Count-Min Sketch

Kurz: CM-Sketch

aus: Cormode und Muthukrishnan: "An improved data stream summary: the count-min sketch and its applications", Journal of Algorithms 55 (2005), 58-75.

Vorteile:

- geringer Speicherplatzbedarf
- schnelle Update-Zeiten
- vielseitig verwendbar (wird mittlerweile von vielen Verfahren als "Basis-Datenstruktur" genutzt)

Szenario:

- Datenstrom besteht aus "updates" einer Datenbank
- $U = \{0, \dots, m-1\}$ ist das Universum potentieller Datenbank-Elemente.
- Zu jedem Zeitpunkt t besteht die Datenbank aus einer Multimenge $D(t)$ über U.
  D.h: $D(t)$ enthält für jedes $u \in U$ die Information darüber, wie oft das Element u in der Datenbank vorkommt

  Bsp: In der Datenbank wird der aktuelle Bestand eines Warenlagers gespeichert.

Für $U = \{0, \dots, m-1\}$ können wir uns $D(t)$ als

Vektor $(f_0, \dots, f_{m-1}) \in \mathbb{N}^m$ vorstellen, wobei f_u angibt, wie oft Element $u \in \mathcal{U}$ in $D(t)$ vorkommt. Notation: $D(t)_u := f_u$ ist die Häufigkeit von u in der DB zum Zeitpunkt t.

- Der <u>Datenstrom</u> besteht aus Tupeln der Form

  (u, z) mit $u \in \mathcal{U}$ und $z \in \mathbb{Z}$
  (i.d.R.: $z \neq 0$)

  Für $z > 0$ bedeutet (u, z), dass das Element u z-mal in die DB eingefügt wird;
  für $z < 0$ bedeutet (u, z), dass u z-mal gelöscht wird.

  <u>Also</u>: Falls das $(t+1)$-te Datenstrom-Element (u, z) ist, so ist

  $$D(t+1)_u := D(t)_u + z \quad \text{und}$$
  $$D(t+1)_{u'} := D(t)_{u'} \quad \text{für alle } u' \in \mathcal{U} \setminus \{u\}.$$

- Wir betrachten hier im Folgenden ausschließlich das <u>strikte Drehkreuzmodell</u> (engl.: strict turnstile model), bei dem davon ausgegangen wird, dass $D(t)_u \geq 0$ für alle $u \in \mathcal{U}$ und alle Zeitpunkte t ist.

- Initialisierung: zum Zeitpunkt 0 ist die DB leer, d.h. $D(0)_u = 0$ für alle $u \in \mathcal{U}$.

Ziel:

Approximative Speicherung von $D(t)$, so dass bei Eingabe eines $u \in \mathcal{U}$ ein Schätzwert $\hat{f}_u$ für f_u ausgegeben wird.

Ansatz:

- Für eine gewünschte "Präzision" $\varepsilon > 0$ wähle die Breite (engl.: width) $w \geq \frac{b}{\varepsilon}$ für eine geeignete Zahl b (z.B.: $b := 2$ oder $b := e$).
- Wähle zufällig, gleichverteilt eine Hash-Funktion
  $$h : \mathcal{U} \to \{0, \dots, w-1\}$$
  aus einer 2-universellen Familie von Hash-Funktionen von $\mathcal{U}$ nach $\{0, \dots, w-1\}$.
- Speichere ein Integer-Array $c[0 \dots w-1]$
- Initialisierung: $c[j] = 0$ f.a. $j \in \{0, \dots, w-1\}$
- Beim Lesen eines Datenstrom-Elements (u, z) tue Folgendes:
  - Berechne $j := h(u)$
  - setze $c[j] := c[j] + z$
- Zum Zeitpunkt t gib als Schätzer für f_u den Wert $f_u^* := c[h(u)]$ aus.

Frage: Wie nah liegt dieser Schätzer am tatsächlichen Wert f_u dran?

(1): **Antwort:** Zu jedem Zeitpunkt t gilt: $f_u^* \geq f_u$, und

mit Wahrscheinlichkeit $\geq 1 - \frac{1}{b}$ ist

$$f_u^* \leq f_u + \varepsilon \| D(t) \|_1$$

wobei $\| D(t) \|_1 = \sum_{u' \in U} f_{u'}$ die Anzahl der

zum Zeitpunkt t insgesamt in der Datenbank gespeicherten Dinge ist.

Beweis: Es gilt: $f_u^* = c[h(u)] = f_u + \sum_{\substack{u' \in U \setminus \{u\} \\ h(u') = h(u)}} f_{u'}$.

Somit ist offensichtlicherweise $f_u^* \geq f_u$.

