Improved Frequency Estimation Algorithms with and without Predictions

Anders Aamand*    Justin Y. Chen*    Huy Nguyen$    Sandeep Silwal*    Ali Vakilian^    

*MIT, $Northeastern, ^TTIC

Thanks to Piotr Indyk for some slides
Will appear at NeurIPS ‘23
TLDR: Better frequency estimation algorithms under natural assumptions

Some background first ...
Frequency Estimation

Data stream $S$: a sequence of items from $[n]$
- E.g.: 8, 1, 7, 4, 6, 4, 10, 4, 4, 6, 8, 7, 5, 4, 2, 5, 6, 3, 9, 2

- Goal: at the end of the stream, given item $i \in [n]$ output an estimation $\tilde{f}_i$ of the frequency $f_i$ in $S$

- Sub-linear space?
Two heroes (assumptions)

- Learning-augmented: Access to heavy hitter predictions

- Zipfian: True frequencies are heavy-tailed (for theory analysis)
• Augment sketching algorithms like CountMin (CM) or CountSketch (CS) with heavy-hitter predictions.

“Learned CM/CS”

Not Heavy

Sketching Alg (e.g. CM)

Learned Oracle

Heavy

Unique Bucket

Stream element

Learned CM/CS
The story so far [Hsu et al.]

- Augment sketching algorithms like CountMin (CM) or CountSketch (CS) with heavy-hitter predictions.

- Short and sweet intuition: heavy elements are responsible for error!

---

Diagram:

- Stream element
- [n]
- Not Heavy
- Unique Bucket
- Heavy
- Learned Oracle
- Sketching Alg (e.g. CM)
- “Learned CM/CS”
Theoretical Analysis of [Hsu et al.]

\[ f_i \propto \frac{1}{i} \]

- Analyzed it under Zipfian distribution

- Error metric: \[ \mathbb{E} \left[ \sum_i f_i \cdot |\tilde{f}_i - f_i| \right] \]
  
  > “Average” error (Want better predictions for larger frequencies)
  
  > Generalizes to other weights such as uniform
  
  > Used in practical works ([Roy et al.’16])
Theoretical Analysis of [Hsu et al.]

- Analyzed it under Zipfian distribution
- Error metric: $\sum_i f_i \cdot |\tilde{f}_i - f_i|$
- B words of space
- Predictor correctly predicts top B heavy elements

<table>
<thead>
<tr>
<th></th>
<th>CountMin (CM)</th>
<th>Learned-CM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>$O \left( \frac{\log(n/B)}{\log n} \right)^2$</td>
</tr>
</tbody>
</table>
Is the story over?

Next Steps Questions:

- Advantage only when space (B) is large?
- What about CountSketch?

<table>
<thead>
<tr>
<th>Method</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountMin (CM)</td>
<td>1</td>
</tr>
<tr>
<td>Learned-CM</td>
<td>$O \left( \frac{\log(n/B)}{\log n} \right)^2$</td>
</tr>
</tbody>
</table>
Is the story over?

Next Steps Questions:

- Advantage only when space ($B$) is large?
- What about CountSketch?

Meta:

- Can we just get better algorithms under Zipfian assumptions?
- Do we even need predictions? “New” algorithms?
Our Results

Space = $B$ words, $n =$ # of elements, Error metric: $\sum_i f_i \cdot |\tilde{f}_i - f_i|$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Error Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountMin (CM)</td>
<td>$1$</td>
</tr>
<tr>
<td>Learned-CM</td>
<td>$\Theta \left( \frac{\log(n/B)}{\log n} \right)^2$</td>
</tr>
<tr>
<td>CountSketch (CS)</td>
<td>$\Theta \left( \frac{1}{\log n} \right)$</td>
</tr>
<tr>
<td>Learned-CS</td>
<td>$\Theta \left( \frac{\log(n/B)}{(\log n)^2} \right)$</td>
</tr>
</tbody>
</table>

Characterize exact performance of classical algos and learned variants (for our error metric and Zipfian data)
Our Results

Space = B words, n = # of elements, Error metric: $\sum_i f_i \cdot |\tilde{f}_i - f_i|$

