

Data Streams:

Probabilistic Counting

Mining Massive Datasets

Prof. Carlos Castillo — <https://chato.cl/teach>



Universitat
Pompeu Fabra
Barcelona

Sources

- Mining of Massive Datasets (2014) by Leskovec et al. (chapter 4)
 - Slides [part 1](#), [part 2](#)
- Tutorial: [Mining Massive Data Streams](#) (2019) by Michael Hahsler

Probabilistic counting

Motivating example

how many neighbors?

- Let $n(u, h)$ be the number of nodes reachable through a path of length up to h from node u
- Naïve method
 - Maintain a set for each node u , initialize $S(u) = \{u\}$
 - Repeat h times:

$$S(u) = S(u) \cup \bigcup_{v \text{ neighbor of } u} S(v)$$

- Answer $n(u, h) = |S(u)|$

What is the problem with this method?

- Let $n(u, h)$ be the number of nodes reachable through a path of length up to h from node u
- Naïve method
 - Maintain a set for each node u , initialize $S(u) = \{u\}$
 - Repeat h times:

$$S(u) = S(u) \cup \bigcup_{v \text{ neighbor of } u} S(v)$$

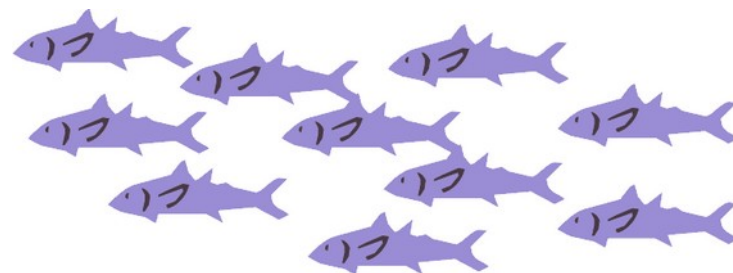
- Answer $n(u, h) = |S(u)|$

Let's look at each node

- We will receive a stream of items
 - Our neighbors at distance $\leq h$
 - Repeated many times because of loops
- We want to use a small amount of memory
- We don't care which items are in the stream
- We just want to know how many are distinct

Counting fishes with pebbles

- Normally, to count with pebbles, you add one pebble every time you see an event
- How do you extend this method to count up to 1000 fishes with 10 pebbles?
- Assume you have access to a random number generator but not to an abacus for ... reasons



Answer in
Nearpod Open-Ended Question

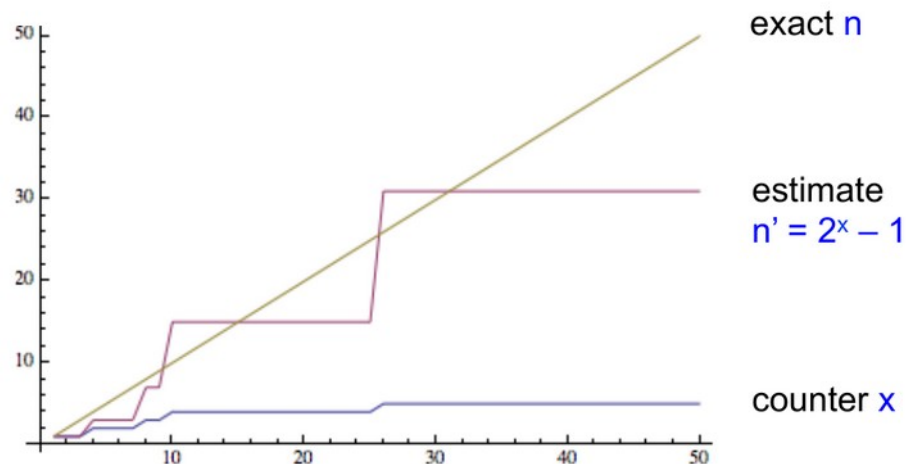
Morris' probabilistic counting (1977)

$x \leftarrow 0$

For each of the n events:

