Probabilistic Techniques in Data Stream Analysis

Conrado Martínez
U. Politècnica de Catalunya

RA-MIRI 2023–2024

Part

Introduction

- A data stream is a (very long) sequence

$$\mathcal{Z} = z_1, z_2, z_3, \ldots, z_N$$

  of elements drawn from a (very large) domain $\mathcal{U}$ ($z_i \in \mathcal{U}$)

- The goal: to compute $f(\mathcal{Z})$, but ...

Introduction

- A data stream is a (very long) sequence

$$\mathcal{Z} = z_1, z_2, z_3, \ldots, z_N$$

  of elements drawn from a (very large) domain $\mathcal{U}$ ($z_i \in \mathcal{U}$)
- The goal: to compute $f(\mathcal{Z})$, but . . .

Introduction

…under rather stringent constraints (data stream model)

- a single pass over the data stream
- extremely short time spent on each single data item
- a limited amount M of auxiliary memory, $M \ll N$; ideally $M = \Theta(1)$ or $M = \Theta(\log N)$
- no statistical hypothesis about the data

Introduction

…under rather stringent constraints (data stream model)

- a single pass over the data stream
- extremely short time spent on each single data item
- a limited amount M of auxiliary memory, $M \ll N$; ideally $M = \Theta(1)$ or $M = \Theta(\log N)$
- no statistical hypothesis about the data

Introduction

…under rather stringent constraints (data stream model)

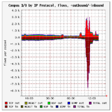
- a single pass over the data stream
- extremely short time spent on each single data item
- a limited amount M of auxiliary memory, $M \ll N$; ideally $M = \Theta(1)$ or $M = \Theta(\log N)$
- no statistical hypothesis about the data

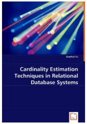
Introduction

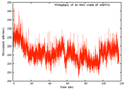
…under rather stringent constraints (data stream model)

- a single pass over the data stream
- extremely short time spent on each single data item
- a limited amount M of auxiliary memory, $M \ll N$; ideally $M = \Theta(1)$ or $M = \Theta(\log N)$
- no statistical hypothesis about the data

Introduction



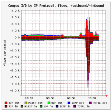


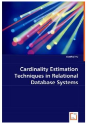


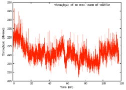
There are a wide range of applications for the data stream model

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

Introduction



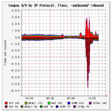


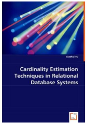


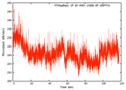
There are a wide range of applications for the data stream model

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

Introduction



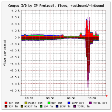


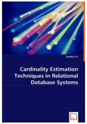


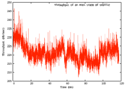
There are a wide range of applications for the data stream model

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

Introduction



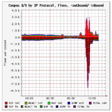


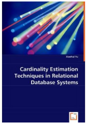


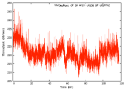
There are a wide range of applications for the data stream model

- Network traffic analysis ⇒ DoS/DDoS attacks, *worms*, …
- Database query optimization
- Information retrieval ⇒ similarity index
- Data mining
- Recommedation systems
- and many more …

Introduction



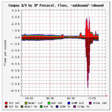


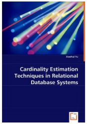


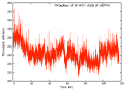
There are a wide range of applications for the data stream model

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

Introduction







There are a wide range of applications for the data stream model

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

Introduction

We'll often look at $\mathcal{Z}$ as a multiset $\{x_1 \circ f_1, \ldots, x_n \circ f_n\}$, where

$$f_i = \text{frequency of the } i\text{-th distinct element } x_i$$

Some fundamental problems in data stream analysis:

- Number of distinct elements: card$(\mathcal{Z}) = n \leqslant N$
- Random samples of distinct elements
- Frequency moments $F_p = \sum_{1 \leqslant i \leqslant n} f_i^p$
  (N.B. $n = F_0, N = F_1$)
- (Number of) Elements x_i such that $f_i \geqslant k$ (k-elephants) or $f_i < k$ (k-mice)
- (Number of) Elements x_i such that $f_i/N \geqslant c$, $0 < c < 1$ (c-icebergs, a.k.a. heavy hitters)
- The k most frequent elements (top-k elements)

Introduction

We'll often look at $\mathcal{Z}$ as a multiset $\{x_1 \circ f_1, \ldots, x_n \circ f_n\}$, where

$$f_i = \text{frequency of the } i\text{-th distinct element } x_i$$

Some fundamental problems in data stream analysis:

- Number of distinct elements: card$(\mathcal{Z}) = n \leqslant N$
- Random samples of distinct elements
- Frequency moments $F_p = \sum_{1 \leqslant i \leqslant n} f_i^p$
  (N.B. $n = F_0, N = F_1$)
- (Number of) Elements x_i such that $f_i \geqslant k$ (k-elephants) or $f_i < k$ (k-mice)
- (Number of) Elements x_i such that $f_i/N \geqslant c$, $0 < c < 1$ (c-icebergs, a.k.a. heavy hitters)
- The k most frequent elements (top-k elements)

Introduction

We'll often look at $\mathcal{Z}$ as a multiset $\{x_1 \circ f_1, \ldots, x_n \circ f_n\}$, where

$$f_i = \text{frequency of the } i\text{-th distinct element } x_i$$

Some fundamental problems in data stream analysis:

- Number of distinct elements: card$(\mathcal{Z}) = n \leqslant N$
- Random samples of distinct elements
- Frequency moments $F_p = \sum_{1 \leqslant i \leqslant n} f_i^p$
  (N.B. $n = F_0, N = F_1$)
- (Number of) Elements x_i such that $f_i \geqslant k$ (k-elephants) or $f_i < k$ (k-mice)
- (Number of) Elements x_i such that $f_i / N \geqslant c$, $0 < c < 1$ (c-icebergs, a.k.a. heavy hitters)
- The k most frequent elements (top-k elements)

Introduction

We'll often look at $\mathcal{Z}$ as a multiset $\{x_1 \circ f_1, \ldots, x_n \circ f_n\}$, where

$$f_i = \text{frequency of the } i\text{-th distinct element } x_i$$

Some fundamental problems in data stream analysis:

- Number of distinct elements: $\text{card}(\mathcal{Z}) = n \leqslant N$
- Random samples of distinct elements
- Frequency moments $F_p = \sum_{1 \leqslant i \leqslant n} f_i^p$
  (N.B. $n = F_0, N = F_1$)
- (Number of) Elements x_i such that $f_i \geqslant k$ (k-elephants) or $f_i < k$ (k-mice)
- (Number of) Elements x_i such that $f_i/N \geqslant c$, $0 < c < 1$ (c-icebergs, a.k.a. heavy hitters)
- The k most frequent elements (top-k elements)

