Counting Distinct Elements

Streaming Statistics

Applications

- How many distinct people visit the website?

- How many different Web pages does each customer request in a week?

- How many distinct products have we sold in the last week?

Counting Distinct Elements

- **Problem:**
  - Data stream consists of a universe of elements chosen from a set of size N
  - Maintain a count of the number of distinct elements seen so far

What can we do?

Using Small Storage

- **Approach with enough space:**
  Maintain the set of elements seen so far
  - E.g. keep a hash table of all the distinct elements seen so far

- **Real problem:** What if we do not have space to maintain the set of elements seen so far?

- **Same philosophy as previous:**
  - Estimate the count in an unbiased way
  - Accept that the count may have a little 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 at least **$\log_2 N$** bits
- For each stream element a, let $r(a)$ be the number of trailing **0s** in $h(a)$
  - **r(a)** = position of first 1 counting from the right
    - E.g., say $h(a) = 12$, then 12 is 1100 in binary, so $r(a) = 2$
- Record **R = the maximum $r(a)$ seen**
  - **$R = \max_a r(a)$**, over all the items a seen so far
- **Estimated number of distinct elements = 2^R**

Why It Works: Intuition

- **Rough and heuristic intuition:**
  - *h(a)* hashes *a* with **equal prob.** to any of *N* values
  - Then *h(a)* is a sequence of $\mathbf{\log_2 N}$ bits,
    where 2^{-r} fraction of all *a*s have a tail of *r* zeros
    - About 50% of *a*s hash to ***0**
    - About 25% of *a*s hash to ****00**
    - So, if we saw the longest tail of *r=2* (i.e., item hash
      ending ***100**) then we have probably seen
      about *4* distinct items so far
  - **So, it takes to hash about 2^r items before we see one with zero-suffix of length *r***

Why It Works: More formally

- **What is the probability that a given *h*(*a*) ends in at least *r* zeros?**
  - **h(a)** hashes elements uniformly at random
  - Probability that a random number ends in at least *r* zeros is $\mathbf{2^{-r}}$
- Then, the probability of **NOT** seeing a tail of length at least *r* among *m* elements:

$$\boxed{\left(\boxed{1-2^{-r}}\right)^{m}}$$

Prob. all end in fewer than ***r*** zeros.

Prob. that given **h(a)** ends in fewer than ***r*** zeros

Why It Works: More formally

- **Note:** $(1-2^{-r})^m = (1-2^{-r})^{2^r(m2^{-r})} \approx e^{-m2^{-r}}$
- **Prob. of NOT finding a tail of length *r* is:**
  - If *m* << *2^r*, then prob. tends to **1**
    - $(1-2^{-r})^m \approx e^{-m2^{-r}} = 1$ as **m/2^r → 0**
    - So, the probability of finding a tail of length *r* tends to **0**
  - If *m* >> *2^r*, then prob. tends to **0**
    - $(1-2^{-r})^m \approx e^{-m2^{-r}} = 0$ as **m/2^r → ∞**
    - So, the probability of finding a tail of length *r* tends to **1**

- **Thus, 2^R will almost always be around *m*!**

Issues to fix

- The estimation is biased
  - Estimated with $\mathbf{2}^{R}/\Phi,$ where $\Phi = 0.77351$ is a correction factor.
- Problems of high variance. Improve accuracy.
  - Use many hash functions with samples of R
  - Partition your samples into small groups
  - Take the median of groups
  - Then take the average of the medians

Further Readings

- Flajolet, Philippe; Fusy, Éric; Gandouet, Olivier; Meunier, Frédéric (2007). "Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm“. Discrete Mathematics and Theoretical Computer Science proceedings. Nancy, France. AH: 127–146.
- Kane, Daniel M.; Nelson, Jelani; Woodruff, David P. (2010). "An optimal algorithm for the distinct elements problem", Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems of data - PODS '10. p. 41.

