CSE 312
Foundations of Computing II

Lecture 19: Application -- Distinct elements

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Anna R. Karlin

Slide Credit: Based on Stefano Tessaro's slides for 312 19au
incorporating ideas from Alex Tsun, Rachel Lin, Hunter Schafer & myself ☺

Data mining – Stream Model

- In many data mining situations, the data is not known ahead of time.
  Examples: Google queries, Twitter or Facebook status updates
  Youtube video views
- In some ways, best to think of the data as an infinite stream that is non-stationary (distribution changes over time)
- 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?

x_1

x_2

Problem Setup

- Input: sequence 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
- 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?

Set .

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 O(m) , where m is the number of distinct IDs*
- *Consider the number of users of youtube, and the number of videos on youtube. This is not feasible.*

Counting distinct elements

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

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

- *How to do this without storing all the elements?*

Yet another super cool application of probability

Counting distinct elements

We will use a hash function $\boxed{h: U \to [0,1]}$

Assumption: For distinct values in U, the function maps to iid (independent and identically distributed) Unif(0,1) random numbers.

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

$$\tilde{h}(x) = i \qquad h(x) = \frac{i}{k} \in \{0, \tfrac{1}{k}, \ldots \tfrac{k-1}{k}\}$$

m different hash value,

Counting distinct elements

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

y_1, y_2, y_3, y_1, y_4, y_2, y_1, y_4, y_1, y_2, y_5

h(32) h(12) ...

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

Assumption: For distinct values in U, the function maps to iid (independent and identically distributed) Unif(0,1) random numbers.

Important: if you were to feed in two equivalent elements, the function returns the **same** number.

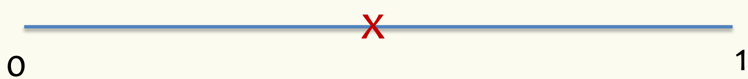
- So m distinct elements → m iid uniform y_i's

Min of IID Uniforms

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

In general, $\boxed{\mathrm{E}[\min(Y_1, \cdots, Y_m)]} = \ ?$

$m = 1$



$\mathrm{E}[\min(Y_1)] = E(Y_1) = \frac{1}{2}$

Min of IID Uniforms

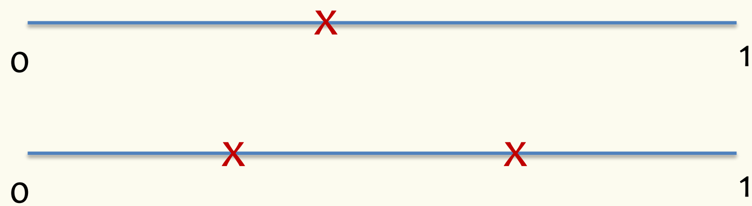
If $Y_1, \cdots, Y_m$ are iid Unif(0,1), where do we expect the points to end up?

In general, $\mathrm{E}[\min(Y_1, \cdots, Y_m)] = ?$

$$\mathrm{E}[\min(Y_1)] = \frac{1}{2}$$

$m = 1$

$m = 2$



$\mathrm{E}[\min(Y_1, Y_2)] = ?$ $\frac{1}{3}$

Min of IID Uniforms

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

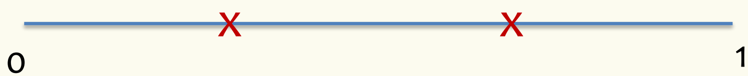
In general, $\mathrm{E}[\min(Y_1, \cdots, Y_m)] = ?$

$$\mathrm{E}[\min(Y_1)] = \frac{1}{2}$$

$m = 1$

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

$m = 2$



$m = 4$

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

Min of IID Uniforms

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

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

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

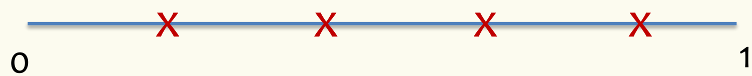
$m = 1$

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

$m = 2$

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

$m = 4$

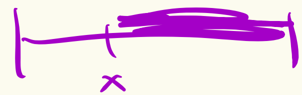


If $Y_1, \cdots, Y_m$ are iid Unif(0,1), then $E[\underbrace{\min(Y_1, \cdots, Y_m)}_{X}] = \frac{1}{m+1}$

$$E(X) = \int_0^1 x \underbrace{f_X(x)} \, dx$$

want to compute $F_X(x) \longrightarrow \frac{d}{dx}$ to get.

