Review CLT

**Theorem. (Central Limit Theorem)** \(X_1, \ldots, X_n\) i.i.d. with mean \(\mu\) and variance \(\sigma^2\). Let \(Y_n = \frac{X_1 + \cdots + X_n - n\mu}{\sigma \sqrt{n}}\). Then,

\[
\lim_{n \to \infty} Y_n \to \mathcal{N}(0,1)
\]

One main application:

Use Normal Distribution to Approximate \(Y_n\)

No need to understand \(Y_n\) !!
Agenda

- Continuity correction
- Application: Counting distinct elements
Example – $Y_n$ is binomial

We understand binomial, so we can see how well approximation works

We flip $n$ independent coins, heads with probability $p = 0.75$.

$X = \# \text{heads} \quad \mu = \mathbb{E}(X) = 0.75n \quad \sigma^2 = \text{Var}(X) = p(1 - p)n = 0.1875n$

$\mathbb{P}(X \leq 0.7n)$

<table>
<thead>
<tr>
<th>$n$</th>
<th>exact</th>
<th>$\mathcal{N}(\mu, \sigma^2)$ approx</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.474407</td>
<td>0.357500327</td>
</tr>
<tr>
<td>20</td>
<td>0.382827</td>
<td>0.302788308</td>
</tr>
<tr>
<td>50</td>
<td>0.251919</td>
<td>0.207108089</td>
</tr>
<tr>
<td>100</td>
<td>0.149541</td>
<td>0.124106539</td>
</tr>
<tr>
<td>200</td>
<td>0.062472</td>
<td>0.051235217</td>
</tr>
<tr>
<td>1000</td>
<td>0.000193</td>
<td>0.000130365</td>
</tr>
</tbody>
</table>
Example – Naive Approximation

Fair coin flipped (independently) 40 times. Probability of 20 or 21 heads?

**Exact.** \( \mathbb{P}(X \in \{20, 21\}) = \binom{40}{20} + \binom{40}{21} \left(\frac{1}{2}\right)^{40} \approx 0.2448 \)

**Approx.** \( X = \# \text{ heads} \) \( \mu = \mathbb{E}(X) = 0.5n = 20 \) \( \sigma^2 = \text{Var}(X) = 0.25n = 10 \)

\[ \mathbb{P}(20 \leq X \leq 21) = \Phi\left( \frac{20 - 20}{\sqrt{10}} \leq \frac{X - 20}{\sqrt{10}} \leq \frac{21 - 20}{\sqrt{10}} \right) \]

\[ \approx \Phi\left( 0 \leq \frac{X - 20}{\sqrt{10}} \leq 0.32 \right) \]

\[ = \Phi(0.32) - \Phi(0) \approx 0.1241 \]
Example – Even Worse Approximation

Fair coin flipped (independently) **40** times. Probability of **20** heads?

**Exact.** \( P(X = 20) = \binom{40}{20} \left(\frac{1}{2}\right)^{40} \approx 0.1254 \)

**Approx.** \( P(20 \leq X \leq 20) = 0 \)

Interval length = 0
Solution – Continuity Correction

Round to next integer!

To estimate probability that discrete RV lands in (integer) interval \( \{a, \ldots, b\} \), compute probability continuous approximation lands in interval \( [a - \frac{1}{2}, b + \frac{1}{2}] \).
Example – Continuity Correction

Fair coin flipped (independently) 40 times. Probability of 20 or 21 heads?

Exact. \( P(X \in \{20, 21\}) = \left[ \binom{40}{20} + \binom{40}{21} \right] \left( \frac{1}{2} \right)^{40} \approx 0.2448 \)

Approx. \( X = \# \text{ heads} \quad \mu = \mathbb{E}(X) = 0.5n = 20 \quad \sigma^2 = \text{Var}(X) = 0.25n = 10 \)

\( P(19.5 \leq X \leq 21.5) = \Phi \left( \frac{19.5 - 20}{\sqrt{10}} \leq \frac{X - 20}{\sqrt{10}} \leq \frac{21.5 - 20}{\sqrt{10}} \right) \)

\approx \Phi \left( -0.16 \leq \frac{X - 20}{\sqrt{10}} \leq 0.47 \right) \approx \Phi(-0.16) - \Phi(0.47) \approx 0.2452 \)
Example – Continuity Correction

Fair coin flipped (independently) 40 times. Probability of 20 heads?

