Streaming Algorithms

By Koko Nanahji
Model

• Input: sequence of integers $x_1, x_2, ..., x_n$
  • $x_i \in U$

• Goal: compute some function $f_i$ on the input stream
Focus

• Main focus:
  • Reduce the amount of memory used to process the input
Outline

Model
Counting
Heavy Hitters
CountMin
CountSketch
Conclusion
Counting

• Input: Sequence of integers $x_1, x_2, ..., x_n$
  • Problem: Find the number of elements in the input stream
Naïve Solution

- Keep a counter
  - Space Complexity: $O(\log_2 n)$ bits
Approx. Solution

• We would like to find an approximate solution $\hat{n}$ such that
  • Given $\epsilon > 0$ and $\delta < 1$
    • Find an estimate $\hat{n}$ such that:
      • $P(|n - \hat{n}| > \epsilon n) < \delta$
Morris’ Algorithm

- Algorithm:
  - $X := 0$
  - For each item seen:
    - Increment $X$ with probability $\frac{1}{2^X}$
  - Output: $\hat{n} := 2^X - 1$

- Intuitively: $X \sim \log_2 n$
Analysis

- Let $X_n$ denote the value of $X$ after seeing the $i^{th}$ item
  - We will show that:
    - $E(2^{X_n}) = n + 1$
Analysis

• Show that $E(2^{X_n}) = n + 1$

• Proof:
  • By induction on $n$
  • Base Case: $X_0 = 0$ => $E(2^{X_0}) = E(2^0) = 1$
  • Induction Hypothesis: Assume the claim is true for $n$ prove for $n + 1$
Analysis

• Inductive Step:
  • \( E(2^{X_{n+1}}) = \sum_{\forall i} 2^i P(X_{n+1} = i) \)
  • \( E(2^{X_{n+1}}) = \sum_{\forall i} 2^i (P(X_n = i - 1) \cdot P(X_n \text{ gets incremented} \mid X_n = i) + P(X_n = i) \cdot P(X_n \text{ does not get incremented} \mid X_n = i)) \)
  • \( E(2^{X_{n+1}}) = \sum_{\forall i} 2^i \left( P(X_n = i - 1) \cdot \frac{1}{2^{i-1}} + P(X_n = i) \cdot \left(1 - \frac{1}{2^i}\right)\right) \)
  • \( E(2^{X_{n+1}}) = \sum_{\forall i} 2^i P(X_n = i - 1) \cdot \frac{1}{2^{i-1}} + \sum_{\forall i} 2^i P(X_n = i) \cdot \left(1 - \frac{1}{2^i}\right) \)
  • \( E(2^{X_{n+1}}) = \sum_{\forall i} 2^i P(X_n = i - 1) \cdot \frac{1}{2^{i-1}} + \sum_{\forall i} 2^i P(X_n = i) - \sum_{\forall i} 2^i P(X_n = i) \cdot \frac{1}{2^i} \)
  • \( E(2^{X_{n+1}}) = \sum_{\forall i} 2 P(X_n = i - 1) + \sum_{\forall i} 2^i P(X_n = i) - \sum_{\forall i} P(X_n = i) \)
  • \( E(2^{X_{n+1}}) = \sum_{\forall i} 2^i P(X_n = i) + \sum_{\forall i} P(X_n = i) \)
  • \( E(2^{X_{n+1}}) = E(2^{X_n}) + 1 = (n + 1) + 1 \)
Analysis

• We can show the following:
  • $E(2^{X_n}) = n + 1$
  • $E(2^{2X_n}) = \frac{3}{2}n^2 + \frac{3}{2}n + 1$ (similar to previous proof)

• Therefore, we have that:
  • $Var(2^{X_n}) < \frac{1}{2}n^2$

• Hence: By Chebyshev’s inequality
  • $P(|n - \hat{n}| > \epsilon n) < \frac{1}{2\epsilon^2}$

• Note: This is not very useful when $\epsilon \geq 1$
  • We will improve this algorithm soon
Space Complexity