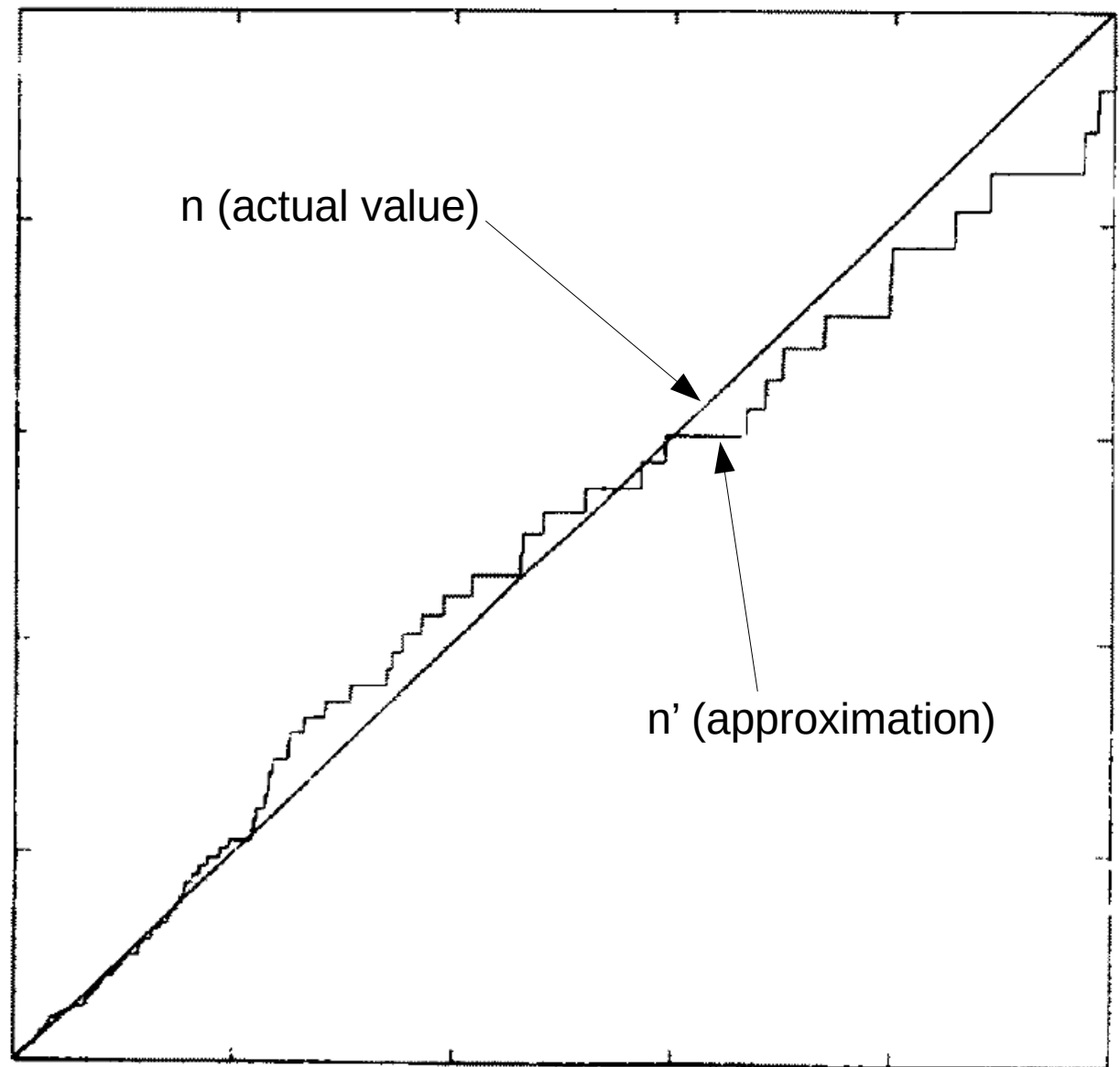
$x \leftarrow x + 1$ with probability $(1/2)^x$

Return estimate $n' = 2^x + 1$



Counter x needs only $\log_2(n)$ bits

Simulation
results by
Flajolet (1985)



Morris' algorithm provides an unbiased estimator

- Init $x=0$, let $p_x = 2^{-x}$, estimate $n' = 2^x - 1$
- $n = 1$
 - before: $x = 0$ $p_0 = 1$;
 - prob. 1: $x \rightarrow 1$
 - estimate $n' = 2^1 - 1 = 1 = n$
- $n = 2$
 - before: $x = 1$; $p_1 = \frac{1}{2}$
 - prob. $\frac{1}{2}$: x stays at 1; $n' = 2^1 - 1 = 1$
 - prob. $\frac{1}{2}$: $x \rightarrow 2$. $n' = 2^2 - 1 = 3$
 - $E[n'] = \frac{1}{2} \times 1 + \frac{1}{2} \times 3 = 2 = n$

Morris' algorithm provides an unbiased estimator (cont.)

