Downtown Bodega Filter Tuning Guide

Calvin Higgins

February 2025

1 Background

Downtown Bodega filters (DBFs) [1] are a secure alternative to Learned Bloom filters (LBFs) [2]. Both data structures require tuning several parameters including a decision threshold τ and backup filter false positive rate(s) to attain the desired false positive rate ϵ. Parameter tuning also impacts the filter's memory footprint and, for DBFs, worst-case false positive rate. Tuning procedures for LBFs do not directly translate to DBFs, as they do not balance the tradeoff between expected and worst-case false positive rates. To be practical, DBFs must have a lower expected false positive rate than Naor-Eylon filters [3], an alternative secure Bloom filter variant, and reasonable worst-case false positive rate.

2 Problem Statement

Given a desired expected false positive rate ϵ and maximum false positive rate $\epsilon_{\max}$, we wish to choose the decision threshold τ, true positive filter false positive rate $\mathrm{TP}_{\mathrm{FPR}}$, and false negative filter false positive rate $\mathrm{FN}_{\mathrm{FPR}}$ such that the resulting Downtown Bodega filter (DBF) has minimal size in bits m.

3 Results

We give a log-linear time vectorizable algorithm to compute the optimal decision threshold τ. First, we compute the optimal backup filter false positive rates, and then optimize the choice of τ with dynamic programming.

3.1 Backup Filter False Positive Rates

We consider the problem of choosing optimal backup filter false positive rates $\mathrm{TP}_{\mathrm{FPR}}$ and $\mathrm{FN}_{\mathrm{FPR}}$, given a learned model with false positive rate $\mathrm{M}_{\mathrm{FPR}}$.

From Theorem 8 in [1], we have that

$$\epsilon = \mathrm{M}_{\mathrm{FPR}}\mathrm{TP}_{\mathrm{FPR}} + \mathrm{M}_{\mathrm{TNR}}\mathrm{FN}_{\mathrm{FPR}} \tag{1}$$

and the total size of the DBF is given by

$$m = -\frac{n}{\ln(2)^2}\left(\mathrm{M_{TPR}}\ln(\mathrm{TP_{FPR}}) + \mathrm{M_{FNR}}\ln(\mathrm{FN_{FPR}})\right) + \mathrm{M}_m \tag{2}$$

where n is the size of the underlying set and M_m is the memory consumption of the learned model in bits. Manipulating (1), we find that

$$\mathrm{FN_{FPR}} = \frac{\epsilon - \mathrm{M_{FPR}}\mathrm{TP_{FPR}}}{\mathrm{M_{TNR}}} \tag{3}$$

so

$$m = -\frac{n}{\ln(2)^2}\left(\mathrm{M_{TPR}}\ln(\mathrm{TP_{FPR}}) + \mathrm{M_{FNR}}\ln\left(\frac{\epsilon - \mathrm{M_{FPR}}\mathrm{TP_{FPR}}}{\mathrm{M_{TNR}}}\right)\right) + \mathrm{M}_m \tag{4}$$

by plugging into (2).

We wish to minimize m by choosing $\mathrm{TP_{FPR}}$, so we differentiate m with respect to $\mathrm{TP_{FPR}}$.

$$\frac{\partial m}{\partial \mathrm{TP_{FPR}}} = -\frac{n}{\ln(2)^2}\left(\frac{\partial}{\partial \mathrm{TP_{FPR}}}\mathrm{M_{TPR}}\ln(\mathrm{TP_{FPR}}) + \frac{\partial}{\partial \mathrm{TP_{FPR}}}\mathrm{M_{FNR}}\ln\left(\frac{\epsilon - \mathrm{M_{FPR}}\mathrm{TP_{FPR}}}{\mathrm{M_{TNR}}}\right)\right) \tag{5}$$

$$= -\frac{n}{\ln(2)^2}\left(\frac{\mathrm{M_{TPR}}}{\mathrm{TP_{FPR}}} + \frac{\mathrm{M_{FNR}}}{\frac{\epsilon - \mathrm{M_{FPR}}\mathrm{TP_{FPR}}}{\mathrm{M_{TNR}}}} \cdot \frac{\mathrm{M_{FPR}}}{\mathrm{M_{TNR}}}\right) \tag{6}$$

$$= -\frac{n}{\ln(2)^2}\left(\frac{\mathrm{M_{TPR}}}{\mathrm{TP_{FPR}}} + \frac{\mathrm{M_{FNR}}\mathrm{M_{FPR}}}{\epsilon - \mathrm{M_{FPR}}\mathrm{TP_{FPR}}}\right) \tag{7}$$

$$\tag{8}$$

Then we set to zero and solve

$$-\frac{n}{\ln(2)^2}\left(\frac{\mathrm{M_{TPR}}}{\mathrm{TP_{FPR}}} + \frac{\mathrm{M_{FNR}}\mathrm{M_{FPR}}}{\epsilon - \mathrm{M_{FPR}}\mathrm{TP_{FPR}}}\right) = 0 \tag{9}$$

$$-\mathrm{M_{TPR}}\left(\epsilon - \mathrm{M_{FPR}}\mathrm{TP_{FPR}}\right) - \mathrm{TP_{FPR}}\mathrm{M_{FNR}}\mathrm{M_{FPR}} = 0 \tag{10}$$

$$(\mathrm{M_{TPR}} - \mathrm{M_{FNR}})\mathrm{M_{FPR}}\mathrm{TP_{FPR}} = \epsilon\mathrm{M_{TPR}} \tag{11}$$

$$\mathrm{M_{FPR}}\mathrm{TP_{FPR}} = \epsilon\mathrm{M_{TPR}} \tag{12}$$

$$\mathrm{TP_{FPR}} = \epsilon\frac{\mathrm{M_{TPR}}}{\mathrm{M_{FPR}}} \tag{13}$$