• Since we have: \( E(2^{X_n}) = n + 1 \).
  • We can show: \( P(2^{X_n} - 1 \geq n^c) \leq \frac{1}{n^{c-1}} \)

• Meaning, with high probability we have
  • \( 2^{X_n} - 1 \geq n^c \)
  • \( 2^{X_n} \geq n^c - 1 \)
  • \( X_n \geq \log_2(n^c - 1) \)

• Hence, to store \( X_n \) we can show that we need
  • \( O(\log_2(\log_2 n)) \) bits with high probability
Morris+ Algorithm

• Improve Morris’ algorithm by using the mean trick
  • Run $s > 1$ independent copies of Morris’ algorithm and average their outputs
Morris+ Algorithm

• Let $X^j$ be the output of the $j^{th}$ copy of Morris’ algorithm after seeing the $i^{th}$ item

  • $Y_i = \frac{1}{s} \sum_j \left(2^{X^j} - 1 \right)$

  • By linearity of expectation we have $E(2^{Y_n}) = n + 1$

  • But

    • $Var(2^{Y_n}) < \frac{1}{2s} n^2 < Var\left(2^{X^j_n}\right) = \frac{1}{2} n^2$

• By Chebyshev’s inequality we have:

  • $P(|n - \hat{n}| > \epsilon n) < \frac{1}{2s \epsilon^2}$

  • Then for $\delta$ error probability we set

    • $s > \frac{1}{2\epsilon^2 \delta}$
Morris+ Space

• Space complexity: $O(s \cdot \log_2(\log_2 n))$ bits with high probability

• For $\delta$ error probability we need
  • $O(\frac{1}{2e^2\delta} \cdot \log_2(\log_2 n))$ bits with high probability

• We will improve this space complexity using the median trick
Morris++ Algorithm

• Improve space complexity by using the median trick
  • Run $t$ independent copies of Morris+ algorithm
    • Such that $s = \frac{3}{2 \cdot \epsilon^2}$
      • Meaning the error probability of each Morris+ is $\frac{1}{3}$
    • Output the median estimate
Morris++ Algorithm

• Note:
  • Since the error probability of each Morris+ is $\frac{1}{3}$
  • Expected number of Morris+ instantiations that succeed is $\frac{2t}{3}$
  • Hence, for the median to be a bad estimate at least half of the Morris+ instantiations must fail
  • We will show that this is not likely
Morris++ Algorithm

• Let $Z_i = 1$ if $i^{th}$ Morris+ instantiation succeeds, otherwise $Z_i = 0$

• We will bound $P \left( \sum_i Z_i \leq \frac{t}{2} \right)$
  
  - $P \left( \sum_i Z_i \leq \frac{t}{2} \right) \leq P \left( \left| \sum_i Z_i - \frac{2t}{3} \right| \leq \frac{t}{2} - \frac{2t}{3} \right)$
  
  - $P \left( \sum_i Z_i \leq \frac{t}{2} \right) \leq P \left( |\sum_i Z_i - E(\sum_i Z_i)| \leq -\frac{t}{6} \right)$

• By Hoeffding bound
  
  - $P \left( \sum_i Z_i \leq \frac{t}{2} \right) \leq e^{-\frac{1}{18t}}$

• Hence, for $t \geq \left[ 18 \ln \frac{1}{\delta} \right]$
  
  - $P \left( \sum_i Z_i \leq \frac{t}{2} \right) \leq \delta$
Morris++ Space

- If we set $s \cdot t = \theta \left( \frac{1}{\epsilon^2} \ln \frac{1}{\delta} \right)$

- We get that we need $O \left( \frac{1}{\epsilon^2} \ln \left( \frac{1}{\delta} \right) \cdot \log_2 \left( \log_2 n \right) \right)$ bits with high probability
Outline

Model  Counting  Heavy Hitters  CountMin  CountSketch  Conclusion
Heavy Hitters

• Input: sequence of integers $x_1, x_2, ..., x_n$
  • $x_i \in \{1,2, ..., m\}$

• Let $f_i$ be the frequency of element $i$ in the given sequence

• Goal: Given some integer $K$ find the elements that have $f_i > \frac{n}{K}$

• There is a simple two pass algorithm named Misra–Gries Algorithm (was covered last time)
CountMin Algorithm