Let $X(n)$ denote random counter x after n^{th} arrival

Initialize $X(0) = 0$; increment w.p. $p_x = 2^{-x}$

Estimate $n' = 2^{X(n)} - 1$

$$\begin{aligned} E[2^{X(n)}] &= \sum_{j=1, \dots, n-1} \Pr[X(n-1) = j] E[2^{X(n)} \mid X(n-1) = j] \\ &= \sum_{j=1, \dots, n-1} \Pr[X(n-1) = j] (p_j 2^{j+1} + (1 - p_j) 2^j) \\ &= \sum_{j=1, \dots, n-1} \Pr[X(n-1) = j] (2^j + 1) \\ &= E[2^{X(n-1)}] + 1 \end{aligned}$$

Iterating: $E[2^{X(n)}] = E[2^{X(0)}] + n = 1 + n$

Therefore: $E[2^{X(n)} - 1] = n$

Flajolet-Martin algorithm for distinct counting

Flajolet-Martin algorithm for counting distinct elements

- For every element u in the stream, compute hash $h(u)$
- Let $r(u)$ be the number of trailing zeros in hash value
 - Example: if $h(u) = 001011101\underline{000}$ then $r(u) = 3$

What is the probability of having
 $r(u)=1?$ $r(u)=2?$ $r(u)=3?$

Flajolet-Martin algorithm for counting distinct elements

- For every element u in the stream, compute hash $h(u)$
- Let $r(u)$ be the number of trailing zeros in hash value
 - Example: if $h(u) = 001011101\underline{000}$ then $r(u) = 3$
- Maintain $R = \max r(u)$ seen so far
- Output 2^R as an estimator of the number of distinct elements seen so far

Flajolet-Martin algorithm

(intuitive, hand-waving explanation)

- Let $r(u)$ be the number of trailing zeros in hash value, keep $R = \max r(u)$, output 2^R as estimate
- Repeated items don't change our estimates because their hashes are equal
- About $\frac{1}{2}$ of distinct items hash to *****0
 - To actually see a *****0, we expect to wait until seeing 2 distinct items
- About $\frac{1}{4}$ of distinct items hash to *****00
 - To actually see a *****00, we expect to wait until seeing 4 items
- ...
- If we actually saw a hash value of ***000...0 (having R trailing zeros) then on expectation we saw 2^R different items

Flajolet-Martin, correctness proof

- Let m be the number of distinct elements
- Let $z(r)$ be the probability of finding a tail of r zeroes
- We will prove that

$$z(r) \rightarrow 1 \text{ if } m \gg 2^r$$

$$z(r) \rightarrow 0 \text{ if } m \ll 2^r$$

- Hence 2^r should be around m

Flajolet-Martin, correctness proof (cont.)

- Probability a hash value ends in r zeroes = $(1/2)^r$
 - Assuming $h(u)$ produces values at random
 - Prob. random binary ends in r zeroes = $(1/2)^r$
- Probability of seeing m distinct elements and NOT seeing a tail of r zeroes = $(1 - (1/2)^r)^m$

Flajolet-Martin, correctness proof (cont.)

- Probability of seeing m distinct elements and NOT seeing a tail of r zeroes $= (1 - (1/2)^r)^m$
- Remember $(1 - \varepsilon)^{1/\varepsilon} \simeq 1/e$ for small ε
- Hence
$$\left(1 - \left(\frac{1}{2}\right)^r\right)^m = \left(1 - \left(\frac{1}{2}\right)^r\right)^{\frac{m \left(\frac{1}{2}\right)^r}{\left(\frac{1}{2}\right)^r}} \approx \left(\frac{1}{e}\right)^{\left(\frac{m}{2^r}\right)}$$

Flajolet-Martin, correctness proof (cont.)

- Probability of seeing m distinct elements and NOT seeing a tail of r zeroes
- If $m \gg 2^r$, this tends to $0 \approx (1/e)^{\left(\frac{m}{2^r}\right)}$
 - We almost certainly will see a tail of r zeroes
- If $m \ll 2^r$, this tends to 1
 - We almost certainly will not see a tail of r zeroes
- Hence, 2^r should be around m

Flajolet-Martin: increasing precision

- Idea: repeat many times or compute in parallel for multiple hash functions
- How to combine?
 - **Average?** $E[2^r]$ is infinite, extreme values will skew the number excessively
 - **Median?** 2^r is always a power of 2
- Solution: group hash functions, take median of values obtained in each group, then average across groups