$$\Pr(\min(Y_1, \ldots, Y_m) > x)$$



$$= \Pr(Y_1 > x, Y_2 > x, \ldots, Y_m > x)$$

$$\underset{\text{indep}}{=} \underline{\Pr(Y_1 > x)} \Pr(Y_2 > x) \cdots \Pr(Y_m > x)$$

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

$$F_X(x) = \Pr(\min\ldots \le x) = 1 - (1-x)^m$$

$$f_X(x) = \frac{d}{dx} F_X(x) = +m(1-x)^{m-1}$$

$$E(X) = \int_0^1 x \, m(1-x)^{m-1} \, dx = \frac{1}{m+1}$$

Counting distinct elements

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

$y_1, y_2, y_3, y_1, y_4, y_2, y_1, y_4, y_1, y_2, y_5$

Hash function $h: U \rightarrow [0,1]$ (hashes to a uniform value).

- So m distinct elements → m iid uniform values.

$$val = \min\left(h(X_1), \cdots, h(X_N)\right) = \min(Y_1, \cdots, Y_m)$$

$$E(val) = \frac{1}{m+1}$$

A super duper clever idea!!!!!

If $Y_1, \cdots, Y_n$ are iid Unif(0,1), where do we expect the points to end up?

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

$$\text{Idea: m} = \frac{1}{\mathrm{E}[\min(Y_1,\cdots,Y_m)]} - 1$$

A super duper clever idea!!!!!

If $Y_1, \cdots, Y_n$ are iid Unif(0,1), where do we expect the points to end up?

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

Idea: $\mathrm{m} = \frac{1}{\mathrm{E}[\min(Y_1,\cdots,Y_m)]} - 1$

Let's keep track of the value val of min of hash values,

and estimate m as Round $\left(\frac{1}{val} - 1\right)$

The Distinct Elements Algorithm

Algorithm 2 Distinct Elements Operations

function INITIALIZE()
  val $\leftarrow \infty$
function UPDATE(x)
  val $\leftarrow$ min {val, hash(x)}
function ESTIMATE()
  return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: **do** ▷ Loop through all stream elements
  update(x_i) ▷ Update our single float variable
return estimate() ▷ An estimate for n, the number of distinct elements.

Distinct Elements Example

val= $\cancel{\infty}$ $\cancel{0.51}$ 0.26

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

Hashes: 0.51 0.26 0.79 0.26 0.79 0.79

Algorithm 2 Distinct Elements Operations

> **function** INITIALIZE()
>   val $\leftarrow \infty$
> **function** UPDATE(x)
>   val $\leftarrow$ min {val, hash(x)}
> **function** ESTIMATE()
>   **return** round $\left(\frac{1}{\text{val}} - 1\right)$
> **for** $i = 1, \ldots, N$: **do** ▷ Loop through all stream elements
>   update(x_i) ▷ Update our single float variable
> **return** estimate() ▷ An estimate for n, the number of distinct elements.

Suppose that

$\boldsymbol{h(13) = 0.51}$

$\boldsymbol{h(25) = 0.26}$

$\boldsymbol{h(19) = 0.79}$

Distinct Elements Example

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

Hashes:

Algorithm 2 Distinct Elements Operations

function INITIALIZE()
  val $\leftarrow \infty$
function UPDATE(x)
  val $\leftarrow$ min {val, hash(x)}
function ESTIMATE()
  return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: **do** ▹ Loop through all stream elements
  update(x_i) ▹ Update our single float variable
return estimate() ▹ An estimate for n, the number of distinct elements.

val = infty

Distinct Elements Example

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

Hashes: 0.51,

Algorithm 2 Distinct Elements Operations

function INITIALIZE()
  val $\leftarrow \infty$
function UPDATE(x)
  val $\leftarrow$ min {val, hash(x)}
function ESTIMATE()
  return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: **do** ▹ Loop through all stream elements
  update(x_i) ▹ Update our single float variable
return estimate() ▹ An estimate for n, the number of distinct elements.

