Bloom Filters & Count-Min Sketch

Anil Maheshwari

School of Computer Science
Carleton University
Canada

August 6, 2018

Outline

Bloom Filters

Problem Definition

Let U be the universe.
Input: A subset $S \subseteq U$.
Query: For any $q \in U$, decide whether $q \in S$.

Objective

Answer queries quickly and use very little extra space.

SPAM Detection

U = All possible email addresses;
S = My collection of non-junk email addresses.
Query: Given any $q \in U$, report whether $q \in S$.

History of Bloom Filters

- Proposed by Bloom in CACM 1970 - *Space/Time tradeoffs in Hash Coding with Allowable Errors*. (7000 Citations)
- Space-Efficient Probabilistic Data Structure for Membership Testing
- May have false positives
- Numerous Variants: Counting Filters, Dynamic Filters with insertion/deletion of elements in S,
- Vast Applications:Estimating size of union/intersection of sets, Avoid cashing 'one-hit wonders', Google Bigtable, Chrome's uses it to detect malicious URLs.
- Refined Analysis in 2008 by Bose et al.

Bloom Filter Data Structure

Data Structure

An array B consisting of m bits and k hash functions $h_1, h_2, \ldots, h_k$, where $h_i : U \rightarrow \{1, \ldots, m\}$

Initialization

$B \leftarrow 0$.
For all $x \in S$, set
$B[h_1(x)] = B[h_2(x)] = \cdots = B[h_k(x)] = 1$.

Queries

Answering Query

For any query $q \in U$,
if $B[h_1(q)] = B[h_2(q)] = \cdots = B[h_k(q)] = 1$, report $q \in S$,
else report $q \notin S$.

Observation

If $q \in S$, the queries are answered correctly.

False Positives

Suppose $q \notin S$
If $B[h_1(q)] = B[h_2(q)] = \cdots = B[h_k(q)] = 1$,
we will report that $q \in S$.

Estimating Probability of False-Positives

Assume $n = |S|$.

- nk times, we attempt to set locations in B to "1".

- What is the probability that $B[l] = 1$?

- Complementary Event: $Pr(B[l] = 0) = (1 - \frac{1}{m})^{nk}$

- $p = Pr(B[l] = 1) = 1 - (1 - \frac{1}{m})^{nk}$

- For False-Positive to occur, all of the k specified locations $B[h_1(q)], \ldots, B[h_k(q)]$ must be "1".

Bloom70

$Pr(B[h_1(q)] = B[h_2(q)] = \cdots = B[h_k(q)] = 1) = p^k$.

Estimating Probability of False-Positives

Assume $n = |S|$.

- nk times, we attempt to set locations in B to "1".

- What is the probability that $B[l] = 1$?

- Complementary Event: $Pr(B[l] = 0) = (1 - \frac{1}{m})^{nk}$

- $p = Pr(B[l] = 1) = 1 - (1 - \frac{1}{m})^{nk}$

- For False-Positive to occur, all of the k specified locations $B[h_1(q)], \ldots, B[h_k(q)]$ must be "1".

Bloom70

$Pr(B[h_1(q)] = B[h_2(q)] = \cdots = B[h_k(q)] = 1) = p^k$.

Estimating Probability of False-Positives

Assume $n = |S|$.

- nk times, we attempt to set locations in B to "1".

- What is the probability that $B[l] = 1$?

- Complementary Event: $Pr(B[l] = 0) = (1 - \frac{1}{m})^{nk}$

- $p = Pr(B[l] = 1) = 1 - (1 - \frac{1}{m})^{nk}$

- For False-Positive to occur, all of the k specified locations $B[h_1(q)], \ldots, B[h_k(q)]$ must be "1".

Bloom70

$Pr(B[h_1(q)] = B[h_2(q)] = \cdots = B[h_k(q)] = 1) = p^k$.

Estimating Probability of False-Positives

Assume $n = |S|$.

- nk times, we attempt to set locations in B to "1".

- What is the probability that $B[l] = 1$?

- Complementary Event: $Pr(B[l] = 0) = (1 - \frac{1}{m})^{nk}$

- $p = Pr(B[l] = 1) = 1 - (1 - \frac{1}{m})^{nk}$

- For False-Positive to occur, all of the k specified locations $B[h_1(q)], \ldots, B[h_k(q)]$ must be "1".

Bloom70

$Pr(B[h_1(q)] = B[h_2(q)] = \cdots = B[h_k(q)] = 1) = p^k$.

