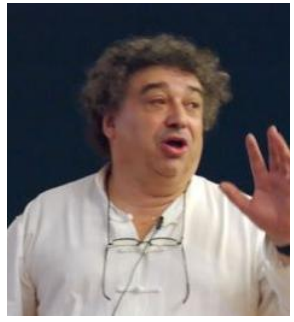
Techniques probabilistes et combinatoires pour l'analyse des flux de données

Dédié à la mémoire de Philippe Flajolet (1948-2011)



Conrado Martínez
Universitat Politècnica de Catalunya

Journées ALÉA, CIRM, Marseille-Luminy
Mars 2012

Introduction

- Un flux de données est une séquence (très longue)

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

  d'éléments tirés d'un (grand) domaine $\mathcal{U}$ ($s_i \in \mathcal{U}$)
- Le but: calculer $y = y(\mathcal{S})$, mais ...

Introduction

- Un flux de données est une séquence (très longue)

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

  d'éléments tirés d'un (grand) domaine $\mathcal{U}$ ($s_i \in \mathcal{U}$)
- Le but: calculer $y = y(\mathcal{S})$, mais ...

Introduction

. . . sous certaines contraintes

- une seule passe sur les données
- court temps de traitement pour chaque élément
- une quantité limitée M de mémoire auxiliaire, $M \ll N$; idéalement $M = \Theta(1)$ où $M = \Theta(\log N)$
- aucune hypothèse statistique (sur les données)

Introduction

. . . sous certaines contraintes

- une seule passe sur les données
- court temps de traitement pour chaque élément
- une quantité limitée M de mémoire auxiliaire, $M \ll N$; idéalement $M = \Theta(1)$ où $M = \Theta(\log N)$
- aucune hypothèse statistique (sur les données)

Introduction

. . . sous certaines contraintes

- une seule passe sur les données
- court temps de traitement pour chaque élément
- une quantité limitée M de mémoire auxiliaire, $M \ll N$; idéalement $M = \Theta(1)$ où $M = \Theta(\log N)$
- aucune hypothèse statistique (sur les données)

Introduction

. . . sous certaines contraintes

- une seule passe sur les données
- court temps de traitement pour chaque élément
- une quantité limitée M de mémoire auxiliaire, $M \ll N$; idéalement $M = \Theta(1)$ où $M = \Theta(\log N)$
- aucune hypothèse statistique (sur les données)

Introduction

Il y a de nombreuses applications pour ce modèle:

- Analyse du trafic de réseau $\Rightarrow$ attaques DoS/DDoS, vers (*worms*), ...
- Optimisation des requêtes de base de données
- Récupération de l'information $\Rightarrow$ indice de similarité
- *Data mining*
- Et beaucoup plus ...

Introduction

Il y a de nombreuses applications pour ce modèle:

- Analyse du trafic de réseau $\Rightarrow$ attaques DoS/DDoS, vers (*worms*), ...
- Optimisation des requêtes de base de données
- Récupération de l'information $\Rightarrow$ indice de similarité
- *Data mining*
- Et beaucoup plus ...

Introduction

Il y a de nombreuses applications pour ce modèle:

- Analyse du trafic de réseau $\Rightarrow$ attaques DoS/DDoS, vers (*worms*), ...
- Optimisation des requêtes de base de données
- Récupération de l'information $\Rightarrow$ indice de similarité
- *Data mining*
- Et beaucoup plus ...

Introduction

Il y a de nombreuses applications pour ce modèle:

- Analyse du trafic de réseau $\Rightarrow$ attaques DoS/DDoS, vers (*worms*), ...
- Optimisation des requêtes de base de données
- Récupération de l'information $\Rightarrow$ indice de similarité
- *Data mining*
- Et beaucoup plus ...

Introduction

Il y a de nombreuses applications pour ce modèle:

- Analyse du trafic de réseau $\Rightarrow$ attaques DoS/DDoS, vers (*worms*), ...
- Optimisation des requêtes de base de données
- Récupération de l'information $\Rightarrow$ indice de similarité
- *Data mining*
- Et beaucoup plus ...

Introduction

On regarde $\mathcal{S}$ comme un multi-ensemble $\{w_1 \circ f_1, \ldots, w_n \circ f_n\}$, où

$$f_i = \text{la fréquence de l'élément } i\text{-ième distincte } w_i$$

- La cardinalité $\text{card}(\mathcal{S}) = n \leq N$
- Les moments de fréquence $F_p = \sum_{1 \leq i \leq n} f_i^P$
  (N.B. $n = F_0, N = F_1$)
- Les éléments w_i tels que $f_i \geq k$ (k-éléphants)
- Les éléments w_i tels que $f_i < k$ (k-souris)
- Les éléments w_i tels que $f_i \geq cN$, $0 < c < 1$ (c-icebergs)
- Les k éléments les plus fréquents
- ...