Computing Moments

Generalization: Moments

- Suppose a stream has elements chosen from a set *A* of *N* values (say 1 to N)

- Let m_i be the number of times item *i* occurs in the stream

- The k^{th} *moment* is

$$\sum_{i \in A} (m_i)^k$$

Special Cases

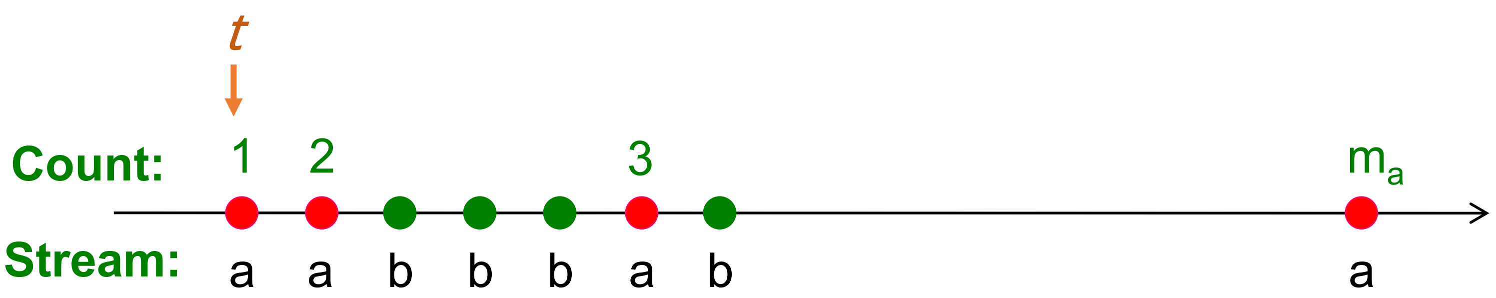
$$\sum_{i \in A} (m_i)^k$$

- **0th moment =** number of **distinct** elements (Flajolet-Martin)
- **1st moment =** count of the **numbers** of elements
- **2nd moment =** a measure of how uneven the distribution is (denoted as S)
  - E.g. **Stream of length 100, 11 distinct values**
    - Item counts: **10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9** S = 910
    - Item counts: **90, 1, 1, 1, 1, 1, 1, 1 ,1, 1, 1** S = 8,110

AMS(Alon-Matias-Szegedy) Method

- Gives an **unbiased estimate** for the **2nd moment** $\boldsymbol{S} = \sum_i \boldsymbol{m}_i^2$

by keeping track of just **one variable** *X:*
  - *X.el* corresponds to a item *i*
    - Pick some random time *t* (*t<n*) to start, **equally likely** in a stream of length *n*
    - If at time *t* the stream have item *i*, we set *X.el = i*
  - *X.val* corresponds to the **count** of the chosen item *i*
    - Count *c (X.val = c*), the number of item *i* starting from the chosen time *t*



AMS(Alon-Matias-Szegedy) Method

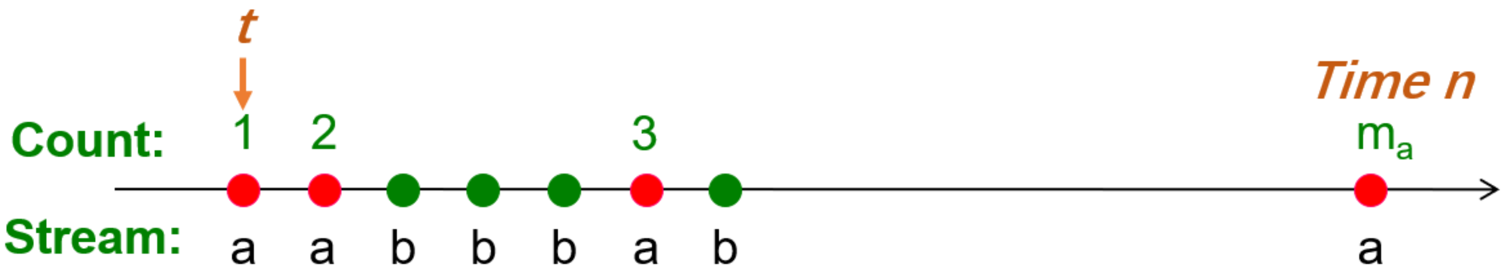
- **The estimate of the 2nd moment ($\sum_i \boldsymbol{m_i^2}$) is:**

$$\boldsymbol{f(X) = n\,(2 \cdot c - 1)}$$

  - Note, we will keep track of multiple Xs, (X_1, X_2,⋯ X_k) and our final estimate will be $\boldsymbol{S = 1/k \sum_j^k f(X_j)}$

