Count-Min Sketch

Anil Maheshwari

School of Computer Science
Carleton University
Canada

Review

Outline

Review

Count-Min Sketch

Complexity Analysis

Probability Preliminaries

Proof of the claim

Conclusions

Misra & Gries [82] Algorithm

Finding Heavy Hitters

Input: A stream consisting of n elements and fixed integer $k < n$.
Output: Report all heavy hitters, i.e. elements that occur $\geq n/k$ times.

---

1. Initialize k bins, each with null element and a counter with 0.
2. **For** each element x in the stream **do**
   if $x \in$ Bin b **then** increment bin b's counter

   elseif find a bin whose counter is 0 and

   - Assign x to this bin
   - Assign 1 to its counter

   else decrement the counter of every bin.
3. Output elements in the bins.

Analysis of Misra and Gries Algorithm

Claim

Let $f_x^* =$ Frequency of x in the stream. Each heavy hitter x is in one of the bins with counter value $\geq f_x^* - n/k$.

Correctness: What can be the minimum value of the counter of a heavy hitter?

Running Time:
Initializing k bins: $O(k)$ time
Processing each element requires looking at $O(k)$ bins.
Total Run Time = $O(nk)$

Space: $O(k)$

Reference: J. Misra and D. Gries,"Finding repeated elements" in Science of Computer Programming, Vol. 2 (2): 143 -152, 1982.

Count-Min Sketch

Generalize More

For a data stream, using very little space, we are interested to report

1. All the elements that occur frequently, e.g at least 2% times.
2. For each element, its (approximate) frequency.

Count-Min Sketch Data Structure

Input: An array (stream) A consisting of n numbers and r hash functions $h_1, \ldots, h_r$, where $h_i : \mathbb{N} \to \{1, \ldots, b\}$

Output: $CMS[\cdot, \cdot]$ table consisting of r rows and b columns

for $i = 1$ *to* r **do**
    for $j = 1$ *to* b **do**
        $CMS[i, j] \leftarrow 0$
    end
end
for $i = 1$ *to* n **do**
    for $j = 1$ *to* r **do**
        $CMS[j, h_j(A[i])] \leftarrow CMS[j, h_j(A[i])] + 1$
    end
end
return $CMS[\cdot, \cdot]$

Illustration of Algorithm

Let $b = 10$ and $r = 3$.
Assume that stream $A = xyy$.
Assume the following h-values for x and y:
For x: $h_1(x) = 3, h_2(x) = 8$, and $h_3(x) = 5$
For y: $h_1(y) = 6, h_2(y) = 8$, and $h_3(y) = 1$

$CMS[*, *] =$

	1	2	3	4	5	6	7	8	9	10
1										
2										
3										

for $i = 1$ *to* n **do**
    for $j = 1$ *to* r **do**
        $CMS[j, h_j(A[i])] \leftarrow CMS[j, h_j(A[i])] + 1$
    end
end

Updating CMS table

An example with $b = 10$ and $r = 3$ and assume that stream $A = xyy$

After Initialization:

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0

Execution of Algorithm

An example with $b = 10$ and $r = 3$ and assume that stream $A = xyy$

Assume the following h-values for x and y:

For x: $h_1(x) = 3, h_2(x) = 8$, and $h_3(x) = 5$
For y: $h_1(y) = 6, h_2(y) = 8$, and $h_3(y) = 1$

	1	2	3	4	5	6	7	8	9	10
1										
2										
3										

Updating CMS table

Insertion of x: $h_1(x) = 3, h_2(x) = 8$, and $h_3(x) = 5$:

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0

After inserting x:

	1	2	3	4	5	6	7	8	9	10
1	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	1	0	0	0	0	0

Updating CMS table

Insertion of 1st y: $h_1(y) = 6, h_2(y) = 8$, and $h_3(y) = 1$ that hashes to locations 6,8, and 1:

	1	2	3	4	5	6	7	8	9	10
1	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	1	0	0	0	0	0

After inserting 1st y:

	1	2	3	4	5	6	7	8	9	10
1	0	0	1	0	0	1	0	0	0	0
2	0	0	0	0	0	0	0	2	0	0
3	1	0	0	0	1	0	0	0	0	0

Updating CMS table

Insertion of 2nd y (hashes to same locations 6,8, and 1):

	1	2	3	4	5	6	7	8	9	10
1	0	0	1	0	0	1	0	0	0	0
2	0	0	0	0	0	0	0	2	0	0
3	1	0	0	0	1	0	0	0	0	0

After inserting 2nd y:

	1	2	3	4	5	6	7	8	9	10
1	0	0	1	0	0	2	0	0	0	0
2	0	0	0	0	0	0	0	3	0	0
3	2	0	0	0	1	0	0	0	0	0

Complexity Analysis

Observations

Let $n =$ Total number of items in the stream.
$f_x^* =$ True frequency of x in the stream.