Introduction

On regarde $\mathcal{S}$ comme un multi-ensemble $\{w_1 \circ f_1, \ldots, w_n \circ f_n\}$, où

$$f_i = \text{la fréquence de l'élément } i\text{-ième distincte } w_i$$

- La cardinalité $\text{card}(\mathcal{S}) = n \leq N$
- Les moments de fréquence $F_p = \sum_{1 \leq i \leq n} f_i^P$
  (N.B. $n = F_0, N = F_1$)
- Les éléments w_i tels que $f_i \geq k$ (k-éléphants)
- Les éléments w_i tels que $f_i < k$ (k-souris)
- Les éléments w_i tels que $f_i \geq cN$, $0 < c < 1$ (c-icebergs)
- Les k éléments les plus fréquents
- ...

Introduction

On regarde $\mathcal{S}$ comme un multi-ensemble $\{w_1 \circ f_1, \ldots, w_n \circ f_n\}$, où

$$f_i = \text{la fréquence de l'élément } i\text{-ième distincte } w_i$$

- La cardinalité $\text{card}(\mathcal{S}) = n \leq N$
- Les moments de fréquence $F_p = \sum_{1 \leq i \leq n} f_i^P$
  (N.B. $n = F_0, N = F_1$)
- Les éléments w_i tels que $f_i \geq k$ (k-éléphants)
- Les éléments w_i tels que $f_i < k$ (k-souris)
- Les éléments w_i tels que $f_i \geq cN$, $0 < c < 1$ (c-icebergs)
- Les k éléments les plus fréquents
- ...

Introduction

On regarde $\mathcal{S}$ comme un multi-ensemble $\{w_1 \circ f_1, \ldots, w_n \circ f_n\}$, où

$$f_i = \text{la fréquence de l'élément } i\text{-ième distincte } w_i$$

- La cardinalité $\text{card}(\mathcal{S}) = n \leq N$
- Les moments de fréquence $F_p = \sum_{1 \leq i \leq n} f_i^P$
  (N.B. $n = F_0, N = F_1$)
- Les éléments w_i tels que $f_i \geq k$ (k-éléphants)
- Les éléments w_i tels que $f_i < k$ (k-souris)
- Les éléments w_i tels que $f_i \geq cN$, $0 < c < 1$ (c-icebergs)
- Les k éléments les plus fréquents
- ...

Introduction

On regarde $\mathcal{S}$ comme un multi-ensemble $\{w_1 \circ f_1, \ldots, w_n \circ f_n\}$, où

$$f_i = \text{la fréquence de l'élément } i\text{-ième distincte } w_i$$

- La cardinalité $\text{card}(\mathcal{S}) = n \leq N$
- Les moments de fréquence $F_p = \sum_{1 \leq i \leq n} f_i^P$
  (N.B. $n = F_0, N = F_1$)
- Les éléments w_i tels que $f_i \geq k$ (k-éléphants)
- Les éléments w_i tels que $f_i < k$ (k-souris)
- Les éléments w_i tels que $f_i \geq cN$, $0 < c < 1$ (c-icebergs)
- Les k éléments les plus fréquents
- ...

Introduction

On regarde $\mathcal{S}$ comme un multi-ensemble $\{w_1 \circ f_1, \ldots, w_n \circ f_n\}$, où

$$f_i = \text{la fréquence de l'élément } i\text{-ième distincte } w_i$$

- La cardinalité $\text{card}(\mathcal{S}) = n \leq N$
- Les moments de fréquence $F_p = \sum_{1 \leq i \leq n} f_i^P$
  (N.B. $n = F_0, N = F_1$)
- Les éléments w_i tels que $f_i \geq k$ (k-éléphants)
- Les éléments w_i tels que $f_i < k$ (k-souris)
- Les éléments w_i tels que $f_i \geq cN$, $0 < c < 1$ (c-icebergs)
- Les k éléments les plus fréquents
- ...

Introduction

On regarde $\mathcal{S}$ comme un multi-ensemble $\{w_1 \circ f_1, \ldots, w_n \circ f_n\}$, où

$$f_i = \text{la fréquence de l'élément } i\text{-ième distincte } w_i$$

- La cardinalité $\text{card}(\mathcal{S}) = n \leq N$
- Les moments de fréquence $F_p = \sum_{1 \leq i \leq n} f_i^P$
  (N.B. $n = F_0, N = F_1$)
- Les éléments w_i tels que $f_i \geq k$ (k-éléphants)
- Les éléments w_i tels que $f_i < k$ (k-souris)
- Les éléments w_i tels que $f_i \geq cN$, $0 < c < 1$ (c-icebergs)
- Les k éléments les plus fréquents
- ...

