Application: Distinct Elements

CSE 312 Spring 26

Lecture 21

Data mining – Stream Model

- In many data mining situations, data often not known ahead of time.
  - Examples: Google queries, Twitter or Facebook status updates, YouTube video views
- Think of the data as an infinite stream
- Input elements (e.g. Google queries) enter/arrive one at a time.
  - We cannot possibly store the stream.

Question: How do we make critical calculations about the data stream using a limited amount of memory?

Stream Model – Problem Setup

Input: sequence (aka. "stream") of N elements $x_1, x_2, \ldots, x_N$ from a known universe U (e.g., 8-byte integers).

Goal: perform a computation on the input, in a single left to right pass, where:

- Elements processed in real time
- Can't store the full data ⇒ use minimal amount of storage while maintaining working "summary"

What can we compute?

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4

Some functions are easy:

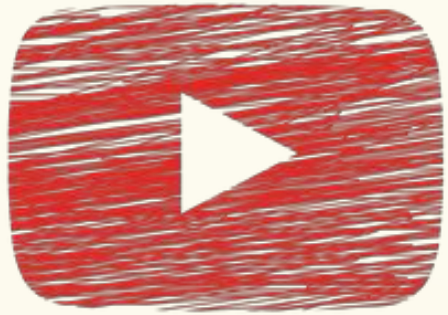
- Min
- Max
- Sum
- Average

Today: Counting distinct elements

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4

Application

You are the content manager at YouTube, and you are trying to figure out the **distinct** view count for a video. How do we do that?



Note: A person can view their favorite videos several times, but they only count as 1 **distinct** view!

Other applications

- IP packet streams: How many distinct IP addresses or IP flows (source+destination IP, port, protocol)
  - Anomaly detection, traffic monitoring
- Search: How many distinct search queries on Google on a certain topic yesterday (users identified by IP address)
- Web services: how many distinct users (cookies) searched/browsed a certain term/item
  - Advertising, marketing trends, etc.

Counting distinct elements

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4

N = # of IDs in the stream = 11, m = # of distinct IDs in the stream = 5

Want to compute number of **distinct** IDs in the stream.

How?

Counting distinct elements

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4

N = # of IDs in the stream = 11, m = # of distinct IDs in the stream = 5

Want to compute number of **distinct** IDs in the stream.

- *Naïve solution: As the data stream comes in, store all distinct IDs in a hash table.*
- *Space requirement:* $\Omega(m)$

YouTube Scenario: m *is huge!*

Counting distinct elements

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4

N = # of IDs in the stream = 11, m = # of distinct IDs in the stream = 5

Want to compute (at least approximately) the number of **distinct** IDs in the stream.

How to do this without storing all the elements?

Detour – I.I.D. Uniforms

If $Y_1, \cdots, Y_m \sim \text{Unif}(0,1)$ (i.i.d.) where do we expect the points to end up?

$Y_1 \sim \text{Unif}(0,1)$ $\quad E(Y_1) = \frac{1}{2}$

$m = 1$

Detour – I.I.D. Uniforms

If $Y_1, \cdots, Y_m \sim \text{Unif}(0,1)$ (i.i.d.) where do we expect the points to end up?

$m = 1$

$E(\min(Y_1, Y_2)) = \frac{1}{3}$

$m = 2$

Detour – I.I.D. Uniforms

If $Y_1, \cdots, Y_m \sim \text{Unif}(0,1)$ (i.i.d.) where do we expect the points to end up?

"Evenly spread out"

$m = 1$

$m = 2$

$m = 4$

$E(\min(Y_1, \cdots Y_4)) = \frac{1}{5}$

Detour – Min of I.I.D. Uniforms

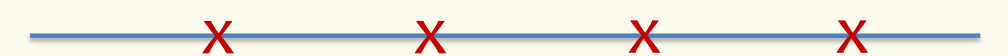
If $Y_1, \cdots, Y_m \sim \text{Unif}(0,1)$ (iid) where do we expect the points to end up?

