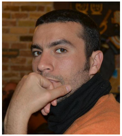
Data Streams as Random Permutations: the Distinct Element Problem

Conrado Martínez,
Univ. Politècnica de Catalunya

VIII JMDA, Almería, Julio 2012

Joint work with:



A. Helmi

J. Lumbroso

A. Viola

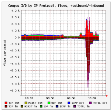
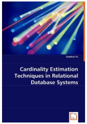
Introduction

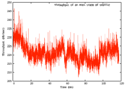
- A data stream is a (very long) sequence

$$\mathcal{S} = s_1, s_2, s_3, \ldots, s_N$$

  of items s_i drawn from some (large) domain $\mathcal{U}$, $s_i \in \mathcal{U}$
- The goal: to compute $\theta = \theta(\mathcal{S})$, but there are limitations to our computational power:
  - a single pass over the sequence
  - very short time for computation on each item
  - very small auxiliary memory: $M \ll N$; ideally $M = \Theta(1)$ or $M = \mathcal{O}(\log N)$
  - no statistical hypothesis on the data

Introduction



There are lots of applications for this data strem model:

- Network traffic analysis ⇒ DoS/DDoS attacks, worms, ...
- Database query optimization
- Information retrieval ⇒ similarity index
- Data mining
- And many more ...

Introduction


We will often see $\mathcal{S}$ as a multiset

$\{y_1{}^{f_1}, \ldots, y_n{}^{f_n}\}$, $f_i =$ frequency of the ith distinct element y_i

Some typical problems:

- The cardinality of $\mathcal{S}$: card$(\mathcal{S}) = n \leqslant N = |\mathcal{S}| \Leftarrow$ This paper
- The elements y_i such that:
  - $f_i \geqslant k$ (k-elephants)
  - $f_i \geqslant c \cdot N$, $0 < c < 1$ (c-icebergs)
  - $f_i < k$ (k-mice)

Introduction

Small auxiliary memory $\Rightarrow$

Exact solution too costly (or impossible) $\Rightarrow$

Randomized algorithms $\Rightarrow$

Estimation $\hat{\theta}$ of the quantity θ

- The estimator $\hat{\theta}$ must be unbiased

$$\mathsf{E}\left[\hat{\theta}\right] = \theta$$

- The estimator must be accurate (small standard error)

$$\mathsf{SE}\left[\hat{\theta}\right] := \frac{\sqrt{\mathsf{Var}\left[\hat{\theta}\right]}}{\mathsf{E}\left[\hat{\theta}\right]} < \epsilon,$$

e.g., $\epsilon = 0.01$ (1%)

Estimating the cardinality

The first ingredient:

- Map each item s_i of the stream to a value in $(0,1)$ using a hash function $h : \mathcal{U} \to (0,1) \Rightarrow$ reproducible randomness
- The multiset $\mathcal{S}$ is mapped to a multiset

$$\mathcal{S}' = h(\mathcal{S}) = \{{x_1}^{f_1}, \ldots, {x_n}^{f_n}\},$$

  with $x_i = \text{hash}(y_i)$, $f_i = \#$ of x_i's
- The set of distinct elements $X = \{x_1, \ldots, x_n\}$ is a set of n independent and uniformly distributed real numbers in $(0,1)$

Estimating the cardinality

The second ingredient:

- Define some easily computable observable R which is insensitive to repetitions, that is, it only depends on the underlying set of distinct elements:

$$R = R(\mathcal{S}) = R(X)$$

- Perform the probabilistic analysis of R for a set X of n random real numbers. If

$$\mathsf{E}_n[R] = \varphi(n)$$

then it is reasonable to assume that the expected value of $\varphi^{-1}(R)$ will be close to n; we will need some correcting factor α to get an (asymptotically) unbiased estimator

$$\mathsf{E}_n\left[\alpha\varphi^{-1}(R)\right] = n + \text{l.o.t.}$$

Probabilistic Counting

- For instance, in Flajolet & Martin's Probabilistic Counting (1985) the observable R is the length of the longest prefix $0.0^{R-1}1$ such that all prefixes 0.0^k1 appear among the hashed values, for $0 \leqslant k \leqslant R-1$
- R is easy to compute and it does not depend on repetitions

$$E_n[R] \approx \log_2 n$$

and

$$E_n\left[\alpha 2^R\right] = n + o(n)$$

for

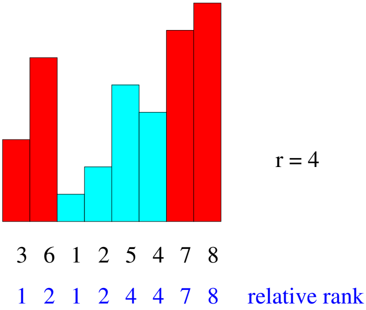
$$\alpha^{-1} = \frac{e^{\gamma}\sqrt{2}}{3} \prod_{k \geqslant 1} \left(\frac{(4k+1)(2k+1)}{2k(4k+3)} \right)^{(-1)^{\nu(k)}} \approx 0.77351\ldots$$

Other estimators

- Based on bit patterns
  - LogLog: Durand, Flajolet (2003)
  - HyperLogLog: Flajolet, Fusy, Gandouet, Meunier (2007)
- Based on order statistics (e.g., the kth smallest in the set of distinct hash values)
  - Bar-Yossef, Kumar & Sivakumar (2002)
  - Bar-Yossef, Jayram, Kumar, Sivakumar & Trevisan (2002)
  - Giroire (2005, 2009)
  - Chassaing & Gérin (2006)
  - Lumbroso (2010)

