Lecture 3: Frequency Moments: $F_2$, Heavy Hitters
Administrivia, Plan

- Piazza: sign-up!
- PS1 releazed
- Scriber?

Plan:
- Frequency Moments
- Heavy Hitters
Part 1: Frequency Moments

- Let $f_i$ be frequency of $i$
  - Lecture 1: count one $f_i$
  - Lecture 2: count # of non-zeros

- Moment 1:
  - $\sum_i f_i$
  - Estimator with low space?
    - Just count

- Moment 2:
  - $\sum_i f_i^2$

\[
\sum_i f_i = 15
\]
\[
\sum_i f_i^2 = 95
\]
2\textsuperscript{nd} Moment: $F_2$

[Alon-Matias-Szegedy 1996]

- **Idea**: Rademacher random variables
  hash function $r: [n] \rightarrow \{-1, +1\}$

- **Algorithm (Tug-of-War)**: 
  store $z = \sum_i r(i) \cdot f_i$

- **Estimator**: $z^2$

Algorithm TOW ($F_2$):
- Init: $z = 0$
- when see element $i$:
  $z = z + r(i)$

Estimator:
$z^2$
Rademacher r.v.

- What if we have $m$ ones?
  - sum of $m$ random $\pm 1$’s

- How much is $z = \sum r(i)$ roughly?
  - Say, $|z|$?
  - $E[z] = 0$
  - $Var[z] = m$
  - Apply Chebyshev:
    - $|z| \leq O(\sqrt{m})$ with constant probability
  - In fact tight

Algorithm TOW ($F_2$):
- Init: $z = 0$
- when see element $i$:
  - $z = z + r(i)$
- Estimator:
  - $z^2$
Analysis

- $E[z^2] = \ldots = \sum_i f_i^2$

- $Var[z^2] \leq E[z^4] = \ldots \leq O\left(\sum f_i^2\right)^2$

- Randomness?
  - $O(\log n)$ for $h$ that is 4-wise independent

- Can apply the average trick:
  - Take $k = O\left(\frac{1}{\epsilon^2}\right)$ counters
  - Obtain: $1 + \epsilon$ approximation in $O\left(\frac{1}{\epsilon^2 \log n}\right)$ space

Algorithm TOW ($F_2$):
- Init: $z = 0$
- when see element $i$:
  $z = z + r(i)$

Estimator:
$z^2$
Linearity

• Important property

Algorithm TOW ($F_2$):
• Init: $z = 0$
• when see element $i$: $z = z + r(i)$
Estimator: $z^2$

$z = z' + z''$ (for $f = f' + f''$)
Similarly for difference!

- Estimate for $\sum (f_i' - f_i'')^2$
  
  $(z' - z'')^2$

- How about $\sum |f_i' - f_i'''|$?  
  – will see later in the class
General streaming model

• At each moment, an update is:
  \((i, \delta_i)\) : increase \(i^{th}\) entry by \(\delta_i\) (may be negative!)

• Linear algorithm \(S\) handles easily:
  \(- S(f + e_i\delta_i) = S(f) + S(e_i\delta_i)\)
  \(- We’ll call \(S\) a sketch\)

• [Nguyen-Li-Woodruff’14]: in fact any algorithm for general streaming might as well be linear!
Part 2: Heavy Hitters

• How about max frequency?

• Impossible to approximate in sublinear space!

• Will settle for an even more modest goal:
  – can detect the max-frequency element if it is very heavy

<table>
<thead>
<tr>
<th>IP</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>𝑛</td>
<td>1</td>
</tr>
</tbody>
</table>
Heavy Hitters: Iteration 1

[Charikar-Chen-FarachColton’04, Cormode-Muthukrishnan’05]

- **Definition:** $i$ is $\phi$-heavy if $f_i \geq \phi \sum_j f_j$
- **Will find them in space** $O(1/\phi)$
- **Idea:** hash functions!
  - $h: [n] \rightarrow [w]$ random
  - Element $i$ goes to bucket $h(i)$
  - In a bucket?
    - Sum frequencies there

Estimator for $f_i$?

$$\hat{f}_i = S(h(i))$$

- $\hat{f}_2 = 2$
- $\hat{f}_5 = 3$
- $\hat{f}_7 = 2$
- $\hat{f}_{11} = 2$
Iteration 1: analysis

• Let’s analyze:
  – Estimator of frequency for element $i$
    \[
    \hat{f}_i = S(h(i)) = f_i + \sum_{\{j : h(j) = h(i)\}} f_j
    \]

• How much extra “chaff” is there?
Iteration 1: extra chaff

- \( S(h(i)) = f_i + \sum_{\{j: h(j) = h(i)\}} f_j \)
- Extra “chaff”:
  - \( E[C] = \sum_j \Pr[h(j) = h(i)] \cdot f_j = \frac{\sum_{j\neq i} f_j}{w} \)
- Is \( S(h(i)) \) an unbiased estimator?
  - No!
    - Bias is at most \( \frac{\sum_j f_j}{w} \): small for \( f_i \gg \frac{\sum_j f_j}{w} \)
- Done?
  - Yes: by Markov \( C \leq \frac{10 \sum_j f_j}{w} \) with 90% prob.
Iteration 1: really done?

