CSE 312
Foundations of Computing II

Lecture 19: Application -- Distinct elements

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?
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

$N = \# \text{ of IDs in the stream} = 11, \quad m = \# \text{ of distinct IDs in the stream} = 5$

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
We will use a hash function \( h: U \rightarrow [0,1] \).

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

\[
\tilde{h}: U \rightarrow \{0, 1, \ldots, k-1\}, \hspace{1cm} h(x) = \frac{i}{k}
\]
Counting distinct elements

Hash function \( h: U \to [0,1] \)
Assumption: For distinct values in \( U \), the function maps to iid (independent and identically distributed) \( \text{Unif}(0,1) \) random numbers.

Important: if you were to feed in two equivalent elements, the function returns the same number.
• So \( m \) distinct elements \( \Rightarrow m \) iid uniform \( y_i \)'s
Min of IID Uniforms

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

In general, $E[\min(Y_1, \ldots, Y_m)] = \frac{1}{2}$

$$E[\min(Y_1)] = E(Y_1) = \frac{1}{2}$$
Min of IID Uniforms

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

In general, $E[\min(Y_1, \ldots, Y_m)] = ?$

$m = 1$

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

$m = 2$

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

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

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

- $m = 4$
  
  \[ E[\min(Y_1, \cdots, Y_4)] = \frac{1}{5} \]
Min of IID Uniforms

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

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

- $m = 1$
  - $E[\min(Y_1)] = \frac{1}{1+1} = \frac{1}{2}$
  - $X_0 = 1$

- $m = 2$
  - $E[\min(Y_1, Y_2)] = \frac{1}{2+1} = \frac{1}{3}$
  - $X_0 = X_1$

- $m = 4$
  - $E[\min(Y_1, \ldots, Y_4)] = \frac{1}{4+1} = \frac{1}{5}$
  - $X_0 = X_1 = X_2 = X_3 = X_4$
If $Y_1, \cdots, Y_m$ are iid Unif(0,1), then $E[\min(Y_1, \cdots, Y_m)] = \frac{1}{m+1}$.

$E(X) = \int_0^x f_X(x) \, dx$

want to compute $F_X(x) \rightarrow \frac{d}{dx}$ together.

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

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

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

$= (1-x)^m$  \hspace{1cm} $F_X(x) = Pr(\min \leq x) = 1 - (1-x)^m$

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

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

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

Hash function $h: U \rightarrow [0,1]$ (hashes to a uniform value).
- So $m$ distinct elements $\Rightarrow m$ iid uniform values.

$$val = \min(h(X_1), \ldots, h(X_N)) = \min(Y_1, \ldots, Y_m)$$

$$E(val) = \frac{1}{m+1}$$
A super duper clever idea!!!!!

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

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

Idea: $m = \frac{1}{E[\min(Y_1, \ldots, Y_m)]} - 1$
A super duper clever idea!!!!

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

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

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

Let’s keep track of the value $\text{val}$ of min of hash values, and estimate $m$ as $\text{Round}(\frac{1}{\text{val}} - 1)$
The Distinct Elements Algorithm

Algorithm 2 Distinct Elements Operations

function INITIALIZE()
    val ← ∞

function UPDATE(x)
    val ← min {val, hash(x)}

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

for i = 1, \ldots, N: do
    update(x_i)

return estimate()

▷ Loop through all stream elements
▷ Update our single float variable
▷ An estimate for n, the number of distinct elements.
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 round \left( \frac{1}{val} - 1 \right) \right)
for i = 1, . . . , N: do
    update(x_i) ▶ Loop through all stream elements
return estimate() ▶ Update our single float variable
    ▶ An estimate for n, the number of distinct elements.

Suppose that
h(13) = 0.51
h(25) = 0.26
h(19) = 0.79
Distinct Elements Example

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

Hashes:

\[
\text{val} = \infty
\]

---

**Algorithm 2** Distinct Elements Operations

```plaintext
function INITIALIZE()
    val ≜ \infty

function UPDATE(x)
    val ≜ \min\{val, \text{hash}(x)\}

function ESTIMATE()
    return round\left(\frac{1}{val} - 1\right)

for i = 1, \ldots, N: do
    update(x_i)
    ▶ Update our single float variable
return estimate()
    ▶ An estimate for \(n\), the number of distinct elements.
```
Distinct Elements Example