Introduction

We'll often look at $\mathcal{Z}$ as a multiset $\{x_1 \circ f_1, \ldots, x_n \circ f_n\}$, where

$$f_i = \text{frequency of the } i\text{-th distinct element } x_i$$

Some fundamental problems in data stream analysis:

- Number of distinct elements: $\text{card}(\mathcal{Z}) = n \leqslant N$
- Random samples of distinct elements
- Frequency moments $F_p = \sum_{1 \leqslant i \leqslant n} f_i^p$
  (N.B. $n = F_0, N = F_1$)
- (Number of) Elements x_i such that $f_i \geqslant k$ (k-elephants) or $f_i < k$ (k-mice)
- (Number of) Elements x_i such that $f_i/N \geqslant c$, $0 < c < 1$ (c-icebergs, a.k.a. heavy hitters)
- The k most frequent elements (top-k elements)

Introduction

We'll often look at $\mathcal{Z}$ as a multiset $\{x_1 \circ f_1, \ldots, x_n \circ f_n\}$, where

$$f_i = \text{frequency of the } i\text{-th distinct element } x_i$$

Some fundamental problems in data stream analysis:

- Number of distinct elements: card$(\mathcal{Z}) = n \leqslant N$
- Random samples of distinct elements
- Frequency moments $F_p = \sum_{1 \leqslant i \leqslant n} f_i^p$
  (N.B. $n = F_0, N = F_1$)
- (Number of) Elements x_i such that $f_i \geqslant k$ (k-elephants) or $f_i < k$ (k-mice)
- (Number of) Elements x_i such that $f_i/N \geqslant c$, $0 < c < 1$ (c-icebergs, a.k.a. heavy hitters)
- The k most frequent elements (top-k elements)

Introduction

Very limited available memory $\Rightarrow$ exact solution too costly or unfeasible
$\Rightarrow$ Randomized algorithms $\Rightarrow$ estimation $\hat{q}$ of the quantity of interest $q = f(\mathbb{Z})$

- $\hat{q}$ must be an unbiased estimator

$$\mathbb{E}[\hat{q}] = q$$

- The estimator must accurate, for example, it must have a small standard error

$$\text{SE}[\hat{q}] := \frac{\sqrt{\mathbb{V}[\hat{q}]}}{\mathbb{E}[\hat{q}]} < \epsilon,$$

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

Introduction

Very limited available memory $\Rightarrow$ exact solution too costly or unfeasible

$\Rightarrow$ Randomized algorithms $\Rightarrow$ estimation $\hat{q}$ of the quantity of interest $q = f(\mathcal{Z})$

- $\hat{q}$ must be an unbiased estimator

$$\mathbb{E}[\hat{q}] = q$$

- The estimator must accurate, for example, it must have a small standard error

$$\text{SE}\,[\hat{q}] := \frac{\sqrt{\mathbb{V}[\hat{q}]}}{\mathbb{E}[\hat{q}]} < \epsilon,$$

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

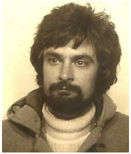
Part I

Cardinality Estimation

Part I

Cardinality Estimation

Probabilistic Counting



G.N. Martin

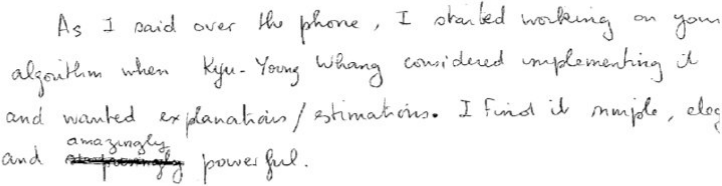
In late 70s G. Nigel Martin invented probabilistic counting to optimize database query performance

To correct the bias that he systematically found in his experiments, he introduced a “fudge” factor in the estimator

Probabilistic Counting

Ph. Flajolet

When Philippe Flajolet learnt about the algorithm, he put it on a solid scientific ground, with a detailed mathematical analysis which delivered the exact value of the correction factor and a tight upper bound on the standard error



As I said over the phone, I started working on your
algorithm when Kyu-Young Whang considered implementing it
and wanted explanations / estimations. I find it simple, eleg
and amazingly powerful.

Probabilistic Counting

- First idea: every element is hashed to a real value in $(0,1)$
  $\Rightarrow$ reproductible randomness
- The "multiset" $\mathcal{Z}$ is mapped by the hash function $h : \mathcal{U} \to (0,1)$ to a multiset

$$\mathcal{Z}' = h(\mathcal{Z}) = \{y_1 \circ f_1, \ldots, y_n \circ f_n\},$$

  with $y_i = \text{hash}(x_i)$, $f_i =$ frequency of x_i in $\mathcal{Z}$
- The set of distinct* elements $Y = \{y_1, \ldots, y_n\}$ is a set of n random numbers, independent and uniformly drawn from $(0,1)$

Probabilistic Counting

- First idea: every element is hashed to a real value in $(0, 1)$
  $\Rightarrow$ reproductible randomness
- The "multiset" $\mathcal{Z}$ is mapped by the hash function $h : \mathcal{U} \to (0, 1)$ to a multiset

$$\mathcal{Z}' = h(\mathcal{Z}) = \{y_1 \circ f_1, \ldots, y_n \circ f_n\},$$

  with $y_i = \text{hash}(x_i)$, $f_i = $ frequency of x_i in $\mathcal{Z}$
- The set of distinct* elements $Y = \{y_1, \ldots, y_n\}$ is a set of n random numbers, independent and uniformly drawn from $(0, 1)$

Probabilistic Counting

- First idea: every element is hashed to a real value in $(0,1)$ $\Rightarrow$ reproductible randomness
- The "multiset" $\mathcal{Z}$ is mapped by the hash function $h : \mathcal{U} \to (0,1)$ to a multiset

$$\mathcal{Z}' = h(\mathcal{Z}) = \{y_1 \circ f_1, \ldots, y_n \circ f_n\},$$

  with $y_i = \text{hash}(x_i)$, $f_i =$ frequency of x_i in $\mathcal{Z}$
- The set of distinct* elements $Y = \{y_1, \ldots, y_n\}$ is a set of n random numbers, independent and uniformly drawn from $(0,1)$

*We'll neglect the probability of collisions, i.e., $h(x_i) = h(x_j)$ for some $x_i \neq x_j$; this is reasonable if $h(x)$ has enough bits

Probabilistic Counting

Flajolet & Martin (JCSS, 1985) proposed to find, among the set of hash values, the length of the largest prefix (in binary) $0.0^{R-1}1\ldots$ such that all shorter prefixes with the same pattern $0.0^{p-1}1\ldots$, $p \leqslant R$, also appear

