[NOTE for Algorithms course of ECE@UTH.GR]

Bloom Filter's False Positives Rate

Dimitrios Katsaros
Electrical & Computer Engineering department, University of Thessaly, Volos, Greece

Abstract

This short note calculates the false positive rate of a Bloom Filter, in addition to the derivation of the optimal number of hash functions, and the appropriate size of the Bloom filter.

Introduction

The classic Bloom Filter (BF) was introduced as a structure to support approximate membership queries [1]. Since then, many variants of the BF have been proposed to support various tasks. In the field of networking, BFs has been employed to enable IP lookup, packet classification, regular expression matching, routing and forwarding in general, Web caching, network monitoring, security enhancement, content delivering, etc. In the area of databases, the BF is a proper option to support indexing, query and search, privacy preservation, key-value stores, content synchronization, duplicate detection, classification and so on. Beyond the general fields of networking and databases, BFs have been recently used to resolve biometric issues and even navigation tasks in the mobile computing scenarios. A not so recent survey can be found in [4]. Apart from the classic membership queries newer variants [3] of the BF support association and multiplicity queries [5], multiset membership queries [2], range searches and so on.

Bloom Filter's optimal number of hash functions, when n and m are given

Let us assume that the number of elements to be inserted into a Bloom filter is n, and the size in bits of the BF is m bits. We are seeking the optimal number k of hash functions that minimizes the false positive rate (i.e., false drop probability) of the BF.

We make the following assumptions:

1. Each hash function sets (activates) only one bit in the BF.
2. Each hash function is "independent" of every other hash functions, i.e., they are not correlated in how the set bits.
3. Each hash function activates bits uniformly at random throughout the m bits of the BF.

When inserting the first element into the BF, the application of the first hash function sets one bit. The probability of setting anyone of the m bit is equal – due to the uniformity assumption – to:

$$P_{1 \text{ hash}}[bit = 1] = \frac{1}{m}. \tag{1}$$

Thus, the probability of the bit not being set is equal to:

$$P_{1 \text{ hash}}[bit = 0] = 1 - \frac{1}{m}. \tag{2}$$

The repeated application of k hash functions, turns the previous probability – due to the independence assumption – into:

$$P_{\text{k hashes}}[bit = 0] = \left(1 - \frac{1}{m}\right)^{k}. \tag{3}$$

Therefore, after the insertion of n elements, the above probability becomes equal to:

$$P_{\text{n insertions}}[bit = 0] = \left(1 - \frac{1}{m}\right)^{kn}. \tag{4}$$

So, the probability of a bit being set after n insertions is equal to:

$$P_{\text{n insertions}}[bit = 1] = 1 - \left(1 - \frac{1}{m}\right)^{kn}. \tag{5}$$

Recall that in order to check whether an element is part of the set of elements being represented by the BF, we hash the element k times using the k hash functions. What is the probabiilty that all these k hashes encounter '1' bits? This is exactly the *false positive rate* or *error probability* of the Bloom Filter, and it can be calculated – due to the uncorrelation assumption of the hash functions – as follows:

$$f_e^{BF}(k, m, n) = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k} \approx \left(1 - e^{\frac{-kn}{m}}\right)^{k}. \tag{6}$$

In order to find the optimal value of k, we take the partial derivative equal to 0, i.e.,

$$\frac{\partial f_e^{BF}(k, m, n)}{\partial k} = 0. \tag{7}$$

The approximation of the function $f_e^{BF}(k, m, n)$ can be written equivalently as:

$$f_e^{BF}(k, m, n) \approx \left(1 - e^{\frac{-kn}{m}}\right)^{k} = e^{ln\left(1 - e^{\frac{-kn}{m}}\right)^{k}} = e^{k * ln\left(1 - e^{\frac{-kn}{m}}\right)}. \tag{8}$$

Instead of working with Equation 7, it easier to work with (minimize) the following function:

$$\frac{\partial g_e^{BF}(k, m, n)}{\partial k} = \frac{\partial\left\{k * ln\left(1 - e^{\frac{-kn}{m}}\right)\right\}}{\partial k} = 0. \tag{9}$$