Für jedes $u' \in U$ mit $u' \neq u$ sei $X_{u'}$ die Zufallsvariable mit

$$X_{u'} := \begin{cases} 1 & \text{falls } h(u') = h(u) \\ 0 & \text{sonst.} \end{cases}$$

Da h zufällig aus einer 2-universellen Familie von Hash-Funktionen von U nach $\{0, \ldots, w-1\}$ gewählt wurde, gilt

jedes feste $u' \in U$ mit $u' \neq u$:

$$Pr(X_{u'} = 1) = Pr(h(u') = h(u)) = \frac{1}{w},$$ und daher

gilt auch für den Erwartungswert:

(*) $$E(X_{u'}) = \frac{1}{w} \quad \text{für alle } u' \in U \setminus \{u\}.$$

Wir betrachten die Zufallsvariable

$$Y := \sum_{u' \in U \setminus \{u\}} f_{u'} \cdot X_{u'}$$

<u>klar:</u> $f_u^* = c[h(u)] = f_u + \sum_{\substack{u' \in U \setminus \{u\}: \\ h(u') = h(u)}} f_{u'}$

$$= f_u + Y$$

<u>D.h.:</u> Y gibt an, um wie weit der Schätzer f_u^* den tatsächlichen Wert f_u übersteigt.

Für den Erwartungswert $E(Y)$ gilt:

$$E(Y) \underset{\substack{\uparrow \\ \text{Linearität des} \\ \text{Erwartungswerts}}}{=} \sum_{u' \in U \setminus \{u\}} f_{u'} \cdot E(X_{u'})$$

$$\overset{(*)}{=} \sum_{u' \in U \setminus \{u\}} f_{u'} \cdot \frac{1}{w}$$

$$\leq \frac{1}{w} \cdot \|D(t)\|_1$$

$$\underset{w \geq \frac{b}{\varepsilon}}{\leq} \frac{\varepsilon}{b} \cdot \|D(t)\|_1$$

Um die Wahrscheinlichkeit dafür abzuschätzen, dass $Y > \varepsilon \cdot \|D(t)\|_1$ ist, nutzen wir nun die

Markov-Ungleichung:

Für jede Zufallsvariable X, die nur Werte ≥ 0 annehmen kann und für jede reelle Zahl $a > 0$ gilt:

$$\Pr(X \geq a) \leq \frac{E(X)}{a}$$

Beweis:

$$E(X) \overset{\text{Def}}{=} \sum_i i \cdot \Pr(X = i)$$

$$\geq \sum_{i:\, i \geq a} a \cdot \Pr(X = i)$$

$$= a \cdot \Pr(X \geq a),$$

und daher gilt: $\Pr(X \geq a) \leq \dfrac{E(X)}{a}$. □

Gemäß Markov-Ungleichung (mit $X := Y$ und $a := \varepsilon \cdot \|D(t)\|_1$) gilt:

$$\Pr(Y \geq \varepsilon \cdot \|D(t)\|_1) \leq \frac{E(Y)}{\varepsilon \cdot \|D(t)\|_1}$$

$$\underset{E(Y) \leq \frac{\varepsilon}{b} \cdot \|D(t)\|_1}{\leq} \frac{1}{b}.$$

Und somit gilt: $\Pr(\underbrace{Y \leq \varepsilon \cdot \|D(t)\|_1}_{\text{dh } f_u^* \leq f_u + \varepsilon \cdot \|D(t)\|_1}) \geq 1 - \frac{1}{b}$

□ Beweis von (*).

Der Count-Min-Sketch verwendet eine geeignete Anzahl d von unabhängigen Instanzen dieses Ansatzes, um die Erfolgswahrscheinlichkeit von $1-\frac{1}{b}$ auf $1-\delta$ anzuheben.

Man kann nachrechnen, dass der verwendete Speicherplatz kleinstmöglich ist, wenn

$$b := e$$

gewählt wird.

<u>Count-Min-Sketch für Parameter ε, δ:</u>

Wähle Breite (engl.: width) und Tiefe (engl. depth)

$$\boxed{w := \left\lceil \frac{e}{\varepsilon} \right\rceil}$$
$$\boxed{d := \left\lceil \ln\left(\frac{1}{\delta}\right) \right\rceil}$$

und verwende ein 2-dimensionales Array

$$C[1 \dots d][0 \dots w-1]$$

der Tiefe d und der Breite w, dessen Einträge auf 0 initialisiert sind.

Wähle zufällig, gleichverteilt und unabhängig voneinander d Hash-Funktionen

$$h_1, \dots, h_d : U \to \{0, \dots, w-1\}$$

aus einer 2-universellen Familie H von Hash-Funktionen.