In general, $\mathbb{E}[\min(Y_1, \cdots, Y_m)] = \frac{1}{m+1}$

$m = 1$ $\quad \mathbb{E}[\min(Y_1)] =$

$m = 2$ $\quad \mathbb{E}[\min(Y_1, Y_2)] =$

$m = 4$ $\quad \mathbb{E}[\min(Y_1, \cdots, Y_4)] =$

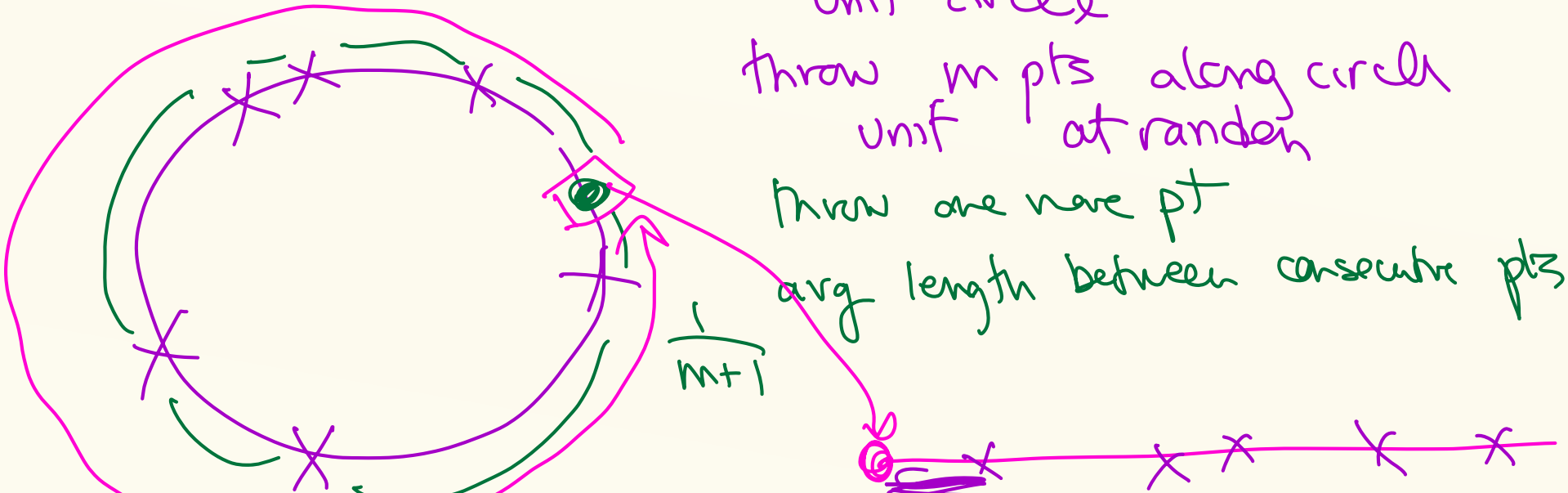


Detour – Min of I.I.D. Uniforms

If $Y_1, \cdots, Y_m \sim \text{Unif}(0,1)$ (iid) where do we expect the points to end up?

In general, $\mathbb{E}[\min(Y_1, \cdots, Y_m)] = \frac{1}{m+1}$

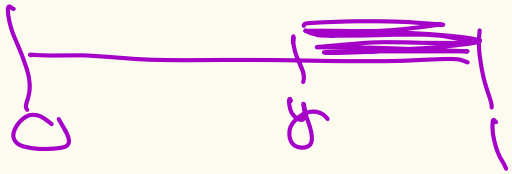
What is some intuition for this?



Detour – Min of I.I.D. Uniforms

If $Y_1, \cdots, Y_m \sim \text{Unif}(0,1)$ (i.i.d.) where do we expect the points to end up?

e.g., what is $\mathbb{E}[\min\{Y_1, \cdots, Y_m\}]$?