Introduction

Mémoire disponible très limitée $\Rightarrow$ solution exacte trop couteuse
$\Rightarrow$ algorithmes randomisés $\Rightarrow$ estimation $\hat{y}$ de la quantité y d'intérêt

- L'estimateur $\hat{y}$ est non biaisé

$$\mathrm{E}\left[\hat{y}\right] = y$$

- Il a une très bonne précision (erreur-type)

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

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

Introduction

Mémoire disponible très limitée $\Rightarrow$ solution exacte trop couteuse
$\Rightarrow$ algorithmes randomisés $\Rightarrow$ estimation $\hat{y}$ de la quantité y d'intérêt

- L'estimateur $\hat{y}$ est non biaisé

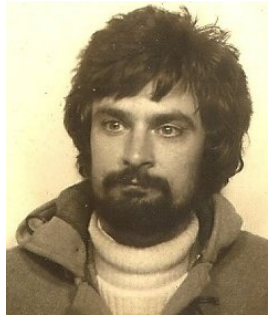
$$\mathrm{E}\left[\hat{y}\right] = y$$

- Il a une très bonne précision (erreur-type)

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

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

Probabilistic Counting



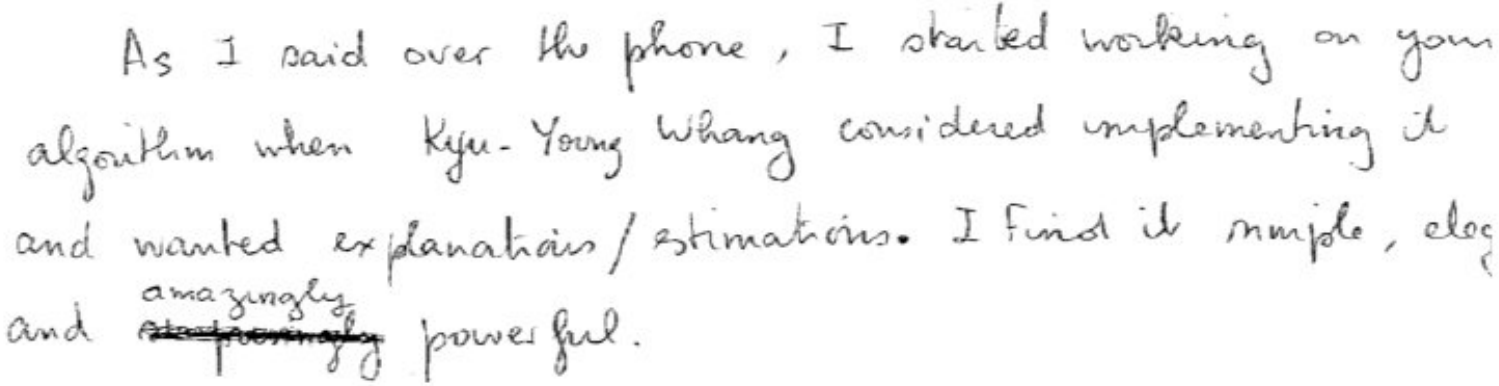
G.N. Martin

À la fin des années 70 G. Nigel N. Martin invente le probabilistic counting pour optimiser les requêtes de bases de données

Il corrige le biais qu'il a trouvé dans son estimateur, en bidouillant l'algorithme

Probabilistic Counting

Lorsque Flajolet découvre l'algorithme, il le met dans des bases solides scientifiques avec une analyse détaillée qui fournit les facteurs de correction et montre la borne supérieure dans l'erreur-type


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, elegant and amazingly ~~stupefyingly~~ powerful.

Probabilistic Counting

- La première idée: chaque élément est haché pour donner un certain nombre dans $(0, 1) \Rightarrow$ aléa reproductible
- Le multi-ensemble $\mathcal{S}$ est mappée par la fonction de hachage* $h : \mathcal{U} \rightarrow (0, 1)$ dans un multi-ensemble

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

  avec $x_i = \text{hash}(w_i)$, $f_i = \#$ de x_i's
- L'ensemble des éléments distincts $X = \{x_1, \ldots, x_n\}$ est donc un ensemble de n numéros aléatoires indépendants répartis uniformément dans $(0, 1)$

Probabilistic Counting

- La première idée: chaque élément est haché pour donner un certain nombre dans $(0, 1) \Rightarrow$ aléa reproductible
- Le multi-ensemble $\mathcal{S}$ est mappée par la fonction de hachage* $h : \mathcal{U} \rightarrow (0, 1)$ dans un multi-ensemble

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

avec $x_i = \text{hash}(w_i)$, $f_i = \#$ de x_i's

- L'ensemble des éléments distincts $X = \{x_1, \ldots, x_n\}$ est donc un ensemble de n numéros aléatoires indépendants répartis uniformément dans $(0, 1)$

