Learning-based frequency estimation algorithms

Chen-Yu Hsu, Piotr Indyk, Dina Katabi, Ali Vakilian

ICLR’19

Frequency estimation

Extremely long sequence of N elements from set U

5 3 4 2 5 8 1 1 5 2 2 7 1 0 8 1 1 3 1 7 6 2 9 2 3 • • •

Goal: for each $i \in U$, estimate fraction of times it appeared, f_i

Challenge: U is huge, so you don't want to $\underbrace{\text{just count elements}}_{|U| \log N \text{ bits}}$

Standard tool: Hashing

Frequency estimation

Extremely long sequence of N elements from set U

5	3	4	2	5	8	1	1	5	2	2	7	1	0	8	1	1	3	1	7	6	2	9	2	3	…

$B \ll |U|$ buckets, uniformly random hash function $h: U \to [B]$

For all $i \in U$ and $j \in [B]$, $\mathbb{P}[h(i) = j] = \frac{1}{B}$

$\boldsymbol{i}$	0	1	2	3	4	5	6	7	8	9
$\boldsymbol{h(i)}$	4	2	2	1	3	4	4	5	4	4

Frequency estimation

Extremely long sequence of N elements from set U

$\boxed{5}$

$\text{counter}_1 = 0$	$\text{counter}_2 = 0$	$\text{counter}_3 = 0$	$\text{counter}_4 = 0$	$\text{counter}_5 = 0$
Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5

$\boldsymbol{i}$	0	1	2	3	4	5	6	7	8	9
$\boldsymbol{h(i)}$	4	2	2	1	3	4	4	5	4	4

Frequency estimation

Extremely long sequence of N elements from set U

5

$\mathrm{counter}_1 = 0$	$\mathrm{counter}_2 = 0$	$\mathrm{counter}_3 = 0$	$\mathrm{counter}_4 = 1$	$\mathrm{counter}_5 = 0$
Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5

$\boldsymbol{i}$	0	1	2	3	4	5	6	7	8	9
$\boldsymbol{h(i)}$	4	2	2	1	3	4	4	5	4	4

Frequency estimation

Extremely long sequence of N elements from set U

5 3

counter$_1$ = 0	counter$_2$ = 0	counter$_3$ = 0	counter$_4$ = 1	counter$_5$ = 0
Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5

$\boldsymbol{i}$	0	1	2	3	4	5	6	7	8	9
$\boldsymbol{h}(\boldsymbol{i})$	4	2	2	1	3	4	4	5	4	4

Frequency estimation

Extremely long sequence of N elements from set U

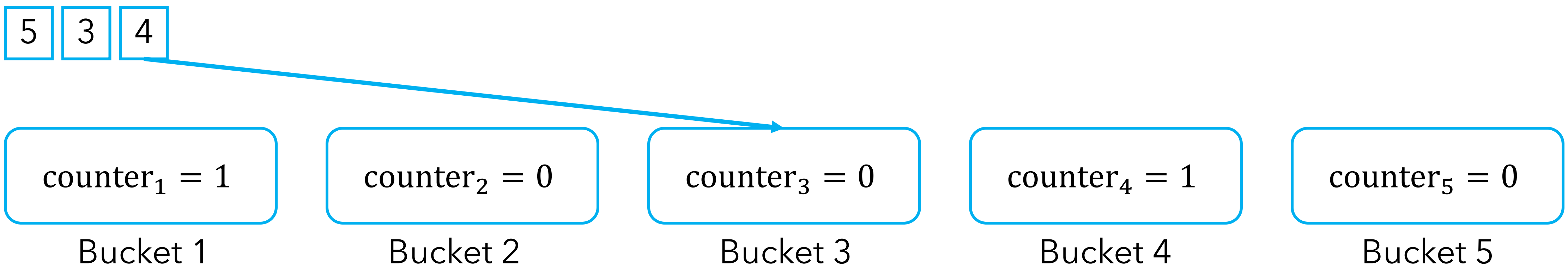
$\boxed{5}$ $\boxed{3}$

$\text{counter}_1 = 1$	$\text{counter}_2 = 0$	$\text{counter}_3 = 0$	$\text{counter}_4 = 1$	$\text{counter}_5 = 0$
Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5