- **Let's prove** $\mathbf{E}[\boldsymbol{f}(\mathbf{X})] = \sum_i (\boldsymbol{m_i})^2 = \boldsymbol{S}$

Expectation Analysis



- c_t ··· number of times item at time t appears from time **t** to n ($c_1 = m_a$, $c_2 = m_a - 1$, $c_3 = m_b$)

m_i … total count of item i in the stream

- $E[f(X)] = \frac{1}{n}\sum_{t=1}^{n} n(2c_t - 1)$

$$= \frac{1}{n}\sum_{i \in A} n\,(1 + 3 + 5 + \cdots + 2m_i - 1)$$

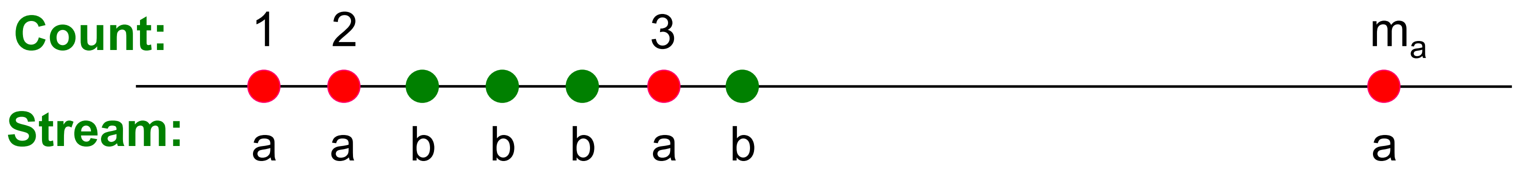
Group times by the value seen

Time t when the last i is seen ($c_t = 1$)

Time t when the penultimate i is seen ($c_t = 2$)

Time t when the first i is seen ($c_t = m_i$)

Expectation Analysis



- $E[f(X)] = \frac{1}{n}\sum_i n\,(1 + 3 + 5 + \cdots + 2m_i - 1)$
  - calculation: $(1 + 3 + 5 + \cdots + 2m_i - 1) = \sum_{i=1}^{m_i}(2i - 1) = 2\frac{m_i(m_i+1)}{2} - m_i = (m_i)^2$
- **Then** $\mathbf{E}[\boldsymbol{f}(\mathbf{X})] = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i}} \boldsymbol{n}\,(\boldsymbol{m_i})^{\mathbf{2}} = \boldsymbol{S}$

- **We have the second moment (in expectation)!**

Higher-Order Moments

- **For estimating k^{th} moment we essentially use the same algorithm but change the estimate:**
  - For **k=2** we used ***n*** **(2·c – 1)** (where **c=X.val**)
  - For **k=3** , can you try to find out what we use?
    - ***n*** **(3·c² – 3c + 1)**
- **Why?**
  - **For k=2:** Remember we had $(1+3+5+\cdots+2m_i-1)$ and we showed terms *2c-1* (for **c=1,⋯,m**) sum to *m²*
    - $\sum_{c=1}^{m} 2c-1 = \sum_{c=1}^{m} c^2 - \sum_{c=1}^{m}(c-1)^2 = m^2$
    - So: $\mathbf{2c-1 = c^2 - (c-1)^2}$
  - **For k=3:** **c^3 - $(c-1)^3$ = $3c^2$ - 3c + 1**
- **Generally:** Estimate $= n\,(c^k - (c-1)^k)$

Combining Samples

- **In practice:**
  - Compute $f(X) = n(2c - 1)$ for as many variables X as you can fit in memory
  - Average them in groups
  - Take median of averages

- **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

Streams Never End: 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 Xs out as time goes on:

- **Objective:** Each starting time t is selected with probability k/n
- **Solution: (fixed-size sampling)**
    - Choose the first k times for k variables
    - When the n^{th} element arrives ($n > k$), choose it with probability k/n
    - If you choose it, throw one of the previously stored variables **X** out, with equal probability

Summary of Streaming Algorithms

- Queries
  - Filtering a data stream
  - Queries over a sliding window
  - Estimating statistics

- Key techniques
  - Hashing functions
  - Approximation with sketch/summarization
  - Theoretical analysis