A Model for Learned Bloom Filters and Optimizing by Sandwiching

Michael Mitzenmacher, Harvard University

Main Questions

How do we analyze and optimize learned Bloom filters?
How do we compare them to standard Bloom filters?

Standard Bloom Filters

- Given a set $S = \{x_1, x_2, x_3, \ldots x_n\}$ on a universe U, want to answer *membership queries* of the form: Is $y \in S$?
- Data structure should be:
  - –Fast (Faster than searching through S).
  - –Small (Smaller than explicit representation).
- To obtain speed and size improvements, allow some probability of error.
  - –False positives: $y \notin S$ but we report $y \in S$
  - –False negatives: $y \in S$ but we report $y \notin S$

Start with an m bit array, filled with 0s.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hash each item x_j in S k times. If $H_i(x_j) = a$, set $B[a] = 1$.

0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

To check if y is in S, check B at $H_i(y)$. All k values must be 1.

0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

Possible to have a false positive; all k values are 1, but y is not in S.

0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	0

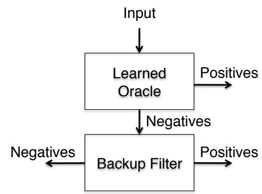
n items $m = cn$ bits k hash functions

$$\text{False positive probability} = (1 - e^{-kn/m})^k$$

Learned Bloom Filters

- Learned Bloom Filters (LBFs) defined/introduced by [Kraska-Beutel-Chi-Dean-Polyzotis, SIGMOD 2018]
- Learn the set: machine learning function (oracle) returns a probability estimate that an element is in the set
- Choose threshold to return positive response
- Need backup to deal with possible false negatives!

Learned Bloom Filter Diagram



Items that the oracle says are very likely positives (above threshold) are treated as positives. This yields some false positives.

Items that the oracle says are negatives might include false negatives. A backup (standard) Bloom filter catches false negatives from the oracle and returns them as positives. This might create additional false positives.

Learned Bloom Filter Analysis

m keys, bm backup filter bits
F_n = false negative fraction from oracle
F_p = false positive rate from oracle
Assume standard Bloom filter false positive probability falls as α^j for j bits per item
Learned Bloom filter false positive rate =

$$F_p + (1 - F_p)\alpha^{b/F_n}$$

Analysis requires standard assumptions; test set has same distribution as future queries.

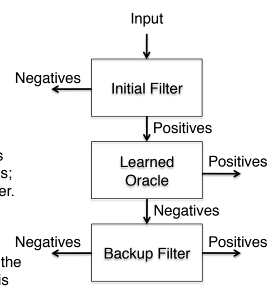
Can be used to derive size of the learned oracle necessary for improved performance.

Sandwiched Learned Bloom Filter Diagram

An initial Bloom filter for the set removes large numbers of potential false positives before reaching the learned.

Items that the oracle says are very likely positives (above threshold) are treated as positives. This yields some false positives; the initial Bloom filter reduces their number.

Items that the oracle says are negatives might include false negatives. A backup Bloom filter catches false negatives from the oracle and returns them as positives. This might create additional false positives.



Sandwiched Learned Bloom Filter Analysis

m keys, F_n, F_p, α as before
bm filter bits, split as $b_1 m$ initial, $b_2 m$ backup
Sandwich learned Bloom filter false positive rate =

$$\alpha^{b_1/F_n}(F_p + (1 - F_p)\alpha^{b_2/F_n})$$

Take derivatives to optimize:

$$b_2^* = F_n \log_\alpha \frac{F_p F_n}{(1 - F_p)(1 - F_n)}$$

- Optimal configuration has small, fixed-sized backup filter
- All remaining bits go to the initial filter
- Better to get rid of false positives early
- Can improve false positives by an order of magnitude

Extension: Learned Bloomier Filters

- ***Bloomier filters*** store a value for each key/element in a set
- Set elements should always return correct value
- Non-set elements should return null
- False positives: get non-null value for non-set element
- Assume an oracle can learn values sufficiently well
- Then can derive a learned Bloomier filter
- Minimize false positives, with no incorrect values on the set

Many Open Problems

- Further optimizations?
- Best implementation? Computational tradeoffs?
- What applications have sets amenable to learning?
- Other algorithms/data structures that can benefit from learning?

Acknowledgements: MM was supported by NSF grants CCF-1563710, CCF-1535795, CCF-1320231, CNS-1228598. Part of this work was done while visiting Microsoft Research.