An Example

Let $n = 1$, $m = 2$, $k = 2$,
$U = \{x, y\}$, $S = \{x\}$ and $q = y \neq x$.

After Initialization B has the following configuration:

B	Pr. of specific config. of B	Given B, Cond. Pr. that $B[h_1(y)] = B[h_2(y)] = 1$
1 0	$1/2 \times 1/2 = 1/4$	$1/2 \times 1/2 = 1/4$
0 1	$1/2 \times 1/2 = 1/4$	$1/2 \times 1/2 = 1/4$
1 1	$2 \times 1/2 \times 1/2 = 1/2$	$1 \times 1 = 1$

Since the three rows are mutually exclusive, the probability of False-Positive is
$1/4 \times 1/4 + 1/4 \times 1/4 + 1/2 \times 1 = 10/16$.

An Example Contd.

$n = 1$, $m = 2$, $k = 2$.

Note that Bloom's result states that the probability of false-positive is p^k, where $p = 1 - (1 - \frac{1}{m})^{kn}$.

From Bloom's computation,
$p = 1 - (1 - \frac{1}{m})^{kn} = 1 - (1 - \frac{1}{2})^{2\times 1} = 3/4$, and
$p^k = p^2 = 9/16$.

But,

$$9/16 \neq 10/16$$

An Example Contd.

$n = 1$, $m = 2$, $k = 2$.

Note that Bloom's result states that the probability of false-positive is p^k, where $p = 1 - (1 - \frac{1}{m})^{kn}$.

From Bloom's computation,
$p = 1 - (1 - \frac{1}{m})^{kn} = 1 - (1 - \frac{1}{2})^{2\times 1} = 3/4$, and
$p^k = p^2 = 9/16$.

But,

$$9/16 \neq 10/16$$

A Possible Fix

We came up with a fairly technical proof and showed that

Theorem

Let $p_{k,n,m}$ be the false-positive rate for a Bloom filter that stores n elements of a set S in a bit-vector of size m using k hash functions.

1. We can express $p_{k,n,m}$ in terms of the Stirling number of second kind as follows:

$$p_{k,n,m} = \frac{1}{m^{k(n+1)}} \sum_{i=1}^{m} i^k i! \binom{m}{i} \left\{ {kn \atop i} \right\}$$

2. Let $p = 1 - (1 - 1/m)^{kn}$, $k \geq 2$ and $\frac{k}{p}\sqrt{\frac{\ln m - 2k \ln p}{m}} \leq c$ for some $c < 1$. Upper and lower bounds on $p_{k,n,m}$ are given by

$$p^k < p_{k,n,m} \leq p^k \left(1 + O\left(\frac{k}{p}\sqrt{\frac{\ln m - 2k \ln p}{m}}\right)\right)$$

Summary of Bloom Filters

Summary

1. A simple scheme for testing membership.
   Has one-sided error, i.e., false positives.
2. How to find the right number of hash functions and right size of the filter?
3. Implemented in various search engines, routers, SPAM filters, ...
4. Unpleasant analysis.
5. Challenge: A nicer analysis.
   Hopefully, this will help with the analysis of variants of Bloom Filters.

An Interview Problem

Finding the Majority Element

Input: A stream consisting of n elements and it is given that it has a majority element.
Output: The majority element.

- Store the stream in an array A. Sort and pick the middle element (if elements can be ordered).
- Count frequency of each element.
- What if we can only use $O(1)$ space!

An Interview Problem

Finding the Majority Element

Input: A stream consisting of n elements and it is given that it has a majority element.
Output: The majority element.

- Store the stream in an array A. Sort and pick the middle element (if elements can be ordered).
- Count frequency of each element.
- What if we can only use $O(1)$ space!

An Interview Problem

Finding the Majority Element

Input: A stream consisting of n elements and it is given that it has a majority element.
Output: The majority element.

- Store the stream in an array A. Sort and pick the middle element (if elements can be ordered).
- Count frequency of each element.
- What if we can only use $O(1)$ space!

An Interview Problem

Finding the Majority Element

Input: A stream consisting of n elements and it is given that it has a majority element.
Output: The majority element.

- Store the stream in an array A. Sort and pick the middle element (if elements can be ordered).
- Count frequency of each element.
- What if we can only use $O(1)$ space!