CDF: Observe that $\min\{Y_1, \cdots, Y_m\} \geq y$ if and only if $Y_1 \geq y, \ldots, Y_m \geq y$

$$P(\min(Y_1 \ldots Y_m) \geq y) = \Pr(Y_1 \geq y, Y_2 \geq y, \ldots, Y_m \geq y)$$

$$\overset{\text{indep}}{=} P(Y_1 \geq y)\, P(Y_2 \geq y) \cdots P(Y_m \geq y) = (1-y)^m$$

$$F_Y(y) = P(\min(Y_1 \ldots Y_m) \leq y) = 1 - (1-y)^m$$

$$f_Y(y) = m(1-y)^{m-1}$$

$$E(Y) = \int_0^1 y\, m(1-y)^{m-1}\, dy = \frac{1}{m+1}$$

Detour – Min of I.I.D. Uniforms

If $Y_1, \cdots, Y_m \sim \text{Unif}(0,1)$ (i.i.d.) where do we expect the points to end up?

e.g., what is $\mathbb{E}[\min\{Y_1, \cdots, Y_m\}]$?

CDF: Observe that $\min\{Y_1, \cdots, Y_m\} \geq y$ if and only if $Y_1 \geq y, \ldots, Y_m \geq y$

$$P(\min\{Y_1, \cdots, Y_m\} \geq y) = P(Y_1 \geq y, \ldots, Y_m \geq y)$$

$y \in [0,1]$

$$= P(Y_1 \geq y) \cdots P(Y_m \geq y) \qquad \text{(Independence)}$$

$$= (1-y)^m$$

$$\Rightarrow P(\min\{Y_1, \cdots, Y_m\} \leq y) = 1 - (1-y)^m$$

$$F_Y(y) = P(\min\{Y_1, \cdots, Y_m\} \leq y) = 1 - (1-y)^m.$$

$$f_Y(y) = \frac{d}{dy} F_Y(y) = m(1-y)^{m-1}.$$

$$\mathbb{E}[Y] = \int_0^1 y\, f_Y(y)\, \mathrm{d}y = \int_0^1 y\, m(1-y)^{m-1} \mathrm{d}y = \frac{1}{m+1}$$

Detour – Min of I.I.D. Uniforms

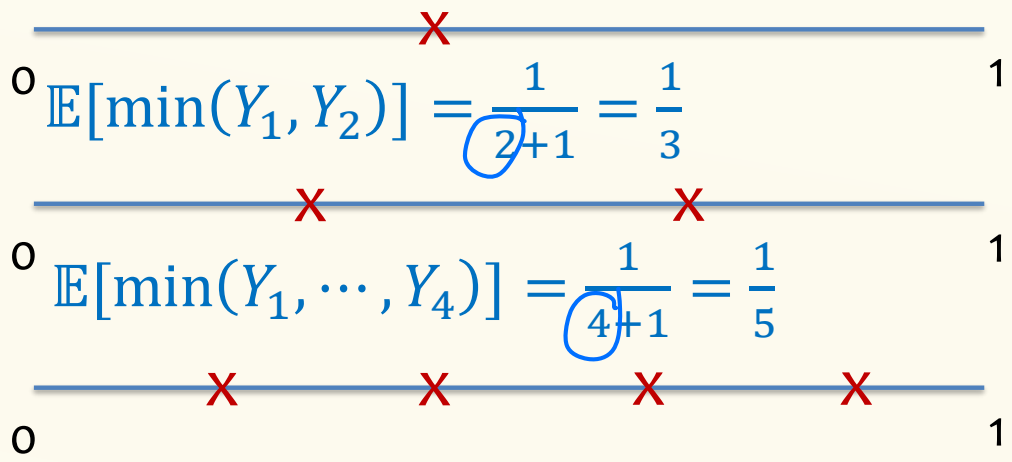
If $Y_1, \cdots, Y_m \sim \text{Unif}(0,1)$ (iid) where do we expect the points to end up?