$\boldsymbol{i}$	0	1	2	3	4	5	6	7	8	9
$\boldsymbol{h(i)}$	4	2	2	1	3	4	4	5	4	4

Frequency estimation

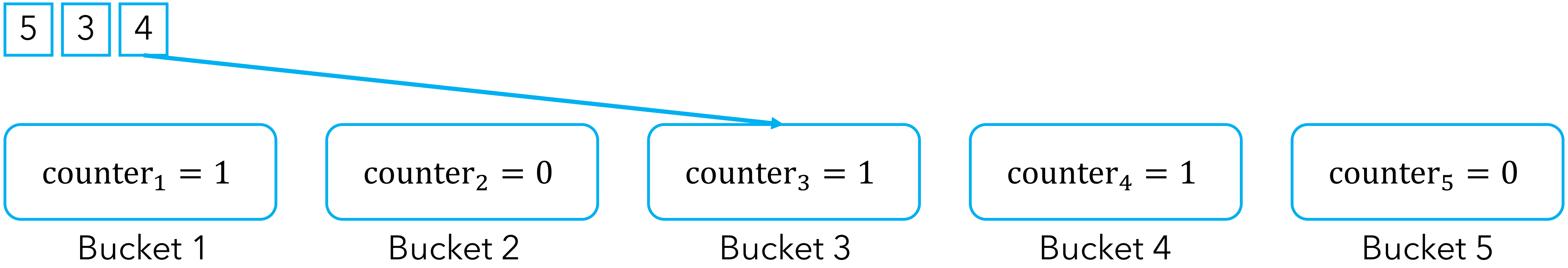
Extremely long sequence of N elements from set U



$\boldsymbol{i}$	0	1	2	3	4	5	6	7	8	9
$\boldsymbol{h(i)}$	4	2	2	1	3	4	4	5	4	4

Frequency estimation

Extremely long sequence of N elements from set U



$\boldsymbol{i}$	0	1	2	3	4	5	6	7	8	9
$\boldsymbol{h(i)}$	4	2	2	1	3	4	4	5	4	4

Frequency estimation

Extremely long sequence of N elements from set U

5	3	4	2	5	8	1	1	5	2	2	7	1	0	8	1	1	3	1	7	6	2	9	2	3	• • •

Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5
$\text{counter}_1 = 3$	$\text{counter}_2 = 11$	$\text{counter}_3 = 1$	$\text{counter}_4 = 8$	$\text{counter}_5 = 2$

$$\tilde{f}_i = \frac{1}{25} \cdot \text{count}_{h(i)} = \sum_{j:h(j)=h(i)} f_j \qquad (\Rightarrow \tilde{f}_i \geq f_i)$$

$\boldsymbol{i}$	0	1	2	3	4	5	6	7	8	9
$\boldsymbol{h(i)}$	4	2	2	1	3	4	4	5	4	4

Frequency estimation

Extremely long sequence of N elements from set U

5	3	4	2	5	8	1	1	5	2	2	7	1	0	8	1	1	3	1	7	6	2	9	2	3	…

$\text{counter}_1 = 3$	$\text{counter}_2 = 11$	$\text{counter}_3 = 1$	$\text{counter}_4 = 8$	$\text{counter}_5 = 2$
Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5

$$\tilde{f}_i = \frac{1}{25} \cdot \text{count}_{h(i)} = \sum_{j:h(j)=h(i)} f_j \qquad (\Rightarrow \tilde{f}_i \geq f_i)$$

$$f_2 = \frac{5}{25}$$

$$\tilde{f}_2 = \frac{1}{25} \cdot \text{count}_{h(2)} = \frac{1}{25} \cdot \text{count}_2 = \frac{11}{25} = f_1 + f_2$$

$\boldsymbol{i}$	0	1	2	3	4	5	6	7	8	9
$\boldsymbol{h(i)}$	4	2	2	1	3	4	4	5	4	4

Frequency estimation

Extremely long sequence of N elements from set U

5	3	4	2	5	8	1	1	5	2	2	7	1	0	8	1	1	3	1	7	6	2	9	2	3	• • •

Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5
$\mathrm{counter}_1 = 3$	$\mathrm{counter}_2 = 11$	$\mathrm{counter}_3 = 1$	$\mathrm{counter}_4 = 8$	$\mathrm{counter}_5 = 2$

$$\tilde{f}_i = \frac{1}{25} \cdot \mathrm{count}_{h(i)} = \sum_{j:h(j)=h(i)} f_j \qquad (\Rightarrow \tilde{f}_i \geq f_i)$$