Majority Algorithm

Input: Array A of size n consisting a majority element

Output: The majority element

```
c ← 0
for i = 1 to n do
    if c = 0 then
        current ← A[i]; c ← c + 1
    end
    else
        if A[i] = current then
            c ← c + 1
        end
        else
            c ← c − 1
        end
    end
end
return current
```

Analysis of Majority Algorithm

Observations

1. Algorithm maintains only two variables: c and current.
2. Correctness: Each non-majority element can 'kill' at most one majority element.

Generalize

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

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

Count-Min Sketch Data Structure

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

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] ← 0
    end
end
for i = 1 to n do
    for j = 1 to r do
        CMS[j, h_j(A[i])] ← CMS[j, h_j(A[i])] + 1
    end
end
return CMS[·, ·]
```

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

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 insertioning 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

CMS Table

$n = \#$items in the stream.
$f_x^* =$frequency of x in the stream.
Let $f_x = \min\{CMS[1, h_1(x)], \ldots, CMS[r, h_r(x)]\}$

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. Therefore, $f_x \geq f_x^*$ (i.e., f_x is an overestimate).

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: $Pr(I_y = 1) = 1/b$, $E[I_y] = 1/b$.
Observe,

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

Proof Contd.

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

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

- Let $X_j = |CMS[j, h_j(x)] - f_x^*|$
- Then, $E[X_j] \leq n/b = \epsilon n/2$.
- Recall Markov's inequality - Pr. that a r.v. deviates from its expectation by a factor of c is at most $1/c$.
- Thus, $Pr(X_j > 2(\epsilon n/2)) \leq 1/2$.
- This also holds for each value of $j = 1, \ldots, r$.
- Furthermore, X_j is independent of X_k as hash functions h_j and h_k are independent for any $k \neq j$.
- Therefore, for
$f_x = \min\{CMS[1, h_1(x)], \ldots, CMS[r, h_r(x)]\}$,
$Pr[|f_x - f_x^*| \geq \epsilon n] \leq \frac{1}{2^r}$ $\square$

Reporting Frequent Elements

Corollary

We have that $f_x^* \leq f_x \leq f_x^* + \epsilon n$ with probability at least $1 - 1/2^r$.

Suppose we are interested to report all elements in A that occur approx. n/k times for some integer k.
- Set $\epsilon = 1/3k$. Then, $b = 2/\epsilon = 6k$.
- Size of CMS table is $br = 6kr$.
- Scan A and update the CMS table as before.
- Also, maintain a set of $O(k)$ items that occur most frequently among all the elements in A scanned so far.
- The items are stored in a heap with their key as their f_x value.
- Assume we have scanned $i - 1$ items and have updated the CMS table and the heap.

Reporting Frequent Elements contd.

Consider the i-th item (say $x = A[i]$) and we perform the following:

1. Update the counts in the CMS table by executing $CMS[j, h_j(x)] \leftarrow CMS[j, h_j(x)] + 1$, for $j = 1$ to r.
2. Let $f_x = \min\{CMS[1, h_1(x)], \ldots, CMS[r, h_r(x)]\}$. If $f_x \geq i/k$, we perform the following heap operations:
   1. If $x \in$ heap, delete x and re-insert it again with the updated f_x value.
   2. If $x \notin$ heap, then we insert it in the heap, but 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 Sketch:
- Note: $f_x^* \leq f_x \leq f_x^* + \epsilon n = f_x^* + n/3k$.
- Thus, heap contains elements whose frequency is at least $n/k - n/3k = 0.667n/k$ with high probability.
- Size of heap = $O(k)$
- Total Time= $O(br + nr + n \log k) = O(kr + nr + n \log k)$, as $b = 6k$.
- Total Space= $O(br + k) = O(kr)$ $\square$

Conclusions on CMS

- Simple idea with important applications in

1. Compressed Sensing, Anomaly Detection, Finding Outliers, $\ldots$
2. Sketch Data Structure:
   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.

References

References

1. *Space/Time Trade-offs in Hash Coding with Allowable Errors*, B. H. Bloom, Communications of ACM, 13 (7): 422-426, 1970.
2. *An improved data stream summary: the count-min sketch and its applications*, G. Cormode and S. Muthukrishnan, J. Algorithms 55(1): 58-75, 2005.
3. *On the false-positive rate of Bloom filters*, Bose et al., Inf. Process. Letters 108(4): 210-213, 2008.
4. mmds.org - Mining Massive Data Sets by Leskovec, Rajaraman, and Ullman.

Thank-you