Let $f_x = \min\{CMS[1, h_1(x)], \ldots, CMS[r, h_r(x)]\}$. Report f_x as the estimate on the frequency of x.

Observations:

1. The size of CMS table ($= br$) is independent of n.
2. CMS table can be computed in $O(br + nr)$ time.
3. For any $x \in A$, and for any $j = 1, \ldots, r$, $CMS[j, h_j(x)] \geq f_x^*$
4. f_x is an overestimate as $f_x \geq f_x^*$

Assume - Proof comes later

Claim

Let $b = \frac{2}{\epsilon}$. Then $Pr[f_x - f_x^* \geq \epsilon n] \leq \frac{1}{2^r}$

Corollary

With probability at least $1 - 1/2^r$, $f_x^* \leq f_x \leq f_x^* + \epsilon n$

Reporting Frequent Elements

Suppose we want to report all the elements of A that occur approximately $\geq n/k$ times for some integer k.

- In the Claim, set $\epsilon = 1/3k$. Then $b = \frac{2}{\epsilon} = 6k$.
- Construct CMS table of size $br = 6kr$.
- Scan A and compute the entries in the CMS table.
- Maintain a set of $O(k)$ items that occur most frequently among all the elements in A scanned so far.
  How?

Heap Data Structure

The items are stored in a HEAP with f_x values as the key.

What is a Heap?

An array that stores n elements and supports:

- Find Max or Min: Report the element with the smallest/largest key value in Heap in $O(1)$ time.
- Insert(x, k): Insert element x with key k in Heap in $O(\log n)$ time.
- Delete(x): Delete element x from Heap in $O(\log n)$ time.
- $\ldots$

Reporting Frequent Elements contd.

Assume we have scanned $i-1$ items and have updated the CMS table and the heap.
Consider the i-th item (say $x = A[i]$) and we perform the following:

1. For $j = 1$ to r: update the CMS table by executing $CMS[j, h_j(x)] \leftarrow CMS[j, h_j(x)] + 1$.
2. Let $f_x = \min\{CMS[1, h_1(x)], \ldots, CMS[r, h_r(x)]\}$.
   If $f_x \geq i/k$, do:
   2.1 If $x \in$ heap, delete x and re-insert it again with the updated f_x value.
   2.2 If $x \notin$ heap, then insert it in the heap and remove all the elements whose count is less than i/k.

Reporting Frequent Elements contd.

Claim

[Cormode and Muthukrishnan 2005] Elements that occur approx. n/k times in a data stream of size n can be reported in $O(kr + nr + n \log k)$ time using $O(kr)$ space with high probability.

Proof.

Recall Corollary: $f_x^* \leq f_x \leq f_x^* + \epsilon n = f_x^* + n/3k$.
This implies:

- Heap contains elements whose frequency is at least $n/k - n/3k = 0.667n/k$ (with high probability).
- Size of heap = $O(k)$
- Time Complexity: $O(br + nr + n \log k) = O(kr + nr + n \log k)$ as $b = 6k$.
- Total Space= $O(br + k) = O(kr)$

□

Probability Preliminaries

Markov's Inequality

Let us recall Markov's inequality:

Theorem

Let X be a non-negative discrete random variable and $s > 0$ be a constant. Then $P(X \geq s) \leq E[X]/s$.