So,

$$\frac{\partial\left\{k * ln\left(1 - e^{\frac{-kn}{m}}\right)\right\}}{\partial k} = 0 \implies ln\left(1 - e^{\frac{-kn}{m}}\right) + k * \frac{\frac{n}{m} * e^{\frac{-kn}{m}}}{1 - e^{\frac{-kn}{m}}} = 0. \tag{10}$$

Let

$$p = e^{\frac{-kn}{m}} \tag{11}$$

then,

$$ln(p) = \frac{-kn}{m} \tag{12}$$

Then, Equation 10 becomes:

$$ln(1-p) - ln(p) * \frac{p}{1-p} = 0 \implies (1-p) * ln(1-p) = p * ln(p) \implies$$
$$(1-p)^{1-p} = p^p \implies 1-p = p \implies p = \frac{1}{2}. \quad (13)$$

Then, from Equation 12 we get

$$ln(\frac{1}{2}) = \frac{-k^\star n}{m} \implies \frac{k^\star n}{m} = ln(2) \implies k^\star = ln(2)\frac{m}{n}. \quad (14)$$

Having found[1] the optimal value $k^\star$, we get back to Equation 6 and substituting for $k^\star$, we can calculate the false positive rate of a BF when the optimal number of hash functions is used, which is equal to:

$$f_e^{BF}(k^\star, m, n) = \left(1 - \frac{1}{2}\right)^{\frac{m*ln(2)}{n}} = 0.6185^{\frac{m}{n}}. \quad (15)$$

Bloom Filter's optimal size, when n and k are given

Suppose now that we wish to decide the size (number of bits) of a Bloom Filter in order to achieve a given false positive rate. So, we are given n, we can compute $k^\star$, and we also can set $f_e^{BF}(k^\star, m, n)$ to a desired value f_e. Then, what is the optimal size $m^\star$ of the BF?

From Equation 6 we have:

$$f_e = \left(1 - e^{\frac{-k^\star n}{m^\star}}\right)^{k^\star} \implies f_e = \left(1 - e^{\frac{-(ln(2)\frac{m^\star}{n})n}{m^\star}}\right)^{ln(2)\frac{m^\star}{n}}$$
$$\implies f_e = \left(1 - e^{ln\frac{1}{2}}\right)^{\frac{m^\star}{n}*ln(2)} \implies f_e = \left(1 - \frac{1}{2}\right)^{\frac{m^\star}{n}*ln(2)}$$
$$\implies ln(f_e) = ln(\frac{1}{2}) * \frac{m^\star}{n} * ln(2) \implies ln(f_e) = -\Big(ln(2)\Big)^2 * \frac{m^\star}{n}. \quad (16)$$

Therefore,

$$m^\star = -n * \frac{ln(f_e)}{(ln(2))^2}. \quad (17)$$

In other words, the number of bits per element $\frac{m^\star}{n}$ of a Bloom filter is equal to $\frac{-ln(f_e)}{(ln(2))^2}$.

Summary

In this short note we developed an analysis of the optimal number of hash functions and optimal size of a Bloom filter that minimize the false positive rate. Our analysis were based on some plausible assumptions; more sophisticated analyses exist in the literature that refrain from making such assumptions.

[1] It might not be an integer. Therefore, we have to round it to the closest integer value.

References

[1] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[2] Yang; D., D. Tian, J. Gong, S. Gao, T. Yang, and X. Li. Difference bloom filter: a probabilistic structure for multi-set membership query. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 1938–1883, 2017.

[3] L. Luo, D. Guo, R. T. B. Ma, O. Rottenstreich, and X. Luo. Optimizing Bloom filter: Challenges, solutions, and comparisons. *IEEE Communications Surveys & Tutorials*, 21(2):1912–1949, 2019.

[4] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz. Theory and practice of Bloom Filters for distributed systems. *IEEE Communications Surveys & Tutorials*, 14(1):131–155, 2011.

[5] T. Yang, A. X. Liu, M. Shahzad, D. Yang, Q. Fu, G. Xie, and X. Li. A shifting framework for set queries. *IEEE/ACM Transactions on Networking*, 25(5):3116–3131, 2017.