<table>
<thead>
<tr>
<th></th>
<th>Learning variants are only useful over classic counterparts for large space</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountMin (CM)</td>
<td>1</td>
</tr>
<tr>
<td>Learned-CM</td>
<td>$\Theta \left( \frac{\log(n/B)}{\log n} \right)^2$</td>
</tr>
<tr>
<td>CountSketch (CS)</td>
<td>$\Theta \left( \frac{1}{\log n} \right)$</td>
</tr>
</tbody>
</table>
| Learned-CS               | $\Theta \left( \frac{\log(n/B)}{(\log n)^2} \right)$
For small space ($B \sim \text{polylog } n$):

- New alg outperforms even prior learned variants *without predictions*

Even better *with predictions*

<table>
<thead>
<tr>
<th>Method</th>
<th>Error Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learned-CS</td>
<td>(\Theta \left( \frac{\log(n/B)}{\log n} \right)^2)</td>
</tr>
<tr>
<td>New Alg</td>
<td>(O \left( \frac{\log B}{(\log n)^2} \right))</td>
</tr>
<tr>
<td>Learned-New Alg</td>
<td>(O \left( \frac{1}{(\log n)^2} \right))</td>
</tr>
</tbody>
</table>

Space = B words, n = # of elements

Error metric: \(\sum_i f_i \cdot |\tilde{f}_i - f_i|\)
<table>
<thead>
<tr>
<th>Alg</th>
<th>Error Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountMin (CM)</td>
<td>1</td>
</tr>
<tr>
<td>Learned-CM</td>
<td>$\Theta\left(\frac{\log(n/B)}{\log n}\right)^2$</td>
</tr>
<tr>
<td>CountSketch (CS)</td>
<td>$\Theta\left(\frac{1}{\log n}\right)$</td>
</tr>
<tr>
<td>Learned-CS</td>
<td>$\Theta\left(\frac{\log(n/B)}{(\log n)^2}\right)$</td>
</tr>
<tr>
<td>New Alg</td>
<td>$O\left(\frac{\log B}{(\log n)^2}\right)$</td>
</tr>
<tr>
<td>Learned-New Alg</td>
<td>$O\left(\frac{1}{(\log n)^2}\right)$</td>
</tr>
</tbody>
</table>

- For small space ($B \sim \text{polylog } n$):
  
  New alg outperforms even prior learned variants *without predictions*

- Even better *with predictions*
Zipfian assumption is also common in theory (e.g. [Milton-Price ’14])

Usually studied for ‘point wise’ error

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Error Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountMin (CM)</td>
<td>$1$</td>
</tr>
<tr>
<td>Learned-CM</td>
<td>$\Theta \left( \frac{\log(n/B)}{\log n} \right)^2$</td>
</tr>
<tr>
<td>CountSketch (CS)</td>
<td>$\Theta \left( \frac{1}{\log n} \right)$</td>
</tr>
<tr>
<td>Learned-CS</td>
<td>$\Theta \left( \frac{\log(n/B)}{(\log n)^2} \right)$</td>
</tr>
<tr>
<td>New Alg</td>
<td>$O \left( \frac{\log B}{(\log n)^2} \right)$</td>
</tr>
<tr>
<td>Learned-New Alg</td>
<td>$O \left( \frac{1}{(\log n)^2} \right)$</td>
</tr>
</tbody>
</table>
New Algorithm Intuition (No predictions)

CS additive error
New Algorithm Intuition

Why would you incur large error for small elements?
Better to predict them as 0!
New Algorithm Intuition

Why would you incur large error for small elements? Better to predict them as 0!

Don’t know beforehand which elements are small
New Algorithm Intuition

Why would you incur large error for small elements? Better to predict them as 0! Use the estimate of CS itself!
New Algorithm Intuition

Note: The additive error of CS (Δ) is a known quantity.
New Algorithm Intuition

Note: The additive error of CS ($\Delta$) is a known quantity.

Problem: Can potentially output 0 for large frequencies! ❗
Many small noisy CS tables

Algorithm 7  Frequency update algorithm

1: procedure UPDATE
2:       $T \leftarrow \Theta(\log \log n)$
3:   for $j = 1$ to $T - 1$ do
4:       $S_j \leftarrow$ CountSketch table with 3 rows and $\frac{B}{6T}$ columns
5:   end for
6:   $S_T \leftarrow$ CountSketch table with 3 rows and $\frac{B}{6}$ columns
7:   Input stream update in all of the $T$ CountSketch tables
8: 
9: end procedure
Idea:
Use estimate of smaller noisy tables to determine if we should output 0 or listen to the large CS table.

