CS60021: Scalable Data Mining

Streaming Algorithms

Sourangshu Bhattacharya

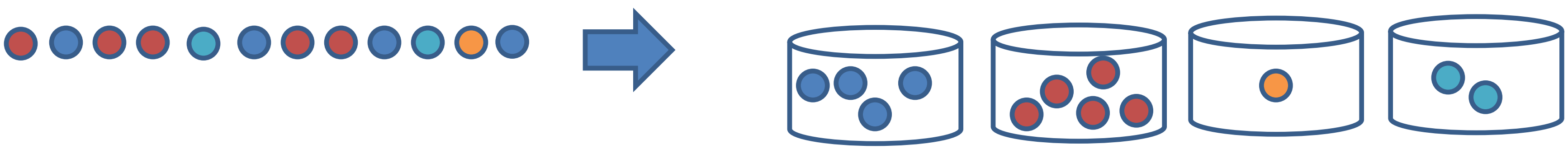
Frequent count

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. monitoring databases.

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, TF-IDF, Language models etc

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 $\pm\epsilon m$

Majority Algorithm

- Whether any item in a stream has majority at a given time:
  - Strict majority: >N/2

- 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 count [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 count

- 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, $\widehat{f_y} =$ counter value, else $\widehat{f_y} = 0$

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 $\widehat{f_x}$ such that

$$f_x - \epsilon m \leq \widehat{f_x} \leq f_x$$

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
- Our next lecture will discuss other algorithms

Space saving

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

- Claim 2: For every element x, the estimate $\hat{f}_x$ satisfies:

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

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

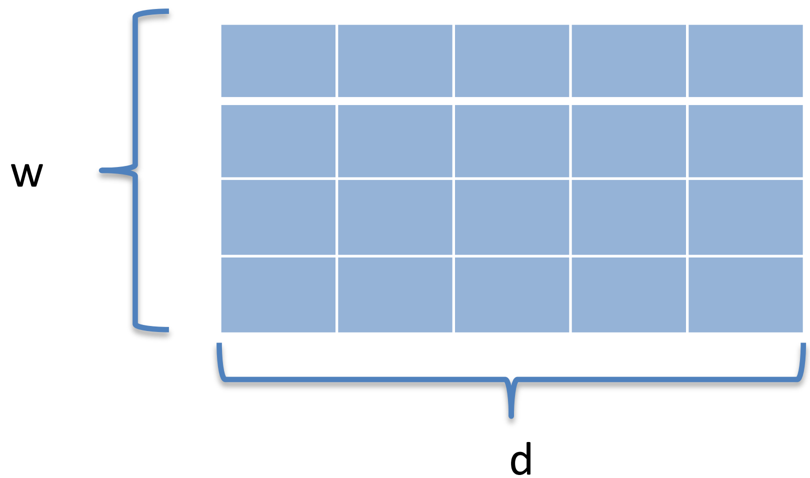
Counter based vs “sketch” based

- 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

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 $U \rightarrow [1, d]$



Count Min Sketch

Initialize

– Choose $h_1, .., h_w,\ A[w, d] \leftarrow 0$

Process(x, c):

– For each $i \in [w],\ A[i, h_i(x)] \mathrel{+}= c$

Query(q):

– Return $\min_i A[i, h_i(x)]$

Example

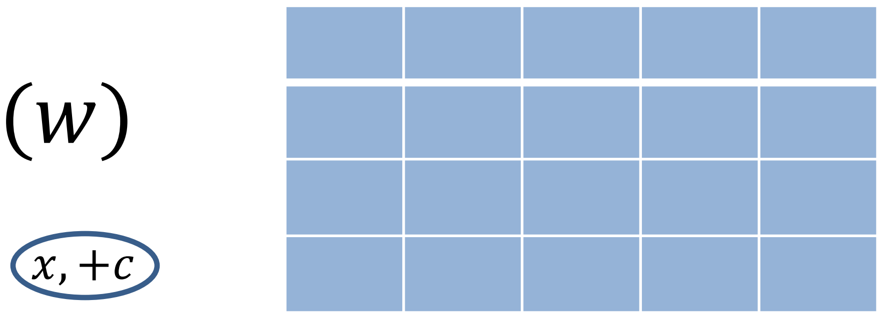
h1			
h2			

	h1	h2
	2	1
	1	2
	1	3
	3	2

Guarantees

Space = $O(wd)$

Update time = $O(w)$



Each item is mapped to one bucket per row

Guarantees

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

$Y_1 \ldots . Y_w$ be the w estimates, i.e. $Y_i = A[i, h_i(x)], \quad \widehat{f_x} = \min_i Y_i$

Each estimate $\widehat{f_x}$ always satisfies $\widehat{f_x} \geq f_x$

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

Guarantees

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

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

Each estimate $\widehat{f_x}$ always satisfies $\widehat{f_x} \geq f_x$

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

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,

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

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

$$f_x \leq \widehat{f_x} \leq f_x + \epsilon m$$

Count-Sketch

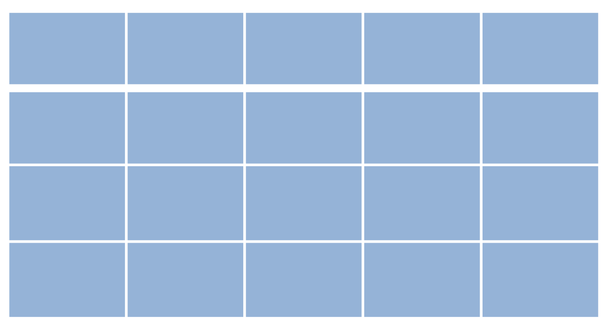
Review: Frequency Estimation

- Given input stream, length m, want a sketch that can answer frequency queries at the end
  - For give item x, return an estimate of the frequency

