Heavy hitter computation over data stream

Slides modified from Rajeev Motwani (Stanford University) and Subhash Suri (University of California)

Frequency Related Problems

Top-k most frequent elements
Find all elements with frequency > 0.1%
What is the frequency of element 3?
What is the total frequency of elements between 8 and 14?
How many elements have non-zero frequency?

An Old Chestnut: Majority

• A sequence of N items.
• You have constant memory.
• In one pass, decide if some item is in majority (occurs > N/2 times)?

N = 12; item 9 is majority

Misra-Gries Algorithm ('82)

• A counter and an ID.
  - If new item is same as stored ID, increment counter.
  - Otherwise, decrement the counter.
  - If counter 0, store new item with count = 1.
• If counter > 0, then its item is the only candidate for majority.

A generalization: Frequent Items (Karp 03)

Find k items, each occurring at least N/(k+1) times.

<table>
<thead>
<tr>
<th>ID</th>
<th>ID2</th>
<th>ID3</th>
<th>ID4</th>
<th>ID5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>count</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Algorithm:
- Maintain k items, and their counters.
- If next item x is one of the k, increment its counter.
- Else if a zero counter, put x there with count = 1
- Else (all counters non-zero) decrement all k counters

Frequent Elements: Analysis

- A frequent item’s count is decremented if all counters are full; it erases k+1 items.
- If x occurs > N/(k+1) times, then it cannot be completely erased.
- Similarly, x must get inserted at some point, because there are not enough items to keep it away.
**Problem of False Positives**

- False positives in Misra-Gries (MG) algorithm
  - It identifies all true heavy hitters, but not all reported items are necessarily heavy hitters.
  - How can we tell if the non-zero counters correspond to true heavy hitters or not?
- A second pass is needed to verify.
- False positives are problematic if heavy hitters are used for billing or punishment.
- What guarantees can we achieve in one pass?

**Approximation Guarantees**

- Find heavy hitters with a guaranteed approximation error [MM02]
- Manku-Motwani (Lossy Counting)
  - Suppose you want \( \phi \)-heavy hitters --- items with freq > \( \phi N \)
  - An approximation parameter \( \epsilon \), where \( \epsilon \ll \phi \)
    (E.g., \( \phi = .01 \) and \( \epsilon = .0001 \); \( \phi = 1\% \) and \( \epsilon = .01\% \))
  - Identify all items with frequency > \( \phi N \)
  - No reported item has frequency < \( (\phi - \epsilon)N \)
- The algorithm uses \( O(1/\epsilon \log(\epsilon N)) \) memory

---

**MM02 Algorithm 1: Lossy Counting**

*Step 1: Divide the stream into ‘windows’*

Window 1  Window 2  Window 3

Window-size \( W \) is a function of support \( s \) – specify later...

---

**Lossy Counting in Action ...**

Empty  First Window

At window boundary, decrement all counters by 1

---

**Lossy Counting continued ...**

Next Window

At window boundary, decrement all counters by 1

---

**Error Analysis**

How much do we undercount?

If current size of stream = \( N \)
and window-size = \( 1/\epsilon \)
then frequency error \( \leq \# \) windows = \( \epsilon N \)

Rule of thumb:
Set \( \epsilon = 10\% \) of support \( s \)
Example:
Given support frequency \( s = 1\% \),
set error frequency \( \epsilon = 0.1\% \)
Putting it all together...

Output:
Elements with counter values exceeding $(s-\epsilon)N$

Approximation guarantees
- Frequencies underestimated by at most $\epsilon N$
- No false negatives
- False positives have true frequency at least $(s-\epsilon)N$

How many counters do we need?
- Worst case bound: $1/\epsilon \log N$ counters

Misra-Gries revisited
- Running MG algorithm with $k = 1/\epsilon$ counters also achieves the $\epsilon$-approximation.
- Undercounts any item by at most $\epsilon N$.
- In fact, MG uses only $O(1/\epsilon)$ memory.
- Lossy Counting slightly better in per-item processing cost
  - MG requires extra data structure for decrementing all counters
  - Lossy Counting is $O(1)$ amortized per item.

Algorithm 2: Sticky Sampling

Stream
- Create counters by sampling
- Maintain exact counts thereafter

Algorithm 2 contd...

For finite stream of length $N$

\[
\text{Sampling rate} = \frac{2}{\epsilon} \log(1/\delta) \quad (3.4)
\]

Output:
Elements with counter values exceeding $(s-\epsilon)N$

Approximation guarantees (probabilistic)
- Frequencies underestimated by at most $\epsilon N$
- No false negatives
- False positives have true frequency at least $(s-\epsilon)N$

Same error guarantees as Lossy Counting but probabilistic

Number of counters?

Finite stream of length $N$
- Sampling rate: $\frac{2}{\epsilon} \log(1/\delta)$

Infinite stream with unknown $N$
- Gradually adjust sampling rate,
- Remove the element with certain probability

In either case,
- Expected number of counters $= \frac{2}{\epsilon} \log 1/\delta$
- Independent of $N$