COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Fall 2021.

Lecture 10

LOGISTICS

- Problem Set 2 is due next Friday at 11:59pm.
- The midterm is will in class on Tuesday 10/19. Midterm study material has been posted in the Schedule Tab and in Moodle.

Suppose a patient's temperature is 98 degrees. Every time their temperature is taken, the reading is <97 degrees with probability 1/3 and is >99 degrees with probability 1/3. Suppose their temperature is taken three times. What's the probability the median of these three readings is ≥97 and ≤99 degrees? Give your answer to two decimal places.

Question **1**

Correct

2.00 points out of 2.00

Flag question

Edit question

[ABC | |]

[AB | C |]

1/3, 1/3, 1/3

97, 98, 99

A, B, C

(1) $= \frac{3!}{27} = \frac{6}{27}$

(2) $= 1/3^3 = \frac{1}{27}$

(3) $= \frac{\binom{3}{1}}{27} = \frac{3}{27}$

(4) $= 3/27$

$$\frac{6+1+3+3}{27} = \frac{13}{27} = .48$$

Last Class:

- Locality sensitive hashing for near neighbor search.
- MinHash as a locality sensitive hash function for Jaccard similarity
- Balancing false positives and negatives with LSH signatures and repeated hash tables.

SUMMARY

Last Class:

- Locality sensitive hashing for near neighbor search.
- MinHash as a locality sensitive hash function for Jaccard similarity
- Balancing false positives and negatives with LSH signatures and repeated hash tables.

This Class:

- Finish up LSH: SimHash for cosine similarity.
- Frequent Items Estimation
- Count-min sketch algorithm

Next Few Classes:

- Random compression methods for high dimensional vectors. The Johnson-Lindenstrauss lemma.
- Connections to the weird geometry of high-dimensional space.

Next Few Classes:

- Random compression methods for high dimensional vectors. The Johnson-Lindenstrauss lemma.
- Connections to the weird geometry of high-dimensional space.

After the Midterm: Spectral Methods

- PCA, low-rank approximation, and the singular value decomposition.
- Spectral clustering and spectral graph theory.

Next Few Classes:

- Random compression methods for high dimensional vectors. The Johnson-Lindenstrauss lemma.
- Connections to the weird geometry of high-dimensional space.

After the Midterm: Spectral Methods

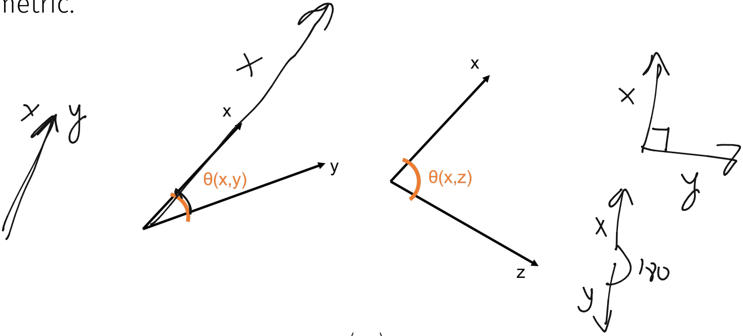
- PCA, low-rank approximation, and the singular value decomposition.
- Spectral clustering and spectral graph theory.

Will use a lot of linear algebra. May be helpful to refresh.

- Vector dot product, addition, Euclidean norm. Matrix vector multiplication.
- Linear independence, column span, orthogonal bases, rank.
- Orthogonal projection, eigendecomposition, linear systems.

SIMHASH FOR COSINE SIMILARITY

Repetition and s-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

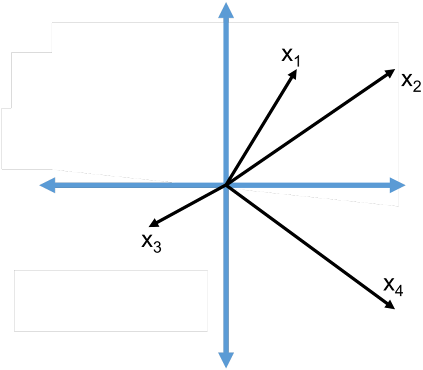


Cosine Similarity: $\cos(\theta(x,y)) = \frac{\langle x,y\rangle}{\|x\|_2 \cdot \|y\|_2}$.

- $\cos(\theta(x,y)) = 1$ when $\theta(x,y) = 0°$ and $\cos(\theta(x,y)) = 0$ when $\theta(x,y) = 90°$, and $\cos(\theta(x,y)) = -1$ when $\theta(x,y) = 180°$.

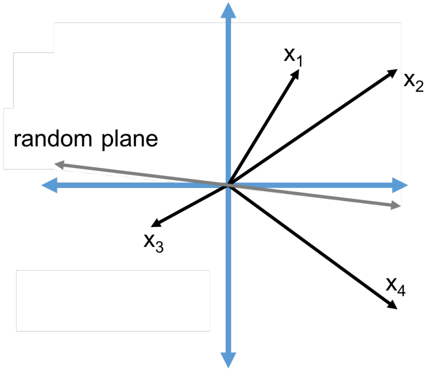
SimHash: LSH for cosine similarity.