The value R is an observable which can be easily be computed using a small auxiliary memory and it is insensitive to repetitions $\leftarrow$ the observable is a function of Y, not of the f_i's

Probabilistic Counting

- For a set of n random numbers in $(0,1) \rightarrow$

$$\mathbb{E}[R] \approx \log_2 n$$

- However $\mathbb{E}\left[2^R\right] \not\sim n$, there is a significant bias and we need ϕ such that

$$\mathbb{E}\left[\phi \cdot 2^R\right] \sim n$$

Probabilistic Counting

- For a set of n random numbers in $(0, 1) \rightarrow$

$$\mathbb{E}[R] \approx \log_2 n$$

- However $\mathbb{E}\left[2^R\right] \not\sim n$, there is a significant bias and we need ϕ such that

$$\mathbb{E}\left[\phi \cdot 2^R\right] \sim n$$

Probabilistic Counting

```
procedure ProbabilisticCounting(𝒵)
    bmap ← ⟨0, 0, . . . , 0⟩
    for z ∈ 𝒵 do
        y ← hash(z)
        p ← lenght of the largest prefix 0.0^(p−1)1 . . . in y
        bmap[p] ← 1
    end for
    R ← largest p such that bmap[i] = 1 for all 1 ⩽ i ⩽ p
▷ ϕ is the correction factor: 𝔼[ϕ · 2^R] = n
    return  Z := ϕ · 2^R
end procedure
```

A very precise mathematical analysis gives:

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

Stochastic averaging

- The standard error of $Z := \phi \cdot 2^R$, despite constant, is too large: $\text{SE}[Z] > 1$
- Second idea: repeat several times to reduce variance and improve precision
- Problem: using m hash functions to generate m streams is too costly and it's very difficult to guarantee independence between the hash values

Stochastic averaging

- The standard error of $Z := \phi \cdot 2^R$, despite constant, is too large: SE $[Z] > 1$
- Second idea: repeat several times to reduce variance and improve precision
- Problem: using m hash functions to generate m streams is too costly and it's very difficult to guarantee independence between the hash values

Stochastic averaging

- The standard error of $Z := \phi \cdot 2^R$, despite constant, is too large: SE $[Z] > 1$
- Second idea: repeat several times to reduce variance and improve precision
- Problem: using m hash functions to generate m streams is too costly and it's very difficult to guarantee independence between the hash values

Stochastic averaging

- Use the first $\log_2 m$ bits of each hash value to "redirect" it (the remaining bits) to one of the m substreams $\rightarrow$ stochastic averaging
- Obtain m observables $R_1, R_2, \ldots, R_m$, one from each substream
- Each R_i gives an estimation for the cardinality of the i-th substream, namely, R_i estimates n/m; the mean value $\overline{R} = 1/m \sum R_i$ also estimates n/m

Stochastic averaging

- Use the first $\log_2 m$ bits of each hash value to "redirect" it (the remaining bits) to one of the m substreams $\rightarrow$ stochastic averaging
- Obtain m observables $R_1, R_2, \ldots, R_m$, one from each substream
- Each R_i gives an estimation for the cardinality of the i-th substream, namely, R_i estimates n/m; the mean value $\overline{R} = 1/m \sum R_i$ also estimates n/m

Stochastic averaging

- Use the first $\log_2 m$ bits of each hash value to "redirect" it (the remaining bits) to one of the m substreams $\rightarrow$ stochastic averaging
- Obtain m observables $R_1, R_2, \ldots, R_m$, one from each substream
- Each R_i gives an estimation for the cardinality of the i-th substream, namely, R_i estimates n/m; the mean value $\overline{R} = 1/m \sum R_i$ also estimates n/m

Stochastic averaging

There are many different options to compute an estimator from the m observables

- Sum of estimators:

$$Z_1 := \phi_1(2^{R_1} + \ldots + 2^{R_m})$$

- Arithmetic mean of observables (as proposed by Flajolet & Martin):

$$Z_2 := m \cdot \phi_2 \cdot 2^{\frac{1}{m} \sum_{1 \leqslant i \leqslant m} R_i}$$

Stochastic averaging

- Harmonic mean (keep tuned):

$$Z_3 := \phi_3 \cdot \frac{m^2}{2^{-R_1} + 2^{-R_2} + \ldots + 2^{-R_m}}$$

Since $2^{-R_i} \approx m/n$, the second factor gives $\approx m^2/(m^2/n) = n$

Stochastic averaging

- All the strategies above yield a standard error of the form

$$\frac{c}{\sqrt{m}} + \text{l.o.t.}$$

  Larger memory $\Rightarrow$ improved precision!

- In *probabilistic counting* the authors used the arithmetic mean of observables

$$\text{SE}\left[Z_{\text{ProbCount}}\right] \approx \frac{0.78}{\sqrt{m}}$$

Stochastic averaging

- All the strategies above yield a standard error of the form

$$\frac{c}{\sqrt{m}} + \text{l.o.t.}$$

  Larger memory $\Rightarrow$ improved precision!

- In *probabilistic counting* the authors used the arithmetic mean of observables

$$\text{SE}\left[Z_{\text{ProbCount}}\right] \approx \frac{0.78}{\sqrt{m}}$$

Part I

Cardinality Estimation

LogLog & HyperLogLog

M. Durand

- Durand & Flajolet (2003) realized that the bitmaps ($\Theta(\log n)$ bits) used by *Probabilistic Counting* can be avoided and propose as observable the largest R such that the pattern $0.0^{R-1}1$ appears
- The new observable is similar to that of *Probabilistic Counting* but not equal: $R(\text{LogLog}) \geqslant R(\text{ProbCount})$

Example

Observed patterns: 0.1101..., 0.010..., 0.0011 ..., 0.00001...

$R(\text{LogLog}) = 5$, $\quad R(\text{ProbCount}) = 3$

LogLog & HyperLogLog

M. Durand

- Durand & Flajolet (2003) realized that the bitmaps ($\Theta(\log n)$ bits) used by *Probabilistic Counting* can be avoided and propose as observable the largest R such that the pattern $0.0^{R-1}1$ appears
- The new observable is similar to that of *Probabilistic Counting* but not equal: $R(\text{LogLog}) \geqslant R(\text{ProbCount})$

Example

Observed patterns: $0.1101\ldots$, $0.010\ldots$, $0.0011\ \ldots$, $0.00001\ldots$

$R(\text{LogLog}) = 5$, $R(\text{ProbCount}) = 3$

LogLog & HyperLogLog

M. Durand

- Durand & Flajolet (2003) realized that the bitmaps ($\Theta(\log n)$ bits) used by *Probabilistic Counting* can be avoided and propose as observable the largest R such that the pattern $0.0^{R-1}1$ appears
- The new observable is similar to that of *Probabilistic Counting* but not equal: $R(\text{LogLog}) \geqslant R(\text{ProbCount})$

