Distinct elements

Anna Karlin

With many slides by Luxi Wang, Shreya Jayaraman,
Alex Tsun and Jeff Ullman

Data mining

- 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)

Stream model

- 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?

Problem

- 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"

Counting distinct elements

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

x_1 x_2 x_3 x_4 ....

output $m = 5$

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,

- Want to compute number of **distinct** keys in the stream.
- *How to do this without storing all the elements?*
- *Yet another super cool application of probability (and hashing)*

A naive solution, counting!

Store the m **distinct** user IDs in a hash table.

Space requirement: O(m)

Considering the number of users of YouTube, and the number of videos on YouTube, this is not feasible.

Consider a hash function $h : \mathcal{U} \to [0, 1]$
For distinct values in $\mathcal{U}$, the function maps to iid (independent and identically distributed) Unif(0,1) random numbers.

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

$h(32) = 0.43$
$h(12) = 0.19$
$h(14) = 0.85$
$h(7) = 0.61$

x_1 x_2 x_3 ...

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

0.43 0.19 0.85 0.43 ...

$\boxed{val = 0.19}$

Track minimum hash value seen so far

$$\min\left(h(x_1), \dots, h(x_N)\right)$$

in this ex. 0.19

$$= \min\left(U_1, U_2, \dots, U_m\right)$$

$U_i \sim U[0,1]$

m # distinct elts

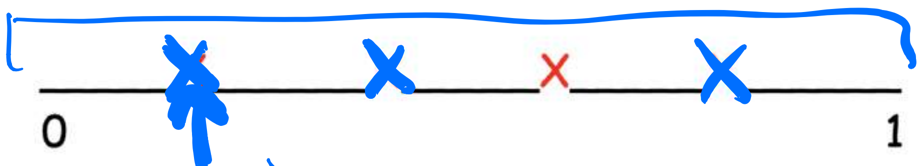
Min of IID Uniforms

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

$\boxed{m = 1}$ $E(Y_1)$

$\boxed{m = 2}$ $E(\min(Y_1, Y_2)) = \frac{1}{3}$

$m = 4$



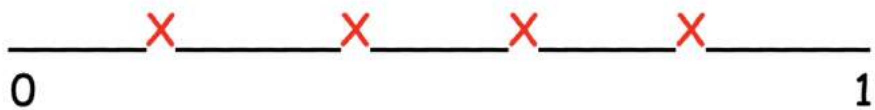
$E(\min(Y_1, Y_2, Y_3, Y_4)) = \frac{1}{5}$

Min of IID Uniforms

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

$$E[\min\{Y_1, \ldots, Y_4\}] = \frac{1}{4+1} = \frac{1}{5}$$

$m = 4$



Min of IID Uniforms

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

$$\boxed{E[\min(Y_1, \ldots, Y_m)] = \frac{1}{m+1} \qquad Y_i \text{: iid } U(0,1)}$$

- proved in section in book
- section worksheets
- CC today

A SUPER DUPER CLEVER IDEA

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

m # distinct els.

val

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

$$\equiv \quad m+1 = \frac{1}{E(\text{val})} \implies \boxed{m = \frac{1}{E(\text{val})} - 1}$$

What if $\boxed{\text{val} \approx E(\text{val})}$

estimate m to be $\frac{1}{\text{val}} - 1$

rounded to nearest int (floor)

$h(32) = 0.43$
$h(5) = 0.19$
$h(17) = 0.85$
$h(14) = 0.61$

x_1 x_2 x_3 x_N

32 5 17 32 14 5 32 32 17

$\min_{1 \le i \le N} h(x_i) = 0.19$ (val)

$\hat{m} = \text{round}\Big(\underbrace{\frac{1}{0.19} - 1}_{4.26}\Big)$

estimate of # distinct elts

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.

val tracks

$\min\left(h(x_1), \ldots, h(x_N)\right)$

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 = infty`

Distinct Elements Example

```
Stream:  13,   25,   19,   25,   19,   19
          h
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 = infty`

Distinct Elements Example

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

h

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 = 0.51

Distinct Elements Example