- Algorithms seen
  - Deterministic counter based algorithms: Misra-Gries, SpaceSaving
  - Count-Min sketch

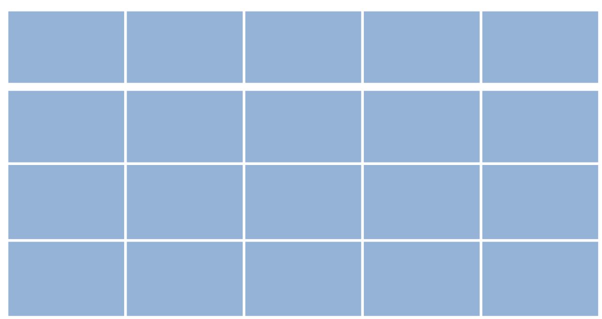
Recall: Count-min sketch

- 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 $U \rightarrow [1, d]$



Count-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, $h_i: U \to [1, d]$
- w sign hash function, each maps $\mathrm{g_i}$: $U \to \{-1, +1\}$



Count Sketch

Initialize

– Choose $h_1, .., h_w$, $A[w, d] \leftarrow 0$

Process(x, c):

– For each $i \in [w]$, $A[i, h_i(x)] \mathrel{+}= c \times g_i(x)$

Query(q):

– Return median$\{g_i(x) A[i, h_i(x)]\}$

Example

h1			
h2			

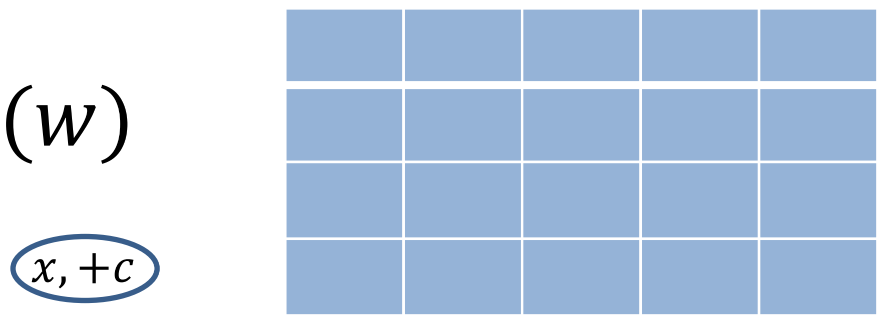
	h1,g1	h2,g2
	2,+	1,+
	3,-	2,+
	1,+	3,-
	2,-	3,+

32

Guarantees

Space = $O(wd)$

Update time = $O(w)$



Each item is mapped to one bucket per row

Guarantees

- d$= \frac{3}{\epsilon^2}$ $\quad \mathrm{w} = \log\left(\frac{1}{\delta}\right)$

$Y_1 \ldots . Y_w$ be the w estimates,
i.e. $Y_i = g_i(x)A[i, h_i(x)], \quad \widehat{f_x} = \operatorname{median}_i Y_i$

$$E[Y_i] = E\left[g_i(x)\, A[i, h_i(x)]\right] = E\left[g_i(x) \sum_{h_i(y)=h_i(x)} f_y\, g_i(y)\right]$$

Guarantees

$$E[Y_i] = E\big[g_i(x)\, A[i, h_i(x)]\big] = E\left[g_i(x) \sum_{h_i(y)=h_i(x)} f_y\, g_i(y)\right]$$

Notice that for $x \neq y$, $E[g_i(x)\ g_i(y)] = 0$!

$$E[Y_i] = g_i(x)^2 f_x = f_x$$

We analyse the variance in order to bound the error

For simplicity assume hash functions all independent

Variance analysis

Using simple algebra, as well as independence of hash functions,

$$var(Y_i) = \frac{\left(\sum_y f_y^2 \; - \; f_x^2\right)}{d} \leq \frac{|f|_2{}^2}{d} \qquad |f|_2^2 = \sum_x f_x^2$$

Using Chebyshev's inequality

$$\Pr[\; |Y_i \; - f_x| > \epsilon |f|_2] \leq \frac{1}{d\epsilon^2} \leq \frac{1}{3} \qquad d = \frac{3}{\epsilon^2}$$

Finally, use analysis of median-trick with $w = \log\left(\frac{1}{\delta}\right)$

Final Guarantees

- Using space $O\left(\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\log(n)\right)$, for any query x, we get an estimate, with prob $1\ -\delta$

$$f_x\ \ -\epsilon|f|_2 \leq f_x \leq f_x + \epsilon|f|_2$$

Comparisons

Algorithm	$\widehat{f_x} - f_x$	Space $\times log(n)$	Error prob	Model
Misra-Gries	$[-\epsilon\|f\|_1, 0]$	$1/\epsilon$	0	Insert Only
SpaceSaving	$[0, \epsilon\|f\|_1]$	$1/\epsilon$	0	Insert Only
CountMin	$[0, \epsilon\|f\|_1]$	$\log\left(\frac{1}{\delta}\right)/\epsilon$	δ	Insert+Delete, strict turnstile
CountSketch	$[-\epsilon\|f\|_2, \epsilon\|f\|_2]$	$\log\left(\frac{1}{\delta}\right)/\epsilon^2$	δ	Insert+Delete

Summary

- CM and Count Sketch to answer point queries about frequencies
  - two user-defined parameters, ϵ and δ
  - Linear sketch, hence can be combined across distributed streams
- Count Sketch handle departures naturally
  - Even if –ve frequencies are present
  - For CM, need strict turnstile
- Extensions to handle range queries and others…
- Actual performance much better than theoretical bound

References:

- Count-sketch:
  - 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

- Moment estimation:
  - Mining massive Datasets by Leskovec, Rajaraman, Ullman

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