Example

Observed patterns: 0.1101..., 0.010..., 0.0011 ...,
0.00001...
$R(\text{LogLog}) = 5$, $R(\text{ProbCount}) = 3$

LogLog & HyperLogLog

- The new observable is simpler to obtain: keep updated the largest R seen so far: $R := \max\{R, p\} \Rightarrow$ only $\Theta(\log\log n)$ bits needed, since $\mathbb{E}[R] = \Theta(\log n)$!
- We have $\mathbb{E}[R] \sim \log_2 n$, but $\mathbb{E}\left[2^R\right] = +\infty$, *stochastic averaging* comes to rescue!
- For LogLog, Durand & Flajolet propose

$$Z_{\text{LogLog}} := \alpha_m \cdot m \cdot 2^{\frac{1}{m}\sum_{1 \leqslant i \leqslant m} R_i}$$

LogLog & HyperLogLog

- The new observable is simpler to obtain: keep updated the largest R seen so far: $R := \max\{R, p\} \Rightarrow$ only $\Theta(\log \log n)$ bits needed, since $\mathbb{E}[R] = \Theta(\log n)$!
- We have $\mathbb{E}[R] \sim \log_2 n$, but $\mathbb{E}\left[2^R\right] = +\infty$, *stochastic averaging* comes to rescue!
- For LogLog, Durand & Flajolet propose

$$Z_{\text{LogLog}} := \alpha_m \cdot m \cdot 2^{\frac{1}{m} \sum_{1 \leqslant i \leqslant m} R_i}$$

LogLog & HyperLogLog

- The new observable is simpler to obtain: keep updated the largest R seen so far: $R := \max\{R, p\} \Rightarrow$ only $\Theta(\log\log n)$ bits needed, since $\mathbb{E}[R] = \Theta(\log n)$!
- We have $\mathbb{E}[R] \sim \log_2 n$, but $\mathbb{E}\left[2^R\right] = +\infty$, *stochastic averaging* comes to rescue!
- For LogLog, Durand & Flajolet propose

$$Z_{\text{LogLog}} := \alpha_m \cdot m \cdot 2^{\frac{1}{m}\sum_{1\leqslant i\leqslant m} R_i}$$

LogLog & HyperLogLog

- The mathematical analysis gives for the correcting factor

$$\alpha_m = \left(\Gamma(-1/m) \frac{1 - 2^{1/m}}{\ln 2} \right)^{-m}$$

  that guarantees that $\mathbb{E}[Z] = n + \mathrm{l.o.t.}$ (asymptotically unbiased) and the standard error is

$$\mathrm{SE}\left[Z_{\mathrm{LogLog}}\right] \approx \frac{1.30}{\sqrt{m}}$$

- Only m counters of size $\log_2 \log_2(n/m)$ bits needed: Ex.: $m = 2048 = 2^{11}$ counters, 5 bits each (1.25 Kbyte in total), are enough to give precise cardinality estimations for n up to $2^{27} \approx 10^8$, with an standard error less than 4%

LogLog & HyperLogLog

- The mathematical analysis gives for the correcting factor

$$\alpha_m = \left(\Gamma(-1/m)\frac{1 - 2^{1/m}}{\ln 2}\right)^{-m}$$

that guarantees that $\mathbb{E}[Z] = n + \text{l.o.t.}$ (asymptotically unbiased) and the standard error is

$$\text{SE}\left[Z_{\text{LogLog}}\right] \approx \frac{1.30}{\sqrt{m}}$$

- Only m counters of size $\log_2 \log_2(n/m)$ bits needed: Ex.: $m = 2048 = 2^{11}$ counters, 5 bits each (1.25 Kbyte in total), are enough to give precise cardinality estimations for n up to $2^{27} \approx 10^8$, with an standard error less than 4%

LogLog & HyperLogLog

É. Fusy

O. Gandouet

F. Meunier

- Flajolet, Fusy, Gandouet & Meunier conceived in 2007 the best algorithm known (cif. Flajolet's *keynote speech* in ITC Paris 2009)
- Briefly: HyperLogLog combines the LogLog observables R_i using the harmonic mean instead of the arithmetic mean

$$\mathrm{SE}\left[Z_{\mathrm{HyperLogLog}}\right] \approx \frac{1.03}{\sqrt{m}}$$

LogLog & HyperLogLog

É. Fusy

O. Gandouet

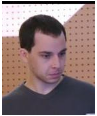
F. Meunier

- Flajolet, Fusy, Gandouet & Meunier conceived in 2007 the best algorithm known (cif. Flajolet's *keynote speech* in ITC Paris 2009)
- Briefly: HyperLogLog combines the LogLog observables R_i using the harmonic mean instead of the arithmetic mean

$$\text{SE}\left[Z_{\text{HyperLogLog}}\right] \approx \frac{1.03}{\sqrt{m}}$$

LogLog & HyperLogLog

P. Chassaing

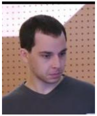


L. Gérin

- The idea of HyperLogLog stems from the analytical study of Chassaing & Gérin (2006) to show the optimal way to combine observables, but in their study the observables were the k-th order statistics of each substream (next!)
- They proved that the optimal way to combine them is to use the harmonic mean

LogLog & HyperLogLog


P. Chassaing


L. Gérin

- The idea of HyperLogLog stems from the analytical study of Chassaing & Gérin (2006) to show the optimal way to combine observables, but in their study the observables were the k-th order statistics of each substream (next!)
- They proved that the optimal way to combine them is to use the harmonic mean

Part I

Cardinality Estimation

Order Statistics

- Bar-Yossef, Kumar & Sivakumar (2002); Bar-Yossef, Jayram, Kumar, Sivakumar & Trevisan (2002) have proposed to use the k-th order statistic $Y_{(k)}$ to estimate cardinality (KMV algorithm); for a set of n random numbers, independent and uniformly distributed in $(0, 1)$

$$\mathbb{E}\left[Y_{(k)}\right] = \frac{k}{n+1} \Rightarrow \mathbb{E}\left[\frac{k-1}{Y_{(k)}}\right] = n$$

- Giroire (2005, 2009) also proposes several estimators combining order statistics via *stochastic averaging*

Order Statistics

- Bar-Yossef, Kumar & Sivakumar (2002); Bar-Yossef, Jayram, Kumar, Sivakumar & Trevisan (2002) have proposed to use the k-th order statistic $Y_{(k)}$ to estimate cardinality (KMV algorithm); for a set of n random numbers, independent and uniformly distributed in $(0, 1)$

$$\mathbb{E}\left[Y_{(k)}\right] = \frac{k}{n+1} \Rightarrow \mathbb{E}\left[\frac{k-1}{Y_{(k)}}\right] = n$$

- Giroire (2005, 2009) also proposes several estimators combining order statistics via *stochastic averaging*