Beim Lesen eines Datenstrom-Elements (u,z)

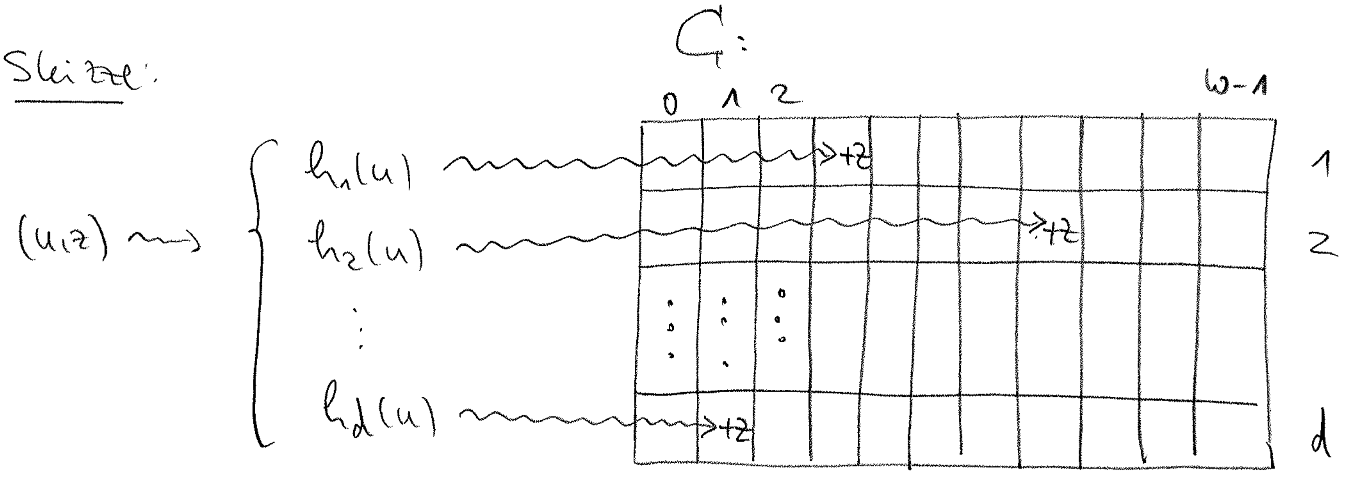
aktualisiere das Array C wie folgt:

Für jedes $i \in \{1,\dots,d\}$:

Berechne $h_i(u)$ und

setze $C[i, h_i(u)] := C[i, h_i(u)] + z$

Skizze:



Zu einem Zeitpunkt t gib als Schätzer für f_u den Wert

$$\hat{f}_u := \min_{i \in \{1,\dots,d\}} C[i, h(u)]$$

aus

Satz: Zu jedem Zeitpunkt t gilt: $\hat{f}_u \geq f_u$, und

mit Wahrscheinlichkeit $\geq 1-\delta$ ist

$$\hat{f}_u \leq f_u + \varepsilon \cdot \|D(t)\|_1$$

Beweis:

Gemäß ⊛ gilt für jedes $i \in \{1, \dots, d\}$:

$$C[i, h_i(u)] \geq f_u \quad \text{und}$$

$$Pr\left(C[i, h_i[u]] \leq f_u + \varepsilon \cdot \|D(t)\|_1 \right) \geq 1 - \frac{1}{b}$$

für $b := e$, also: $Pr\left(C[i, h_i(u)] > f_u + \varepsilon \cdot \|D(t)\|_1 \right) \leq \frac{1}{e}$

Aus der ersten Aussage folgt direkt, dass $\hat{f}_u \geq f_u$.

Außerdem gilt:

$$Pr\left(\hat{f}_u > f_u + \varepsilon \|D(t)\|_1 \right)$$

$$= Pr\left(\text{für alle } i \in \{1..d\} \text{ ist } C[i, h_i(u)] > f_u + \varepsilon \cdot \|D(t)\|_1 \right)$$

$$\underset{\substack{h_1..h_d \text{ sind} \\ \text{unabhängig} \\ \text{voneinander} \\ \text{gewählt}}}{=} \prod_{i=1}^{d} \underbrace{Pr\left(C[i, h_i(u)] > f_u + \varepsilon \|D(t)\|_1 \right)}_{\leq \frac{1}{e}}$$

$$\leq \left(\frac{1}{e}\right)^d$$

$$\underset{d \geq \ln(\frac{1}{\delta})}{\leq} \left(\frac{1}{e}\right)^{\ln(\frac{1}{\delta})} = \left(e^{-1}\right)^{-\ln(\delta)} = e^{\ln(\delta)} = \delta$$

Somit ist $Pr\left(\hat{f}_u \leq f_u + \varepsilon \cdot \|D(t)\|_1 \right) \geq 1 - \delta$.

□

Der Count-Min-Sketch mit Parameter ε, δ benötigt folgenden Speicherplatz

- zum Speichern des Arrays $C[1..d][0..w-1]$:

  $\underbrace{d \cdot w \cdot}\max_{t' \le t} \log \|D(t')\|_1$ Bits

  $= \lceil \ln(\frac{1}{\delta}) \rceil \cdot \lceil \frac{e}{\varepsilon} \rceil \cdot \max_{t' \le t} \log \|D(t')\|_1$

- zum Speichern der d Hash-Funktionen $h_1, \dots, h_d$ insgesamt $2d$ Zahlen aus $\{0, \dots, p-1\}$ für eine Primzahl p mit $m \le p \le 2m$, also $\le 2d \cdot \lceil \log p \rceil \le 2d \lceil \log(2m) \rceil \le 2d(1 + \lceil \log m \rceil)$ Bits.