*On fait abstraction de la possibilité ici de collisions

Probabilistic Counting

- La première idée: chaque élément est haché pour donner un certain nombre dans $(0, 1) \Rightarrow$ aléa reproductible
- Le multi-ensemble $\mathcal{S}$ est mappée par la fonction de hachage* $h : \mathcal{U} \rightarrow (0, 1)$ dans un multi-ensemble

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

  avec $x_i =$ hash(w_i), $f_i =$ # de x_i's
- L'ensemble des éléments distincts $X = \{x_1, \ldots, x_n\}$ est donc un ensemble de n numéros aléatoires indépendants répartis uniformément dans $(0, 1)$

*On fait abstraction de la possibilité ici de collisions

Probabilistic Counting

Flajolet & Martin (JCSS, 1985) ont proposé de trouver, parmi les valeurs hachées, le plus grand préfixe (en binaire) de la forme $0.0^{p-1}1\ldots$; R est la valeur maximale de sorte que les motifs $0.1\ldots$, $0.01\ldots$, $0.001\ldots$, $\ldots$, $0.0^{R-1}1\ldots$ sont tous apparus dans le flux

La valeur R est une observable qui peut être facilement trouvé avec très peu de mémoire & il est insensible aux répétitions $\leftarrow$ observables doivent être des fonctions de X seulement

Probabilistic Counting

- Pour un ensemble de n nombres aléatoires dans $(0, 1) \rightarrow$

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

- Toutefois, $\mathsf{E}\left[2^R\right] \not\sim n$, outre les écarts sont significatifs

Probabilistic Counting

- Pour un ensemble de n nombres aléatoires dans $(0, 1) \rightarrow$

$$\mathsf{E}\left[R\right] \approx \log_2 n$$

- Toutefois, E $\left[2^R\right] \not\sim n$, outre les écarts sont significatifs

Probabilistic Counting

procedure PROBABILISTICCOUNTING($\mathcal{S}$)
  $bmap \leftarrow \langle 0, 0, \ldots, 0 \rangle$
  for $s \in \mathcal{S}$ **do**
    $y \leftarrow$ hash(s)
    $p \leftarrow$ long. de plus grand préfixe $0.0^{p-1}1\ldots$ dans y
    $bmap[p] \leftarrow 1$
  $R \leftarrow$ long. de plus grand préfixe $0^{R-1}1\ldots$ dans $bmap$
▷ ϕ est le facteur de correction
  return $Z := \phi \cdot 2^R$

L'analyse fournit la valeur de ϕ nécessaire pour corriger le biais:

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

$$\Rightarrow \mathsf{E}\left[\phi \cdot 2^R\right] = n$$

Stochastic averaging

- Malheureusement l'erreur-type de l'estimateur $Z := \phi \cdot 2^R$, malgré constante, est trop grande: SE $[Z] > 1$
- Le deuxième idée: répéter pour améliorer la précision
- Mais . . . utiliser m fonctions de hachage pour produire m flux est un calcul coûteux et implique de graves difficultés pour garantir l'indépendance

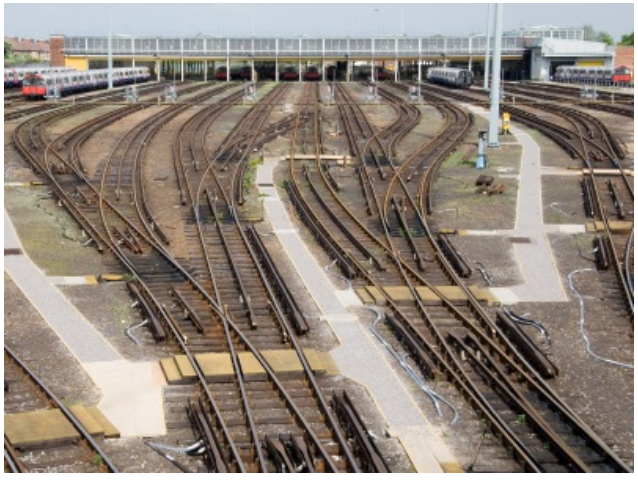
Stochastic averaging

- Malheureusement l'erreur-type de l'estimateur $Z := \phi \cdot 2^R$, malgré constante, est trop grande: SE $[Z] > 1$
- Le deuxième idée: répéter pour améliorer la précision
- Mais . . . utiliser m fonctions de hachage pour produire m flux est un calcul coûteux et implique de graves difficultés pour garantir l'indépendance