Order Statistics

J. Lumbroso

- The minimum of the set $(k = 1)$ does not allow a feasible estimator, but again *stochastic averaging* comes to rescue
- Lumbroso uses the mean of m minima, one for each substream

$$Z_{\text{MinCount}} := \frac{m(m-1)}{M_1 + \ldots + M_m},$$

where M_i is the minimum hash value of the i-th substream

Order Statistics

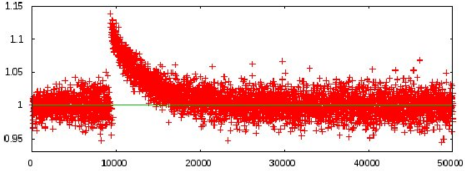
J. Lumbroso

- The minimum of the set $(k = 1)$ does not allow a feasible estimator, but again *stochastic averaging* comes to rescue
- Lumbroso uses the mean of m minima, one for each substream

$$Z_{\text{MinCount}} := \frac{m(m-1)}{M_1 + \ldots + M_m},$$

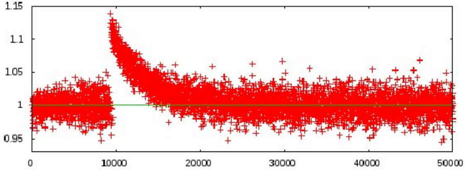
where M_i is the minimum hash value of the i-th substream

Order Statistics



- MinCount is an unbiased estimator with standard error $1/\sqrt{m-2}$
- Lumbroso also succeeds to compute the probability distribution of Z_{MinCount} and the small corrections needed to estimate small cardinalities (too few elements hashing to one particular substream)

Order Statistics

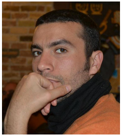


- MinCount is an unbiased estimator with standard error $1/\sqrt{m-2}$
- Lumbroso also succeeds to compute the probability distribution of Z_{MinCount} and the small corrections needed to estimate small cardinalities (too few elements hashing to one particular substream)

Part I

Cardinality Estimation

Recordinality

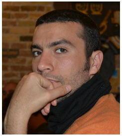


A. Helmi

A. Viola

- RECORDINALITY (Helmi, Lumbroso, M., Viola, 2012) is the most recent proposed estimator (already 10 years ago!), loosely related to order statistics, but based in completely different principles and it exhibits several unique features
- Some of the ideas where very useful to develop Affirmative Sampling, stay tuned!

Recordinality



A. Helmi

A. Viola

- RECORDINALITY (Helmi, Lumbroso, M., Viola, 2012) is the most recent proposed estimator (already 10 years ago!), loosely related to order statistics, but based in completely different principles and it exhibits several unique features
- Some of the ideas where very useful to develop Affirmative Sampling, stay tuned!

Recordinality

Given the data stream $\mathcal{Z} = z_1, \dots, z_N$, consider the substream

$$\mathcal{Z}_u = x_1, \dots, x_n$$

with x_i the i-th distinct element in $\mathcal{Z}$ in order of appearence

Example

$$\mathcal{Z} = 3, 14, 1, 593, 26, 53, 5, 8979, 3, 23, 8, 46, 26, 433, 8, 3, 2, 8$$
$$\mathcal{Z}_u = 3, 14, 1, 593, 26, 53, 5, 8979, 23, 8, 46, 433, 2$$

Introduction

Applying a hash function h on $\mathcal{Z}_u$ allows us to see the data stream as a permutation $\mathcal{P}_u$:

Example

$$\mathcal{Z} = 3, 14, 1, 593, 26, 53, 5, 8979, 3, 23, 8, 46, 26, 433, 8, 3, 2, 8$$

$$\mathcal{Z}_u = 3, 14, 1, 593, 26, 53, 5, 8979, 23, 8, 46, 433, 2$$

$$\mathcal{P}_u = 3, 6, 1, 12, 8, 10, 4, 13, 7, 5, 9, 11, 2$$

To simplify this example take $h(x) = x$

Recordinality

- RECORDINALITY counts the number of records (more generally, k-records) in the sequence of hash values
- It depends in the underlying permutation of the first occurrences of distinct values, very different from the other estimators
- If we assume that the first occurrences of distinct values form a random permutation then there's no need for hash values!

Recordinality

- RECORDINALITY counts the number of records (more generally, k-records) in the sequence of hash values
- It depends in the underlying permutation of the first occurrences of distinct values, very different from the other estimators
- If we assume that the first occurrences of distinct values form a random permutation then there's no need for hash values!

Recordinality

- RECORDINALITY counts the number of records (more generally, k-records) in the sequence of hash values
- It depends in the underlying permutation of the first occurrences of distinct values, very different from the other estimators
- If we assume that the first occurrences of distinct values form a random permutation then there's no need for hash values!

Recordinality

- $\sigma(i)$ is a record of the permutation σ if $\sigma(i) > \sigma(j)$ for all $j < i$
- This notion is generalized to k-records: $\sigma(i)$ is a k-record if there are at most $k - 1$ elements $\sigma(j)$ larger than $\sigma(i)$ for $j < i$; in other words, $\sigma(i)$ is among the k largest elements in $\sigma(1), \ldots, \sigma(i)$

Example

This example permutation contains six 2-records

$$\mathcal{P}_u = 3, 6, 1, 12, 8, 10, 4, 13, 7, 5, 9, 11, 2$$

Recordinality

- $\sigma(i)$ is a record of the permutation σ if $\sigma(i) > \sigma(j)$ for all $j < i$
- This notion is generalized to k-records: $\sigma(i)$ is a k-record if there are at most $k-1$ elements $\sigma(j)$ larger than $\sigma(i)$ for $j < i$; in other words, $\sigma(i)$ is among the k largest elements in $\sigma(1), \ldots, \sigma(i)$

Example

This example permutation contains six 2-records

$$\mathcal{P}_u = 3, 6, 1, 12, 8, 10, 4, 13, 7, 5, 9, 11, 2$$

Recordinality

- $\sigma(i)$ is a record of the permutation σ if $\sigma(i) > \sigma(j)$ for all $j < i$
- This notion is generalized to k-records: $\sigma(i)$ is a k-record if there are at most $k - 1$ elements $\sigma(j)$ larger than $\sigma(i)$ for $j < i$; in other words, $\sigma(i)$ is among the k largest elements in $\sigma(1), \ldots, \sigma(i)$