SimHash: LSH for cosine similarity.



SimHash: LSH for cosine similarity.

$h(x_i) \to$ scalar



SIMHASH FOR COSINE SIMILARITY

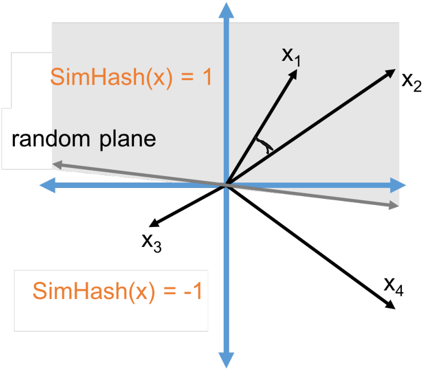
SimHash: LSH for cosine similarity.

$h(x_1) = 1$

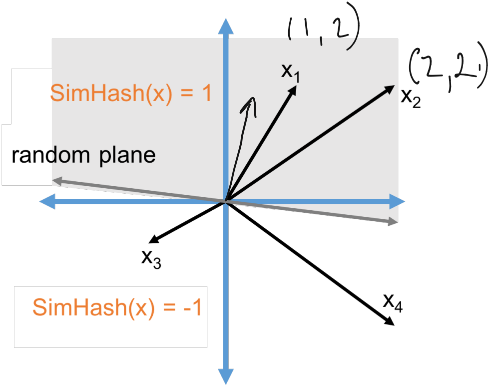
$h(x_2) = 1$

$h(x_3) = -1$

$h(x_4) = -1$



SimHash: LSH for cosine similarity.

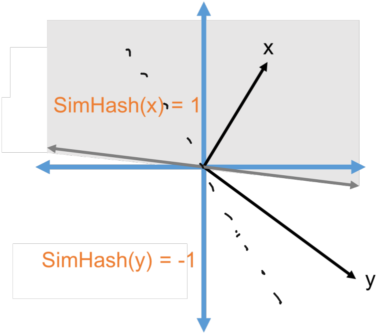


$SimHash(x) = \text{sign}(\langle x, t\rangle)$ for a random vector t.

What is $\Pr[SimHash(x) = SimHash(y)]$?

What is $\Pr[SimHash(x) = SimHash(y)]$?

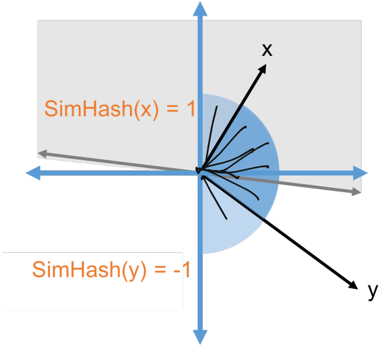
$SimHash(x) \neq SimHash(y)$ when the plane separates x from y.

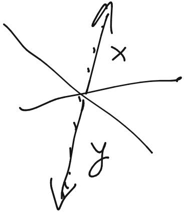


What is $\Pr[SimHash(x) = SimHash(y)]$?

$SimHash(x) \neq SimHash(y)$ when the plane separates x from y.

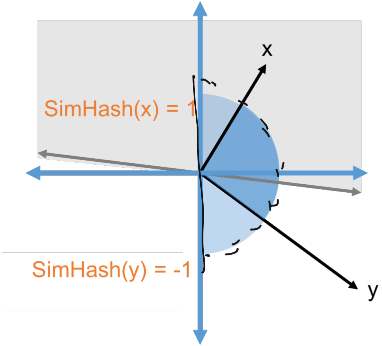
cos(θ(x,y)) = -1
P(SH(x) = SH(y))
= 0





What is $\Pr[SimHash(x) = SimHash(y)]$?

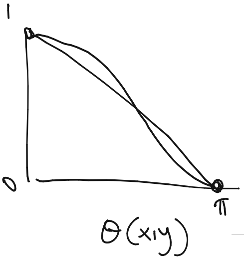
$SimHash(x) \neq SimHash(y)$ when the plane separates x from y.

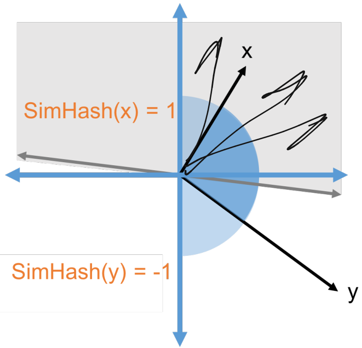


- $\Pr[SimHash(x) \neq SimHash(y)] = \frac{\theta(x,y)}{\pi}$

What is $\Pr[SimHash(x) = SimHash(y)]$?

$SimHash(x) \neq SimHash(y)$ when the plane separates x from y.





