Sampling And Streaming

LD Input is too large to store/access

Plan

1) Sampling
2) Streaming Model
3) Reservoir Sampling
4) Distinct Elements
Sampling:

Poll: “Do you use daylight savings?”

YES = 1    NO = 0

Goal: Estimate the fraction of population voting YES

Algo: Sample \( k \) people at random

Collect their answers \( X_1, \ldots, X_k \in \{0,1\} \)

Output: \[ \hat{p} = \frac{1}{k} \sum_{i=1}^{k} X_i \]

\( \hat{p} \) = estimate of fraction of YES
Pick $k$ large enough
With prob. $1 - \delta (0.999)$

\[ \hat{p} - \text{true value} \mid p \leq 0.001 \]

**Question**

Suppose for 300 people, $k$ samples are needed to get 0.01 approximate estimate w.p. 0.999.

How many samples for 300 million people?

1) $k$ samples
2) 100 $k$ samples
3) 1000 $k$ samples
Suppose $X_1, \ldots, X_t$ are random variables that take \{0, 1\} values
so that each $\Pr[X_i = 1] = p$, $\Pr[X_i = 0] = 1 - p$.

\[
\Pr\left\{ \left| \frac{1}{t} \sum_{i=1}^{t} X_i - p \right| \geq \varepsilon \right\} \leq 2e^{-2\varepsilon^2 t}
\]

The average of samples expectation decreases exponentially in decaying number of samples.

To get: $\Pr[\text{deviation} \geq \varepsilon] \leq \delta$ you pick

\[
t = \frac{1}{2} \varepsilon^2 \log_e \frac{1}{2\delta}
\]
\[ X_i = \text{answer of } i\text{th person} \in \{0, 1\} \]

\[ P(X_i = 1) = \text{fraction of YES votes in population} \]

\[ = p \]
Streaming:

1) What fraction of cars are red?
2) How many cars travelled?
3) How many distinct cars travelled?

Data:
1) Too large to store
2) Comes in a stream,
3) No rewind.
Streaming Model

Input: a stream of $s_1, s_2, \ldots, s_n$...

$s_i \in \{1, \ldots, N\}$

1) read the stream only once, from left to right

2) don't know how long the stream is.

Goal:
Algorithm running using space $\text{poly}(\log N, \log n)$
Sampling from a Stream

**Input:** A stream $s_1, s_2, \ldots \in \mathbb{E}^k$. 

**Goal:** Pick one random element from it.

If $n = \text{length of stream is known}$

1) Pick a random index $i \in \{1, \ldots, n\}$

2) Output $s_i$
Reservoir Sampling

"Reservoir" \( r = 8 \),

for each element \( 8_i \):

- Pick random number \( q \in \{1, \ldots, i\} \)

\[ w = 0 \cdot 1/i \]

\[ g = i \]

\[ g + 1 \]

\[ w = w - 1/i \]

Throw away what's in reservoir \( r \), replace with \( 8_i \)

Ignore \( 8_i \)

Whenever stream stops, Output \( r \)
Claim: At end of $i$th step, $\forall j \leq i$

\[
\Pr \{ \text{reservoir} = 8j \} = \frac{1}{i}
\]

Proof: Induction:

Base case: $i = 1$. True.

Assume claim true for $i-1$ steps.

After $i$th step, consider $j = 1, \ldots, i - 1$.

\[
\Pr \{ \text{reservoir} = 8j \} = \Pr \{ \text{reservoir} = 8j \text{ after } i-1 \text{ steps} \} \cdot \Pr \{ 8j \text{ was not thrown out in } i\text{th step} \}
\]

\[
= \frac{1}{i} (1 - \frac{1}{i-1}) = \frac{1}{i}
\]
Pr(\text{reservoir} = 8i) = Pr(\text{replace reservoir with } 8i \text{ in the last step})

= \frac{1}{n}
Distinct Elements:

Input: A stream $S_1, S_2, \ldots, S_l, E\{1, N\}$

Goal: Estimate the number of distinct elements in stream $S$.
Algorithm (Ideal)

At the beginning, pick a random hash function

\[ h : \{ \ldots \} \rightarrow [0, 1] \]

by remembering only one number!

Compute (as the stream goes by)

\[ \text{Minimum}(h(s_1), h(s_2), \ldots, h(s_n)) \]

Output = \[ \frac{1}{\text{minimum}(h(s_1), \ldots, h(s_n))} \]
Intuition: $S_1, S_2, \ldots, S_n$

Suppose there are $k$ distinct elements in stream

$\rightarrow$ $1, 2, 1, 3, 5, \ldots, k$

All numbers

$\rightarrow$ minimum $(h(S_1), \ldots, h(S_n))$

$\uparrow$

$k$ different hash values.

$\{\text{Every hash value is uniformly random i.i.d. in } [0, 1]\}$
Proof: not in.
"Pseudorandom Hash Function"

**Want:** For each \( s_i \), \( h(s_i) \approx \text{uniformly random from } \{0,1\} \)

**Impossible**

**Expect:** \( h(1) \approx \text{uniformly random in } [0,1] \)

"Pairwise independent": \( \forall i, j \)

\( h(i), h(j) \approx \text{same distribution uniformly random as pair } (0,1) \)