Stochastic averaging

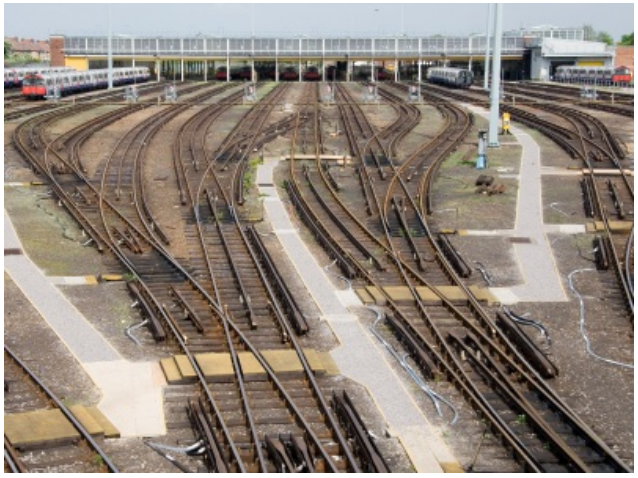
- Malheureusement l'erreur-type de l'estimateur $Z := \phi \cdot 2^R$, malgré constante, est trop grande: SE $[Z] > 1$
- Le deuxième idée: répéter pour améliorer la précision
- Mais . . . utiliser m fonctions de hachage pour produire m flux est un calcul coûteux et implique de graves difficultés pour garantir l'indépendance

Stochastic averaging



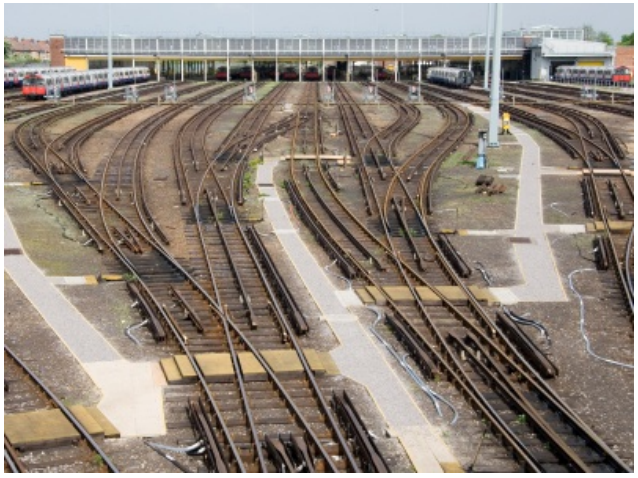
- Utilisez les premiers $\log_2 m$ bits de chaque valeur hachée pour le rediriger vers l'un des m sous-flux $\rightarrow$ stochastic averaging
- Obtenir m observables $R_1, R_2, \ldots, R_m$ et la moyenne!
- Chaque R_i fournit une estimation pour le nombre d'éléments qui vont au sous-flux i-ième, à savoir, R_i estime n/m

Stochastic averaging



- Utilisez les premiers $\log_2 m$ bits de chaque valeur hachée pour le rediriger vers l'un des m sous-flux $\rightarrow$ stochastic averaging
- Obtenir m observables $R_1, R_2, \ldots, R_m$ et la moyenne!
- Chaque R_i fournit une estimation pour le nombre d'éléments qui vont au sous-flux i-ième, à savoir, R_i estime n/m

Stochastic averaging



- Utilisez les premiers $\log_2 m$ bits de chaque valeur hachée pour le rediriger vers l'un des m sous-flux $\rightarrow$ stochastic averaging
- Obtenir m observables R_1, R_2, ..., R_m et la moyenne!
- Chaque R_i fournit une estimation pour le nombre d'éléments qui vont au sous-flux i-ième, à savoir, R_i estime n/m

Stochastic averaging

Il ya plusieurs façons de calculer un estimateur à partir des m observables

- Somme des estimateurs:

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

- Moyenne arithmétique des observables (proposé par Flajolet et Martin):

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

Stochastic averaging

- Moyenne harmonique (voir plus loin):

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

Ici nous nous attendons à $2^{-R_i} \approx m/n$, d'où pour le deuxième facteur on aura $m^2/(m^2/n) = n$

Stochastic averaging

- Toutes les stratégies ci-dessus nous donnent des estimateurs dont l'erreur-type est de la forme

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

  Augmenter la mémoire $\Rightarrow$ précision beaucoup meilleure!

- Dans le cas du *probabilistic counting*, on utilise la moyenne arithmétique des observables

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

Stochastic averaging

- Toutes les stratégies ci-dessus nous donnent des estimateurs dont l'erreur-type est de la forme

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

  Augmenter la mémoire $\Rightarrow$ précision beaucoup meilleure!
- Dans le cas du *probabilistic counting*, on utilise la moyenne arithmétique des observables

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

LogLog & HyperLogLog

M. Durand

- Durand & Flajolet (2003) ont réalisé que on peut dispenser des bitmaps du *Probabilistic Counting* et proposent comme observable le plus grand R tel que le motif $0.0^{R-1}1$ apparaît
- La nouvelle observable est similaire à celle du *Probabilistic Counting*, mais pas égale: $R(\text{LogLog}) \geq R(\text{ProbCount})$

Example

Motifs observés: 0.1101..., 0.010..., 0.0011..., 0.00001...

