

# More Stream-Mining

Counting How Many Elements  
Computing “Moments”

# Counting Distinct Elements

- ◆ **Problem**: a data stream consists of elements chosen from a set of size  $n$ . Maintain a count of the number of distinct elements seen so far.
- ◆ **Obvious approach**: maintain the set of elements seen.

# Applications

- ◆ How many different words are found among the Web pages being crawled at a site?
  - ◆ Unusually low or high numbers could indicate artificial pages (spam?).
- ◆ How many different Web pages does each customer request in a week?

# Using Small Storage

- ◆ **Real Problem:** what if we do not have space to store the complete set?
- ◆ Estimate the count in an unbiased way.
- ◆ Accept that the count may be in error, but limit the probability that the error is large.

# Flajolet-Martin\* Approach

- ◆ Pick a hash function  $h$  that maps each of the  $n$  elements to  $\log_2 n$  bits, uniformly.
  - ◆ Important that the hash function be (almost) a random permutation of the elements.
- ◆ For each stream element  $a$ , let  $r(a)$  be the number of trailing 0's in  $h(a)$ .
- ◆ Record  $R =$  the maximum  $r(a)$  seen.
- ◆ Estimate  $= 2^R$ .

\* Really based on a variant due to AMS (Alon, Matias, and Szegedy)

# Why It Works

- ◆ The probability that a given  $h(a)$  ends in at least  $r$  0's is  $2^{-r}$ .
- ◆ If there are  $m$  different elements, the probability that  $R \geq r$  is  $1 - \left(1 - 2^{-r}\right)^m$ .

Prob. all  $h(a)$ 's  
end in fewer than  
 $r$  0's.

Prob. a given  $h(a)$   
ends in fewer than  
 $r$  0's.

## Why It Works --- (2)

- ◆ Since  $2^{-r}$  is small,  $(1-2^{-r})^m \approx e^{-m2^{-r}}$ .
- ◆ If  $2^r \gg m$ ,  $1 - (1 - 2^{-r})^m \approx 1 - (1 - m2^{-r}) \approx m/2^r \approx 0$ .  
First 2 terms of the Taylor expansion of  $e^x$
- ◆ If  $2^r \ll m$ ,  $1 - (1 - 2^{-r})^m \approx 1 - e^{-m2^{-r}} \approx 1$ .
- ◆ Thus,  $2^R$  will almost always be around  $m$ .

# Why It Doesn't Work

- ◆  $E(2^R)$  is actually infinite.
  - ◆ Probability halves when  $R \rightarrow R + 1$ , but value doubles.
- ◆ Workaround involves using many hash functions and getting many samples.
- ◆ How are samples combined?
  - ◆ **Average**? What if one very large value?
  - ◆ **Median**? All values are a power of 2.

# Solution

- ◆ Partition your samples into small groups.
- ◆ Take the average of groups.
- ◆ Then take the median of the averages.

# Moments (New Topic)

- ◆ Suppose a stream has elements chosen from a set of  $n$  values.
- ◆ Let  $m_i$  be the number of times value  $i$  occurs.
- ◆ The  $k^{\text{th}}$  *moment* is the sum of  $(m_i)^k$  over all  $i$ .

# Special Cases

- ◆ 0<sup>th</sup> moment = number of different elements in the stream.
  - ◆ The problem just considered.
- ◆ 1<sup>st</sup> moment = sum of the numbers of elements = length of the stream.
  - ◆ Easy to compute.
- ◆ 2<sup>nd</sup> moment = *surprise number* = a measure of how uneven the distribution is.

# Example: Surprise Number

- ◆ Stream of length 100; 11 values appear.
- ◆ **Unsurprising**: 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9. Surprise # = 910.
- ◆ **Surprising**: 90, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1. Surprise # = 8,110.

# AMS Method

- ◆ Works for all moments; gives an unbiased estimate.
- ◆ We'll just concentrate on 2<sup>nd</sup> moment.
- ◆ Based on calculation of many random variables  $X$ .
  - ◆ Each requires a count in main memory, so number is limited.

# One Random Variable

- ◆ Assume stream has length  $n$ .
- ◆ Pick a random time to start, so that any time is equally likely.
- ◆ Let the chosen time have element  $a$  in the stream.
- ◆  $X = n * ((\text{twice the number of } a \text{'s in the stream starting at the chosen time}) - 1)$ .
  - ◆ **Note:** just store the count.

# Expected Value of $X$

- ◆ 2<sup>nd</sup> moment is  $\sum_a (m_a)^2$ .
- ◆  $E(X) = (1/n) \left( \sum_{\text{all times } t} n * (\text{twice the number of times the stream element at time } t \text{ appears from that time on}) - 1 \right)$ .
- ◆  $= \sum_a (1/n)(n)(1+3+5+\dots+2m_a-1)$ .
- ◆  $= \sum_a (m_a)^2$ .

# Combining Samples

- ◆ Compute as many variables  $X$  as can fit in available memory.
- ◆ Average them in groups.
- ◆ Take median of averages.
- ◆ Proper balance of group sizes and number of groups assures not only correct expected value, but expected error goes to 0 as number of samples gets large.

# Problem: Streams Never End

- ◆ We assumed there was a number  $n$ , the number of positions in the stream.
- ◆ But real streams go on forever, so  $n$  is a variable --- the number of inputs seen so far.

# Fixups

1. The variables  $X$  have  $n$  as a factor --- keep  $n$  separately; just hold the count in  $X$ .
2. Suppose we can only store  $k$  counts. We must throw some  $X$ 's out as time goes on.
  - ◆ Objective: each starting time  $t$  is selected with probability  $k / n$ .

## Solution to (2)

- ◆ Choose the first  $k$  times for  $k$  variables.
- ◆ When the  $n^{\text{th}}$  element arrives ( $n > k$ ), choose it with probability  $k / n$ .
- ◆ If you choose it, throw one of the previously stored variables out, with equal probability.