• Pick $t$ hash functions such that $h_i : [m] \rightarrow [w]$ from a universal family of hash functions

• Create a 2D array $C[t][w]$ initially all cells set to 0

• Algorithm:
  • For each item $x$:
    • For $i$ from 1 to $t$
      • Increment $C[i][h_i(x)]$

• Then the frequency of item $x$ is $\min_{\forall i} C[i][h_i(x)]$
CountMin Algorithm

<table>
<thead>
<tr>
<th>Fun</th>
<th>$C[t][0]$</th>
<th>$C[t][1]$</th>
<th>$C[t][2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$h_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$h_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Insert 3

<table>
<thead>
<tr>
<th>Fun</th>
<th>$C[t][0]$</th>
<th>$C[t][1]$</th>
<th>$C[t][2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$h_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$h_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fun</td>
<td>$C[t][0]$</td>
<td>$C[t][1]$</td>
<td>$C[t][2]$</td>
</tr>
<tr>
<td>-----</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>$h_0$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$h_1$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$h_2$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Insert 5

<table>
<thead>
<tr>
<th>Fun</th>
<th>$C[t][0]$</th>
<th>$C[t][1]$</th>
<th>$C[t][2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$h_1$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$h_2$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
## Insert 5

<table>
<thead>
<tr>
<th>Fun</th>
<th>$C[t][0]$</th>
<th>$C[t][1]$</th>
<th>$C[t][2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$h_1$</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$h_2$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
### Insert 10

<table>
<thead>
<tr>
<th>Fun</th>
<th>$C[t][0]$</th>
<th>$C[t][1]$</th>
<th>$C[t][2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$h_1$</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$h_2$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Insert 10

<table>
<thead>
<tr>
<th>Fun</th>
<th>$C[t][0]$</th>
<th>$C[t][1]$</th>
<th>$C[t][2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$h_1$</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>$h_2$</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Query 3

Minimum

<table>
<thead>
<tr>
<th>Fun</th>
<th>$C[t][0]$</th>
<th>$C[t][1]$</th>
<th>$C[t][2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$h_1$</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>$h_2$</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Query 5

Minimum

<table>
<thead>
<tr>
<th>Fun</th>
<th>$C[t][0]$</th>
<th>$C[t][1]$</th>
<th>$C[t][2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$h_1$</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>$h_2$</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
CountMin Algorithm

• We will do some analysis on this algorithm

• Let $f_x$ be the actual count of $x$

• Let $\hat{f}_x$ be the estimated count of $x$

• Note: $f_x \leq \hat{f}_x$

• We will show that $P(\hat{f}_x \geq f_x + \epsilon n) \leq \delta$
CountMin Algorithm

• We will compute the $E(C[j][h_j(x)])$
  
  $E(C[j][h_j(x)]) = E\left(\sum_{\forall s:h_j(s)=h_j(x)} f_s\right)$

• $E(C[j][h_j(x)]) = f_x + \frac{1}{w}\sum_{\forall s\neq x} f_s$

• $E(C[j][h_j(x)]) < f_x + \frac{n}{w}$
CountMin Algorithm

• We have
  • \( E(C[j][h_j(x)]) < f_x + \frac{n}{w} \)

• We will bound \( P(C[j][h_j(x)] \geq f_x + \frac{2n}{w}) \)
  • \( P(C[j][h_j(x)] \geq f_x + \frac{2n}{w}) \leq P(C[j][h_j(x)] - f_x \geq \frac{2n}{w}) \)
  • By Chebyshev’s inequality:
    • \( P(C[j][h_j(x)] \geq f_x + \frac{2n}{w}) \leq \frac{E(C[j][h_j(x)]-f_x)}{\frac{2n}{w}} \)
    • \( P(C[j][h_j(x)] \geq f_x + \frac{2n}{w}) \leq \frac{f_x+\frac{n}{w}-f_x}{\frac{2n}{w}} \leq \frac{1}{2} \)
CountMin Algorithm