val = infty

Distinct Elements Example

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

Hashes: 0.51,

Algorithm 2 Distinct Elements Operations

function INITIALIZE()
  val $\leftarrow \infty$
function UPDATE(x)
  val $\leftarrow$ min {val, hash(x)}
function ESTIMATE()
  return round $\left(\frac{1}{\text{val}} - 1\right)$

for $i = 1, \ldots, N$: **do** ▹ Loop through all stream elements
  update(x_i) ▹ Update our single float variable
return estimate() ▹ An estimate for n, the number of distinct elements.

val = 0.51

Distinct Elements Example

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

Hashes: 0.51, 0.26,

Algorithm 2 Distinct Elements Operations

```
function INITIALIZE()
    val ← ∞
function UPDATE(x)
    val ← min {val, hash(x)}
function ESTIMATE()
    return round(1/val − 1)
for i = 1, . . . , N: do                ▹ Loop through all stream elements
    update(x_i)                         ▹ Update our single float variable
return estimate()     ▹ An estimate for n, the number of distinct elements.
```

val = 0.26

Distinct Elements Example

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

Hashes: 0.51, 0.26, 0.79,

Algorithm 2 Distinct Elements Operations

function INITIALIZE()
  val $\leftarrow \infty$
function UPDATE(x)
  val $\leftarrow$ min {val, hash(x)}
function ESTIMATE()
  return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: **do** ▷ Loop through all stream elements
  update(x_i) ▷ Update our single float variable
return estimate() ▷ An estimate for n, the number of distinct elements.

val = 0.26

Distinct Elements Example

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

Hashes: 0.51, 0.26, 0.79, 0.26,

Algorithm 2 Distinct Elements Operations

function INITIALIZE()
  val $\leftarrow \infty$
function UPDATE(x)
  val $\leftarrow$ min {val, hash(x)}
function ESTIMATE()
  return round $\left(\frac{1}{\text{val}} - 1\right)$

for $i = 1, \ldots, N$: **do** ▷ Loop through all stream elements
  update(x_i) ▷ Update our single float variable
return estimate() ▷ An estimate for n, the number of distinct elements.

val = 0.26

Distinct Elements Example

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

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

Algorithm 2 Distinct Elements Operations

function INITIALIZE()
  val $\leftarrow \infty$
function UPDATE(x)
  val $\leftarrow$ min {val, hash(x)}
function ESTIMATE()
  return round $\left(\frac{1}{\text{val}} - 1\right)$

for $i = 1, \ldots, N$: **do** ▹ Loop through all stream elements
  update(x_i) ▹ Update our single float variable
return estimate() ▹ An estimate for n, the number of distinct elements.

val = 0.26

Distinct Elements Example

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

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

Algorithm 2 Distinct Elements Operations

function INITIALIZE()
  val $\leftarrow \infty$
function UPDATE(x)
  val $\leftarrow$ min {val, hash(x)}
function ESTIMATE()
  return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: **do** ▹ Loop through all stream elements
  update(x_i) ▹ Update our single float variable
return estimate() ▹ An estimate for n, the number of distinct elements.

val = 0.26

Distinct Elements Example

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

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

Algorithm 2 Distinct Elements Operations

function INITIALIZE()
  val ← ∞
function UPDATE(x)
  val ← min {val, hash(x)}
function ESTIMATE()
  return $\boxed{\text{round}\left(\frac{1}{\text{val}} - 1\right)}$
for $i = 1, \ldots, N$: **do** ▷ Loop through all stream elements
  update(x_i) ▷ Update our single float variable
return estimate() ▷ An estimate for n, the number of distinct elements.

val = 0.26

Return

round(1/0.26 - 1) =

round(2.846) = 3

Diy: Distinct Elements Example II

$\boxed{\text{val}}$

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

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

Algorithm 2 Distinct Elements Operations

function INITIALIZE()
  val $\leftarrow \infty$
function UPDATE(x)
  val $\leftarrow$ min {val, hash(x)}
function ESTIMATE()
  return round $\left(\frac{1}{\text{val}} - 1\right)$
for $i = 1, \ldots, N$: **do** ▹ Loop through all stream elements
  update(x_i) ▹ Update our single float variable
return estimate() ▹ An estimate for n, the number of distinct elements.