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

LogLog & HyperLogLog

M. Durand

- Durand & Flajolet (2003) ont réalisé que on peut dispenser des bitmaps du *Probabilistic Counting* et proposent comme observable le plus grand R tel que le motif $0.0^{R-1}1$ apparaît
- La nouvelle observable est similaire à celle du *Probabilistic Counting*, mais pas égale: $R(\text{LogLog}) \geq R(\text{ProbCount})$

Example

Motifs observés: 0.1101..., 0.010..., 0.0011 ..., 0.00001...

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

LogLog & HyperLogLog

M. Durand

- Durand & Flajolet (2003) ont réalisé que on peut dispenser des bitmaps du *Probabilistic Counting* et proposent comme observable le plus grand R tel que le motif $0.0^{R-1}1$ apparaît
- La nouvelle observable est similaire à celle du *Probabilistic Counting*, mais pas égale: $R(\text{LogLog}) \geq R(\text{ProbCount})$

Example

Motifs observés: 0.1101..., 0.010..., 0.0011 ..., 0.00001...

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

LogLog & HyperLogLog

- Nous avons besoin seulement de maintenir la R maximal en cours $R := \max\{R, p\} \Rightarrow$ il suffit de stocker $\Theta(\log \log n)$ bits comme $\mathsf{E}[R] = \Theta(\log n)$!
- Maintenant nous avons $\mathsf{E}[R] \sim \log_2 n$, mais $\mathsf{E}\left[2^R\right] = +\infty$, le *stochastic averaging* vient à la rescousse!
- Pour LogLog, Durand & Flajolet propose

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

LogLog & HyperLogLog

- Nous avons besoin seulement de maintenir la R maximal en cours $R := \max\{R, p\} \Rightarrow$ il suffit de stocker $\Theta(\log\log n)$ bits comme $\mathsf{E}[R] = \Theta(\log n)$!
- Maintenant nous avons $\mathsf{E}[R] \sim \log_2 n$, mais $\mathsf{E}\left[2^R\right] = +\infty$, le *stochastic averaging* vient à la rescousse!
- Pour LogLog, Durand & Flajolet propose

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

LogLog & HyperLogLog

- Nous avons besoin seulement de maintenir la R maximal en cours $R := \max\{R, p\} \Rightarrow$ il suffit de stocker $\Theta(\log\log n)$ bits comme E $[R] = \Theta(\log n)$!
- Maintenant nous avons E $[R] \sim \log_2 n$, mais E $\left[2^R\right] = +\infty$, le *stochastic averaging* vient à la rescousse!
- Pour LogLog, Durand & Flajolet propose

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

LogLog & HyperLogLog

- L'analyse fournit la forme exacte du facteur de correction

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

pour garantir non-biaisage asymptotique, et l'erreur-type est

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

- Seulement m compteurs de $\log_2 \log_2 N$ bits sont nécessaires:
  Ex.: $m = 2048$ compteurs de 5 bits pour estimer cardinalités jusqu'à $2^{27} \approx 10^8$, avec une erreur inférieure à 4%

LogLog & HyperLogLog

- L'analyse fournit la forme exacte du facteur de correction

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

pour garantir non-biaisage asymptotique, et l'erreur-type est

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

- Seulement m compteurs de $\log_2 \log_2 N$ bits sont nécessaires:
  Ex.: $m = 2048$ compteurs de 5 bits pour estimer cardinalités jusqu'à $2^{27} \approx 10^8$, avec une erreur inférieure à 4%

LogLog & HyperLogLog


É. Fusy


O. Gandouet


F. Meunier

- Flajolet, Fusy, Gandouet & Meunier conçoivent en 2007 le *best algorithm known* (cif. PF's *keynote speech* à ITC Paris 2009)
- En bref, HyperLogLog combine les observables R_i à la LogLog de m sous-flux avec la moyenne harmonique

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

LogLog & HyperLogLog


É. Fusy


O. Gandouet


F. Meunier

- Flajolet, Fusy, Gandouet & Meunier conçoivent en 2007 le *best algorithm known* (cif. PF's *keynote speech* à ITC Paris 2009)
- En bref, HyperLogLog combine les observables R_i à la LogLog de m sous-flux avec la moyenne harmonique

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

LogLog & HyperLogLog

P. Chassaing

L. Gérin

- L'idée a été inspirée par l'étude analytique de Chassaing & Gérin (2006) de quelle etait la manière optimale de combiner observables; dans leur cas, les observables étaient les valeurs k-èmes de chaque sous-flux
- Chassaing & Gérin montrent que la meilleure façon de faire ceci est avec la moyenne harmonique