In general, $\mathbb{E}[\min(Y_1, \cdots, Y_m)] = \frac{1}{m+1}$

$m = 1$ $\quad \mathbb{E}[\min(Y_1)] = \frac{1}{1+1} = \frac{1}{2}$

$m = 2$ $\quad \mathbb{E}[\min(Y_1, Y_2)] = \frac{1}{2+1} = \frac{1}{3}$

$m = 4$ $\quad \mathbb{E}[\min(Y_1, \cdots, Y_4)] = \frac{1}{4+1} = \frac{1}{5}$



Back to counting distinct elements

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4

N = # of IDs in the stream = 11, m = # of distinct IDs in the stream = 5

Want to compute number of **distinct** IDs in the stream, at least approximately.

How to do this without storing all the elements?

$h: U \longrightarrow \{0, 1, \ldots, n-1\}$

$h'(x) = \frac{h(x)}{n}$

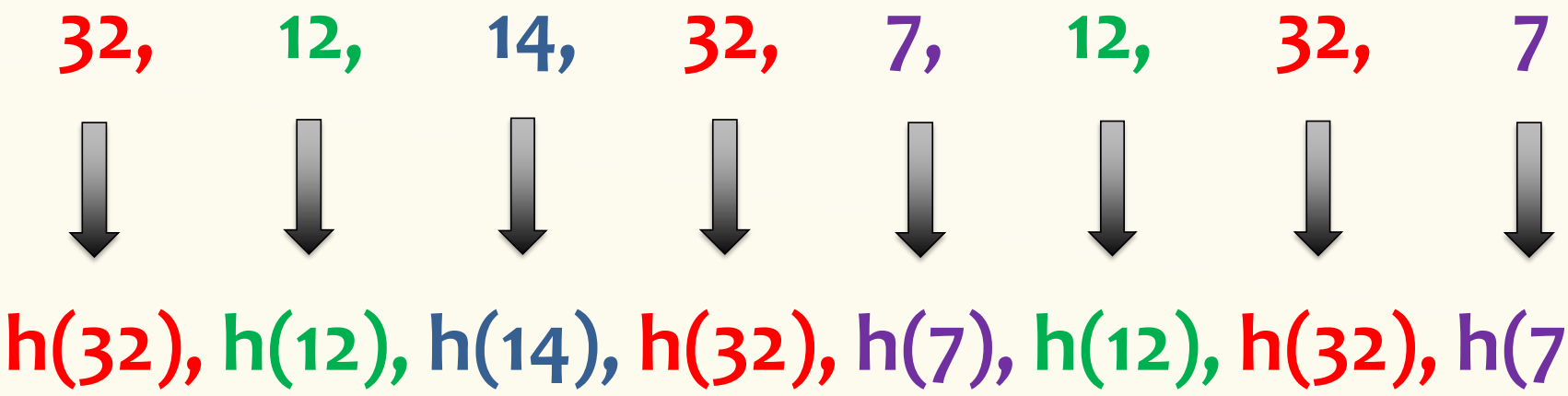
Distinct Elements – Hashing into $[0, 1]$

stream has length N

m distinct elts.

Hash function $h: U \to [0,1]$

Key property: Every time you hash the same element, you get the same value.

