Sampling And Streaming

↳ Input is too large to store/access

Plan

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

Sampling:

Poll: "Do you like daylight savings?"

YES = 1 NO = 0

Goal: Estimate the fraction of population voting YES

Algo: Sample K - people at random

Collect their answers $X_1 \ldots X_K \in \{0, 1\}$

$$\text{Output} = \frac{1}{K} \sum_{i=1}^{K} X_i = \hat{p} = \text{estimate of fraction of YES}$$

Pick k large enough

With prob. $1-\delta$ (0.999)

estimate $|\hat{p} - \text{true value } p| \leq \varepsilon \approx 0.001$

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) 10 k

2) 100 k samples

3) 1000 k samples

[Chernoff Bound]

Suppose $X_1 \ldots X_t$ are that take $\{0,1\}$ values

i.i.d (independent & identically random variables distributed)

so that each $P_r[X_i = 1] = p$ $P_r[X_i = 0] = 1-p$

$$P_r\left[\left|\underbrace{\frac{1}{t}\sum_{i=1}^{t} X_i}_{\text{average of samples}} - \underbrace{p}_{\text{expectation}}\right| \geq \varepsilon\right] \leq 2e^{-2\varepsilon^2 t} = \delta$$

exponentially in decaying number of samples

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

$$t = 1/2\varepsilon^2 \log_e \lceil 1/2\delta \rceil$$

X_i = answer of i^{th} person $\in \{0,1\}$

$\Pr[X_i = 1]$ = fraction of YES votes in population

= "p"

Streaming:

At end of day 

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 $s_1 \ldots s_n \ldots$ numbers

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

[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 $s_1, \ldots \in \{1..N\}$

GOAL: Pick one random element from it.

If n = length of stream is known

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

2) Output s_i

Reservoir Sampling

"Reservoir" $r = s_1$

for each element s_i

Pick random number $q \in \{1 \ldots i\}$

$q = i$ w.p. $1/i$ → Throw away what's in reservoir r, replace with s_i

$q \neq i$ w.p. $1 - 1/i$ → Ignore s_i

Whenever stream stops, Output r

Claim: At end of i^{th} step, $\forall\ j \leq i$

$$\Pr\{\text{reservoir} = s_j\} = 1/i$$

Proof: Induction:

Base Case $= 1$: True.

Assume claim true for $i-1$ steps.

After i^{th} step, Consider $j = 1, \ldots, i-1$

$$\Pr(\text{reservoir} = s_j) = \Pr\left(\begin{array}{c}\text{reservoir} = s_j \\ \text{after } i-1 \text{ steps}\end{array}\right) \cdot \Pr\left(\begin{array}{c}s_j \text{ was} \\ \text{not thrown} \\ \text{out in} \\ i^{th} \text{ step}\end{array}\right)$$

$$= \frac{1}{(i-1)} \quad \text{// (induction hypothesis)} \cdot \left(1 - \frac{1}{i}\right) = \frac{1}{i}$$

$$Pr\left[\text{reservoir} = s_i\right] = Pr\left[\text{replace reservoir with } s_i \text{ in the last step}\right]$$

$$= 1/i$$

Distinct Elements:

INPUT: A stream $s_1 \ldots s_n \in \{1 \ldots N\}$

GOAL: Estimate the number of distinct elements in stream.

ALGORITHM (IDEAL)

→ At the beginning, pick a random hash function

$$\left[h : \{1 \ldots N\} \longrightarrow \underbrace{[0, 1]}_{\text{real number}} \right] \times$$

→ Compute (as the stream goesby)

Minimum $\left(h(s_1), h(s_2) \ldots h(s_n) \right)$

{ by remembering only one number!! }

→ Output = $\left\lceil 1 \Big/ \text{minimum}\left(h(s_1) \ldots h(s_n)\right) \right\rceil$

INTUITION: $s_1, s_2 \ldots s_n$

Suppose there are k distinct elements in stream

→ 1, 2, 1, 3, 5, ... k

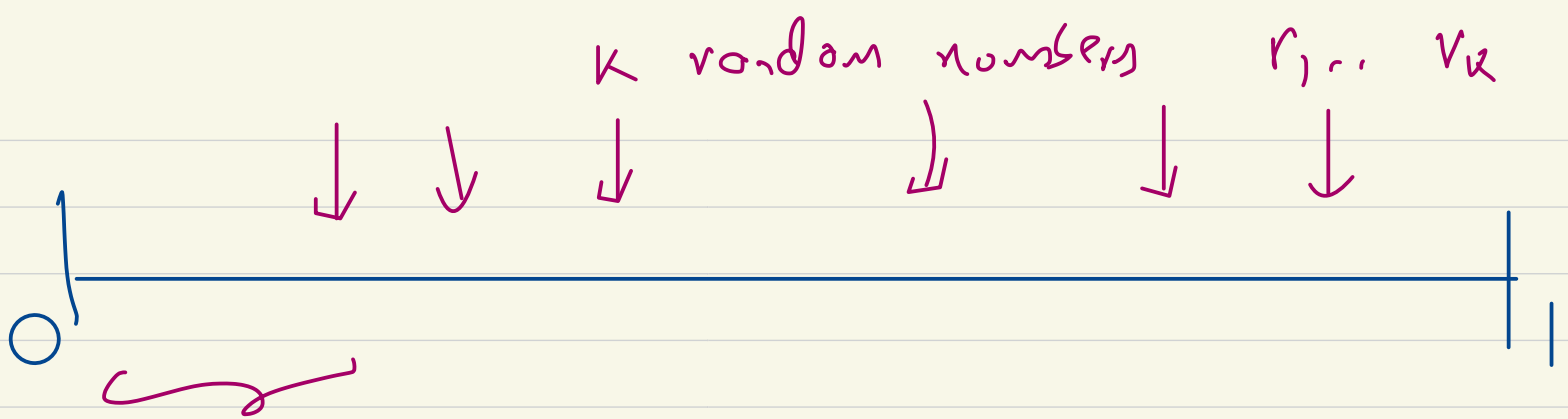
all numbers.

→ minimum $(h(s_1) \ldots h(s_n))$

$=$ minimum $(r_1 \ldots r_k)$

↑ k different hash values.

[Every hash value is uniformly random i.i.d in $[0,1]$]



$\approx 1/K+1$

$\therefore 1/\text{minimum}(r_1 .. r_K) \approx K$

Lemma:

Pick $r_1 .. r_K$ uniformly at random from $[0,1]$

$$\mathbb{E}\left[\min(r_1 .. r_K)\right] = 1/K+1$$

Proof: notes.

"Pseudorandom Hash Function"

WANT: · For each s_i, $h(s_i) \approx$ uniformly random from $[0,1]$

IMPOSSIBLE!

· EXPECT: $h[1], h[2], \ldots \overset{\text{looks}}{\approx}$ uniformly random in $(0,1)$

"Pairwise independent": $\forall\ i, j$

$h[i], h[j] \approx$ same distribution as uniformly random pair $(0,1)$