Space = B words, n = # of elements
Algorithm 7 Query

1: procedure QUERY
2: for $j = 1$ to $T - 1$ do
3: $\hat{f}_i^j \leftarrow$ estimate of the $i$th frequency given by table $S_j$
4: end for
5: $\hat{f}_i \leftarrow$ Median($\hat{f}_i^1, \ldots, \hat{f}_i^{T-1}$)
6: if $\hat{f}_i < \text{‘Appropriate Threshold’}$ then
7: Return 0
8: else
9: Return $\hat{f}_i^T$, the estimate given by table $S_T$
10: end if
11: end procedure

Idea:

Use estimate of smaller noisy tables to determine if we should output 0 or listen to the large CS table

Space = B words, n = # of elements
Algorithm 7  Frequency update algorithm

1: procedure UPDATE
2: \( T \leftarrow \Theta(\log \log n) \)
3: for \( j = 1 \) to \( T - 1 \) do
4: \( S_j \leftarrow \text{CountSketch table with 3 rows and } \frac{B}{6T} \text{ columns} \)
5: end for
6: \( S_T \leftarrow \text{CountSketch table with 3 rows and } \frac{B}{6} \text{ columns} \)
7: Input stream update in all of the \( T \) CountSketch tables
8: 
9: end procedure

Algorithm 7  Query

1: procedure QUERY
2: for \( j = 1 \) to \( T - 1 \) do
3: \( \hat{f}_i^j \leftarrow \text{estimate of the } i\text{th frequency given by table } S_j \)
4: end for
5: \( \hat{f}_i \leftarrow \text{Median}(\hat{f}_i^1, \ldots, \hat{f}_i^{T-1}) \)
6: if \( \hat{f}_i < \text{‘Appropriate Threshold’} \) then
7: Return 0
8: else
9: Return \( \hat{f}_i^T \), the estimate given by table \( S_T \)
10: end if
11: end procedure
Some Empirical Results

Test on two real-world datasets

- Internet search queries (AOL Dataset)
- Internet Traffic/IP (CAIDA Dataset)
- [Hsu et al.] obtained predictions (predictor trained on past versions of the data)
Why it works

Quick stretch time!
Algorithm 7 Frequency update algorithm

1: procedure UPDATE
2:    $T \leftarrow \Theta(\log \log n)$
3: for $j = 1$ to $T - 1$ do
4:    $S_j \leftarrow$ CountSketch table with 3 rows and $\frac{B}{6T}$ columns
5: end for
6: $S_T \leftarrow$ CountSketch table with 3 rows and $\frac{B}{6}$ columns
7: Input stream update in all of the $T$ CountSketch tables
8: 
9: end procedure

Algorithm 7 Query

1: procedure QUERY
2: for $j = 1$ to $T - 1$ do
3:    $\hat{f}_i^j \leftarrow$ estimate of the $i$th frequency given by table $S_j$
4: end for
5: $\tilde{f}_i \leftarrow$ Median($\hat{f}_i^1, \ldots, \hat{f}_i^{T-1}$)
6: if $\tilde{f}_i < \text{‘Appropriate Threshold’}$ then
7:    Return 0
8: else
9:    Return $\hat{f}_i^T$, the estimate given by table $S_T$
10: end if
11: end procedure
Why it works

Ideal Analysis: Pay CS error for ‘large’ frequencies, pay $f_i$ error for ‘small’ frequencies
Why it works

Ideal: Pay CS error for ‘large’ frequencies, pay $f_i$ error for ‘small’ frequencies

But it’s a bit asymmetric.
Why it works

Ideal: Pay CS error for ‘large’ frequencies, pay \( f_i \) error for ‘small’ frequencies

But it’s a bit asymmetric.

- Small elements: Even if something goes ‘wrong’, we only pay CS (= bounded) error

- Probability of going wrong \( \sim 1/\log n \) \([O(\log \log n)\] small tables for boosting)
Ideal: Pay CS error for ‘large’ frequencies, pay $f_i$ error for ‘small’ frequencies

But it’s a bit asymmetric.