- $\Pr[SimHash(x) \neq SimHash(y)] = \frac{\theta(x,y)}{\pi}$
- $\Pr[SimHash(x) = SimHash(y)] = 1 - \frac{\theta(x,y)}{\pi} \approx \frac{\cos(\theta(x,y))+1}{2}$

k-Frequent Items (Heavy-Hitters) Problem: Consider a stream of n items $x_1, \ldots, x_n$ (with possible duplicates). Return any item at appears at least $\frac{n}{k}$ times.

k-Frequent Items (Heavy-Hitters) Problem: Consider a stream of n items $x_1, \ldots, x_n$ (with possible duplicates). Return any item at appears at least $\frac{n}{k}$ times.

$k = 3 \quad \frac{n}{k} = \frac{9}{3} = 3$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
5	12	3	3	4	5	5	10	3

***k*-Frequent Items (Heavy-Hitters) Problem**: Consider a stream of n items $x_1, \ldots, x_n$ (with possible duplicates). Return any item at appears at least $\frac{n}{k}$ times.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
5	12	3	3	4	5	5	10	3

***k*-Frequent Items (Heavy-Hitters) Problem**: Consider a stream of n items $x_1, \ldots, x_n$ (with possible duplicates). Return any item at appears at least $\frac{n}{k}$ times. k=1, at least n times

k=n

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
5	12	3	3	4	5	5	10	3

- What is the maximum number of items that can be returned?

a) n b) k c) n/k d) $\log n$

k-**Frequent Items (Heavy-Hitters) Problem**: Consider a stream of n items $x_1, \ldots, x_n$ (with possible duplicates). Return any item at appears at least $\frac{n}{k}$ times.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
5	12	3	3	4	5	5	10	3

- What is the maximum number of items that can be returned?
  a) n b) k c) n/k d) $\log n$
- Trivial with $O(n)$ space – store the count for each item and return ~~the~~ any one that appears $\geq n/k$ times.
- Can we do it with less space? I.e., without storing all n items?

Applications of Frequent Items:

- Finding top/viral items (i.e., products on Amazon, videos watched on Youtube, Google searches, etc.)
- Finding very frequent IP addresses sending requests (to detect DoS attacks/network anomalies).
- ‘Iceberg queries’ for all items in a database with frequency above some threshold.

Generally want very fast detection, without having to scan through database/logs. I.e., want to maintain a running list of frequent items that appear in a stream.

Association rule learning: A very common task in data mining is to identify common associations between different events.

- Identified via frequent itemset counting. Find all sets of t items that appear many times in the same basket.
- Frequency of an itemset is known as its support.
- A single basket includes many different itemsets, and with many different baskets an efficient approach is critical. E.g., baskets are Twitter users and itemsets are subsets of who they follow.

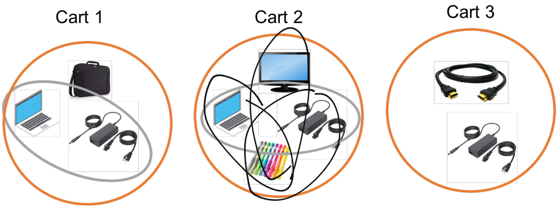
Association rule learning: A very common task in data mining is to identify common associations between different events.

MMDS

- Identified via frequent itemset counting. Find all sets of *t* items that appear many times in the same basket.
- Frequency of an itemset is known as its support.
- A single basket includes many different itemsets, and with many different baskets an efficient approach is critical. E.g., baskets are Twitter users and itemsets are subsets of who they follow.

FREQUENT ITEMSET MINING

Association rule learning: A very common task in data mining is to identify common associations between different events.



- Identified via frequent itemset counting. Find all sets of *t* items that appear many times in the same basket.
- Frequency of an itemset is known as its support.
- A single basket includes many different itemsets, and with many different baskets an efficient approach is critical. E.g., baskets are Twitter users and itemsets are subsets of who they follow.

Issue: No algorithm using $o(n)$ space can output just the items with frequency $\geq n/k$. Hard to tell between an item with frequency n/k (should be output) and $n/k - 1$ (should not be output).

x_1	x_2	x_3	x_4	x_5	x_6	...	$x_{n-n/k+1}$	...	x_n
3	12	9	27	4	101		3		3

n/k-1 occurrences

Issue: No algorithm using $o(n)$ space can output just the items with frequency $\geq n/k$. Hard to tell between an item with frequency n/k (should be output) and $n/k - 1$ (should not be output).

x_1	x_2	x_3	x_4	x_5	x_6	...	$x_{n-n/k+1}$	...	x_n
3	12	9	27	4	101		3		3

n/k-1 occurrences

(ϵ, k)-Frequent Items Problem: Consider a stream of n items $x_1, \ldots, x_n$. Return a set F of items, including all items that appear at least $\frac{n}{k}$ times and only items that appear at least $(1-\epsilon) \cdot \frac{n}{k}$ times.