Example

This example permutation contains six 2-records

$$\mathcal{P}_u = 3, 6, 1, 12, 8, 10, 4, 13, 7, 5, 9, 11, 2$$

Recordinality

procedure RECORDINALITY($\mathcal{Z}$, k)
  fill $\mathcal{S}$ with the first k distinct elements (hash values) of the stream $\mathcal{Z}$
  $R \leftarrow k$
  for all $z \in \mathcal{Z}$ **do**
    $y \leftarrow h(z)$
    if $y > \min\{h(x) \mid x \in \mathcal{S}\} \wedge z \notin \mathcal{S}$ **then**
      $z^* \leftarrow$ the element in $\mathcal{S}$ with min. hash value
      $R \leftarrow R + 1$; $\mathcal{S} \leftarrow \mathcal{S} \cup \{z\} \setminus z^*$
    end if
  end for
  return $Z = k\left(1 + \frac{1}{k}\right)^{R-k+1} - 1$
end procedure

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

Analysis of k-Records

The behavior of $R = R_n$, the number of k-records in a random permutation of size n, is very well understood[1]

$$\mathbb{E}[R] = k(H_n - H_k + 1) = k \ln(n/k) + O(1)$$

Likewise

$$\mathbb{V}[R] = k(H_n - H_k) - k^2(H_n^{(2)} - H_k^{(2)}) = k \ln(n/k) + O(1)$$

and we also know exact and asymptotic estimates for $\mathbb{P}[R = j]$.

---

[1] $H_n = 1 + 1/2 + 1/3 + \cdots + 1/n \sim \ln n + \mathcal{O}(1)$ denotes the n-th harmonic number, and $H_n^{(2)} = 1 + 1/4 + 1/9 + \cdots + 1/n^2 \leqslant \pi^2/6$.

The Estimator for Recordinality

Let us assume for the moment that $k \leqslant R \leqslant n$. If $R < k$ then we are sure that $n = R$. Otherwise, since $\mathbb{E}[R] = k \ln(n/k) + O(1)$ we can take

$$Z = \exp(\phi \cdot R)$$

for some correcting factor ϕ to be determined and such that $\mathbb{E}[Z]$ is (asymptotically?) n. Our knowledge of the probability distribution of R furnishes the exact form for Z.

The Estimator for Recordinality

Theorem

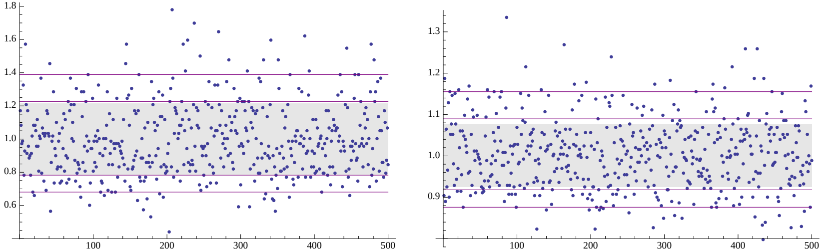
Let R be the number of k-records seen while processing the data stream $\mathcal{Z}$. Then

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

is an unbiased estimator of the cardinality (number of distinct elements) of $\mathcal{Z}$, that is,

$$\mathbb{E}[Z] = n$$

Recordinality in Practice



Two plots showing the accuracy of 500 estimates of the number of distinct elements contained in Shakespeare's *A Midsummer Night's Dream*. Left: $k = 64$. Right: $k = 256$. Above the top and below the bottom line: 5% of the estimates. Area within centermost lines: 70% estimates. Gray rectangle: area within one standard deviation from the mean.

Recordinality in Practice

k	Recordinality		*Adaptive Sampling*		k-th Order Statistic		HyperLogLog	
	Avg.	Error	Avg.	Error	Avg.	Error	Avg.	Error
4	2737	1.04	3047	0.70	4050	0.89	2926	0.61
8	2811	0.73	3014	0.41	3495	0.44	3147	0.42
16	3040	0.54	3012	0.31	3219	0.28	2981	0.26
32	3010	0.34	3078	0.20	3159	0.18	3001	0.18
64	3020	0.22	3020	0.15	3071	0.12	3011	0.13
128	3042	0.14	3032	0.11	3070	0.10	3031	0.09
256	3044	0.08	3027	0.07	3037	0.06	3025	0.06
512	3043	0.04	3043	0.05	3046	0.04	2975	0.08

Table: Estimating the number of distinct elements in Shakespeare's *A Midsummer Night's Dream* ($n = 3031$). Normalized average and the empirical standard deviation divided by n. 10 000 simulations.

Recordinality in Practice

k	Recordinality		*Adaptive Sampling*		k-th Order Statistic		HyperLogLog	
	Avg.	Error	Avg.	Error	Avg.	Error	Avg.	Error
4	43658	1.19	59474	0.94	81724	1.30	44302	0.42
8	35230	0.52	47432	0.38	57028	0.41	52905	0.39
16	57723	0.98	49889	0.29	52990	0.23	51522	0.27
32	48686	0.45	49480	0.23	50556	0.18	48009	0.16
64	47617	0.34	50524	0.14	51146	0.13	49345	0.14
128	50097	0.17	50452	0.09	50947	0.08	51531	0.10
256	51742	0.11	50857	0.06	50348	0.06	49287	0.06
512	49496	0.09	49920	0.06	50084	0.04	49916	0.04

Table: Experiments for a random stream containg $n = 50\,000$ distinct elements—here 25 000 simulations were run.

Part II

Distinct Sampling and Applications

Drawing Random Samples

- In a random sample from the data stream (e.g., using the reservoir method) each distinct element x_j appears with relative frequency in the sample equal to its relative frequency f_j/N in the data stream ⇒ needle-on-a-haystack
- Elements of low frequency will seldom be sampled, and we cannot keep exact counts as we don't know if the sampled elements have been "monitorized" from the beginning

Drawing Random Samples

- In a random sample from the data stream (e.g., using the reservoir method) each distinct element x_j appears with relative frequency in the sample equal to its relative frequency f_j/N in the data stream $\Rightarrow$ needle-on-a-haystack
- Elements of low frequency will seldom be sampled, and we cannot keep exact counts as we don't know if the sampled elements have been "monitorized" from the beginning

Drawing Random Samples

- The distinct sampling problem is to draw a random sample of distinct elements and it has many applications in data stream analysis
- For example, to estimate the number of k-elephants or k-mice in the stream we can draw a random sample of S distinct elements, together with their frequency counts
- Let S_P be the number of mice (or elephants) in the sample, and n_P the number of mice (or elephants) in the data stream. Then

$$\mathbb{E}\left[\frac{S_P}{S}\right] = \frac{n_P}{n}$$

Drawing Random Samples

- The distinct sampling problem is to draw a random sample of distinct elements and it has many applications in data stream analysis
- For example, to estimate the number of k-elephants or k-mice in the stream we can draw a random sample of S distinct elements, together with their frequency counts
- Let S_P be the number of mice (or elephants) in the sample, and n_P the number of mice (or elephants) in the data stream. Then

$$\mathbb{E}\left[\frac{S_P}{S}\right] = \frac{n_P}{n}$$

Drawing Random Samples

- The distinct sampling problem is to draw a random sample of distinct elements and it has many applications in data stream analysis
- For example, to estimate the number of k-elephants or k-mice in the stream we can draw a random sample of S distinct elements, together with their frequency counts
- Let S_P be the number of mice (or elephants) in the sample, and n_P the number of mice (or elephants) in the data stream. Then

$$\mathbb{E}\left[\frac{S_P}{S}\right] = \frac{n_P}{n}$$

Drawing Random Samples

Let P some property.