LogLog & HyperLogLog

P. Chassaing

L. Gérin

- L'idée a été inspirée par l'étude analytique de Chassaing & Gérin (2006) de quelle etait la manière optimale de combiner observables; dans leur cas, les observables étaient les valeurs k-èmes de chaque sous-flux
- Chassaing & Gérin montrent que la meilleure façon de faire ceci est avec la moyenne harmonique

Statistiques d'ordre

- Bar-Yossef, Kumar & Sivakumar (2002); Bar-Yossef, Jayram, Kumar, Sivakumar & Trevisan (2002) proposent d'utiliser la statistique d'ordre k-ième $X_{(k)}$ pour estimer la cardinalité; pour un ensemble de n nombres aleatoires indépendantes et uniformément distribués

$$\mathsf{E}\left[X_k\right] = \frac{k}{n+1}$$

- Giroire (2005, 2009) propose aussi quelques estimateurs qui combinent le *stochastic averaging* et les statistiques d'ordre k-ième

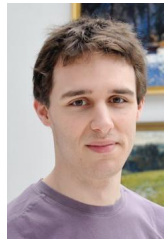
Statistiques d'ordre

- Bar-Yossef, Kumar & Sivakumar (2002); Bar-Yossef, Jayram, Kumar, Sivakumar & Trevisan (2002) proposent d'utiliser la statistique d'ordre k-ième $X_{(k)}$ pour estimer la cardinalité; pour un ensemble de n nombres aleatoires indépendantes et uniformément distribués

$$\mathsf{E}\left[X_k\right] = \frac{k}{n+1}$$

- Giroire (2005, 2009) propose aussi quelques estimateurs qui combinent le *stochastic averaging* et les statistiques d'ordre k-ième

Statistiques d'ordre



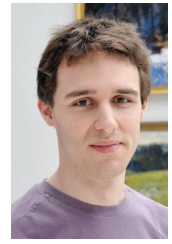
J. Lumbroso

- Bien que le minimum ($k = 1$) ne permet pas un estimateur viable, le *stochastic averaging* évite le problème
- Ainsi, Lumbroso utilise la moyenne harmonique des minimums de m sous-flux

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

où M_i est le minimum de l'i-ième sous-flux

Statistiques d'ordre



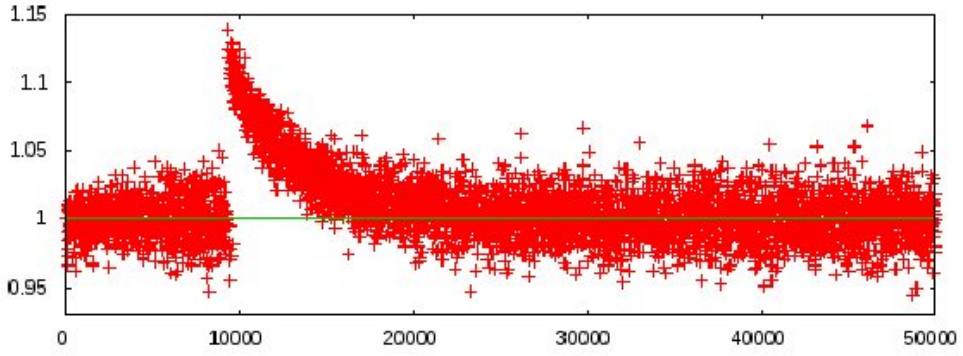
J. Lumbroso

- Bien que le minimum ($k = 1$) ne permet pas un estimateur viable, le *stochastic averaging* évite le problème
- Ainsi, Lumbroso utilise la moyenne harmonique des minimums de m sous-flux

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

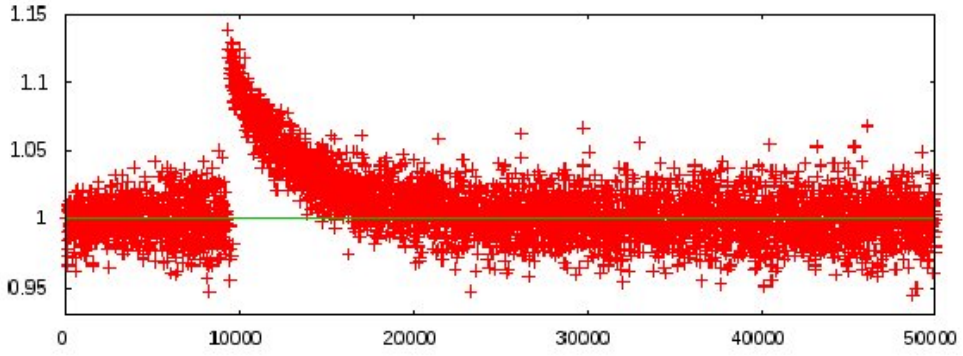
où M_i est le minimum de l'i-ième sous-flux