Issue: No algorithm using $o(n)$ space can output just the items with frequency $\geq n/k$. Hard to tell between an item with frequency n/k (should be output) and $n/k - 1$ (should not be output).

x_1	x_2	x_3	x_4	x_5	x_6	...	$x_{n-n/k+1}$	...	x_n
3	12	9	27	4	101		3		3

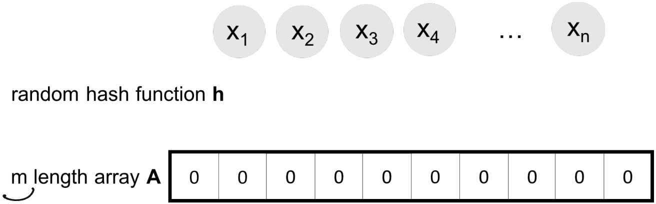
n/k-1 occurrences

(ϵ, k)-Frequent Items Problem: Consider a stream of n items $x_1, \ldots, x_n$. Return a set F of items, including all items that appear at least $\frac{n}{k}$ times and only items that appear at least $(1-\epsilon) \cdot \frac{n}{k}$ times.

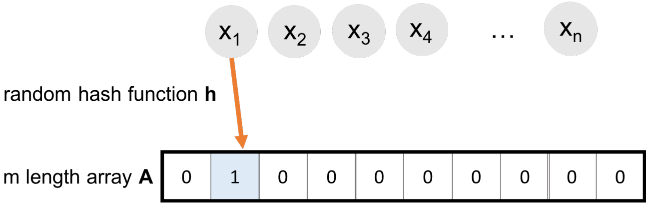
- An example of relaxing to a 'promise problem': for items with frequencies in $[(1-\epsilon) \cdot \frac{n}{k}, \frac{n}{k}]$ no output guarantee.

Today: Count-min sketch – a random hashing based method closely related to bloom filters.

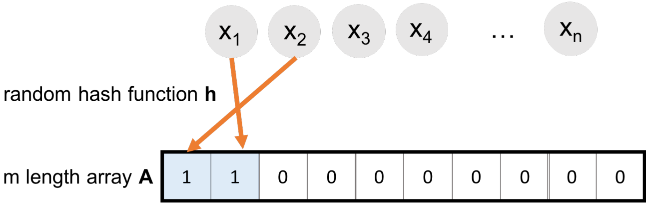
Today: Count-min sketch – a random hashing based method closely related to bloom filters.



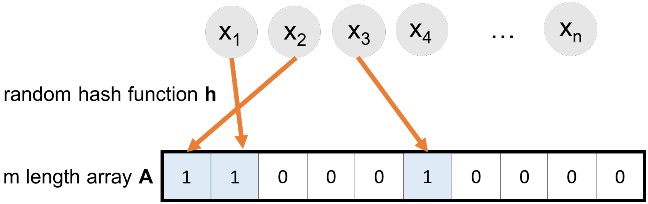
Today: Count-min sketch – a random hashing based method closely related to bloom filters.



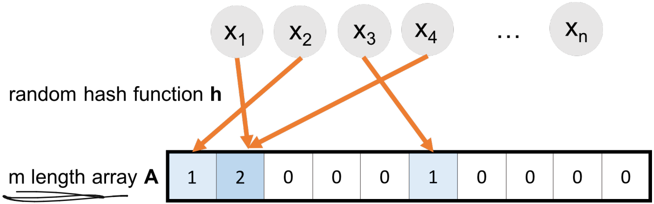
Today: Count-min sketch – a random hashing based method closely related to bloom filters.



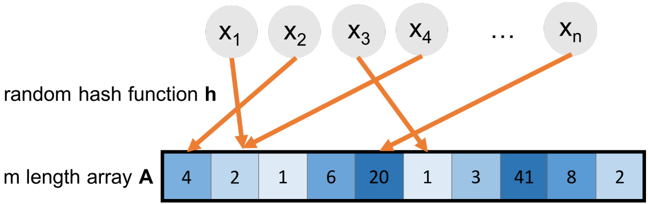
Today: Count-min sketch – a random hashing based method closely related to bloom filters.



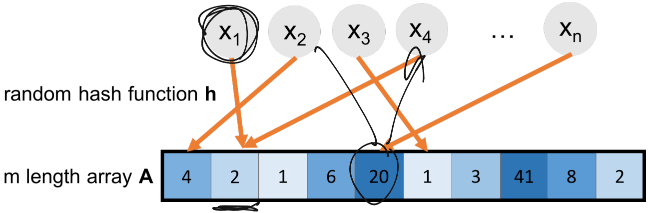
Today: Count-min sketch – a random hashing based method closely related to bloom filters.



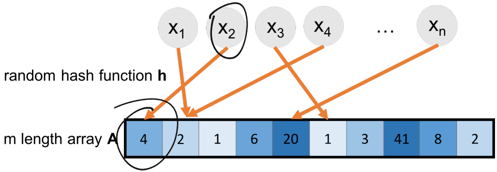
Today: Count-min sketch – a random hashing based method closely related to bloom filters.