$\boldsymbol{i}$	0	1	2	3	4	5	6	7	8	9
$\boldsymbol{h(i)}$	4	2	2	1	3	4	4	5	4	4

$$f_3 = \frac{3}{25}$$

$$\tilde{f}_3 = \frac{1}{25} \cdot \mathrm{count}_{h(3)} = \frac{1}{25} \cdot \mathrm{count}_1 = \frac{3}{25}$$

Overview

1. Frequency estimation
   i. **Analysis of a single hash function**
2. Improving estimation with domain knowledge

Model

Elements drawn from distribution D over $U = [n]$

$f_i = \mathbb{P}_{j \sim D}[j = i]$

Error: $\mathbb{E}_{i \sim D}\left[\left|\tilde{f}_i - f_i\right|\right] = \sum_{i=1}^{n} f_i \, \mathbb{E}\left[\left|\tilde{f}_i - f_i\right|\right]$

Error of a single hash function

Theorem: For a single hash, error $= \sum_{i=1}^{n} f_i \, \mathbb{E}\left[\left|\tilde{f}_i - f_i\right|\right] \leq \frac{1}{B}$

Proof:

$$\sum_{i=1}^{n} f_i \, \mathbb{E}\left[\left|\tilde{f}_i - f_i\right|\right] = \sum_{i=1}^{n} f_i \mathbb{E}\left[\sum_{j:h(j)=h(i)} f_j - f_i\right]$$

- Randomness is only over the hash function h:
    - for all $i \in U$ and $j \in [B], \mathbb{P}[h(i) = j] = \frac{1}{B}$
- Ignoring randomness of the sequence (assume it's really long)

Error of a single hash function

$$\sum_{i=1}^{n} f_i \, \mathbb{E}\left[\left|\tilde{f}_i - f_i\right|\right] = \sum_{i=1}^{n} f_i \mathbb{E}\left[\sum_{j:h(j)=h(i)} f_j - f_i\right]$$

$$= \sum_{i=1}^{n} f_i \mathbb{E}\left[\sum_{j\neq i:h(j)=h(i)} f_j\right]$$

$$= \sum_{i=1}^{n} f_i \sum_{j\neq i} f_j \mathbb{P}[h(j) = h(i)]$$

Error of a single hash function

$$\sum_{i=1}^{n} f_i \, \mathbb{E}\left[\left|\tilde{f}_i - f_i\right|\right] = \sum_{i=1}^{n} f_i \sum_{j \neq i} f_j \mathbb{P}[h(j) = h(i)]$$

$$\mathbb{P}[h(j) = h(i)] = \sum_{k=1}^{B} \mathbb{P}[h(j) = h(i) = k]$$

$$= \sum_{k=1}^{B} \mathbb{P}[h(j) = k] \cdot \mathbb{P}[h(i) = k] = \sum_{k=1}^{B} \left(\frac{1}{B} \cdot \frac{1}{B}\right) = \frac{1}{B}$$

Error of a single hash function

$$\sum_{i=1}^{n} f_i \, \mathbb{E}\left[\left|\tilde{f}_i - f_i\right|\right] = \sum_{i=1}^{n} f_i \sum_{j \neq i} f_j \mathbb{P}[h(j) = h(i)] \leq \left(\sum_{i=1}^{n} f_i\right)^2 \cdot \frac{1}{B} = \frac{1}{B}$$

$$\mathbb{P}[h(j) = h(i)] = \sum_{k=1}^{B} \mathbb{P}[h(j) = h(i) = k]$$

$$= \sum_{k=1}^{B} \mathbb{P}[h(j) = k] \cdot \mathbb{P}[h(i) = k] = \sum_{k=1}^{B} \left(\frac{1}{B} \cdot \frac{1}{B}\right) = \frac{1}{B}$$

Count-min

Extremely long sequence of N elements from set U

5	3	4	2	5	8	1	1	5	2	2	7	1	0	8	1	1	3	1	7	6	2	9	2	3	• • •

Hash function h_1	$\text{counter}_{1,1}$	$\text{counter}_{2,1}$	$\text{counter}_{3,1}$	$\text{counter}_{4,1}$	$\text{counter}_{5,1}$
	Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5
Hash function h_2	$\text{counter}_{1,2}$	$\text{counter}_{2,2}$	$\text{counter}_{3,2}$	$\text{counter}_{4,2}$	$\text{counter}_{5,2}$
Hash function h_3	$\text{counter}_{1,3}$	$\text{counter}_{2,3}$	$\text{counter}_{3,3}$	$\text{counter}_{4,3}$	$\text{counter}_{5,3}$

$$\tilde{f}_i = \frac{1}{25}\min\{\text{count}_{h_1(i),1}, \text{count}_{h_2(i),2}, \text{count}_{h_3(i),3}\}$$