**Exact.** \[ P(X = 20) = \binom{40}{20} \left(\frac{1}{2}\right)^{20} \approx 0.1254 \]

**Approx.** \[
P(19.5 \leq X \leq 20.5) = \Phi\left(\frac{19.5 - 20}{\sqrt{10}} \leq \frac{X - 20}{\sqrt{10}} \leq \frac{20.5 - 20}{\sqrt{10}}\right) \]

\[
\approx \Phi\left(-0.16 \leq \frac{X - 20}{\sqrt{10}} \leq 0.16\right) \]

\[
= \Phi(-0.16) - \Phi(0.16) \approx 0.1272
\]
Agenda

- Continuity correction
- Application: Counting distinct elements
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 $\Rightarrow$ 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

\[32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4\]

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

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:** \(\Omega(m)\)

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 number of distinct IDs in the stream.

*How to do this* without storing all the elements?
Detour – I.I.D. Uniforms

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

“Evenly spread out”

$m = 1$

$m = 2$

$m = 4$

What is some intuition for this?
If $Y_1, \ldots, Y_m \sim \text{Unif}(0,1)$ (i.i.d.) where do we expect the points to end up?

$m = 1$

\[ \begin{array}{ccccccc}
0 & & & & & & 1 \\
\end{array} \]

$Y_1$ has expected value $1/2$

... but probably isn’t very close to the middle

... and $Y_2$ is more likely to be in the bigger gap

$m = 2$

\[ \begin{array}{ccccccc}
0 & & & & & & 1 \\
\end{array} \]
Detour – Min of I.I.D. Uniforms

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

\( \text{e.g., what is } \mathbb{E}[\min\{Y_1, \ldots, Y_m\}]? \)

**CDF:** Observe that \( \min\{Y_1, \ldots, Y_m\} \geq y \) if and only if \( Y_1 \geq y, \ldots, Y_m \geq y \)

(Similar to Section 6)

\[
P(\min\{Y_1, \ldots, Y_m\} \geq y) = P(Y_1 \geq y, \ldots, Y_m \geq y) \\
= P(Y_1 \geq y) \cdots P(Y_m \geq y) \quad \text{(Independence)} \\
= (1 - y)^m \\
\Rightarrow P(\min\{Y_1, \ldots, Y_m\} \leq y) = 1 - (1 - y)^m
\]
Detour – Min of I.I.D. Uniforms

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

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

**Proof** (Not covered)

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

$$
= \int_0^\infty \int_y^\infty f_Y(x) \, dx \, dy = \int_0^\infty P(Y \geq y) \, dy
$$
Detour – Min of I.I.D. Uniforms

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

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

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

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

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

$Y = \min\{Y_1, \ldots, Y_m\}$
Detour – Min of I.I.D. Uniforms

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

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

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

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

- $m = 4$
  
  $\mathbb{E}[\min(Y_1, \ldots, Y_4)] = \frac{1}{4+1} = \frac{1}{5}$
Distinct Elements – Hashing into $[0, 1]$

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

**Assumption:** For all $x \in U$, $h(x) \sim \text{Unif}(0,1)$ and mutually independent

\[
\begin{align*}
x_1 &= 5 & x_2 &= 2 & x_3 &= 27 & x_4 &= 35 & x_5 &= 4 \\
h(5) &= h(2) & h(27) & h(35) & h(4)
\end{align*}
\]

5 distinct elements

$\to$ 5 i.i.d. RVs $h(x_1), \ldots, h(x_5) \sim \text{Unif}(0,1)$

$\to \mathbb{E}[\min\{h(x_1), \ldots, h(x_5)\}] = \frac{1}{5+1} = \frac{1}{6}$
Distinct Elements – Hashing into $[0, 1]$