Today: Count-min sketch – a random hashing based method closely related to bloom filters.



Will use $A[\mathbf{h}(x)]$ to estimate $f(x)$, the frequency of x in the stream. I.e., $|\{x_i : x_i = x\}|$.

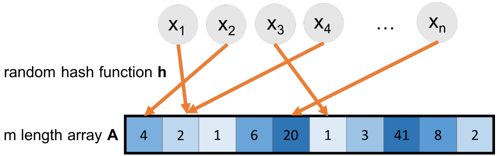


Use $A[\mathbf{h}(x)]$ to estimate $f(x)$.

Claim 1: We always have $A[\mathbf{h}(x)] \geq f(x)$. Why?

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

COUNT-MIN SKETCH ACCURACY

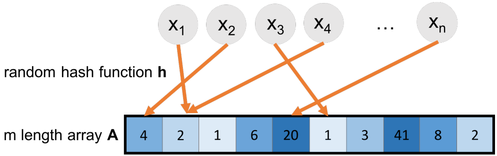


Use $A[\mathbf{h}(x)]$ to estimate $f(x)$.

Claim 1: We always have $A[\mathbf{h}(x)] \geq f(x)$. Why?

- $A[\mathbf{h}(x)]$ counts the number of occurrences of any y with $\mathbf{h}(y) = \mathbf{h}(x)$, including x itself.

> $f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.



Use $A[\mathbf{h}(x)]$ to estimate $f(x)$.

Claim 1: We always have $A[\mathbf{h}(x)] \geq f(x)$. Why?

- $A[\mathbf{h}(x)]$ counts the number of occurrences of any y with $\mathbf{h}(y) = \mathbf{h}(x)$, including x itself.
- $A[\mathbf{h}(x)] = f(x) + \sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)$.

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

COUNT-MIN SKETCH ACCURACY

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}} .$$

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}}.$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] =$$

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

COUNT-MIN SKETCH ACCURACY

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}} .$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] = \sum_{y \neq x} \Pr(\mathbf{h}(y) = \mathbf{h}(x)) \cdot f(y) = \sum_{y \neq x} \frac{1}{m} \cdot f(y)$$

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}} .$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] = \sum_{y \neq x} \Pr(\mathbf{h}(y) = \mathbf{h}(x)) \cdot f(y)$$

$$= \sum_{y \neq x} \frac{1}{m} \cdot f(y)$$

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}}.$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] = \sum_{y \neq x} \Pr(\mathbf{h}(y) = \mathbf{h}(x)) \cdot f(y)$$

$$= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}$$

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

COUNT-MIN SKETCH ACCURACY

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}}.$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] = \sum_{y \neq x} \Pr(\mathbf{h}(y) = \mathbf{h}(x)) \cdot f(y)$$

$$= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}$$

What is a bound on probability that the error is $\geq \frac{2n}{m}$?

$$\frac{\mathbb{E}(\text{error})}{2n/m} = \frac{n/m}{2n/m} = \frac{1}{2}$$

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}} .$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] = \sum_{y \neq x} \Pr(\mathbf{h}(y) = \mathbf{h}(x)) \cdot f(y)$$

$$= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}$$

What is a bound on probability that the error is $\geq \frac{2n}{m}$?

Markov's inequality: $\Pr\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y) \geq \frac{2n}{m}\right] \leq \frac{1}{2}$.

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}}.$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] = \sum_{y \neq x} \Pr(\mathbf{h}(y) = \mathbf{h}(x)) \cdot f(y)$$

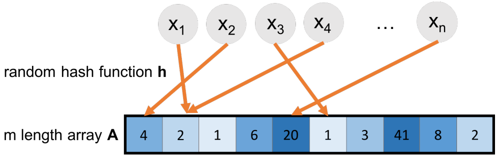
$$= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}$$

What is a bound on probability that the error is $\geq \frac{2n}{m}$?

Markov's inequality: $\Pr\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y) \geq \frac{2n}{m}\right] \leq \frac{1}{2}$.

What property of **h** is required to show this bound? a) fully random b) pairwise independent c) 2-universal d) locality sensitive

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). **h**: random hash function. m: size of Count-min sketch array.



Claim: For any x, with probability at least 1/2,

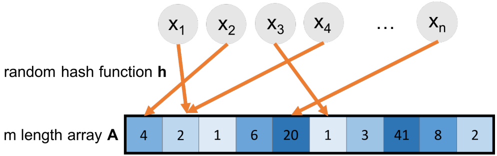
$$f(x) \leq A[\mathbf{h}(x)] \leq f(x) + \frac{2n}{m}.$$

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

COUNT-MIN SKETCH ACCURACY

$O\left(\frac{k}{\varepsilon}\right)$

$\frac{n}{k}$ $\quad (1-\varepsilon)\frac{n}{k}$



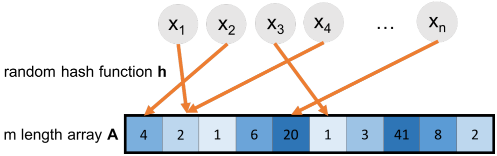
Claim: For any x, with probability at least 1/2,

$$f(x) \le A[\mathbf{h}(x)] \le f(x) + \frac{2n}{m}.$$

$$\frac{2n}{m} = \frac{2n}{\frac{2k}{\varepsilon}} = \boxed{\frac{\varepsilon n}{k}}$$

To solve the (ϵ, k)-Frequent elements problem, set $m = \frac{2k}{\epsilon}$.

