Heavy Hitters

Lecture 08
September 17, 2020
Models

Richer model:

- Want to estimate a function of a vector $x \in \mathbb{R}^n$ which is initially assume to be the all $0$’s vector.
- Each element $e_j$ of a stream is a tuple $(i_j, \Delta_j)$ where $i_j \in [n]$ and $\Delta_j \in \mathbb{R}$ is a real-value: this updates $x_{i_j}$ to $x_{i_j} + \Delta_j$. ($\Delta_j$ can be positive or negative)
Models

Richer model:

- Want to estimate a function of a vector $x \in \mathbb{R}^n$ which is initially assume to be the all 0’s vector.

- Each element $e_j$ of a stream is a tuple $(i_j, \Delta_j)$ where $i_j \in [n]$ and $\Delta_j \in \mathbb{R}$ is a real-value: this updates $x_{i_j}$ to $x_{i_j} + \Delta_j$. ($\Delta_j$ can be positive or negative)

- $\Delta_j > 0$: cash register model. Special case is $\Delta_j = 1$.

- $\Delta_j$ arbitrary: turnstile model

- $\Delta_j$ arbitrary but $x \geq 0$ at all times: strict turnstile model

- Sliding window model: interested only in the last $W$ items (window)
Frequent Items Problem

What is $F_k$ when $k = \infty$?

$$F_\infty = \max_i f_i$$
Frequent Items Problem

What is $F_k$ when $k = \infty$? Maximum frequency.
Frequent Items Problem

What is $F_k$ when $k = \infty$? Maximum frequency.

$F_\infty$ very brittle and hard to estimate with low memory. Can show strong lower bounds for very weak relative approximations.

If $n = \infty$ is very large.
Frequent Items Problem

What is $F_k$ when $k = \infty$? Maximum frequency.

$F_\infty$ very brittle and hard to estimate with low memory. Can show strong lower bounds for very weak relative approximations.

Hence settle for weaker (additive) guarantees.
Frequent Items Problem

What is $F_k$ when $k = \infty$? Maximum frequency.

$F_\infty$ very brittle and hard to estimate with low memory. Can show strong lower bounds for very weak relative approximations.

Hence settle for weaker (additive) guarantees.

**Heavy Hitters Problem:** Find all items $i$ such that $f_i > m/k$ for some fixed $k$.

Heavy hitters are very frequent items.
Finding Majority Element

Majority element problem:

- Offline: given an array/list $A$ of $m$ integers, is there an element that occurs more than $m/2$ times in $A$?
- Streaming: is there an $i$ such that $f_i > m/2$?
Finding Majority Element

Streaming-Majority:

\[ c = 0, \ s \leftarrow \text{null} \]

While (stream is not empty) do

If \((e_j = s)\) do

\[ c \leftarrow c + 1 \]

ElseIf \((c = 0)\)

\[ c = 1 \]

\[ s = e_j \]

Else

\[ c \leftarrow c - 1 \]

endWhile

Output \(s, c\)

Claim:

If there is a majority element \(i\) then algorithm outputs \(s = i\) and \(c = i + \frac{1}{2}\).

Caveat:

Algorithm may output incorrect element if no majority element. Can verify correctness in a second pass.
Finding Majority Element

**Streaming-Majority:**

\[
c = 0, \quad s \leftarrow \text{null}
\]

While (stream is not empty) do

If \( (e_j = s) \) do

\[
c \leftarrow c + 1
\]

ElseIf \( (c = 0) \)

\[
c = 1
\]

\[
s = e_j
\]

Else

\[
c \leftarrow c - 1
\]

endWhile

Output \( s, c \)

**Claim:** If there is a majority element \( i \) then algorithm outputs \( s = i \) and \( c \geq f_i - m/2 \).
Finding Majority Element

**Streaming-Majority:**

\[ c = 0, \ s \leftarrow \text{null} \]

While (stream is not empty) do

If \( e_j = s \) do

\[ c \leftarrow c + 1 \]

ElseIf \( c = 0 \)

\[ c = 1 \]

\[ s = e_j \]

Else

\[ c \leftarrow c - 1 \]

endWhile

Output \( s, c \)

**Claim:** If there is a majority element \( i \) then algorithm outputs \( s = i \) and \( c \geq f_i - m/2 \).

**Caveat:** Algorithm may output incorrect element if no majority element. Can verify correctness in a second pass.
Misra-Gries Algorithm

Heavy Hitters Problem: Find all items $i$ such that $f_i > m/k$.