Insgesamt kommt der Count-Min-Sketch also mit

$$\boxed{O\left(\frac{1}{\varepsilon} \cdot \ln\left(\frac{1}{\delta}\right) \cdot \max_{t' \le t} \log \|D(t')\|_1\right) \text{ Speicherbits}}$$

aus.

Anwendungsbeispiele für den Count-Min-Sketch

Beispiel 1:

Wähle $\varepsilon = \frac{1}{10}$, $\delta = \frac{1}{100}$.

Dann ist $d \cdot w = \underbrace{\lceil \ln(100) \rceil}_{\approx 4{,}6} \cdot \underbrace{\lceil 10 \cdot e \rceil}_{\approx 27{,}18} = 5 \cdot 28 = 140.$

Betrachte folgendes Szenario:

- zu Beginn ist die Datenbank leer
- dann werden nach und nach 1 Million IP-Adressen (alle verschieden, jede 1×) in die DB eingefügt
- dann werden alle bis auf 9 Stück wieder gelöscht.

Sei t der Zeitpunkt am Ende dieses Szenarios.

Dann ist $\|D(t)\|_1 = 9$ und $\varepsilon \cdot \|D(t)\|_1 = \frac{9}{10} < 1$

Für jede IP-Adresse u gilt für den Schätzer $\hat{f}_u$ zum Zeitpunkt t:

$$\Pr\Big(\hat{f}_u \leq f_u + \underbrace{\varepsilon \|D(t)\|_1}_{9/10}\Big) \geq 1 - \delta = \frac{99}{100},$$

also $\Pr\left(\hat{f}_u > f_u + \frac{9}{10}\right) \leq \frac{1}{100}$

Falls $u \in D(t)$, so ist $f_u = 1 \leq \hat{f}_u$ (da $\hat{f}_u \geq f_u$ ist).

Falls $u \notin D(t)$, so ist $f_u = 0$ und daher

$$\Pr\left(\hat{f}_u > \frac{9}{10}\right) \leq \frac{1}{100}.$$

Die Frage "Gehört u zu $DB(t)$?" können wir wie folgt beantworten:

Falls $\hat{f}_u < 1$, so antworte "nein";
falls $\hat{f}_u \geq 1$, so antworte "möglicherweise ja"

Die Antwort ist korrekt; und die Wahrscheinlichkeit, eine "falsche positive Antwort" für ein $u \notin D(t)$ zu bekommen, ist $\leq \frac{1}{100} \mathrel{\hat{=}} 1\%$.

Wie viele Speicherbits verwendet der Count-Min-Sketch dazu während der gesamten Berechnung?

- für die Hash-Funktionen: $\leq 2d \lceil \log(2m) \rceil$ Bits
- für das Array C $\leq d \cdot w \cdot \max_{t' \leq t} \log \|D(t')\|_1$ Bits

Hier ist $d = \lceil \ln(100) \rceil = 5$, $w = \lceil 10e \rceil = 28$.
Also: $2d\lceil \log(2m) \rceil = 10 \cdot \lceil \log(2m) \rceil$; $d \cdot w = 5 \cdot 28 = 140$.
Und:
$$\max_{t' \leq t} \|D(t')\| = 10^6, \text{ also}$$

$$\max_{t' \leq t} \log \|D(t')\| = \log(\underbrace{10^6}_{\substack{= 10^{3 \cdot 2} \\ \leq (2^{10})^2}}) \leq \log(2^{20}) = 20$$

Also werden für die Hash-Funktionen $\leq 2d \lceil \log(2m) \rceil = 10 \cdot \lceil \log(2m) \rceil$ Bits benötigt, und für das Array C werden

$$\leq d \cdot w \cdot \max_{t' \leq t} \log \| D(t') \|_1 \leq 140 \cdot 20 = \underbrace{2800 \text{ Bits}}_{= 350 \text{ Byte}} \text{ benötigt.}$$

In unserem Bsp ist $m = 2^{32}$ (da gemäß IPv4 jede IP-Adresse mit $a = 32$ Bits gespeichert werden kann. Also ist

$10 \cdot \lceil \log(2m) \rceil = 10 \cdot \log(2^{33}) = 10 \cdot 33 = 330$ Bits < 42 Byte

gesamter Speicherverbrauch für den CM-Sketch: < 392 Byte

$\underline{\underline{< 1 \text{ kB}}}$

Zum Vergleich:

Wenn wir an Stelle des Count-Min-Sketches zu jedem Zeitpunkt $t' \leq t$ die gesamte Datenbank $D(t')$ exakt speichern, verbrauchen wir

$$10^6 \cdot a \text{ Speicherbits,}$$

wobei a die zur Anzahl einer einzelnen IP-Adresse nötigen Bits ist.