Proof.

$$\begin{aligned} E[X] &= \sum_{i=0}^{\infty} i.P(X = i) \\ &\geq \sum_{i=s}^{\infty} i.P(X = i) \\ &\geq s \sum_{i=s}^{\infty} P(X = i) \\ &= sP(X \geq s). \end{aligned}$$

Hence, $P(X \geq s) \leq E[X]/s$. □

Linearity of Expectation

Let X and Y be two random variables mapping elements of a sample space S to real numbers. Assume that expected values $E[X]$ and $E[Y]$ are finite. Linearity of Expectation says that $E[X+Y] = E[X] + E[Y]$ (Note that X and Y need not be independent.).

$$\begin{aligned} E[X+Y] &= \sum_{\omega \in S} (X+Y)[\omega] \cdot P(\omega) \\ &= \sum_{\omega \in S} (X[\omega] + Y[\omega]) \cdot P(\omega) \\ &= \sum_{\omega \in S} (X[\omega] \cdot P(\omega) + Y[\omega] \cdot P(\omega)) \\ &= \sum_{\omega \in S} X[\omega] \cdot P(\omega) + \sum_{\omega \in S} Y[\omega] \cdot P(\omega) \\ &= E[X] + E[Y] \end{aligned}$$

This generalizes to the sum of n random variables: $E[X_1 + \cdots + X_n] = E[X_1] + \cdots + E[X_n]$.

An Example

1. Roll a fair die n times and sum total the outcomes. What is the expected value of this sum?
2. You toss a fair coin n times. What is the expected number of Heads?
3. What is the probability that the number of Heads is at least $\frac{4}{5}n$?

Proof of the claim

Bounding f_x

Claim

Let $b = \frac{2}{\epsilon}$. Then $Pr[f_x - f_x^* \geq \epsilon n] \leq \frac{1}{2^r}$

Proof Sketch: Let V be the set of different values in the stream A.
Define indicator r.v. I_y corresponding to each value $y \in A$ as follows:

$$I_y = \begin{cases} 1 & \text{if } h_j(y) = h_j(x) \\ 0, & \text{otherwise} \end{cases}$$

Note: $Pr(I_y = 1) = 1/b$, as the hash function h_j maps y uniformly at random in one of the m buckets of row j in the CMS table.
Thus, $E[I_y] = 1 \cdot Pr(I_y = 1) + 0 \cdot Pr(I_y = 0) = 1/b$.

Proof Contd.

$$CMS[j, h_j(x)] = f_x^* + \sum_{\substack{y \in V \\ y \neq x}} I_y * f_y^* \tag{1}$$

$$E[CMS[j, h_j(x)]] = f_x^* + E[\sum_{\substack{y \in V \\ y \neq x}} I_y * f_y^*] \tag{2}$$

Proof Contd.

By Linearity of Expectation, we have

$$E[CMS[j, h_j(x)]] = f_x^* + \sum_{\substack{y \in V \\ y \neq x}} E[I_y] * f_y^* \tag{3}$$

$$= f_x^* + \sum_{\substack{y \in V \\ y \neq x}} \frac{1}{b} f_y^* \tag{4}$$

$$= f_x^* + \frac{1}{b} \sum_{\substack{y \in V \\ y \neq x}} f_y^* \tag{5}$$

$$\leq f_x^* + \frac{n}{b} \tag{6}$$

Proof Contd.

By setting $b = \frac{2}{\epsilon}$, we obtain

$$E[CMS[j, h_j(x)]] \leq f_x^* + \frac{n}{b} = f_x^* + \epsilon n/2 \tag{7}$$

Define r. v.: $X_j = |CMS[j, h_j(x)] - f_x^*|$

- $E[X_j] \leq n/b = \epsilon n/2$.
- Apply Markov's inequality: $Pr(X_j > 2(\epsilon n/2)) \leq 1/2$.
- Therefore, $Pr(X_j > \epsilon n) \leq 1/2$ holds for all $j \in \{1, \ldots, r\}$.
- X_j is independent of X_k as hash functions h_j and h_k are independent for any $k \neq j$.
- For $f_x = \min\{CMS[1, h_1(x)], \ldots, CMS[r, h_r(x)]\}$, $Pr[|f_x - f_x^*| \geq \epsilon n] \leq \frac{1}{2^r}$

□

Conclusions

Conclusions on CMS

Simple idea with important applications.

Consider a vector $v = (v_1, v_2, \ldots, v_n)$.
Initially $v = 0$.
Update at time t is a pair (j, c): $v_j \leftarrow v_j + c$.
Using only small space, answer queries of the form

1. Point Query: Report v_i
2. Range Query $[l, r]$: Report $\sum_{i=l}^{r} v_i$
3. Inner product of two vectors: $u \cdot v$
4. In general, c can be positive or negative - replace $\min$ by median.

Reference: An improved data stream summary: the count-min sketch and its applications, G. Cormode and S. Muthukrishnan, J. Algorithms 55(1): 58-75, 2005.