$

(rows independent so ok to multiply probabilities $\Rightarrow (1/e)^{\#\text{rows}} \leq \delta$)

Overestimate comes from collisions with other elements

Expected number of collisions $= \epsilon N/e$. Instead of Chernoff, use Markov's inequality: $Pr[X > c\,E[X]] \leq 1/c$

Turnstile medians

Data structure for approximate medians in the turnstile model, of a collection S of integers $0, 1, 2, \ldots n$ (allowing repetitions):

For each $i = 0 \ldots \log_2 n$, round each number down to a multiple of 2^i, using the formula `round(x,i) = x &~ ((1<<i)-1)`

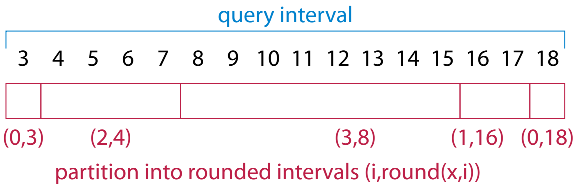
Construct $\log n$ count-min sketches,
one for each sequence of rounded numbers

(but strengthen accuracy and failure probability bounds by a factor of $\log n$ so that when we query all of them and combine the answers, the overall accuracy is what we want)

Finding the approximate median

To estimate the number of elements in the interval $[\ell, r]$:

Partition the interval into $O(\log n)$ rounded intervals (sequences of 2^i elements that all round to the same value in S_i)



Use the rounded sketches to count elements in each rounded interval and sum the results

To find median: binary search for an interval $[0, m]$ containing approximately half of the elements

Set reconciliation

Find the missing element

Toy problem:

I give you a sequence of 999 different numbers, from 1 to 1000

Which number is missing?

Streaming solution: return $500500 - \sum(\text{sequence})$

Straggler sketching

Given a parameter k, maintain a turnstile collection of objects (allowing repetitions), such that

- Total space is $O(k)$
- The set can have arbitrarily many elements
- Whenever it has $\leq k$ distinct elements, we can list them all (with their numbers of repetitions)

E.g. could solve the toy problem by:

- Initialize structure with $k = 1$
- Insert all numbers from 1 to 1000
- Remove all members of sequence
- Ask which element is left

Invertible Bloom filter

Structure:

- Table with $O(k)$ cells
- $O(1)$ hash functions $h_i(x)$ mapping elements to cells (like a Bloom filter)
- "Fingerprint" hash function $f(x)$ (like a cuckoo filter)

Each cell stores three values:

- Number of elements mapped to it (like count-min sketch)
- Sum of elements mapped to it
- Sum of fingerprints of elements mapped to it

Eppstein & Goodrich, "Space-efficient straggler identification in round-trip data streams via Newton's identities and invertible Bloom filters", 2011

How to list the elements

"Pure cell": stores only a single distinct element (but maybe more than one copy of it)

If a pure cell stores element x, we can compute x as (sum of elements)/(number of elements)

Test whether a cell is pure: compute x and check that stored sum of fingerprints equals number of copies times fingerprint(x)

Decoding algorithm:

- While there is a pure cell, decode it, output it, and remove its elements from the collection
- If we reach an empty table (all cells zero), return
- Otherwise, decoding fails (table is too full)

Application: Set reconciliation

Problem: Two internet servers both have slightly different copies of the same collection (e.g. versions of a git repository)

We want to find how they differ, using communication proportional to the size of the difference (rather than the size of the whole set)

If we already know the size of the difference, k:

- Server A builds an invertible Bloom filter for its collection, with parameter k, and sends it to server B
- Server B removes everything in its collection from the filter and then decodes the result
- (Some elements might have negative numbers of copies! But the data structure still works)

Eppstein, Goodrich, Uyeda, and Varghese, "What's the difference? Efficient set reconciliation without prior context", 2011

How to estimate the size of the difference?

Use MinHash to find samples of server B's set with numbers of elements smaller by factors of 1/2, 1/4, 1/8, ...

Sample i: elements whose hash has first i bits = zero

Server B chooses parameter $k = O(1)$ and sends invertible Bloom filters of all the samples in a single message to server A

Server A uses the same hash function to sample its set, and removes sampled subsets from B's filters

Estimate of the size of the difference = 1/sampling rate of largest sample whose difference can be decoded

Summary

Summary

- Sketch: Structure with useful information in sublinear space
- Stream: Loop through data once, using a sketch
- Cash register (addition only) and turnstile (addition and removal) models of sketching and streaming
- Mean and median; impossibility of exact median
- Maintaining a random sample of a stream and using it for approximate medians
- Majority voting, heavy hitters, and frequency estimation
- MinHash-based sampling and estimation of Jaccard distance
- Count-min sketch turnstile-model frequency estimation
- Using count-min sketch for turnstile median estimation
- Stragglers, invertible Bloom filters, and set reconciliation