

CS60021: Scalable Data Mining

Streaming Algorithms

Sourangshu Bhattacharya

Frequent count

Streaming model revisited

- Data is seen as incoming sequence
 - can be just element-ids, or ids +frequency updates
- Arrival only streams
- Arrival + departure
 - Negative updates to frequencies possible
 - Can represent fluctuating quantities, e.g.

Frequency Estimation

- Given the input stream, answer queries about item frequencies at the end
 - Useful in many practical applications e.g. finding most popular pages from website logs, detecting DoS attacks, database optimization



- Also used as subroutine in many problems
 - Entropy estimation, itemset mining etc

Frequency estimation

Q1. Can we create a data structure, sketch, sublinear in the data size to answer all frequency queries accurately?

Frequency estimation in one pass

Q1. Can we create a data structure, sketch, sublinear in the data size to answer all frequency queries accurately?

– No

Q2. Can we create a sketch to estimate frequencies of the “most frequent” elements exactly?

Frequency estimation in one pass

Q1. Can we create a data structure, sketch, sublinear in the data size to answer all frequency queries exactly?

– No

Q2. Can we create a sketch to answer frequencies of the “most frequent” elements exactly?

– No

Q3. Sketch to estimate frequencies of “most frequent” elements approximately?

Frequency estimation in one pass

Q1. Can we create a data structure, sketch, sublinear in the data size to answer all frequency queries exactly?

– No

Q2. Can we create a sketch to answer frequencies of the “most frequent” elements exactly?

– No

Q3. Sketch to estimate frequencies of “most frequent” elements approximately?

– YES!

Approximate Heavy Hitters

- Given an update stream of length m , find out all elements that occur “frequently”
 - e.g. at least 1% of the time
 - cannot be done in sublinear space, one pass
- Find out elements that occur at least ϕm times, and none that appears $< (\phi - \epsilon)m$ times
 - Error ϵ
 - Related question: estimate each frequency with error $\pm \epsilon m$

Starting with a puzzle

[J. Algorithms, 1981] Suppose we have a list of N numbers, representing votes of N processors on result of some computation. We wish to decide if there is a majority vote and what that vote is.

- By J.S. Moore
- Did not talk about streaming solution, but proposed solution is
- Strict majority: $>N/2$

Majority Algorithm

- Arrivals only model
- Start with a counter set to zero
- For each item
 - if counter = 0, pick new item and increment counter
 - else if new item is same as item in hand, increment counter
 - else decrement counter



Majority Algorithm

- Start with a counter set to zero
- For each item
 - if counter = 0, pick new item and increment counter
 - else if new item is same as item in hand, increment counter
 - else decrement counter
- If there is a majority item, it is in hand at the end
- Proof: Since majority occurs $> N/2$ times, not all occurrences can be cancelled out

Frequent [Misra-Gries]

- Keep k counters and items in hand

Initialize:

- Set all counters to 0

Process(x)

- if x is same as any item in hand, increment its counter
- else if number of items $< k$, store x with counter = 1
- else drop x and decrement all counters

Query(q)

- If q is in hand return its counter, else 0

Frequent

- f_x be the true frequency of element x
- At the end, some set of elements is stored with counter values
- If *query* y in hand, $\hat{f}_y =$ counter value, else $\hat{f}_y = 0$

Theoretical Bound

Claim: No element with frequency $> m/k$ is missed at the end

Theoretical Bound

Claim: No element with frequency $> m/k$ is missed at the end

Intuition: Each decrement (including drop) is charged with k arrivals. Therefore, will have some copy of an item with frequency $> m/k$

Stronger Claim

Choose $k = \frac{1}{\epsilon}$. For every item x , with frequency f_x the algo can return an estimate \hat{f}_x such that

$$f_x - \epsilon m \leq \hat{f}_x \leq f_x$$

Stronger Claim

Choose $k = \frac{1}{\epsilon}$. For every item x , with frequency f_x the algo can return an estimate \hat{f}_x such that

$$f_x - \epsilon m \leq \hat{f}_x \leq f_x$$

Same intuition, whenever we drop a copy of item x , we also drop $k - 1$ copies of other items

Summary

- Simple deterministic algorithm to estimate heavy hitters
 - Works only in the arrival model
- Proposed in 1982, rediscovered multiple times with modifications
- Also basis of matrix low rank approximation
- Our next lecture will discuss other algorithms

Space Saving Algorithm

- Keep k counters and items in hand

Initialize:

- Set all counters to 0

Process(x)

- if x is same as any item in hand, increment its counter
- else if number of items $< k$, store x with counter = 1
- else replace item with smallest counter by x , increment counter

Query(q)

- If q is in hand return its counter, else 0

Analysis

Claim 1: All items with true count $> \epsilon m$ are present in hand at the end

Analysis

Claim 1: All items with true count $> \epsilon m$ are present in hand at the end

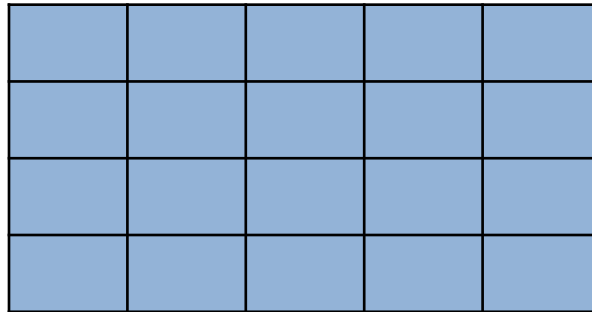
- Smallest counter value, min , is at most ϵm
 - Counters sum to m , by induction
 - $1/\epsilon$ counters, so average is ϵm , hence smallest is less
- True count of an uncounted item is between 0 and min
 - Proof by induction, true initially, min increases monotonically
 - Consider last time the item was dropped