**MisraGreis($k$):**

1. $D$ is an empty associative array
2. While (stream is not empty) do
   - $e_j$ is current item
   - If ($e_j$ is in keys($D$))
     - $D[e_j] \leftarrow D[e_j] + 1$
   - Else if ($|\text{keys}(A)| < k - 1$) then
     - $D[e_j] \leftarrow 1$
   - Else
     - for each $\ell \in \text{keys}(D)$ do
       - $D[\ell] \leftarrow D[\ell] - 1$
     - Remove elements from $D$ whose counter values are 0
3. endWhile
4. For each $i \in \text{keys}(D)$ set $\hat{f}_i = D[i]$
5. For each $i \not\in \text{keys}(D)$ set $f_i = 0$
\[ k = 3 \]

\[ \begin{bmatrix} 2 \\mid 1 \\mid 1 \end{bmatrix} \]

\[ \hat{f}_1 = 2 \quad \hat{f}_2 = 0 \quad \hat{f}_3 = 0 \quad \hat{f}_4 = 0 \quad \hat{f}_5 = 1 \]
Analysis

Space usage $O(k)$.

**Theorem**

For each $i \in [n]$: $f_i - \frac{m}{k+1} \leq \hat{f}_i \leq f_i$.

**Corollary**

Any item with $f_i > \frac{m}{k}$ is in $D$ at the end of the algorithm.

A second pass to verify can be used to verify correctness of elements in $D$. 
Proof of Correctness

Theorem

For each $i \in [n]$: $f_i - \frac{m}{k+1} \leq \hat{f}_i \leq f_i$.

\[ \hat{f}_i \geq \max\{0, f_i - \frac{m}{k+1}\} \]
Proof of Correctness

**Theorem**

For each $i \in [n]$: $f_i - \frac{m}{k+1} \leq \hat{f}_i \leq f_i$.

Easy to see: $\hat{f}_i \leq f_i$. Why?
Proof of Correctness

**Theorem**

For each $i \in [n]$: $f_i - \frac{m}{k+1} \leq \hat{f}_i \leq f_i$.

Easy to see: $\hat{f}_i \leq f_i$. Why?

Alternative view of algorithm:

- Maintains counts $C[i]$ for each $i$ (initialized to 0). Only $k$ are non-zero at any time.
- When new element $e_j$ comes
  - If $C[e_j] > 0$ then increment $C[e_j]$
  - Elself less then $k$ positive counters then set $C[e_j] = 1$
  - Else decrement all positive counters (exactly $k$ of them)
- Output $\hat{f}_i = C[i]$ for each $i$
Proof of Correctness

Want to show: \( f_i - \hat{f}_i \leq m/(k + 1) \):

\[
\hat{f}_i \geq f_i - \frac{m}{k+1}
\]

\[
\hat{f}_i \leq f_i
\]
Proof of Correctness

Want to show: \( f_i - \hat{f}_i \leq m/(k + 1) \):

- Suppose we have \( \ell \) occurrences of \( k \) counters being decremented.
Proof of Correctness

Want to show: $f_i - \hat{f}_i \leq m/(k + 1)$:

- Suppose we have $\ell$ occurrences of $k$ counters being decremented. Then $\ell k + \ell \leq m$ which implies $\ell \leq m/(k + 1)$.
- Consider $\alpha = (f_i - \hat{f}_i)$ as items are processed. Initially 0. How big can it get?
Proof of Correctness