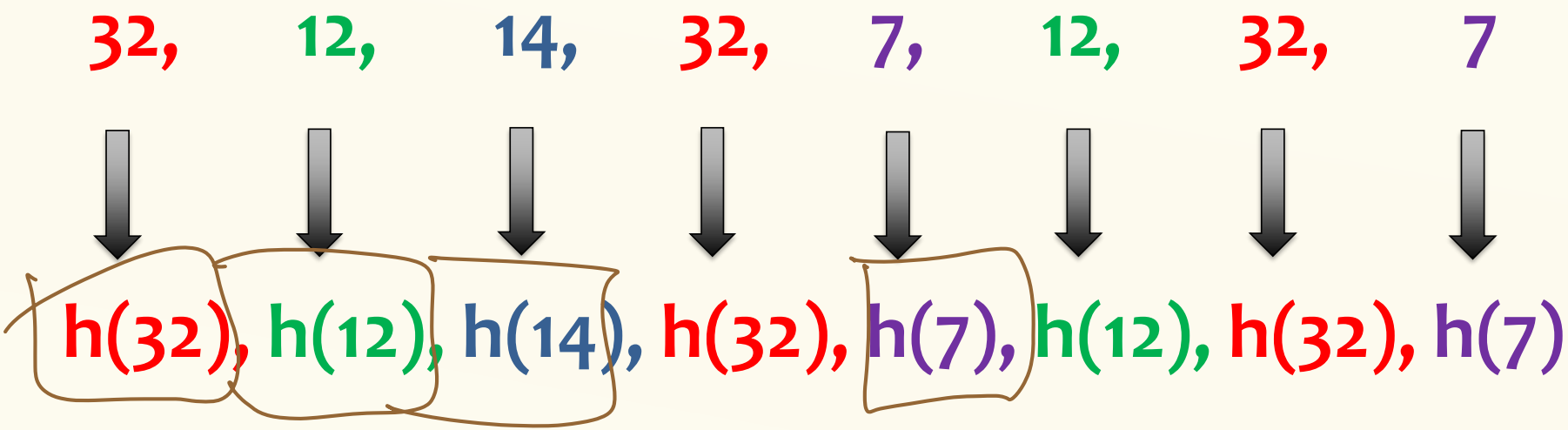


Distinct Elements – Hashing into $[0, 1]$

Hash function $h: U \to [0,1]$

Key Assumptions:

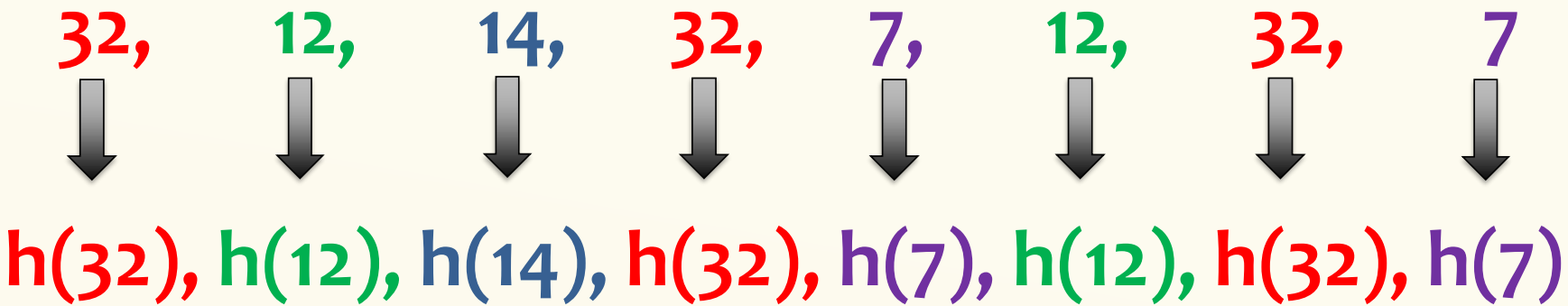
- Hash values are uniformly random.
- Distinct elements are hashed independently.



Hash function $h: U \to [0,1]$

Key Assumptions:

- Hash values are uniformly random.
- Distinct elements are hashed independently.



Stream of length N=8, m=4 distinct elements

$\to$ 4 i.i.d. RVs $\quad h(32), h(12), h(14), h(7) \sim \text{Unif}(0,1)$

$$\to \mathbb{E}[\min\{h(32), h(12), h(14), h(7)\}] = \frac{1}{4+1} = \frac{1}{5}$$

Distinct Elements – Hashing into $[0, 1]$

Hash function $h: U \to [0,1]$

Key Assumptions:

- Hash values are uniformly random.
- Distinct elements are hashed independently.