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

Hashes: 0.51,
Distinct Elements Example

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

Hashes: 0.51,

```
Algorithm 2 Distinct Elements Operations

function INITIALIZE()
  val ← ∞

function UPDATE(x)
  val ← min {val, hash(x)}

function ESTIMATE()
  return round \left( \frac{1}{\text{val}} - 1 \right) \right.

for i = 1, \ldots, N: do
  update(x_i)  \text{  \small (Update our single float variable)}
return estimate()  \text{  \small (An estimate for } n, \text{ 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 \left( \frac{1}{val} - 1 \right)

for i = 1, \ldots, N: do
    update(x_i)
    ▷ Loop through all stream elements
    ▷ 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, \textbf{0.79},

\begin{algorithm}
\begin{verbatim}
Algorithm 2 Distinct Elements Operations

function INITIALIZE()
    val ← ∞

function UPDATE(x)
    val ← min \{val, hash(x)\}

function ESTIMATE()
    return round \(\frac{1}{val} - 1\)

for i = 1, \ldots, N: do
    update(\(x_i\)) \quad \triangleright \text{Update our single float variable}

return estimate() \quad \triangleright \text{An estimate for } n, \text{ the number of distinct elements.}
\end{verbatim}
\end{algorithm}

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 ← ∞

function UPDATE(x)
    val ← min {val, hash(x)}

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

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

function UPDATE(x)
    val ← min {val, hash(x)}

function ESTIMATE()
    return round \left( \frac{1}{val} - 1 \right)

for i = 1, \ldots, N: do
    update(x_i)  
    \quad \triangleright \text{Update our single float variable}
return estimate()  
    \quad \triangleright \text{An estimate for } n, \text{ the number of distinct elements.}

\text{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

val = 0.26

Algorithm 2 Distinct Elements Operations

function INITIALIZE()
  val ← ∞

function UPDATE(x)
  val ← min {val, hash(x)}

function ESTIMATE()
  return round \left( \frac{1}{val} - 1 \right) \hspace{1cm} \triangleright \text{Loop through all stream elements}

for i = 1, \ldots, N: do
  update(x_i) \hspace{1cm} \triangleright \text{Update our single float variable}

return estimate() \hspace{1cm} \triangleright \text{An estimate for } n, \text{ the number of distinct elements.}
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 round \left( \frac{1}{\text{val}} - 1 \right)

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

val = 0.26

Return

dround(1/0.26 - 1) =
dround(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

val = 0.1
Return = 9
Problem

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

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

Algorithm:

Track $\text{val} = \min(h(X_1), \cdots, h(X_N)) = \min(Y_1, \cdots, Y_m)$

estimate $m = 1/\text{val} - 1$

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

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

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

Algorithm:
Track \( \text{val} = \min(h(X_1), \ldots, h(X_N)) = \min(Y_1, \ldots, Y_m) \)
estimate \( m = 1/\text{val} - 1 \)

But, \( \text{val} \) is not \( \text{E}[\text{val}] \)! How far is \( \text{val} \) from \( \text{E}[\text{val}] \)?

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

What can we do to fix this?
How can we reduce the variance?

Idea: Repetition to reduce variance!
How can we reduce the variance?

**Idea: Repetition to reduce variance!**

Use \( k \) independent hash functions \( h^1, h^2, \ldots, h^k \)

Keep track of \( k \) independent min hash values

\[
val^1 = \min(h^1(x_1), \ldots, h^1(x_N)) = \min(Y_1, \ldots, Y_m)
\]

\[
val^2 = \min(h^2(x_1), \ldots, h^2(x_N)) = \min(Y_1, \ldots, Y_m)
\]

\[
\vdots
\]

\[
val^k = \min(h^k(x_1), \ldots, h^k(x_N)) = \min(Y_1, \ldots, Y_m)
\]

\[
val = \frac{1}{k} \sum_{i=1}^{k} val_i,
\]

Estimate \( m = \frac{1}{val} - 1 \)

\[
E(val) = \frac{1}{k} \sum_{i=1}^{k} E(val_i) = \frac{1}{k} \sum_{i=1}^{k} \frac{k}{m+1} = \frac{1}{m+1} \cdot \frac{1}{m}
\]
\[
\text{Var}(\text{val}) = \text{Var} \left( \frac{1}{k} \sum_{i=1}^{N} \text{val}_i \right) = \frac{1}{k^2} \text{Var}(\text{val}_i) \\
= \frac{1}{k^2} \left[ k \cdot \text{Var}(\text{val}_i) \right]
\]

\[
= \frac{1}{k^2} \cdot \frac{1}{k} \cdot \text{Var}(\text{val}_i)
\]
Construct a family \( \mathcal{H} \) of hash functions

\[ \mathcal{H} = \{ h_0, h_1, h_2, \ldots, h_m \} \]

Pick \( h \in \mathcal{H} \) uniformly at random.

For all \( x \in \mathcal{U} \),

\[ \Pr( h(x) = i ) = \frac{1}{m} \]

For all \( x \neq 0 \),

\[ h_i(x) = i \]

For all \( x, y \in \mathcal{U} \),

\[ \Pr( h(x) = i, h(y) = j ) = \frac{1}{m^2} \]

Choose \( h \in \mathcal{H} \).