```
Stream:   13,   25,   19,   25,   19,   19
                 h
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

h

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
                              |
                              h
                              |
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, $\boxed{19}$, 19

h

Hashes: 0.51, 0.26, 0.79, 0.26, $\boxed{0.79}$, 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, $\boxed{19}$

$$\boxed{19} \xrightarrow{h} \boxed{0.79}$$

Hashes: 0.51, 0.26, 0.79, 0.26, 0.79, $\boxed{0.79}$

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.
```

$val_1, \ldots, val_k$

$val_i := \min(val_i, h_i(x))$

using $\widehat{val}$

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

val = 0.26

h: maps $\mathcal{U}$ to $\mathbb{R}$

random value in $(0,1)$

Distinct Elements Example

Stream: 13, 25, 19, 25, 19, $\boxed{19}$

$\downarrow$ h $\downarrow$

Hashes: 0.51, 0.26, 0.79, 0.26, 0.79, $\boxed{0.79}$

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

Return
round(1/0.26 - 1) =
round(2.846) =
3

DIY: Distinct Elements Example II

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

SUMMARY SO FAR

$h : \underset{\text{universe}}{U} \rightarrow [0,1)$

Assume h is really great in that $\forall x \leftarrow U$ $h(x) \sim U(0,1)$ indep.

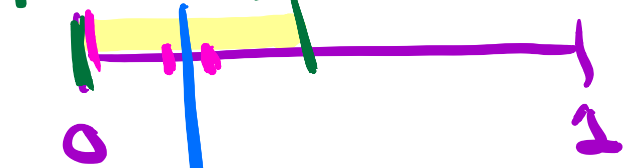
track $val := \min_{1 \le i \le N} h(x_i)$

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

$h' : U \rightarrow \{0, 1, \ldots, H-1\}$

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

m distinct elts $x_1, \dots, x_N$



PROBLEM

Is val likely to close to E(val)?

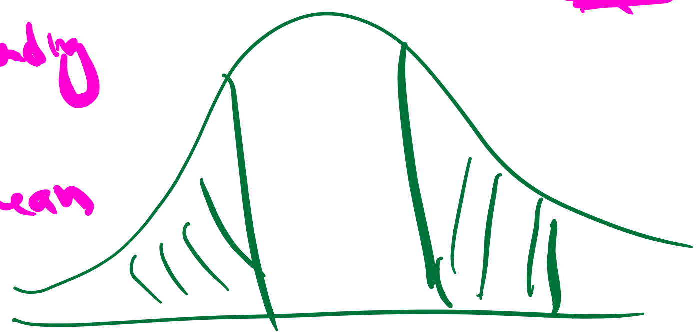
$$\mathrm{Var}(val) \approx \frac{1}{(m+1)^2}$$

$$\sigma(val) \approx \frac{1}{m+1}$$

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

$$\hat{m} = \mathrm{round}\left(\frac{1}{val} - 1\right)$$

for many random vars including val, likely to be > 0.999σ away from mean



How can we reduce the variance?

Repetition

Let $h_1, \ldots, h_k$ k indep hash fns.

$h_i(x_j) \sim U(0,1)$ [assume all of these indep]

track

$val_1 = \min(h_1(x_1) \ldots, h_1(x_N)) = \min(U_1^1, \ldots, U_m^1)$

$val_2 = \min(h_2(x_1) \ldots h_2(x_N)) = \min(U_1^2, \ldots, U_m^2)$

$\vdots$

$val_k = \min(h_k(x_1) \ldots, h_k(x_N)) = \min(U_1^k \ldots U_m^k)$

$$\hat{val} = \frac{1}{K}\left(val_1 + val_2 + \dots + val_K\right)$$

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

$$E(\hat{val}) = E\left(\frac{1}{K}(val_1 + val_2 + \dots + val_K)\right)$$

$$= \frac{1}{K}\Big[\underbrace{E(val_1)}_{\frac{1}{m+1}} + \underbrace{E(val_2)}_{\frac{1}{m+1}} + \dots E(val_K)\Big]$$

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

$$Var(\hat{val}) = Var\left(\frac{1}{K}(val_1 + \dots + val_K)\right)$$

$$= \frac{1}{K^2} Var(val_1 + \dots + val_K)$$

$$\underset{\text{indep}}{=} \frac{1}{K^2}\Big[\underbrace{Var(val_1)}_{\approx\frac{1}{(m+1)^2}} + \dots + \underbrace{Var(val_K)}\Big]$$

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