$x_1, x_2, \ldots, x_N$ contains m distinct elements

$h(x_1), h(x_2), \ldots, h(x_N)$ contains m i.i.d. rvs $\sim \text{Unif}(0,1)$

and $N - m$ repeats

$$\mathbb{E}[\min\{h(x_1), \ldots, h(x_N)\}] = \frac{1}{m+1}$$

A super duper clever idea!!!!

We can track $\min\{h(x_1), \dots, h(x_N)\}$ by storing only a single element.

Suppose that $\min\{h(x_1), \dots, h(x_N)\}$ is $\approx \mathbb{E}[\min\{h(x_1), \dots, h(x_N)\}]$.

Then we would be golden because

$$\mathbb{E}[\min\{h(x_1), \dots, h(x_N)\}] = \frac{1}{m+1}$$

$$m+1 \approx \frac{1}{\mathbb{E}(\min(h(x_1), \dots, h(x_N)))}$$

$$\Rightarrow m \approx \frac{1}{\mathbb{E}(\min(\quad))} - 1$$

A super duper clever idea!!!!

We can track $\min\{h(x_1), \ldots, h(x_N)\}$ by storing only a single element.

Suppose that $\min\{h(x_1), \ldots, h(x_N)\}$ is $\approx \mathbb{E}[\min\{h(x_1), \ldots, h(x_N)\}]$.

Then we would be golden!

$$\mathbb{E}[\min\{h(x_1), \ldots, h(x_N)\}] = \frac{1}{m+1}$$

$$\text{So } m = \frac{1}{\mathbb{E}[\min\{h(x_1),\ldots,h(x_N)\}]} - 1$$

The MinHash Algorithm – Idea

$$m = \frac{1}{\mathbb{E}[\min\{h(x_1), \dots, h(x_N)\}]} - 1$$

Val.

1. Compute $\text{val} = \min\{h(x_1), \dots, h(x_N)\}$
2. Assume that $\text{val} \approx \mathbb{E}[\min\{h(x_1), \dots, h(x_N)\}]$
3. Output as estimate for m: $\text{round}\left(\frac{1}{\text{val}} - 1\right)$

The MinHash Algorithm – Implementation

Algorithm **MinHash**$(x_1, x_2, \ldots, x_N)$

$\text{val} \leftarrow \infty$

for $i = 1$ **to** N **do**

    $\text{val} \leftarrow \min\{\text{val}, h(x_i)\}$

return $\text{round}\left(\frac{1}{\text{val}} - 1\right)$

Memory cost = just remember val
(with sufficient precision)

MinHash Example

1. Compute $\text{val} = \min\{h(x_1), \ldots, h(x_N)\}$
2. Assume that $\text{val} \approx \mathbb{E}[\min\{h(x_1), \ldots, h(x_N)\}]$
3. Output $\text{round}\left(\frac{1}{\text{val}} - 1\right)$

Stream: 13, 25, 19, 25, 19, 19

Hashes: 0.51, 0.26, 0.79, 0.26, 0.79, 0.79

val 0.51 0.26 0.26 0.26 0.26 0.26

$\text{round}\left(\frac{1}{0.26} - 1\right) \approx 3.$

What does MinHash return?

MinHash Example II

Stream: 11, 34, 89, 11, 89, 23

Hashes: 0.5, 0.21, 0.94, 0.5, 0.94, 0.1

Output is $\frac{1}{0.1} - 1 = 9$

Clearly, not a very good answer!