Derzeit (IPv4) ist $a = 32$.

Dh wir benötigen folgenden Speicherplatz zum Speichern von $D(t')$:

$$32 \cdot 10^6 \text{ Bits} \hat{=} 4 \cdot 10^6 \text{ Bytes} \hat{=} \underline{4 \text{ MB}}.$$

Beispiel 2: Bereichsanfragen

Wir betrachten jetzt wieder das allgemeine Szenario bei dem der Zustand $D(t)$ einer Datenbank zum Zeitpunkt t durch einen Count-Min-Sketch $C[1..d][0..w-1]$ repräsentiert wird. Als Schätzer dafür, wie oft ein Wert $u \in U$ in $D(t)$ vorkommt, geben wir den Wert

$$\hat{f}_u := \min_{i \in \{1,\dots,d\}} C[i, h_i(u)]$$ aus

Ziel jetzt: Beantworte Bereichsanfragen, bei denen für Werte ℓ, r mit $0 \le \ell \le r \le m-1$ (zur Erinnerung: $U = \{0,\dots,m-1\}$) ein Schätzer

$$f_{[\ell,\dots,r]} := \sum_{\substack{u \in U \text{ mit} \\ \ell \le u \le r}} f_u$$, d.h. ein Schätzer für

dafür, wie oft $D(t)$ Werte zwischen ℓ und r enthält, ausgegeben werden soll.

1. Idee: Gib $\sum_{u=\ell}^{r} \hat{f}_u$ als Schätzer aus

Problem: Wenn der Bereich $\{\ell,\dots,r\}$ sehr groß ist, dauert das zu lang, und der Fehler wird zu groß

Lösung:

Nutze $\lceil \log|U| \rceil = \lceil \log m \rceil$ viele Count-Min-Sketches

Für jedes $k \in \{1, \dots, \lceil \log m \rceil\}$ nutze einen Count-Min-Sketch C_k, der an Stelle des Vektors $D(t) = (f_0, f_1, \dots, f_{m-1})$ den Vektor $D(t)^{(k)} := \left(f^{(k)}_1, \dots, f^{(k)}_{\lceil \frac{m}{2^k} \rceil}\right)$

repräsentiert, wobei

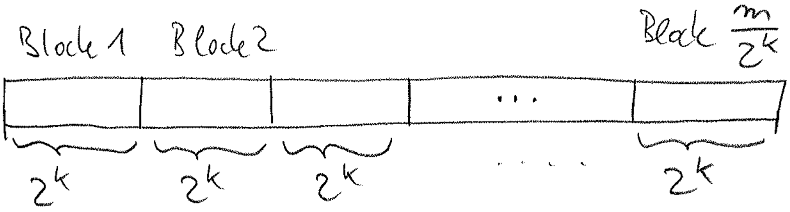
$$f^{(k)}_b := \sum_{u=(b-1)\cdot 2^k - 1}^{b\cdot 2^k} f_u$$

für jedes $b \in \{1, \dots, \lceil \frac{m}{2^k} \rceil\}$ ist.

Skizze:

$U = \{0, \dots, m-1\}$:

aufgeteilt in Blöcke der Länge 2^k



$f^{(k)}_b$ gibt an, wie viele Elemente von $D(t)$ zum b-ten Block der Länge 2^k gehören.

Der b-te Block besteht hier aus allen Elementen $u \in U$ mit $(b-1)\cdot 2^k \leq u < b \cdot 2^k$.

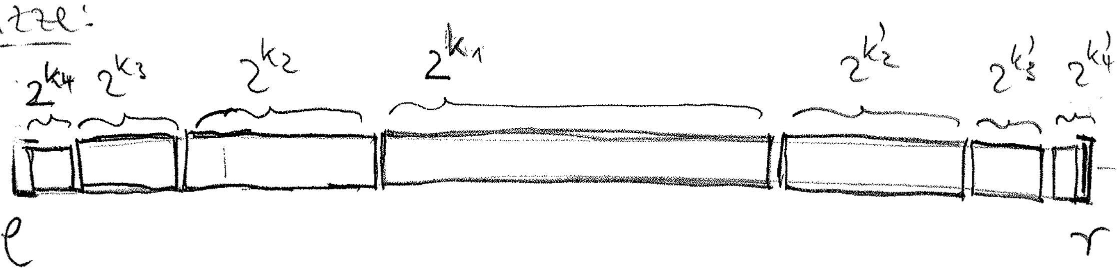
Ein solcher Block wird auch dyadischer Bereich genannt.