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.



Claim: For any x, with probability at least 1/2,

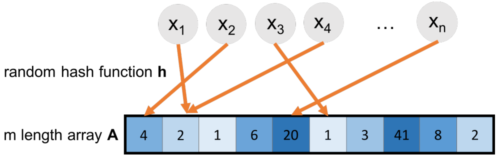
$$f(x) \leq A[\mathbf{h}(x)] \leq f(x) + \frac{2n}{m}.$$

To solve the (ϵ, k)-Frequent elements problem, set $m = \frac{2k}{\epsilon}$.

How can we improve the success probability?

> $f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

COUNT-MIN SKETCH ACCURACY



Claim: For any x, with probability at least 1/2,

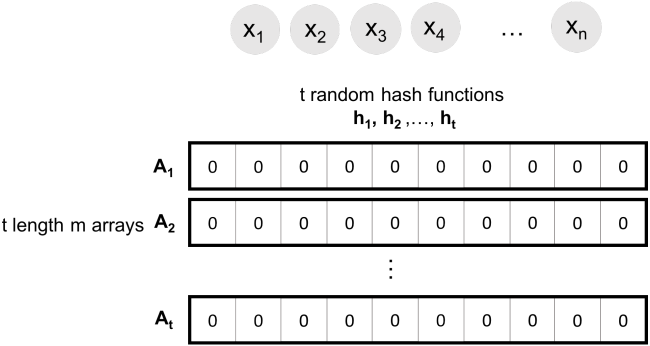
$$f(x) \leq A[\mathbf{h}(x)] \leq f(x) + \frac{2n}{m}.$$

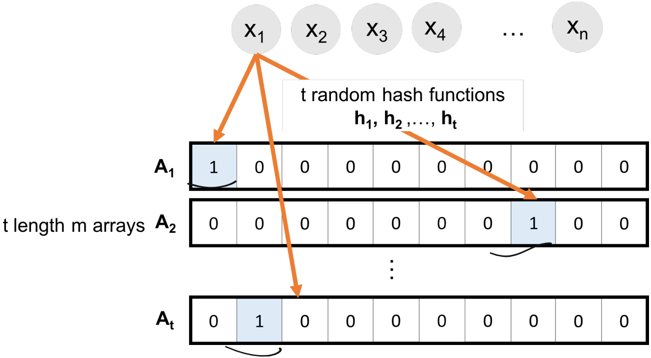
To solve the (ϵ, k)-Frequent elements problem, set $m = \frac{2k}{\epsilon}$.

How can we improve the success probability? Repetition.

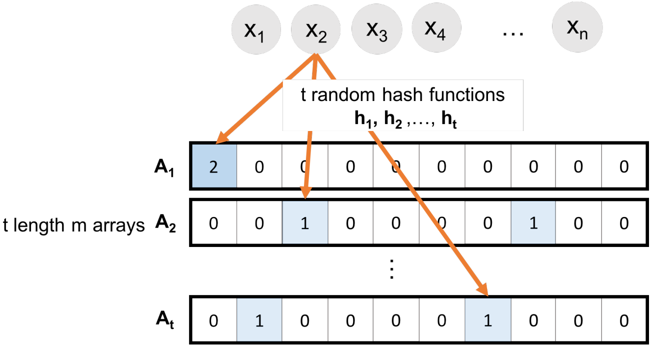
> $f(x)$: frequency of x in the stream (i.e., number of items equal to x). **h**: random hash function. m: size of Count-min sketch array.