Let's go back to counting neighbors

Naïve method:

Maintain a set for each node u , initialize $S(u) = \{u\}$

Repeat h times: $S(u) = S(u) \cup \bigcup_{v \text{ neighbor of } u} S(v)$

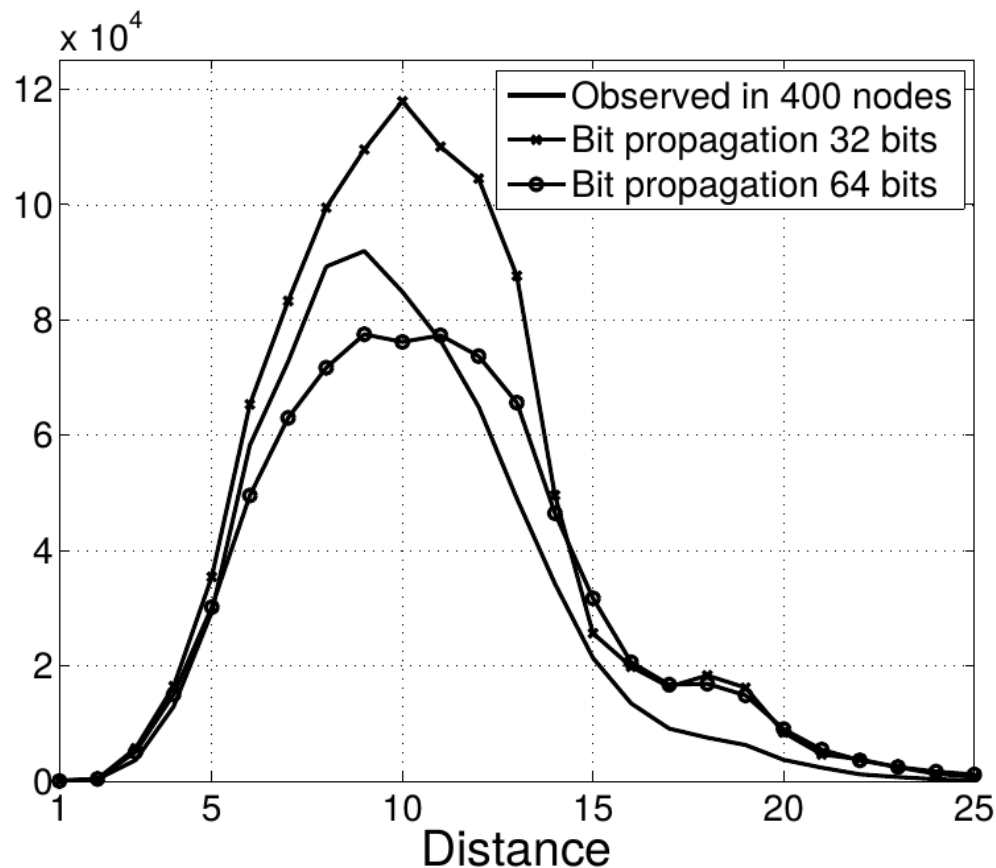
Answer $n(u, h) = |S(u)|$

ANF method:

```
// Set  $\mathcal{M}(x, 0) = \{x\}$ 
FOR each node  $x$  DO
     $M(x, 0) =$  concatenation of  $k$  bitmasks
                  each with 1 bit set ( $P(\text{bit } i) = .5^{i+1}$ )
FOR each distance  $h$  starting with 1 DO
    FOR each node  $x$  DO  $M(x, h) = M(x, h - 1)$ 
    // Update  $\mathcal{M}(x, h)$  by adding one step
    FOR each edge  $(x, y)$  DO
         $M(x, h) = (M(x, h) \text{ BITWISE-OR } M(y, h - 1))$ 
    // Compute the estimates for this  $h$ 
    FOR each node  $x$  DO
        Individual estimate  $\hat{IN}(x, h) = (2^b)/.77351$ 
        where  $b$  is the average position of the least zero bits
        in the  $k$  bitmasks
```

Example of another variant of the same type of algorithm

- More repetitions of the algorithm yield better precision



Summary

Things to remember

- Probabilistic counting algorithms:
 - Morris
 - Flajolet-Martin

Exercises for TT22-T26

- Mining of Massive Datasets (2014) by Leskovec et al.
 - Exercises 4.2.5
 - Exercises 4.3.4
 - Exercises 4.4.5
 - Exercises 4.5.6