$E(\hat{val})$

a) $\boxed{\frac{1}{m+1}}$

b) $\frac{1}{K(m+1)}$

c) $\frac{K}{(m+1)}$

d) I don't know

$Var(X)$

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

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

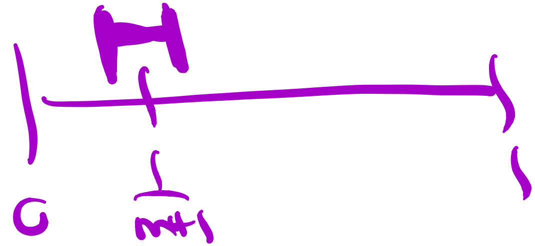
c) $\boxed{\approx \frac{1}{K(m+1)^2}}$

d) I don't know

$$= \frac{1}{k^2} \; k \cdot \frac{1}{(m+1)^2} = \frac{1}{k(m+1)^2}$$

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

$$\sigma(\widehat{val}) = \frac{1}{\sqrt{k}\,(m+1)}$$



$$val_i = \min\left(h(x_1), \ldots, h(x_N)\right)$$

$x_i = x_j$

$h(x_i) = h(x_j)$

$x \ldots, x_N$

$$= \min\left(U_1, \ldots, U_m\right)$$

$U_1 \downarrow h(x_{i_1})$, $\quad U_m \downarrow h(x_{i_m})$

m distinct elts

$x_{i_1}, \ldots, x_{i_m}$

$$E(val) = E\left[\min(U_1, \ldots, U_m)\right] = \frac{1}{m+1}$$

Coding on pset 6

You will use a hash function $h : \mathcal{U} \to [0,1]$
For distinct values in $\mathcal{U}$, the function maps to iid (independent and identically distributed) Unif(0,1) random numbers.

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

We will implement the hash function for you! Just know that you can consider it an iid uniform continuous random variables for each of the values being hashed.

To do better...

1. we will keep track of K DistElts classes each with its own independent hash function
2. take the mean of our K mins to get a better estimate of the min
3. and then apply the same trick as earlier to give an estimate for the number of distinct elements based on this min that we saw.

Joint Distributions

Anna Karlin
Most slides by Alex Tsun

5.1 Joint Discrete Distributions

Agenda

- Motivation
- Cartesian Products of Sets
- Joint PMFs and Expectation
- Marginal PMFs

Naive Bayes Classifier - What we Calculate

$$\mathbb{P}(\text{spam} \mid \text{"You buy Viagra!"}) = \frac{\mathbb{P}(\text{"You buy Viagra!"} \mid \text{spam})\ \mathbb{P}\ (\text{spam})}{\mathbb{P}(\text{"You buy Viagra!"})}$$

$$= \frac{\mathbb{P}(\{\text{"you"},\text{"buy"},\text{"viagra"}\} \mid \text{spam})\ \mathbb{P}(\text{spam})}{\mathbb{P}(\{\text{"you"},\text{"buy"},\text{"viagra"}\} \mid \text{spam})\ \mathbb{P}(\text{spam}) + \mathbb{P}(\{\text{"you"},\text{"buy"},\text{"viagra"}\} \mid \text{ham})\ \mathbb{P}(\text{ham})}$$ [LTP]

Naive Bayes Classifier - The naive part

$$\mathbb{P}(\{\text{"you"}, \text{"buy"}, \text{"viagra"}\} \mid \text{spam})$$
$$\approx \mathbb{P}(\text{"you"} \mid \text{spam})\mathbb{P}(\text{"buy"} \mid \text{spam})\mathbb{P}(\text{"viagra"} \mid \text{spam})$$

Why is this Naive?

"!!!Lunch free for You. Viagra included. $$$!!!!!"

Ubiquitous in ML

$Pr(\underbrace{x_1=35, BP=95, \ldots, disease=No})$

- Given "labeled data" X_1 X_2 X_3 X_5

Temp.	BP.	Sore Throat	...	Colour	diseaseX
35	95	Y	...	Pale	No
22	110	N	...	Clear	Yes
:	:			:	:
10	87	N	...	Pale	No

Learner

- Learn CLASSIFIER, that can predict label of *NEW* instance

Temp	BP	Sore-Throat	...	Color	diseaseX
32	90	N	...	Pale	?

Classifier

diseaseX
No