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

**Assumption:** For all $x \in U$, $h(x) \sim \text{Unif}(0,1)$ and mutually independent

\[
x_1 = 5 \quad x_2 = 2 \quad x_3 = 27 \quad x_4 = 5 \quad x_5 = 4
\]

\[
h(5) \quad h(2) \quad h(27) \quad h(5) \quad h(4)
\]

4 distinct elements

$\Rightarrow$ 4 i.i.d. RVs $h(x_1), h(x_2), h(x_3), h(x_5) \sim \text{Unif}(0,1)$ and $h(x_1) = h(x_4)$

$\Rightarrow \mathbb{E}[\min\{h(x_1), \ldots, h(x_5)\}] = \mathbb{E}[\min\{h(x_1), h(x_2), h(x_3), h(x_5)\}] = \frac{1}{4+1}$
Distinct Elements – Hashing into $[0, 1]$ 

**Hash function** $h: U \rightarrow [0,1]$  
**Assumption:** For all $x \in U$, $h(x) \sim \text{Unif}(0,1)$ and mutually independent $h(A), h(A), \ldots, h(B)$ contains $F_i.i.d. \ rvs \sim \text{Unif}(0,1)$ and $N - m$ repeats

\[
\begin{align*}
    x_1, x_2, \ldots, x_N \text{ contains } m \text{ distinct elements} \\
h(x_1), h(x_2), \ldots, h(x_N) \text{ contains } m \text{ i.i.d. rvs } \sim \text{Unif}(0,1)
\end{align*}
\]

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

26
The MinHash Algorithm – Idea

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 $\text{round}\left(\frac{1}{\text{val}} - 1\right)$
The MinHash Algorithm – Implementation

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

\[
\text{val} \leftarrow \infty \\
\text{for } i = 1 \text{ to } N \text{ do} \\
\quad \text{val} \leftarrow \min\{\text{val}, h(x_i)\} \\
\text{return } \text{round}\left(\frac{1}{\text{val}} - 1\right)
\]

Memory cost = just remember val (with sufficient precision)
MinHash Example

Stream: 13, 25, 19, 25, 19, 19
Hashes: 0.51, 0.26, 0.79, 0.26, 0.79, 0.79

Poll: pollev.com/paulbeame028
a. 1
b. 3
c. 5
d. No idea

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 unlikely: $P(h(x) < 0.1) = 0.1$
The MinHash Algorithm – Problem

Algorithm MinHash\((x_1, x_2, ..., x_N)\)

\[
\text{val} \leftarrow \infty \\
\text{for } i = 1 \text{ to } N \text{ do} \\
\quad \text{val} \leftarrow \min\{\text{val}, h(x_i)\} \\
\text{return round}\left(\frac{1}{\text{val}} - 1\right)
\]

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

\[
\text{Var}(\text{val}) \approx \frac{1}{(m + 1)^2}
\]
How can we reduce the variance?

Idea: Repetition to reduce variance!
Use $k$ independent hash functions $h^1, h^2, \ldots h^k$

Algorithm $\text{MinHash}(x_1, x_2, \ldots, x_N)$

\[
\text{val}_1, \ldots, \text{val}_k \leftarrow \infty \\
\text{for } i = 1 \text{ to } N \text{ do} \\
\quad \text{val}_1 \leftarrow \min\{\text{val}_1, h^1(x_i)\}, \ldots, \text{val}_k \leftarrow \min\{\text{val}_k, h^k(x_i)\} \\
\text{val} \leftarrow \frac{1}{k} \sum_{i=1}^{k} \text{val}_i \\
\text{return round} \left( \frac{1}{\text{val}} - 1 \right)
\]

\[
\text{Var(Val)} = \frac{1}{k} \frac{1}{(m + 1)^2}
\]
MinHash and Estimating # of Distinct Elements in Practice

• MinHash in practice:
  – One also stores the element that has the minimum hash value for each of the $k$ hash functions
    • Then, just given separate MinHashes for sets $A$ and $B$, can also estimate
      – what fraction of $A \cup B$ is in $A \cap B$; i.e., how similar $A$ and $B$ are

• Another randomized data structure for distinct elements in practice:
  – HyperLoglog - even more space efficient but doesn’t have the set combination properties of MinHash