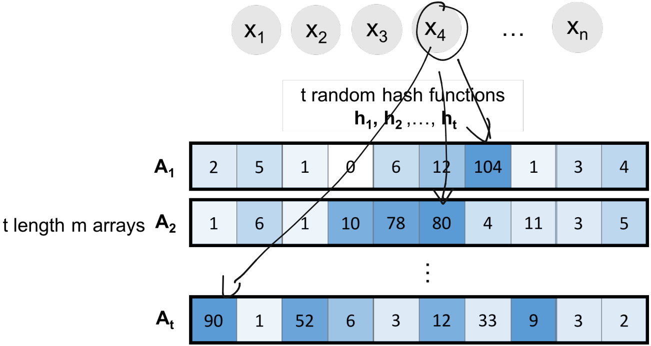
COUNT-MIN SKETCH ACCURACY



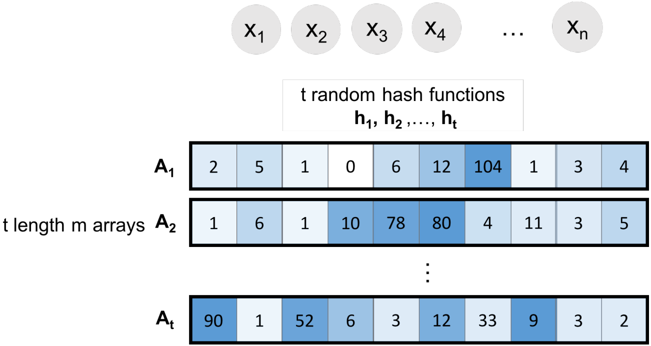


x_1
x_2
x_3
x_4
…
x_n
t random hash functions
$h_1, h_2, \ldots, h_t$
A_1
1 0 0 0 0 0 0 0 0 0
t length m arrays
A_2
0 0 0 0 0 0 0 1 0 0
A_t
0 1 0 0 0 0 0 0 0 0

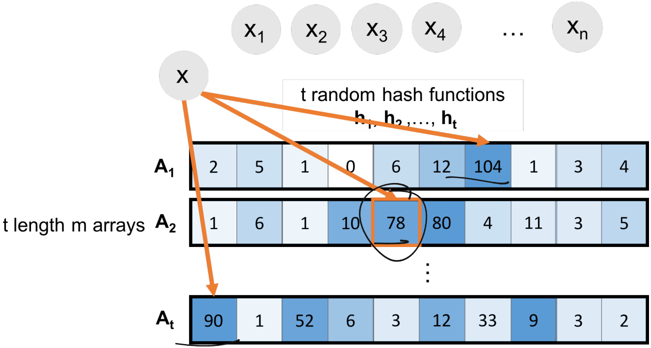

x1
x2
x3
x4
…
xn
t random hash functions
h1, h2 ,…, ht
A1
2 0 0 0 0 0 0 0 0 0
t length m arrays
A2
0 0 1 0 0 0 0 1 0 0
⋮
At
0 1 0 0 0 0 1 0 0 0


x1
x2
x3
x4
...
xn
t random hash functions
h1, h2 ,..., ht
A1
2 5 1 0 6 12 104 1 3 4
t length m arrays
A2
1 6 1 10 78 80 4 11 3 5
At
90 1 52 6 3 12 33 9 3 2

COUNT-MIN SKETCH ACCURACY

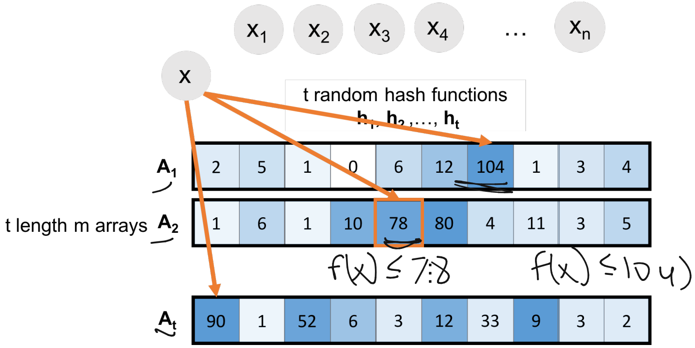


Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$. (count-min sketch)



Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$. (count-min sketch)

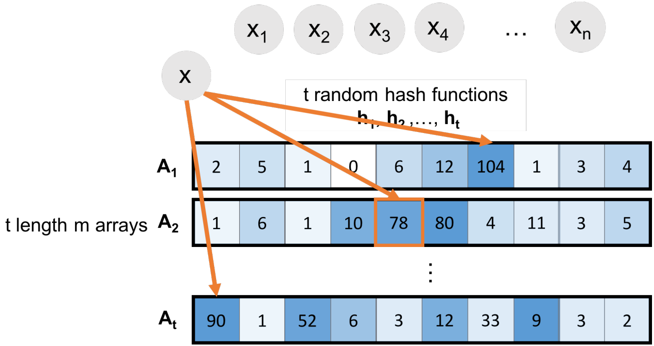
$\hat{f}(x) = 78$



Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$. (count-min sketch)

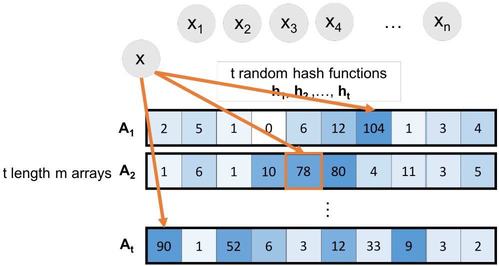
Why min instead of mean or median?