- Small elements: Even if something goes ‘wrong’, we only pay CS (= bounded) error

- Probability of going wrong $\sim 1/\log n$ [$O(\log \log n)$ small tables for boosting]

<table>
<thead>
<tr>
<th>CountSketch (CS)</th>
<th>$\Theta \left( \frac{1}{\log n} \right)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Alg</td>
<td>$O \left( \frac{\log B}{(\log n)^2} \right)$</td>
</tr>
</tbody>
</table>
Why it works

Ideal: Pay CS error for ‘large’ frequencies, pay $f_i$ error for ‘small’ frequencies.

Large elements: If something goes ‘wrong’, we pay huge error: $f_i$!
Handling large elements

To output 0 on a large element, all estimates of small tables are *wayyyyy off*

---

**Algorithm 7 Query**

1: procedure QUERY
2:     for $j = 1$ to $T - 1$ do
3:         $\hat{f}_i^j \leftarrow$ estimate of the $i$th frequency given by table $S_j$
4:     end for
5:     $\tilde{f}_i \leftarrow$ Median($\hat{f}_i^1, \ldots, \hat{f}_i^{T-1}$)
6:     if $\tilde{f}_i <$ ‘Appropriate Threshold’ then
7:         Return 0
8:     else
9:         Return $\hat{f}_i^T$, the estimate given by table $S_T$
10:    end if
11: end procedure
Handling large elements

To output 0 on a element i, all estimates of small tables are wayyyy off

A majority of small estimates need to be off by $O(f_i)$.

Bound “probability deviation is very large” for single small table:
Handling large elements

To output 0 on an element i, all estimates of small tables are \textit{wayyyy off}

A majority of small estimates need to be off by $O(f_i)$.

Bound “probability deviation is very large” for single small table:

- Event 1: An element with frequency $> s$ collides with our element i [few elements due to Zipfian assumption!]
Handling large elements

To output 0 on a element i, all estimates of small tables are \textit{wayyyy off}

A majority of small estimates need to be off by \(O(f_i)\).

Probability deviation is very large for single small table:

- Event 1: An element with frequency > s collides with our element i [few elements due to Zipfian assumption!]
- Event 2: “Many” elements with frequency < s collide with element i
Handling large elements

To output 0 on a element i, all estimates of small tables are wayyyy off

A majority of small estimates need to be off by $O(f_i)$.

- Event 1: An element with frequency $> s$ collides with our element i [few elements due to Zipfian assumption!]

- Event 2: “Many” elements with frequency $< s$ collide with element i [Small probability due to many]

Has to be “many” because we know the error is huge

Small probability of colliding on “many” elements!
Why it works

Ideal: Pay CS error for ‘large’ frequencies, pay $f_i$ error for ‘small’ frequencies

But it’s a bit asymmetric.

• Small elements: Even if something goes ‘wrong’, we only pay CS (= bounded) error

• Probability of going wrong $\sim 1/\log n$

• Large elements: If something goes ‘wrong’, we pay huge error: $f_i$!

• Probability estimate is “$O(f_i)$ off” exactly cancels $f_i$ factor
Thank you!

1) Better understanding of frequency estimation for Zipfian data?

2) ‘Better’ ways to use predictions?

3) Other problems? Note: lots of work on learning-augmented algo design: https://algorithms-with-predictions.github.io/
Beyond Zipfian

Note:
Can obtain good errors without Zipfian assumption by estimating the ‘additive error’ of CountSketch on the fly
Our Results

Space = B words, n = # of elements, Error metric: \( \sum_i f_i \cdot |\tilde{f}_i - f_i| \)

<table>
<thead>
<tr>
<th>Method</th>
<th>Error Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountMin (CM)</td>
<td>1</td>
</tr>
<tr>
<td>Learned-CM</td>
<td>( \Theta \left( \frac{\log(n/B)}{\log n} \right)^2 )</td>
</tr>
<tr>
<td>CountSketch (CS)</td>
<td>( \Theta \left( \frac{1}{\log n} \right) )</td>
</tr>
<tr>
<td>Learned-CS</td>
<td>( \Theta \left( \frac{\log(n/B)}{(\log n)^2} \right) )</td>
</tr>
</tbody>
</table>

Note: Our paper is a merger with an older manuscript “(Learned) Frequency Estimation Algorithms under Zipfian Distribution”

Which analyzed the tight behavior of CS/CM and their learned variants (under Zipfian)