Counter based vs “sketch” based

- Counter based methods
 - Misra-Gries, Space-Saving,
 - Work for arrival only streams
 - In practice somewhat more efficient: space, and especially update time
- Sketch based methods
 - “Sketch” is informally defined as a “compact” data structure that allows both inserts and deletes
 - Use hash functions to compute a linear transform of the input
 - Work naturally for arrivals + departure

Count-min sketch

- Model input stream as a vector over U
 - f_x is the entry for dimension x
- Creates a small summary $w \times d$
- Use w hash functions, each maps $U \rightarrow [1, d]$



A 4x5 grid of light blue squares, representing a summary matrix of size $w \times d$. The grid is composed of 4 rows and 5 columns of squares, totaling 20 cells.

Count Min Sketch

Initialize

- Choose h_1, \dots, h_w , $A[w, d] \leftarrow 0$

Process(x, c):

- For each $i \in [w]$, $A[i, h_i(x)] += c$





Query(q):

- Return $\min_i A[i, h_i(x)]$

Example



h1			
h2			

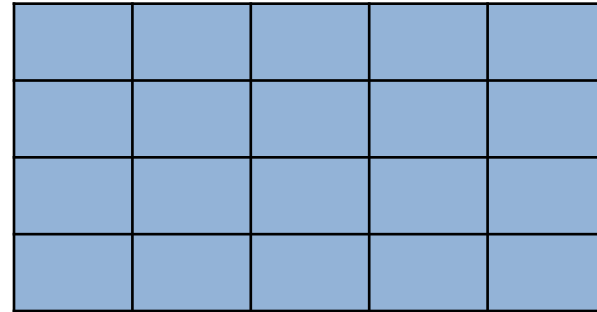
	h1	h2
	2	1
	1	2
	1	3
	3	2

Guarantees

Space = $O(wd)$

Update time = $O(w)$

$x, +c$



Each item is mapped to one bucket per row

Guarantees

- $d = \frac{2}{\epsilon}$ $w = \log\left(\frac{1}{\delta}\right)$

$Y_1 \dots Y_w$ be the w estimates, i.e. $Y_i = A[i, h_i(x)]$, $\hat{f}_x = \min_i Y_i$

Each estimate \hat{f}_x always satisfies $\hat{f}_x \geq f_x$

Guarantees

- $d = \frac{2}{\epsilon}$ $w = \log\left(\frac{1}{\delta}\right)$

$Y_1 \dots Y_w$ be the w estimates, i.e. $Y_i = A[i, h_i(x)]$, $\hat{f}_x = \min_i Y_i$

Each estimate \hat{f}_x always satisfies $\hat{f}_x \geq f_x$

$$E[Y_i] = \sum_{y: h_i(y)=h_i(x)} f_y = f_x + \epsilon(m - f_x)/2$$

Guarantees

- $d = \frac{2}{\epsilon}$ $w = \log\left(\frac{1}{\delta}\right)$

$Y_1 \dots Y_w$ be the w estimates, i.e. $Y_i = A[i, h_i(x)]$, $\hat{f}_x = \min_i Y_i$

Each estimate \hat{f}_x always satisfies $\hat{f}_x \geq f_x$

$$E[Y_i] = \sum_{y: h_i(y)=h_i(x)} f_y = f_x + \epsilon(m - f_x)/2$$

Applying Markov's inequality,

$$\Pr[Y_i - f_x > \epsilon m] \leq \frac{\epsilon(m - f_x)}{2\epsilon m} \leq \frac{1}{2}$$

Guarantee

- Since we are taking minimum of $\log\left(\frac{1}{\delta}\right)$ such random variables,

$$\Pr[\hat{f}_x > f_x + \epsilon m] \leq 2^{-\log\left(\frac{1}{\delta}\right)} \leq \delta$$

Guarantee

- Since we are taking minimum of $\log\left(\frac{1}{\delta}\right)$ such random variables,

$$\Pr[\hat{f}_x > f_x + \epsilon m] \leq 2^{-\log\left(\frac{1}{\delta}\right)} \leq \delta$$

- Hence, with probability $1 - \delta$, for any query x

$$f_x \leq \hat{f}_x \leq f_x + \epsilon m$$

Summary

- Two algorithms for frequency estimation
 - Counter based: Space Saving
 - Sketch based: Count-Min
- Guiding principle: use error bounds as design parameters of the data structure
- More to come...

References:

- Primary references for this lecture
 - Lecture slides by Graham Cormode
<http://dmac.rutgers.edu/Workshops/WGUnifyingTheory/Slides/cormode.pdf>
 - Lecture notes by Amit Chakrabarti: <http://www.cs.dartmouth.edu/~ac/Teach/data-streams-lecnotes.pdf>
 - Sketch techniques for approximate query processing, Graham Cormode.
<http://dimacs.rutgers.edu/~graham/pubs/papers/sk.pdf>