Man kann sich leicht davon überzeugen, dass jeder beliebige Bereich

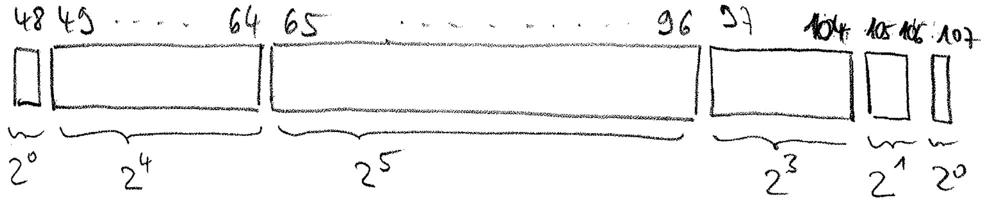
$$\{\ell, \ell+1, \dots, r\}$$

aus höchstens $2\lceil \log m \rceil$ dyadischen Bereichen (mit verschiedenen Parametern b, k) besteht

Skizze:



Bsp: $\ell = 48$, $r = 107$:



Für eine Bereichsanfrage für ℓ, r, zerlege den Bereich $\{\ell, \dots, r\}$ in die entsprechenden dyadischen Bereiche, nutze die entsprechenden Count-Min-Sketches, um Schätzer $\hat{f}_b^{(k)}$ für die zu den dyadischen Bereichen gehörenden Werte $f_b^{(k)}$ zu erhalten, addiere all diese Schätzer, und gib sie als Schätzer für $f_{[\ell \dots r]}$ aus. **Details: Übung!**

Man kann zeigen, dass dieser Schätzwert stets mindestens so groß ist wie die tatsächliche Antwort auf die Bereichsanfrage, und dass es mit Wahrscheinlichkeit $\geq 1-\delta$ die tatsächliche Antwort um höchstens

$$2 \cdot \varepsilon \cdot (\log m) \cdot \|D(t)\|_1$$

überschreitet.

Beispiel 3: Schätzung von Join-Größen

Szenario: Für zwei Datenbank-Relationen R und S und ein Attribut A soll die Größe des Joins $R \bowtie S := R \bowtie_{R.A = S.A} S$ abgeschätzt werden

(Informationen dieser Art werden von DBMS zur Anfrageoptimierung verwendet).

Sei $U = \{0, \ldots, m-1\}$ so, dass das Attribut A nur Werte aus dem Bereich U annehmen kann.

Für jedes $u \in U$ sei

r_u die Anzahl der Tupel in R, bei denen Attribut A den Wert u annimmt,
d.h.: $r_u := |\sigma_{A=u}(R)|$.

Analog sei
$s_u := |\sigma_{A=u}(S)|$ die Anzahl der Tupel in S, bei denen Attribut A den Wert u annimmt.

Die Anzahl $|R \bowtie S|$ der Tupel im Join von R und S bzgl. A ist dann

$$|R \bowtie S| = \sum_{u \in U} r_u \cdot s_u = r^T \cdot s, \quad \text{wobei}$$

$$r := \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix} \quad \text{und} \quad s := \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{pmatrix} \quad \text{ist.}$$

Um einen Schätzer für $|R \bowtie S| = r^T \cdot s$ zu erhalten, nutzen wir 2 Count-Min-Sketches $C_{R,A}$ und $C_{S,A}$, die wie folgt erzeugt wurden:

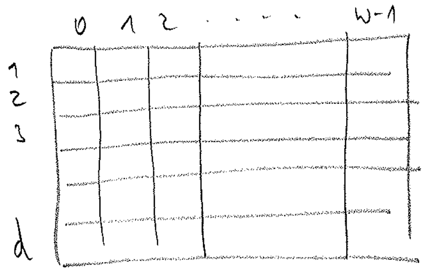
- Starte mit $C_{R,A}$ als leerem Sketch (d.h. alle Werte sind auf 0 initialisiert).
- Für jedes Tupel t in R tue Folgendes:
  - Sei u der Wert von t bzgl. Attribut A
  - Füge in $C_{R,A}$ das "update" $(u,1)$ ein.

$C_{R,A}$ repräsentiert also den Vektor $r^T = (r_1, \dots, r_m)$

Analog gehen wir für $C_{S,A}$ vor, so dass $C_{S,A}$ den Vektor $s^T = (s_1, \dots, s_m)$ repräsentiert.

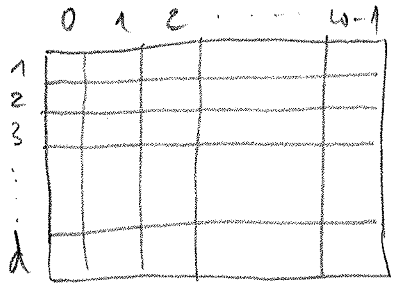
Skizze:

$C_{R,A}$:



repräsentiert $(r_1, r_2, \dots, r_m)$

$C_{S,A}$:



repräsentiert $(s_1, s_2, \dots, s_m)$

Die beiden Count-Min-Sketches $C_{R,A}$ und $C_{S,A}$ haben dabei die gleiche Breite w und die gleiche Tiefe d und nutzen dieselben Hash-Funktionen $h_1, \dots, h_d$.