- $n = \#$ of distinct elements in $\mathcal{Z}$
- $n_P = \#$ of distinct elements in $\mathcal{Z}$ that satisfy P
- $S =$ size of the sample $\Leftarrow$ in general, a r.v., assume $2 \leqslant S \leqslant n$
- $S_P = \#$ of elements in the sample that satisfy P

Theorem

1. $\mathbb{E}\left[\frac{S_P}{S}\right] = \frac{n_P}{n}$
2. $\mathbb{V}\left[\frac{S_P}{S}\right] \sim \frac{n_p}{n} \cdot \left(1 - \frac{n_p}{n}\right) \cdot \mathbb{E}\left[\frac{1}{S}\right]$

Drawing Random Samples

Let P some property.

- $n = \#$ of distinct elements in $\mathcal{Z}$
- $n_P = \#$ of distinct elements in $\mathcal{Z}$ that satisfy P
- $S =$ size of the sample $\Leftarrow$ in general, a r.v., assume $2 \leqslant S \leqslant n$
- $S_P = \#$ of elements in the sample that satisfy P

Theorem

1. $\mathbb{E}\left[\frac{S_P}{S}\right] = \frac{n_P}{n}$
2. $\mathbb{V}\left[\frac{S_P}{S}\right] \sim \frac{n_p}{n} \cdot \left(1 - \frac{n_p}{n}\right) \cdot \mathbb{E}\left[\frac{1}{S}\right]$

Part II

Distinct Sampling and Applications

Adaptive Sampling

M. Wegman

G. Louchard

- *Adaptive sampling* (Wegman, 1980; Flajolet, 1990; Louchard, 1997) is the first algorithm proposed specifically for distinct sampling
- It also gives an estimation of the cardinality, as the size S of the returned sample is itself a random variable, but it is always bounded by a fixed constant $maxS$

Adaptive Sampling

```
procedure ADAPTIVESAMPLING(𝒵, maxS)
    𝒮 ← ∅; p ← 0
    for z ∈ 𝒵 do
        if hash(z) = 0^p ... ∧ z ∉ 𝒮  then
            𝒮 ← 𝒮 ∪ {z}
            if |𝒮| > maxS then
                p ← p + 1
                𝒮 ← 𝒮 \ {z ∈ 𝒮 | h(z) = 0^{p−1}1 ...} ▷ Filter 𝒮
            end if
        end if
    end for
    return 𝒮
end procedure
```

The set $\mathcal{S}$ is a random sample (because we can assume hash values behave as random uniform numbers) of $S = |\mathcal{S}|$ distinct elements; if n is large enough, $maxS/2 \leqslant \mathbb{E}[S] \leqslant maxS$

Adaptive Sampling

At the end of the algorithm, S is the number of distinct elemnts with hash value starting $.0^p \equiv$ the number of strings in the subtree rooted at 0^p in a binary trie for n random binary strings. There are 2^p subtrees rooted at depth p

$$S = |\mathcal{S}| \approx n/2^p \Rightarrow \mathbb{E}[2^p \cdot S] \approx n$$

Part II

Distinct Sampling and Applications

Distinct Sampling in Recordinality and Order Statistics

- Recordinality and KMV collect the elements with the k largest (smallest) hash values
- Such k elements constitute a random sample of k distinct elements, because hash values behave as random numbers; but the value k is fixed in advance and might be too small for the sample to be representative
- Recordinality can be easily adapted to collect random samples of expected size $\Theta(\log n)$ or $\Theta(n^{\alpha})$, with $0 < \alpha < 1$ and without prior knowledge of n! $\Rightarrow$ Affirmative Sampling $\Rightarrow$ variable-size samples, growing with n, better precision in inferences about the full data stream

Distinct Sampling in Recordinality and Order Statistics

- Recordinality and KMV collect the elements with the k largest (smallest) hash values
- Such k elements constitute a random sample of k distinct elements, because hash values behave as random numbers; but the value k is fixed in advance and might be too small for the sample to be representative
- Recordinality can be easily adapted to collect random samples of expected size $\Theta(\log n)$ or $\Theta(n^\alpha)$, with $0 < \alpha < 1$ and without prior knowledge of n! $\Rightarrow$ Affirmative Sampling $\Rightarrow$ variable-size samples, growing with n, better precision in inferences about the full data stream

Distinct Sampling in Recordinality and Order Statistics

- Recordinality and KMV collect the elements with the k largest (smallest) hash values
- Such k elements constitute a random sample of k distinct elements, because hash values behave as random numbers; but the value k is fixed in advance and might be too small for the sample to be representative
- Recordinality can be easily adapted to collect random samples of expected size $\Theta(\log n)$ or $\Theta(n^{\alpha})$, with $0 < \alpha < 1$ and without prior knowledge of n! $\Rightarrow$ Affirmative Sampling $\Rightarrow$ variable-size samples, growing with n, better precision in inferences about the full data stream

Affirmative Sampling

- Early ideas date back to the original paper on Recordinality (2012); developed and analyzed in detail in (Lumbroso, M., 2019)
- The larger the cardinality (n) the larger the samples $\Rightarrow$ samples better represent diversity
- All distinct elements have the same opportunity to be sampled, and if sampled they can be “monitorized” from their first appearance

Affirmative Sampling

```
procedure AFFIRMATIVESAMPLING(k, Z)
    fill S with the first k distinct elements
    (and hash values) of the stream Z
    for z ∈ Z do
        if z ∈ S then
            Update z stats; continue
        end if
        if HASH(z) > k-th largest hash value in S then
            S ← S ∪ {z}
        else if HASH(()z) > min hash value in S then
            ▷ replace elem of min. hash in S with z
            S ← S \ {elem. with min. hash in S} ∪ {z}
        end if
    end for
    return S
end procedure
```

Affirmative Sampling

- The size S of the sample $\mathcal{S}$ is a random variable = the number of k-records in a random permutation of size n $\Rightarrow$ $\mathbb{E}[S] = k \ln(n/k) + \mathcal{O}(1)$
- The sample does not contain the k-records, but the S elements with the largest hash values seen so far $\Rightarrow$ $\mathcal{S}$ is a random sample
- If $x \in \mathcal{S}$ then x has been added to $\mathcal{S}$ in its very first occurrence and it has remained in $\mathcal{S}$ ever since $\Rightarrow$ can collect exact stats (e.g. frequency counts) for x