Statistiques d'ordre



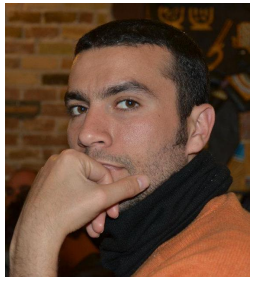
- L'estimateur, non-biasé, a erreur-type $1/\sqrt{m-2}$
- Lumbroso parvient aussi à calculer la distribution de probabilité du estimateur, et quelles sont les corrections nécessaires pour estimer les petites cardinalités (qui présentent habituellement des déformations)

Statistiques d'ordre



- L'estimateur, non-biasé, a erreur-type $1/\sqrt{m-2}$
- Lumbroso parvient aussi à calculer la distribution de probabilité du estimateur, et quelles sont les corrections nécessaires pour estimer les petites cardinalités (qui présentent habituellement des déformations)

Recordinality



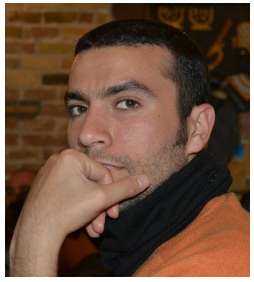
A. Helmi

J. Lumbroso

A. Viola

- RECORDINALITY est un nouvel estimateur vaguement lié à les statistiques d'ordre, mais il fonctionne sur des principes très différents de ceux d'autres estimateurs, avec des propriétés remarquables
- Ne manquez pas l'exposé de Jérémie juste après!

Recordinality



A. Helmi

J. Lumbroso

A. Viola

- RECORDINALITY est un nouvel estimateur vaguement lié à les statistiques d'ordre, mais il fonctionne sur des principes très différents de ceux d'autres estimateurs, avec des propriétés remarquables
- Ne manquez pas l'exposé de Jérémie juste après!

D'autres problèmes

- Pour trouver les k-éléphants ou k-souris on prend un échantillon aléatoire de taille T d'éléments différents, avec leurs compteurs de fréquence
- Alors, si il y a une proportion p_k de k-souris (k-éléphants) dans l'échaintillon alors notre estimation est

$$\#\text{ de } k\text{-souris} = p_k \hat{n},$$

avec n l'estimation de la cardinalité ...

D'autres problèmes

- Pour trouver les k-éléphants ou k-souris on prend un échantillon aléatoire de taille T d'éléments différents, avec leurs compteurs de fréquence
- Alors, si il y a une proportion p_k de k-souris (k-éléphants) dans l'échaintillon alors notre estimation est

$$\text{\# de } k\text{-souris} = p_k \hat{n},$$

avec n l'estimation de la cardinalité ...

D'autres problèmes

- Le problème de l'obtention des échantillons aléatoires d'éléments du flux est clairement l'un des plus important et intéressant
- L'échantillons est un multi-ensemble, chaque élément x_j apparaît avec fréquence proportionnelle a $\nu_j \Rightarrow$ straight sampling $\Rightarrow$ needle-on-a-haystack

D'autres problèmes

- Le problème de l'obtention des échantillons aléatoires d'éléments du flux est clairement l'un des plus important et intéressant
- L'échantillons est un multi-ensemble, chaque élément x_j apparaît avec fréquence proportionnelle a $\nu_j \Rightarrow$ straight sampling $\Rightarrow$ needle-on-a-haystack

D'autres problèmes

Wegman

G. Louchard

- Ce qu'il faut sont des échantillons d'éléments différents $\Rightarrow$ distinct sampling
- *Adaptive sampling* (Wegman, 1980; Flajolet, 1990; Louchard, 1997) nous donne exactement ce que on a besoin

D'autres problèmes

Wegman

G. Louchard

- Ce qu'il faut sont des échantillons d'éléments différents $\Rightarrow$ distinct sampling
- *Adaptive sampling* (Wegman, 1980; Flajolet, 1990; Louchard, 1997) nous donne exactement ce que on a besoin

D'autres problèmes

procedure AdaptiveSampling($\mathcal{S}$, $maxC$)
  $C \leftarrow \emptyset$; $p \leftarrow 0$
  for $x \in \mathcal{S}$ **do**
    if hash$(x) = 0^p \ldots$ **then**
      $C \leftarrow C \cup \{x\}$
      if $|C| > maxC$ **then**
        $p \leftarrow p + 1$; filter C
  return C

Quand on a fini, $|C|$ est le nombre d'éléments que on a, dans un trie aléatoire de taille n, dans le sous-arbre avec racine 0^p; il a exactement 2^p sous-arbres avec racines à profondeur p (0 c'est un d'eux).
Donc il y aura

$$|C| \approx n/2^p$$

$\Rightarrow$ la taille $|C|$ nous donne une estimation d'n!!

Merci beaucoup
de votre attention!