Für jede Zeile $i \in \{1,\dots,d\}$ setzen wir

$$\boxed{g_i := \sum_{j=0}^{w-1} C_{R,A}[i,j] \cdot C_{S,A}[i,j]}\,.$$

Als Schätzer für $|R \bowtie S| = r^T \cdot s$ geben wir aus:

$$\boxed{\widehat{r^T \cdot s} := \min_{i \in \{1,\dots,d\}} g_i}\,.$$

Verbrauchter Speicherplatz für die beiden Count-Min-Sketches:
$O\left(\frac{1}{\varepsilon} \cdot \ln\left(\frac{1}{\delta}\right) \cdot \log\left(\max(\|r\|_1, \|s\|_1)\right)\right)$

Es gilt:

Satz: $\widehat{r^T \cdot s} \geq r^T \cdot s$, und

mit Wahrscheinlichkeit $\geq 1-\delta$ ist

$$\underbrace{\widehat{r^T \cdot s}}_{\text{Schätzer für } |R \bowtie S|} \leq \underbrace{r^T \cdot s}_{|R \bowtie S|} + \varepsilon \cdot \underbrace{\|r\|_1 \cdot \|s\|_1}_{\text{Anzahl der Tupel in } R \text{ und } S}$$

(dabei sind ε, δ die Parameter bzgl denen die beiden Count-Min-Sketches konstruiert werden).

Beweis: Es gilt:

$$g_i \overset{\text{Def}}{=} \sum_{j=0}^{w-1} \underbrace{C_{R,A}[i,j]} \cdot \underbrace{C_{S,A}[i,j]}$$

$$= \sum_{j=0}^{w-1} \overbrace{\left(\sum_{\substack{u \in U \text{ mit} \\ h_i(u)=j}} r_u\right)} \cdot \overbrace{\left(\sum_{\substack{u' \in U \text{ mit} \\ h_i(u')=j}} s_{u'}\right)}$$

$$= \sum_{j=0}^{w-1} \sum_{\substack{u,u' \in U \text{ mit} \\ h_i(u)=h_i(u')=j}} r_u \cdot s_{u'}$$

$$= \sum_{j=0}^{w-1} \left(\sum_{\substack{u \in U \text{ mit} \\ h_i(u)=j}} r_u \cdot s_u + \sum_{\substack{u,u' \in U \text{ mit} \\ u \neq u' \text{ und} \\ h_i(u)=h_i(u')=j}} r_u \cdot s_{u'} \right)$$

$$= \sum_{j=0}^{w-1} \sum_{\substack{u \in U \text{ mit} \\ h_i(u)=j}} r_u \cdot s_u + \sum_{j=0}^{w-1} \sum_{\substack{u,u' \in U \text{ mit} \\ u \neq u' \text{ und} \\ h_i(u)=h_i(u')=j}} r_u \cdot s_{u'}$$

$$= \sum_{u \in U} r_u \cdot s_u + \sum_{\substack{u,u' \in U \text{ mit} \\ u \neq u' \text{ und} \\ h_i(u)=h_i(u')}} r_u \cdot s_{u'}$$

$$= r^T \cdot s + \sum_{\substack{u,u' \in U \text{ mit} \\ u \neq u' \text{ und} \\ h_i(u)=h_i(u')}} r_u \cdot s_{u'} .$$

Insbes gilt also: $g_i \geq r^T \cdot s$, und daher auch

$$\widehat{r^T s} \overset{\text{Def}}{=} \min_{i \in \{1,\dots,d\}} g_i \geq r^T \cdot s .$$

Außerdem gilt für den Erwartungswert der Differenz von g_i und $r^T s$:

$$E\left(g_i - r^T s \right) = E\left(\sum_{\substack{u,u' \in U \text{ mit} \\ u \neq u' \text{ und} \\ h_i(u)=h_i(u')}} r_u \cdot s_{u'} \right) =$$

$$\sum_{\substack{u,u' \in U \text{ mit} \\ u \neq u'}} r_u \cdot s_{u'} \cdot \underbrace{\Pr\left(h_i(u) = h_i(u') \right)}_{= \frac{1}{w}\text{, da } h_i \text{ aus einer 2-universellen Familie von Hash-Funktionen gewählt wurde}} \overset{w = \lceil \frac{e}{\varepsilon} \rceil}{\leq} \underbrace{\sum_{\substack{u,u' \in U \\ \text{mit } u \neq u'}} r_u \cdot s_{u'} \cdot \frac{\varepsilon}{e}}_{\leq \|r\|_1 \cdot \|s\|_1 \cdot \frac{\varepsilon}{e}}$$