Not particularly unlikely: $P(h(x) < 0.1) = 0.1$

The MinHash Algorithm – Problem

Algorithm `MinHash`$(x_1, x_2, \dots, x_N)$

  val $\leftarrow \infty$

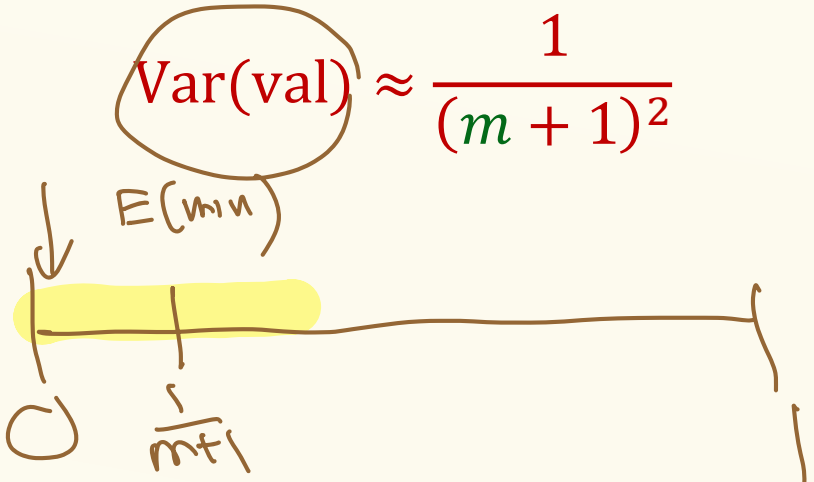
  for $i = 1$ **to** N **do**

    val $\leftarrow \min\{\text{val}, h(x_i)\}$

  return round $\left(\frac{1}{\text{val}} - 1\right)$

Problem: val is not $\mathbb{E}[\text{val}]$!
How far is val from $\mathbb{E}[\text{val}]$?

$$\text{Var}(\text{val}) \approx \frac{1}{(m+1)^2}$$



$\text{val} = \min\{h(x_1), \dots, h(x_N)\}$

$$\mathbb{E}[\text{val}] = \frac{1}{m+1}$$

How can we reduce the variance?

Idea: Repetition to reduce variance!

Use k **independent** hash functions $h^1, h^2, \cdots h^k$

$$\text{val}_1 = \min\{h^1(x_1), \ldots, h^1(x_N)\}$$
$$\text{val}_2 = \min\{h^2(x_1), \ldots, h^2(x_N)\}$$
$$\ldots$$
$$\text{val}_k = \min\{h^k(x_1), \ldots, h^k(x_N)\}$$

$$\widehat{\text{val}} \leftarrow \frac{1}{k}\sum_{i=1}^{k} \text{val}_i$$

$$E(\widehat{val}) = \frac{1}{k}\sum_{i=1}^{k} E(val_i)$$

$$E(val_i) = E(\min(U_1, \ldots, U_m)) = \frac{1}{m+1}$$

$$= \frac{1}{m+1}$$

$$Var(\widehat{val}) = \frac{1}{k^2}\sum_{i=1}^{k} Var(val_i) = \frac{1}{k}\,\frac{1}{(m+1)^2}$$

$$Var(val_i) = \frac{1}{(m+1)^2}$$

Output as estimate for m: $\text{round}(1/\widehat{\text{val}} - 1)$

How can we reduce the variance?

Idea: Repetition to reduce variance!

Use k **independent** hash functions $h^1, h^2, \cdots h^k$

Algorithm `MinHash`$(x_1, x_2, \ldots, x_N)$

$\text{val}_1, \ldots, \text{val}_k \leftarrow \infty$

for $i = 1$ **to** N **do**

  for $j = 1$ **to** k **do** $\quad \text{val}_j \leftarrow \min\{\text{val}_j, h^j(x_i)\}$

$$\widehat{\text{val}} \leftarrow \frac{1}{k}\sum_{i=1}^{k} \text{val}_i$$

return $\text{round}\left(1/\widehat{\text{val}} - 1\right)$

$$\text{Var}\left(\widehat{\text{val}}\right) = \frac{1}{k}\frac{1}{(m+1)^2}$$

$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} X_i$

$X_1 \ldots X_n$ i.i.d.

How do we get a handle on how many hash functions to use?