Cormode, Muthukrishnan, J. of Algorithms ‘05

Overview

1. Frequency estimation
2. **Improving estimation with domain knowledge**

Heavy hitters

Extremely long sequence of N elements from set U

5	3	4	2	5	8	1	1	5	2	2	7	1	0	8	1	1	3	1	7	6	2	9	2	3	…

$\text{counter}_1 = 3$	$\text{counter}_2 = 11$	$\text{counter}_3 = 1$	$\text{counter}_4 = 8$	$\text{counter}_5 = 2$
Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5

$\boldsymbol{i}$	0	1	2	3	4	5	6	7	8	9
$\boldsymbol{h(i)}$	4	2	2	1	3	4	4	5	4	4

Key insight:

Heavy hitters increase error of elements they collide with

Heavy hitters

Key insight:

Heavy hitters increase error of elements they collide with

Algorithm IDEAL COUNT-MIN:

- Suppose you know the top-B_r most frequent elements
- Reserve B_r buckets to count individual frequencies
- Use $\underbrace{\text{hash function}}_{\text{Range is } [B - B_r]}$ to estimate other elements' frequencies

Algorithm Ideal Count-Min

Extremely long sequence of N elements from set U

5	3	4	2	5	8	1	1	5	2	2	7	1	1	8	1	1	3	1	7	6	2	9	2	3	…

Heavy hitters: 1 and 2

$\text{counter}_1 = 7$	$\text{counter}_2 = 5$	$\text{counter}_3 = 1$	$\text{counter}_4 = 7$	$\text{counter}_5 = 5$
Bucket for HH 1	**Bucket for HH 2**	Bucket 1	Bucket 2	Bucket 3