Want to show: \( f_i - \hat{f}_i \leq m/(k + 1) \):

- Suppose we have \( \ell \) occurrences of \( k \) counters being decremented. Then \( \ell k + \ell \leq m \) which implies \( \ell \leq m/(k + 1) \).
- Consider \( \alpha = (f_i - \hat{f}_i) \) as items are processed. Initially 0. How big can it get?
  - If \( e_j = i \) and \( C[i] \) is incremented \( \alpha \) stays same
Proof of Correctness

Want to show: \( f_i - \hat{f}_i \leq m/(k + 1) \):

- Suppose we have \( \ell \) occurrences of \( k \) counters being decremented. Then \( \ell k + \ell \leq m \) which implies \( \ell \leq m/(k + 1) \).
- Consider \( \alpha = (f_i - \hat{f}_i) \) as items are processed. Initially 0. How big can it get?
  - If \( e_j = i \) and \( C[i] \) is incremented \( \alpha \) stays same
  - If \( e_j = i \) and \( C[i] \) is not incremented then \( \alpha \) increases by one and \( k \) counters decremented — charge to \( \ell \)
Proof of Correctness

Want to show: \( f_i - \hat{f}_i \leq m/(k + 1) \):

- Suppose we have \( \ell \) occurrences of \( k \) counters being decremented. Then \( \ell k + \ell \leq m \) which implies \( \ell \leq m/(k + 1) \).
- Consider \( \alpha = (f_i - \hat{f}_i) \) as items are processed. Initially 0. How big can it get?
  - If \( e_j = i \) and \( C[i] \) is incremented \( \alpha \) stays same
  - If \( e_j = i \) and \( C[i] \) is not incremented then \( \alpha \) increases by one and \( k \) counters decremented — charge to \( \ell \)
  - If \( e_j \neq i \) and \( \alpha \) increases by 1 it is because \( C[i] \) is decremented — charge to \( \ell \)

Hence total number of times \( \alpha \) increases is at most \( \ell \).
Proof of Correctness

Want to show: \( f_i - \hat{f}_i \leq m/(k + 1) \):

- Suppose we have \( \ell \) occurrences of \( k \) counters being decremented. Then \( \ell k + \ell \leq m \) which implies \( \ell \leq m/(k + 1) \).

- Consider \( \alpha = (f_i - \hat{f}_i) \) as items are processed. Initially \( 0 \). How big can it get?
  - If \( e_j = i \) and \( C[i] \) is incremented \( \alpha \) stays same
  - If \( e_j = i \) and \( C[i] \) is not incremented then \( \alpha \) increases by one and \( k \) counters decremented — charge to \( \ell \)
  - If \( e_j \neq i \) and \( \alpha \) increases by \( 1 \) it is because \( C[i] \) is decremented — charge to \( \ell \)

- Hence total number of times \( \alpha \) increases is at most \( \ell \).
Deterministic to Randomized Sketches

Cannot improve $O(k)$ space if one wants additive error of at most $m/k$. Nice to have a deterministic algorithm that is near-optimal

Why look for randomized solution?

- Obtain a sketch that allows for deletions
- Additional applications of sketch based solutions
- Will see Count-Min and Count sketches

\[
\begin{align*}
1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, & \ldots, \\
X, X, X, 3, 2, 1, 2, 3, 3, X, X, & \ldots,
\end{align*}
\]
Basic Hashing/Sampling Idea

**Heavy Hitters Problem:** Find all items $i$ such that $f_i > m/k$.

- Let $b_1, b_2, \ldots, b_k$ be the $k$ heavy hitters
- Suppose we pick $h : [n] \rightarrow [ck]$ for some $c > 1$
- $h$ spreads $b_1, \ldots, b_k$ among the buckets ($k$ balls into $ck$ bins)
- In ideal situation each bucket can be used to count a separate heavy hitter