Substituting into (3), we find that

$$\mathrm{FN_{FPR}} = \epsilon\frac{\mathrm{M_{FNR}}}{\mathrm{M_{TNR}}} \tag{14}$$

3.2 Learned Model False Positive Rate

In Section 3.1, we found the optimal backup filter false positive rates given a fixed decision threshold τ. Now we consider the problem of choosing the optimal decision threshold.

Plugging (13) and (14) into (2), we find that

$$m = -\frac{n}{\ln(2)^2}\left(\mathrm{M}_{\mathrm{TPR}} \ln\left(\epsilon \frac{\mathrm{M}_{\mathrm{TPR}}}{\mathrm{M}_{\mathrm{FPR}}}\right) + \mathrm{M}_{\mathrm{FNR}} \ln\left(\epsilon \frac{\mathrm{M}_{\mathrm{FNR}}}{\mathrm{M}_{\mathrm{TNR}}}\right)\right) + \mathrm{M}_m \tag{15}$$

$$= -\frac{n}{\ln(2)^2}\left(\mathrm{M}_{\mathrm{TPR}} \ln\left(\frac{\mathrm{M}_{\mathrm{TPR}}}{\mathrm{M}_{\mathrm{FPR}}}\right) + \mathrm{M}_{\mathrm{TPR}} \ln(\epsilon) + \mathrm{M}_{\mathrm{FNR}} \ln\left(\frac{\mathrm{M}_{\mathrm{FNR}}}{\mathrm{M}_{\mathrm{TNR}}}\right) + \mathrm{M}_{\mathrm{FNR}} \ln(\epsilon)\right) + \mathrm{M}_m \tag{16}$$

$$= -\frac{n}{\ln(2)^2}\left(\mathrm{M}_{\mathrm{TPR}} \ln\left(\frac{\mathrm{M}_{\mathrm{TPR}}}{\mathrm{M}_{\mathrm{FPR}}}\right) + \mathrm{M}_{\mathrm{FNR}} \ln\left(\frac{\mathrm{M}_{\mathrm{FNR}}}{\mathrm{M}_{\mathrm{TNR}}}\right) + \ln(\epsilon)\right) + \mathrm{M}_m \tag{17}$$

Dropping constants and constant factors, we get the following cost function

$$C(\tau) = -\mathrm{M}_{\mathrm{TPR}} \ln\left(\frac{\mathrm{M}_{\mathrm{TPR}}}{\mathrm{M}_{\mathrm{FPR}}}\right) - \mathrm{M}_{\mathrm{FNR}} \ln\left(\frac{\mathrm{M}_{\mathrm{FNR}}}{\mathrm{M}_{\mathrm{TNR}}}\right)$$

which we can optimize with dynamic programming in $\Theta(n \lg n)$ time. We can restrict the choice of τ such that

$$\max\left\{\epsilon \frac{\mathrm{M}_{\mathrm{TPR}}}{\mathrm{M}_{\mathrm{FPR}}}, \epsilon \frac{\mathrm{M}_{\mathrm{FNR}}}{\mathrm{M}_{\mathrm{TNR}}}\right\} \leq \epsilon_{\max} \tag{18}$$

and the resulting DBF will have worst-case false positive rate $\epsilon_{\max}$.

Here is one sample procedure to optimize the cost function.

1. Sort the learned model confidence scores A on the training set from least to greatest.
2. For each index i,
   (a) Compute the number of elements with negative class in $A[: i]$ and with positive class in $A[i :]$ with a rolling sum.
   (b) Compute the false/true negative/positive rates of the model given that $A[: i]$ is classified as negative and $A[i :]$ is classified as positive.
   (c) Verify that the optimal backup filter false positive rates are less than $\epsilon_{\max}$.
   (d) Track the index with the lowest cost i^*.
3. Output the average between $A[i^* - 1]$ and $A[i^*]$.

4 Future Work

Rather than outright rejecting decision thresholds τ, where the worst-case false positive rate is greater than $\epsilon_{\max}$, it might be possible to compute optimal backup filter false positive rates that obey this constraint. It might also be possible to derive optimal parameter values given a fixed memory budget and maximum false positive rate.

References

[1] Allison Bishop and Hayder Tirmazi. "Adversary Resilient Learned Bloom Filters". In: (2025). arXiv: 2409.06556.

[2] Tim Kraska et al. "The Case for Learned Index Structures". In: (2018).

[3] Moni Naor and Yogev Eylon. "Bloom Filters in Adversarial Environments". In: (2019). DOI: 10.1145/3306193.