CLT

Tail bounds

Tail Bounds (Idea)

Bounding the probability a random variable is far from its mean. Usually statements of the form:

$$\Pr(X \geq a) \leq b$$
$$\Pr(|X - E[X]| \geq a) \leq b$$

Useful tool when

- An approximation that is easy to compute is sufficient
- The process is too complex to analyze exactly

Key point

- A random variable might almost never be at least its expectation.
- Similarly, a random variable might almost always be at least its expectation or even much bigger.

Tail Bounds

Generally, the more you know about your random variable the better tail bounds you can get.

- Markov's Inequality
- Chebyshev's Inequality

Markov's Inequality

Theorem. Let X be a random variable taking only non-negative values. Then, for any $t > 0$,

$$\mathbb{P}(X \geq t) \leq \frac{\mathbb{E}(X)}{t}.$$

$$\mathbb{P}\big(X \geq t \cdot \mathbb{E}(X)\big) \leq \frac{1}{t}.$$

Incredibly simplistic – only requires that the random variable is non-negative and only needs you to know expectation. You don't need to know **anything else** about the distribution of X.

Some tail bounds

- Markov's Inequality
- Chebyshev's Inequality

Using variance

- If we know more about the random variable, e.g. its variance, we can get a better bound!

Chebyshev's Inequality

Markov's inequality
$\mathbb{P}(X \geq t) \leq \frac{\mathbb{E}(X)}{t}$.

Theorem. Let X be a random variable. Then, for any $t > 0$,

$$\mathbb{P}(|X - \mathbb{E}(X)| \geq t) \leq \frac{\mathrm{Var}(X)}{t^2}.$$

Proof: Define $Z = X - \mathbb{E}(X)$

Definition of Variance

$$\mathbb{P}(|Z| \geq t) = \mathbb{P}(Z^2 \geq t^2) \leq \frac{\mathbb{E}(Z^2)}{t^2} = \frac{\mathrm{Var}(X)}{t^2}$$

$|Z| \geq t$ iff $Z^2 \geq t^2$

Markov's inequality ($Z^2 \geq 0$)

Tail Bounds

Useful for approximations of complex systems. How good the approximation is depends on the actual distribution and the context you are using it in.

- Usually loose upper-bounds are okay when designing for worst-case

Generally, the more you know about your random variable the better tail bounds you can get.

And one more thing….

Detour – Min of I.I.D. Uniforms

Useful fact. For any random variable Y taking non-negative values

$$\mathbb{E}[Y] = \int_0^\infty P(Y \geq y)\mathrm{d}y$$

Proof

$$\mathbb{E}[Y] = \int_0^\infty x \cdot f_Y(x)\,\mathrm{d}x = \int_0^\infty \left(\int_0^x 1\,\mathrm{d}y\right) \cdot f_Y(x)\,\mathrm{d}x = \int_0^\infty \int_0^x f_Y(x)\,\mathrm{d}y\,\mathrm{d}x$$

$$= \int_0^\infty \int_y^\infty f_Y(x)\,\mathrm{d}x\,\mathrm{d}y = \int_0^\infty P(Y \geq y)\,\mathrm{d}y$$

Detour – Min of I.I.D. Uniforms

$Y_1, \cdots, Y_m \sim \text{Unif}(0,1)$ (i.i.d.)

$Y = \min\{Y_1, \cdots, Y_m\}$

Useful fact. For any continuous random variable Y taking non-negative values

$$\mathbb{E}[Y] = \int_0^\infty P(Y \geq y)\mathrm{d}y$$

$$\mathbb{E}[Y] = \int_0^\infty P(Y \geq y)\mathrm{d}y = \int_0^1 (1-y)^m \mathrm{d}y$$

$$= -\frac{1}{m+1}(1-y)^{m+1}\Big|_0^1 = 0 - \left(-\frac{1}{m+1}\right) = \frac{1}{m+1}$$

Useful fact. For any discrete random variable Y taking non-negative values $\sum_{k=0}^{\infty} P(Y > \mathrm{k})$

$$\mathbb{E}[Y] = \sum_{k=0}^{\infty} P(Y > \mathrm{k})$$