Affirmative Sampling

- We also understand fairly well F = number of times an element substitutes another in the sample (not a k-record, but larger than some k-record):

$$\mathbb{E}[F] = k \ln^2(n/k) + \text{l.o.t.}$$

- Expected cost $C_{N,n}$ of Affirmative Sampling

$$\begin{aligned}\mathbb{E}[C_{N,n}] &= \Theta(N + (\mathbb{E}[S] + \mathbb{E}[F]) \log \mathbb{E}[S]) \\ &= \Theta(N + (\log^2 n) \cdot (\log \log n))\end{aligned}$$

using appropriate data structures for the sample $\mathcal{S}$

Part II

Distinct Sampling and Applications

Similarity Estimation

Consider two data streams $\mathcal{Z}_A$ and $\mathcal{Z}_B$. Let A and B denote their respective sets of distinct elements. Similarity between the two sets is often measured by their Jaccard index

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

The containment index measures how much "$A \subseteq B$" and it is given by

$$c(A, B) = \frac{|A \cap B|}{|A|}$$

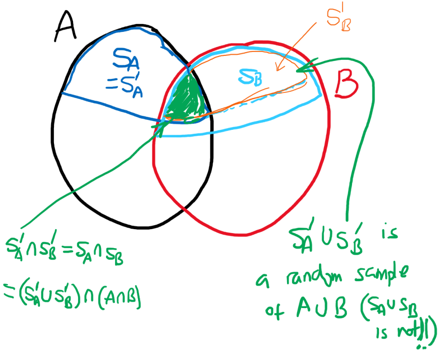
Similarity Estimation

We can estimate similarity and containment from random samples S_A and S_B of the two streams. If the samples are drawn using Affirmative Sampling then

Theorem

1. $\mathbb{E}\left[J(S'_A, S'_B)\right] = J(A, B) = \frac{|A \cap B|}{|A \cup B|}$
2. $\mathbb{V}\left[J(S'_A, S'_B)\right] \sim \frac{J(A,B) \cdot (1 - J(A,B))}{k \ln(|A \cup B|/k)}$

Similarity Estimation



Estimating the size of the intersection

We can estimate the size of the intersection with:

$$Z_1 = \frac{|S_A \cap S_B|}{|S_A|} \cdot \left(k \left(1 + \frac{1}{k} \right)^{|S_A|-k+1} - 1 \right)$$

$$Z_2 = \frac{|S_A \cap S_B|}{|S_A|} \cdot \frac{|S_A| - 1}{1 - M_{S_A}}, \qquad M_{S_A} = \min\{h(z) \mid z \in S_A\}$$

$$\mathbb{E}[Z_1] = \mathbb{E}[Z_2] = |A \cap B|$$

N.B. No need to “filter” the samples

Other similarity measures

Jaccard's index	$\frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$
Otsuka-Ochiai (a.k.a. Cosine)	$\frac{\lvert A \cap B \rvert}{\sqrt{\lvert A \rvert \cdot \lvert B \rvert}}$
Sørensen-Dice	$2\frac{\lvert A \cap B \rvert}{\lvert A \rvert + \lvert B \rvert}$
Kulczynski 1	$\frac{\lvert A \cap B \rvert}{\lvert A \triangle B \rvert}$
Kulczynski 2	$\frac{1}{2}\left(\frac{\lvert A \cap B \rvert}{\lvert A \rvert} + \frac{\lvert A \cap B \rvert}{\lvert B \rvert}\right)$
Simpson	$\frac{\lvert A \cap B \rvert}{\min(\lvert A \rvert, \lvert B \rvert)}$
Braun-Blanquet	$\frac{\lvert A \cap B \rvert}{\max(\lvert A \rvert, \lvert B \rvert)}$
Correlation	$\cos^2(A, B) = \frac{\lvert A \cap B \rvert^2}{\lvert A \rvert \cdot \lvert B \rvert}$
...	...

Other similarity measures

The same proof that works for Jaccard's similarity also works for containment and many other similarity measures:

1. $\mathbb{E}[c(S_A, S_B)] = c(A, B) = |A \cap B|/|A|$
2. If σ is any of Jaccard, Simpson, Braun-Blanquet, Kulczynski 2, correlation or Sørensen-Dice:

$$\mathbb{E}\left[\sigma(S'_A, S'_B)\right] = \sigma(A, B)$$

3. We conjecture this also holds (asymptotically) for cosine and Kulczynski 1 and maybe others

To Know More

[1] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan.
Counting Distinct Elements in a Data Stream.
Randomization and Approximation Techniques (RANDOM), pages 1–10. 2002.

[2] Andrei Broder.
On the resemblance and containment of documents.
Proc. Compression and Complexity of Sequences (SEQUENCES), pages 21–29. 1997.

[3] Marianne Durand and Philippe Flajolet.
LogLog Counting of Large Cardinalities.
Proc. European Symposium on Algorithms (ESA), volume 2832 of *Lecture Notes in Computer Science*, pages 605–617, 2003.

To Know More

[4] Philippe Chassaing and Lucas Gerin.
Efficient Estimation of the Cardinality of Large Data Sets.
Proc. Int. Col. Mathematics and Computer Science (MathInfo), pages 419–422, 2007.

[5] Philippe Flajolet.
On adaptive sampling.
Computing, 34:391–400, 1990.

[6] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier.
HyperLoglog: the analysis of a near-optimal cardinality estimation algorithm.
Proceedings of Int. Conf. Analysis of Algorithms (AofA), pages 127–146, 2007.

To Know More

[7] Philippe Flajolet and G. Nigel Martin.
Probabilistic Counting Algorithms for Data Base Applications.
Journal of Computer and System Sciences, 31(2):182–209, 1985.

[8] A. Helmi, J. Lumbroso, C. Martínez, and A. Viola.
Counting distinct elements in data streams: the random permutation viewpoint.
Proc. of Int. Conf. Analysis of Algorithms (AofA), pages 323–338, 2012.

[9] Jérémie Lumbroso.
An optimal cardinality estimation algorithm based on order statistics and its full analysis.
In *Proc. Analysis of Algorithms (AofA)*, pages 489–504, 2010.

To Know More

[10] Jérémie Lumbroso.
How Flajolet Processed Stream with Coin Flips.
`arXiv:1805.00612v1 [cs.DS] 2 May 2018`.
December 2013.

[11] M. Monenizadeh and D. Woodruff.
1-Pass Relative-Error L_p*-Sampling with Applications*.
In *Proc. Symp. Discrete Algorithms (SODA)*, pages 1143–1160, 2010.

[12] S. Muthu Muthukrishnan.
Data streams: Algorithms and applications.
Foundations and Trends in Theoretical Computer Science, 1(2):117–236, 2005.