• So far we have:
  - \( P\left( C[j][h_j(x)] \geq f_x + \frac{2n}{w} \right) \leq \frac{1}{2} \)

• We will bound \( P\left( \hat{f}_x \geq f_x + \frac{2n}{w} \right) \)
  - \( P\left( \hat{f}_x \geq f_x + \frac{2n}{w} \right) = P\left( \min_{\forall j} C[j][h_j(x)] \geq f_x + \frac{2n}{w} \right) \)
  - \( P\left( \hat{f}_x \geq f_x + \frac{2n}{w} \right) = \prod_j P\left( C[j][h_j(x)] \geq f_x + \frac{2n}{w} \right) \)
  - \( P\left( \hat{f}_x \geq f_x + \frac{2n}{w} \right) \leq \left( \frac{1}{2} \right)^t \)

• If we set \( w = \frac{2}{\epsilon} \) and \( t = \log_2 \frac{1}{\delta} \) we will have
  - \( P\left( \hat{f}_x \geq f_x + \epsilon n \right) \leq \delta \)
CountMin Algorithm

• Space complexity:
  • $O(w \cdot t) = O\left(\frac{2}{\epsilon} \cdot \log_2 \frac{1}{\delta}\right)$
Heavy Hitters with CountMin

- We extend CountMin as follows:
  - For each row of intervals in figure, we store a separate count-min structure
  - For each row, count-min of that row treats two elements that fall into the same interval as the same element
  - Note that the value at any ancestor of a node is at least as big as the value at that node
Heavy Hitters with CountMin

To get the $K$ heavy-hitters:

- Explore the tree starting from the root
  - Only explore the children of intervals that have frequency at least $\frac{n}{K}$
Heavy Hitters with CountMin

Analysis:

- Space complexity \( O\left(\frac{2}{\epsilon} \cdot \log_2 \frac{1}{\delta} \cdot \log_2 n\right) \)
- Time complexity to get \( K \) heavy hitters is \( O(K \cdot \log_2 n) \)
  - For any given row, the sum over all frequencies in that row is \( n \)
  - Thus, in any row, there are at most \( K \) intervals with frequency \( \frac{n}{K} \)
  - Therefore, we only explore the children of at most \( K \) intervals in any given row
CountSketch Algorithm

• Pick $t$ hash functions such that $h_i : [m] \rightarrow [w]$ from a universal family of hash functions
• Pick $t$ hash functions such that $s_i : [m] \rightarrow \{-1, +1\}$ from a universal family of hash functions
• Create a 2D array $C[t][w]$ initially all cells set to 0
• Algorithm:
  • For each item $x$:
    • For $i$ from 1 to $t$
      • $C[i][h_i(x)] = C[i][h_i(x)] + s_i(x)$
  • Then the frequency of item $x$ is $\hat{f}_x = \text{median} \{C[i][h_i(x)] \cdot s_i(x)\}$
CountSketch Algorithm

• We can show that
  • When we set $t = O(\log n)$ and $w = \frac{3}{\epsilon^2}$
  • Then, with high probability
    • $|\hat{f}_x - f_x| \leq \epsilon \cdot (\sum_j f_j^2)$
    • $(\sum_j f_j^2) \ll n$ for skewed distributions
CountSketch Algorithm

• Space complexity:
  • $O\left(\frac{1}{\varepsilon^2} \cdot \log_2 \frac{1}{\delta}\right)$
Outline

- Model
- Counting
- Heavy Hitters
- CountMin
- CountSketch
- Conclusion
Conclusion

• Randomized approximate algorithms provide simple solutions to important problems

• Mean and median tricks help us improve the error probability and space complexity of algorithm
References

• The material presented is from the following source:
  • https://www.sketchingbigdata.org/fall20/lec/notes.pdf

• I have used the following resources to understand some of the proofs better:
  • http://web.stanford.edu/class/cs369g/files/lectures/lec7.pdf
  • http://web.stanford.edu/class/cs369g/files/lectures/lec8.pdf