val = 0.1

Return= 9

round $\left(\frac{1}{0.1} - 1\right)$

Problem

$\text{val} = \min(Y_1, \cdots, Y_m)$

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

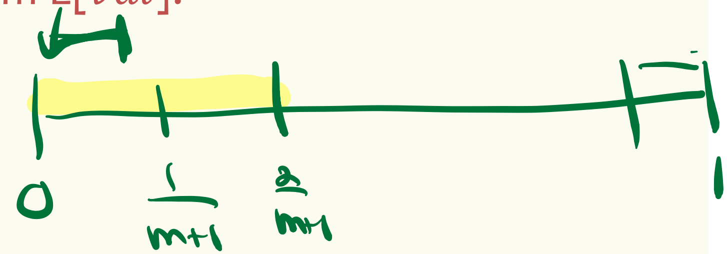
Algorithm:

Track $val = \min\big(h(X_1), \cdots, h(X_N)\big) = \min(Y_1, \cdots, Y_m)$

estimate m = 1/ val -1

But, val is not $\mathrm{E}[val]$! How far is val from $\mathrm{E}[val]$?

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



Problem

$\text{val} = \min(Y_1, \cdots, Y_m)$ $\quad E[val] = \frac{1}{m+1}$

Algorithm:

Track $val = \min\big(h(X_1), \cdots, h(X_N)\big) = \min(Y_1, \cdots, Y_m)$

estimate m = 1/ val -1

But, val is not E[val]! How far is val from E[val]?

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

What can we do to fix this?

How can we reduce the variance?

Idea: Repetition to reduce variance!

indep.

How can we reduce the variance?

Idea: Repetition to reduce variance!

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

Keep track of k independent min hash values

$$val^1 = \min\left(h^1(x_1), \cdots, h^1(x_N)\right) = \min(Y_1^1, \cdots, Y_m^1)$$

$$val^2 = \min\left(h^2(x_1), \cdots, h^2(x_N)\right) = \min(Y_1^2, \cdots, Y_m^2)$$

... ...

$$val^k = \min\left(h^k(x_1), \cdots, h^k(x_N)\right) = \min(Y_1^k, \cdots, Y_m^k)$$

$$\boxed{val = \frac{1}{k}\Sigma_i val_i,}$$ Estimate $m = \frac{1}{val} - 1$

$Var(aX) = a^2 Var(X)$

$$E(val) = \frac{1}{k}\sum_{i=1}^{k} E(val_i) = \frac{1}{k}\boxed{\sum_{i=1}^{k} \frac{1}{m+1}} = \frac{1}{\cancel{k}} \cdot \cancel{k} \cdot \frac{1}{m+1}$$

$$\mathrm{Var}(val) = \mathrm{Var}\left(\frac{1}{K}\sum_i val_i\right) = \frac{1}{K^2}\mathrm{Var}\left(\sum_i val_i\right)$$

$$= \frac{1}{K^2}\left[K \cdot \mathrm{Var}(val_1)\right]$$

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

$$= \frac{1}{K}\mathrm{Var}(val_1)$$

$h : U \to \{0, 1, \ldots, m-1\}$

Construct a family $\mathcal{H}$ of hash fns

$\mathcal{H} = \{ \underline{h_0, h_1, h_2, \ldots, h_{m-1}} \}$

pick $h \in \mathcal{H}$ u.a.r.

$i \in \{0, 1, \ldots, m-1\}$

$$\boxed{\forall x \in U \qquad \Pr(h(x) = i) = \frac{1}{m}}$$

$h_i(x) = i \qquad \forall x \in U$

$\forall x \in U, \ \forall y \in U$

$\forall \ 0 \le i \le m-1$

$\quad 0 \le j \le m-1$

$$\boxed{\Pr(h(x) = i, \underline{h(y) = j}) = \frac{1}{m^2}}$$

choice of $h \in \mathcal{H}$