Our estimator: Recordinality

- RECORDINALITY counts the number of records (local maxima) in the sequence



- It depends in the underlying permutation of the first occurrences of distinct values, very different from the other estimators

Our estimator: Recordinality

- $\sigma(i)$ is a record of the permutation σ if $\sigma(i) > \sigma(j)$ for all $j < i$
- It is well known that the number r of records satisfies

$$\mathsf{E}_n[r] = \log n + \mathcal{O}(1)$$

  hence we anticipate that e^r should give us an estimate of n
- The notion is generalized to k-records: $\sigma(i)$ is a k-record if $\sigma(i)$ is among the k largest elements in $\sigma(1), \ldots, \sigma(i)$

Our estimator: Recordinality

procedure RECORDINALITY($\mathcal{S}$, k)
  fill T with the first k distinct elements (hash values) of the stream $\mathcal{S}$
  $r \leftarrow k$
  for all $s \in \mathcal{S}$ **do**
    $x \leftarrow \text{hash}(s)$
    if $x > \min(T) \wedge x \notin T$ **then**
      $r \leftarrow r + 1$; $T \leftarrow T \cup \{x\} \setminus \min(T)$
    end if
  end for
  return $Z = \alpha_k \cdot e^r$ ▷ α_k is a correcting factor
end procedure

Memory: k hash values ($k \log n$ bits) + 1 counter ($\log \log n$ bits)

Our estimator: Recordinality

Theorem (Helmi, Martínez and Panholzer, 2012)

Let r_k *denote the number of* k*-records in a permutation of size* n*. The exact distribution of* r_k *is*

$$\textit{Prob}_n\{r_k = j\} = \begin{cases} [\![n = j]\!] & \textit{if } k > n, \\ k^{j-k}\dfrac{k!}{n!}\begin{bmatrix} n-k+1 \\ j-k+1 \end{bmatrix} & \textit{if } k \leqslant j \leqslant n \end{cases}$$

$\begin{bmatrix} n \\ j \end{bmatrix}$ *= signless Stirling numbers of the first kind;* $[\![P]\!] = 1$ *if* P *true,* $= 0$ *otherwise*

Our estimator: Recordinality

- The expected value of r_k is $k \log(n/k) + \text{l.o.t.}$; it is reasonable then to assume that for

$$Z := k \exp(\alpha_k \cdot r_k)$$

  we should have $\mathsf{E}_n[Z] \sim n$ for some suitable correcting factor α_k
- We can use the formula for $\mathsf{Prob}_n\{r_k = j\}$ to explictly compute $\mathsf{E}_n[Z]$ and to determine ϕ, and then compute the standard error

Our estimator: Recordinality

Theorem

The RECORDINALITY *estimator*

$$Z := k\left(1 + \frac{1}{k}\right)^{r_k - k + 1} - 1$$

is an *unbiased* *estimator of* n*:* $E_n[Z] = n$.

Our estimator: Recordinality

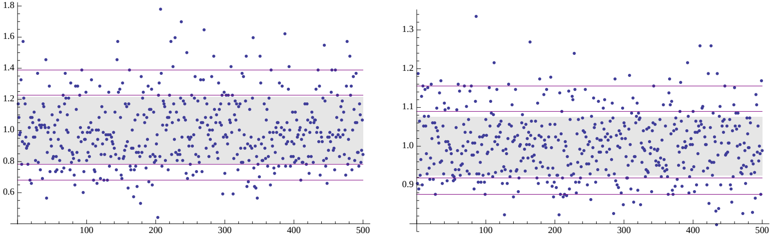
Theorem

The accuracy of RECORDINALITY, *expressed in terms of standard error, asymptotically satisfies*

$$SE_n [Z] \sim \sqrt{\left(\frac{n}{ke}\right)^{\frac{1}{k}} - 1}$$

Our estimator: Recordinality

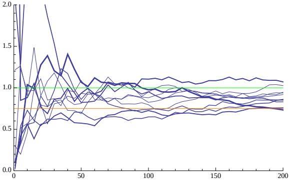
For practical values of n, even for small k, the estimates may be significantly concentrated.
For instance, for $k = 10$, the estimates are within σ, 2σ, 3σ of the exact count in respectively 91%, 96% and 99% of all cases.



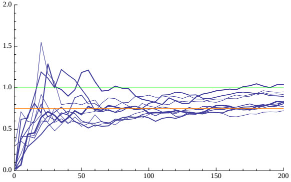
$k = 64$ $\qquad$ $k = 256$

500 estimates of cardinality in Shakespare's *A Midsummer Night's Dream*; top and bottom lines (5%), centermost lines (70%); gray area (1 standard deviation)

Other issues



Original texts



Randomly permuted texts

- RECORDINALITY does not depend on the hash values, only the relative ordering ⇒ we can avoid using the hash function, provided the distinct elements appear (for the first time) in random order
- We can combine RECORDINALITY with any of the exisiting estimators since they are independent; a suitably weighted sum of the estimations will have less variance ⇒ better accuracy

Other issues

- The table of k largest hash values gives us a random sample of k distinct elements out of the $n \Rightarrow$ distinct sampling for free
- Indeed, distinct elements "enter" the table or not according to their hash value, a random uniform number
- An easy modification allows us to have a random sample of distinct elements with expected size $k \log(n/k) \Rightarrow$ variable-size sampling

Concluding remarks

- First (?) application of combinatorics of random permutations to data stream algorithms
- Simple and elegant algorithms
- Nice combinatorics and mathematical analysis
- Many extensions to explore: sampling, sliding windows, similarity index, ...

Thanks a lot for your attention!

¡Gracias por vuestra atención!