Die Markov-Ungleichung $\leftarrow$ $\boxed{Pr(X \geq a) \leq \frac{E(X)}{a}}$

liefert für $X := g_i - r^T s$ und $a := \varepsilon \cdot \|r\|_1 \cdot \|s\|_1$:

$$Pr\left((g_i - r^T s) \geq \varepsilon \cdot \|r\|_1 \cdot \|s\|_1\right) \leq \frac{E(X)}{\varepsilon \cdot \|r\|_1 \cdot \|s\|_1} \leq \frac{\|r\|_1 \cdot \|s\|_1 \cdot \frac{\varepsilon}{e}}{\varepsilon \cdot \|r\|_1 \cdot \|s\|_1}$$

$$= \frac{1}{e}$$

Da die Hash-Funktionen $h_1, \ldots, h_d$ unabhängig voneinander gewählt wurden, gilt

$$Pr\left(\forall i \in \{1, \ldots, d\} \text{ ist } (g_i - r^T s) \geq \varepsilon \cdot \|r\|_1 \cdot \|s\|_1\right) \leq \left(\frac{1}{e}\right)^d$$

$$\boxed{\begin{aligned} &\Leftrightarrow \widehat{r^T s} - r^T s \geq \varepsilon \cdot \|r\|_1 \cdot \|s\|_1 \\ &\Leftrightarrow \widehat{r^T s} \geq r^T s + \varepsilon \|r\|_1 \|s\|_1 \end{aligned}}$$

$$\underset{d = \lceil \ln(\frac{1}{\delta}) \rceil}{\leq} \left(\frac{1}{e}\right)^{\ln(\frac{1}{\delta})}$$

$$= \left(e^{-1}\right)^{-\ln(\delta)}$$

$$= e^{\ln \delta}$$

$$= \delta$$

Also gilt:

$$Pr\left(\widehat{r^T s} \geq r^T s + \varepsilon \|r\|_1 \|s\|_1\right) \leq \delta \quad , \quad \text{d.h.}$$

$$Pr\left(\widehat{r^T s} \leq r^T s + \varepsilon \|r\|_1 \|s\|_1\right) \geq 1 - \delta \; .$$

$\square$

Beispiel 4: Sicherheit von Passwörtern

aus: "New passwords approach" von Phil Scott in Communications of the ACM, Vol. 53, No. 9, 2010, Seite 15.

nach: "Popularity is Everything: A new approach to protecting passwords from statistical-guessing attacks" von S. Schechter, C. Herley und M. Mitzenmacher, Hot Topics in Security Conference 2010.

Ansatz:

Nutzer dürfen jedes mögliche Wort als Passwort wählen, vorausgesetzt, dass das Wort nicht schon von vielen anderen Nutzern als Passwort genutzt wird.

Realisierung:

Nutze einen Count-Min-Sketch zur Repräsentation von Informationen darüber, welches Wort von wie vielen Nutzern bereits als Passwort verwendet wird.

Jedesmal, wenn ein Nutzer ein neues Passwort wählt, schau im Count-Min-Sketch nach, um einen Schätzer dafür zu erhalten, von wie vielen Nutzern das Passwort schon verwendet wird.

Falls dieser Schätzer unterhalb eines vorher festgelegten Schwellwerts liegt, darf der Nutzer das Passwort nehmen; ansonsten muss er ein anderes Wort als Passwort wählen.

Vorteile:

- Durch "Wörterbuch-Attacken" können Angreifer nicht auf einen Schlag eine große Anzahl von Accounts knacken, da jedes Wort nur von wenigen Nutzern als Passwort verwendet wird
- "Gefährlich populäre Passwörter" werden vermieden.
- Im Count-Min-Sketch werden Passwörter nicht als Klartext gespeichert

Bemerkung:
z.B. verbot Twitter im Jahr 2010 die Nutzung der 390 beliebtesten Passwörter, darunter "ABCD", "1234" und "p@ssword".

~~Man kann zeigen, dass dieser Schätzwert stets mindestens so groß wie die tatsächliche Antwort auf die Bereichsanfrage ist, und dass er mit Wahrscheinlichkeit $\geq 1-\delta$ die tatsächliche Antwort um höchstens $2 \cdot \varepsilon \cdot (\log m) \cdot \|D(t)\|_1$ übersteigt~~

Beispiel 5: "Heavy Hitters"

Für eine feste Zahl α mit $0 < \alpha < 1$ wollen wir alle $u \in U$ finden, die mindestens

$\alpha \cdot \|D(t)\|_1$ oft in der Datenbank vorkommen.

Ein solches $u \in U$ wird "heavy hitter" bzgl α genannt.

Durch Verwenden der in Bsp 2 beschriebenen Methode für Bereichsanfragen kann man ein Verfahren konstruieren, das "approximative heavy hitters" ausgibt: Jedes $u \in U$, das mindestens

$\alpha \cdot \|D(t)\|_1$ oft in der DB vorkommt, wird garantiert ausgegeben; und mit Wahrscheinlichkeit $\geq 1-\delta$ wird kein Element ausgegeben, das weniger als

$(\alpha - \varepsilon) \cdot \|D(t)\|_1$ oft in der DB vorkommt.

Details dazu finden sich in der Originalarbeit von Cormode und Muthukrishnan.