i	**1**	**2**	3	4	5	6	7	8	9
$h(i)$	**1**	**2**	3	1	2	2	3	2	2

Overview

1. Frequency estimation
2. Improving estimation with domain knowledge
    i. **Analysis of IDEAL COUNT-MIN**

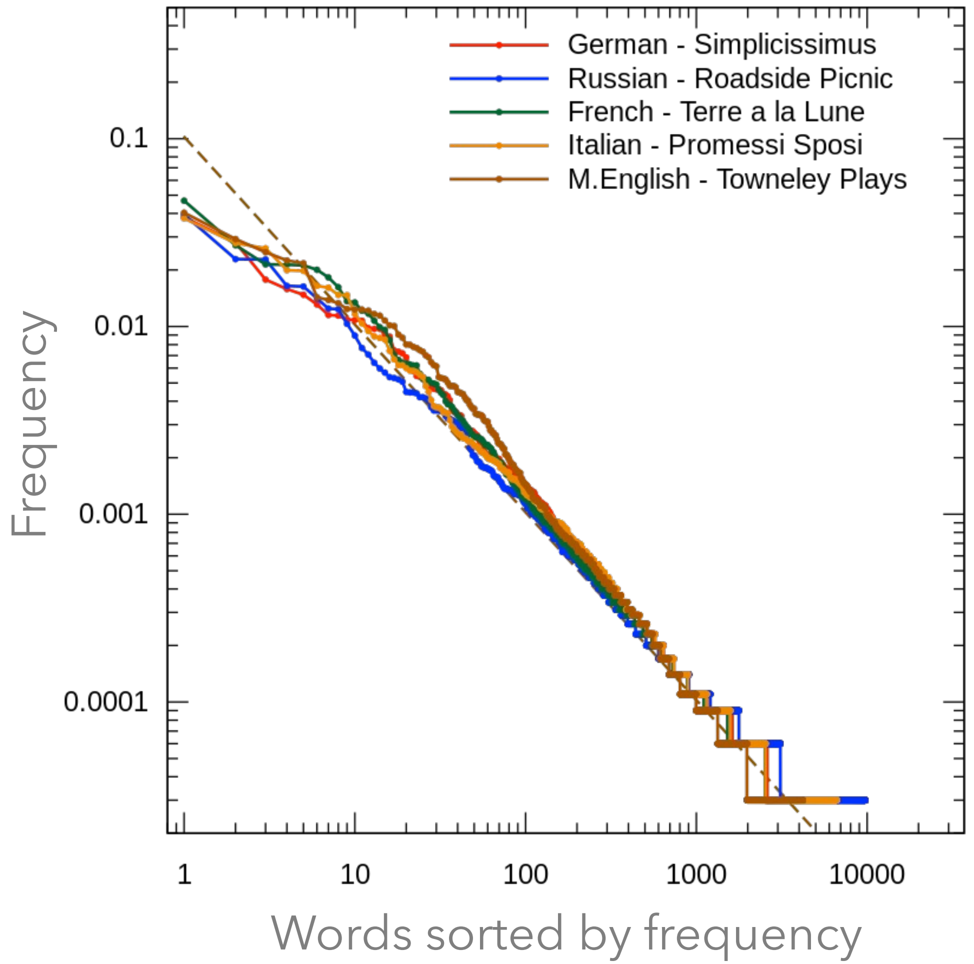
Model

D is a **Zipfian distribution**

Means elements can be sorted:

$$f_{i_1} \geq f_{i_2} \geq \cdots \geq f_{i_n} \text{ with } f_{i_j} \propto \frac{1}{j}$$

For ease of notation, assume $f_i \propto \frac{1}{i}$



Disclaimer: the paper has $f_i = \frac{1}{i}$, but here I'm sticking with $f_i \propto \frac{1}{i}$

As a result, the results in these slides are slightly different, but equivalent, to those in the paper

IDEAL COUNT-MIN error

Theorem: IDEAL COUNT-MIN has error

$$\sum_{i=1}^{n} f_i \, \mathbb{E}\left[\left|\tilde{f}_i - f_i\right|\right] = O\left(\frac{\log^2 \frac{n}{B_r}}{(B - B_r)\log^2 n}\right)$$

Example: if $B_r = \Theta(B) = \Theta(n)$

- Error of IDEAL COUNT-MIN = $O\left(\frac{1}{n \log^2 n}\right)$
- In contrast, error of single hash function = $O\left(\frac{1}{n}\right)$

Ideal Count-Min error

Theorem: Ideal Count-Min has error

$$\sum_{i=1}^{n} f_i \, \mathbb{E}\left[\left|\tilde{f}_i - f_i\right|\right] = O\left(\frac{\log^2 \frac{n}{B_r}}{(B - B_r)\log^2 n}\right)$$

Proof idea: For $i \leq B_r$, $\tilde{f}_i = f_i$, so

$$\sum_{i=1}^{n} f_i \, \mathbb{E}\left[\left|\tilde{f}_i - f_i\right|\right] = \sum_{i > B_r} f_i \mathbb{E}\left[\left|\tilde{f}_i - f_i\right|\right] \leq \left(\sum_{i > B_r} f_i\right)^2 \cdot \boxed{\frac{1}{B - B_r}}$$

By same exact argument as before, except hash function maps to $[B - B_r]$, not $[B]$

IDEAL COUNT-MIN error

$$\sum_{i=1}^{n} f_i \, \mathbb{E}\left[\left|\tilde{f}_i - f_i\right|\right] \leq \left(\sum_{i>B_r} f_i\right)^2 \cdot \frac{1}{B - B_r} = O\left(\left(\frac{\log \frac{n}{B_r}}{\log n}\right)^2 \cdot \frac{1}{B - B_r}\right)$$

Follows from harmonic number inequalities $H_n = \sum_{i=1}^{n} \frac{1}{i} = \Theta(\log n)$

Heavy hitters

Key insight:

Heavy hitters increase error of elements they collide with

Algorithm IDEAL COUNT-MIN:

- Suppose you **know** the top-B_r most frequent elements
- Reserve B_r buckets to count individual frequencies
- Use hash function to estimate other elements' frequencies

Also study setting with only a noisy predictor of heavy elements

- E.g., a machine-learned model
- Similar analysis

Overview

Improve error of **frequency estimation** algorithms

- Use *a priori* knowledge of **heaviest elements**,
- Or **predict** which are heaviest

Paper mostly focuses on **experiments**

Hopefully these slides help you understand the **theory** too!