• Estimator:
\[
\hat{f}_i = S(h(i)) = f_i + \sum_{\{j: h(j) = h(i)\}} f_j \\
= f_i + C
\]
where \( C \leq O(\sum_j f_j / w) \) with 90% prob

– for \( w = O\left(\frac{1}{\epsilon \phi}\right) \), and \( f_i \geq \phi \sum_j f_j \)

\[
C \leq \epsilon f_i \Rightarrow \hat{f}_i \text{ is a } 1 + \epsilon \text{ approximation!}
\]

• Issues?
  – Only constant probability
  – For many indices, it is an overestimate!

Fundamental issue: if \( i \) and \( j \) collide, can’t know if it’s \( i \) or \( j \) with high frequency;
but must have many collisions to reduce space
Iteration 2: **CountMin**

- **Median trick!**
  - Use $L = O(\log n)$ hash tables with hash functions $h_j$

```
Algorithm CountMin:

Initialize(r, L):
    array S[L][w]
    L hash functions $h_1 \ldots h_L$, into {1,...w}

Process(int i):
    for(j=0; j<L; j++)
        S[j][ h_j(i) ] += 1;

Estimator:
    foreach i in PossibleIP {
        $\hat{f}_i = median_j(S[j][h_j(i)])$;
    }
```
Algorithm CountMin:

**Initialize** \((r, L)\):
- array \(S[L][w]\)
- \(L\) hash functions \(h_1 \ldots h_L\), into \(\{1, \ldots w\}\)

**Process** \((\text{int } i)\):
- for \((j=0; j<L; j++)\)
  - \(S[j][h_j(i)] += 1;\)

**Estimator**:
- foreach \(i \in \text{PossibleIP}\) {
  - \(\hat{f}_i = \text{median}_j(S[j][h_j(i)]);\)
  - \(\text{min} \)
}

**CountMin: analysis**

- Consider an index \(i\)
- Each table gives
  - \(\hat{f}_i = f_i \pm \epsilon \phi\) with 90% probability
- Median is a \(\pm \epsilon \phi\) with \(1 - 1/n^2\) probability
  - Apply union bound over all \(i \in [n]\)
  - All are \(\pm \epsilon \phi\), with \(1 - 1/n\) probability
- Alternative estimator?
  - Take \(\text{MIN}\) instead of median
CountMin: overall

• Iterate over all $i$’s

• Heavy hitters: $\frac{f_i}{\sum f_j} \geq \phi$
  – If $\frac{f_i}{\sum f_j} \leq \phi(1 - \epsilon)$, not in the output
  – If $\frac{f_i}{\sum f_j} \geq \phi(1 + \epsilon)$, reported as heavy hitter

• Space: $O \left( \frac{\log^2 n}{\epsilon \phi} \right)$ bits

• Issues?
  – Time: to iterate $\Omega(n)$

Algorithm CountMin:

Initialize($r$, $L$):
  array $S[L][w]$
  $L$ hash functions $h_1 \ldots h_L$, into $\{1, \ldots w\}$

Process(int $i$):
  for($j=0$; $j<L$; $j++$)
    $S[j][h_j(i)] += 1$;

Estimator:
  foreach $i$ in PossibleIP {
    $\hat{f}_i = \text{median}_j(S[j][h_j(i)])$;
  } min
CountMin: time

• Can improve time; space degrades to $O\left(\frac{\log^3 n}{\epsilon \phi}\right)$ bits
• **Idea:** dyadic intervals
  – Each level with its own sketch
  – Find heavy hitters by following down the tree all the heavy hitters (in intermediary)
A variant: CountSketch

- Is CountMin linear?
  - CountMin($f' + f''$) from CountMin($f'$) and CountMin($f''$)?
  - Just sum the two!
    - sum the 2 arrays, assuming we use the same hash function $h_j$

- What about $f = f' - f''$?
  - “Heavy hitter”: if $|f_i| \geq \phi \sum_j |f_j| = \phi \cdot ||f||_1$
  - “min” is an issue
  - But median is still ok
  - Ideas to improve it further?
    - Use Tug of War $r$ in each bucket => CountSketch
    - Better in certain cases
Recap

• 2\textsuperscript{nd} moment:
  – Tug-Of-War (sum of random ±1’s)

• Linearity:
  – Can add/subtract sketches easily

• Max-frequency:
  – Can only do heavy hitters
  – Hash functions to distribute elements
  – CountMin
    • [https://sites.google.com/site/countminsketch/](https://sites.google.com/site/countminsketch/)
  – CountSketch: CountMedian+TugOfWar