$m = \frac{2k}{\varepsilon}$

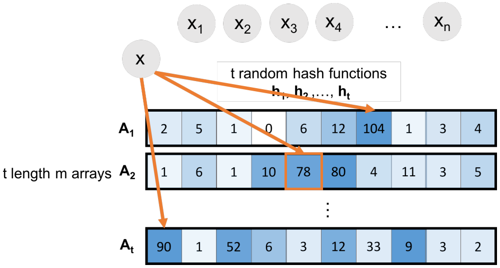


Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$. (count-min sketch)

Why min instead of mean or median? The minimum estimate is always the most accurate since they are all overestimates of the true frequency!



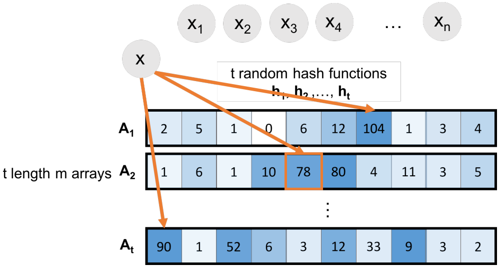
Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$



Estimate $f(x)$ by $\tilde{f}(x) = \min_{i\in[t]} A_i[\mathbf{h}_i(x)]$

- For every x and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$, with probability $\geq 1/2$:

$$f(x) \leq A_i[\mathbf{h}_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$

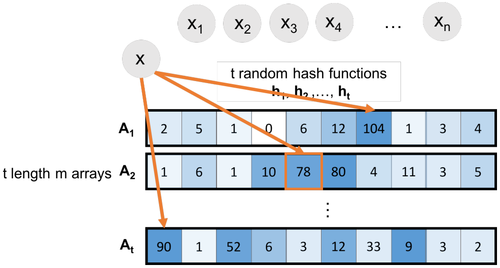


Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$

- For every x and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$, with probability $\geq 1/2$:

$$f(x) \leq A_i[\mathbf{h}_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$

- What is $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}]$? $1 - \frac{1}{2^t}$

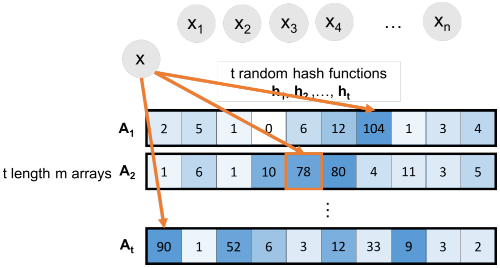


Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$

- For every x and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$, with probability $\geq 1/2$:

$$f(x) \leq A_i[\mathbf{h}_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$

- What is $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}]$? $1 - 1/2^t$.



Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$

- For every x and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$, with probability $\geq 1/2$:

$$f(x) \leq A_i[\mathbf{h}_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$

$1 - \frac{1}{2^{\log(1/\delta)}} = 1 - \delta$

- What is $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}]$? $1 - 1/2^t$.
- To get a good estimate with probability $\geq 1 - \delta$, set $t = \log(1/\delta)$.

Upshot: Count-min sketch lets us estimate the frequency of every item in a stream up to error $\frac{\epsilon n}{k}$ with probability $\geq 1 - \delta$ in $O\left(\log(1/\delta) \cdot k/\epsilon\right)$ space.

Upshot: Count-min sketch lets us estimate the frequency of every item in a stream up to error $\frac{\epsilon n}{k}$ with probability $\geq 1 - \delta$ in $O\left(\log(1/\delta) \cdot k/\epsilon\right)$ space.

$O(t \cdot m)$

- Accurate enough to solve the (ϵ, k)-Frequent elements problem – distinquish between items with frequency $\frac{n}{k}$ and those with frequency $(1 - \epsilon)\frac{n}{k}$.

COUNT-MIN SKETCH

Upshot: Count-min sketch lets us estimate the frequency of every item in a stream up to error $\frac{\epsilon n}{k}$ with probability $\geq 1 - \delta$ in $O(\log(1/\delta) \cdot k/\epsilon)$ space.

- Accurate enough to solve the (ϵ, k)-Frequent elements problem – distinquish between items with frequency $\frac{n}{k}$ and those with frequency $(1 - \epsilon)\frac{n}{k}$.
- How should we set δ if we want a good estimate for all items at once, with 99% probability?

$\frac{.01}{n} \cdot n$ $\delta = \frac{.01}{n}$

$O(t \cdot m) = O(\log(n) \cdot k/\epsilon)$

Count-min sketch gives an accurate frequency estimate for every item in the stream. But how do we identify the frequent items without having to store/look up the estimated frequency for all elements in the stream?

Count-min sketch gives an accurate frequency estimate for every item in the stream. But how do we identify the frequent items without having to store/look up the estimated frequency for all elements in the stream?

One approach:

- When a new item comes in at step i, check if its estimated frequency is $\geq i/k$ and store it if so.
- At step i remove any stored items whose estimated frequency drops below i/k.
- Store at most $O(k)$ items at once and have all items with frequency $\geq n/k$ stored at the end of the stream.

Questions on Frequent Elements?