Truly Adaptive Bloom Filters with Monotone Streaming Updates

Devan Shah
ds6237@princeton.edu

David Yan
yan.david@princeton.edu

COS 598D

Abstract

The bloom filter is an efficient and probabilistic insertion-only data structure for testing set inclusion that offers improved run time and lower memory at the cost of occasional false positive queries. As false positive rate directly correlates with the number of insertions into the filter, learned models [1] have reduced insertions by training a classifier to identify whether *x* in the set, with the bloom filter acting as simply an overflow buffer for keys the model incorrectly classifies. Yet the classical method of training filters [1,2] cannot handle distribution shifts, offers "over-eager" estimations of the true key space, and requires distribution knowledge that may be simply unavailable. We leverage key embeddings to produce an unsupervised streaming algorithm with suitable guarantees and consistently high performance despite distribution shifts.

Introduction

- Bloom Filter:
  - Init(M, k): m = [0] * M, h = [new_hash_function() for _ in k]
  - Insert(key): m[h[i](key)] = 1 for i in range(k)
  - Query(key): return all(m[h[i](key)] == 1 for i in range(k))
- Learned Bloom Filter:
  - Init(M, k, f, t): bf = BloomFilter(M, k)
  - Insert(key): if (not f(key) > t) bf.insert(key)
  - Query(key): return f(key) > t or bf.query(key)

LBF Limitations:

Training Data Since Bloom Filters do not store keys, training f on the key-distribution is practically difficult.	**Distribution Shift** If we adapt f to f*, we require f(x) > t => f*(x) > t, which can typically not be ensured.
Over-Eager How can we model complex key distributions with static and constant-size space-partitioning models?	**Unstandardized** Requires data scientist to provide model, tune parameters and maintain.

Approach

We provide a function class and streaming update mechanism enabling unsupervised learning of the key space
1. Map to semantic embeddings to detect general patterns in filter inserts and JL transform to reduce dim.
2. Determine the likelihood the key belongs to an existing cluster or cluster set, representing a pattern.
3. Expand the cluster to optimize density or consider adding an additional cluster to combat shifts.

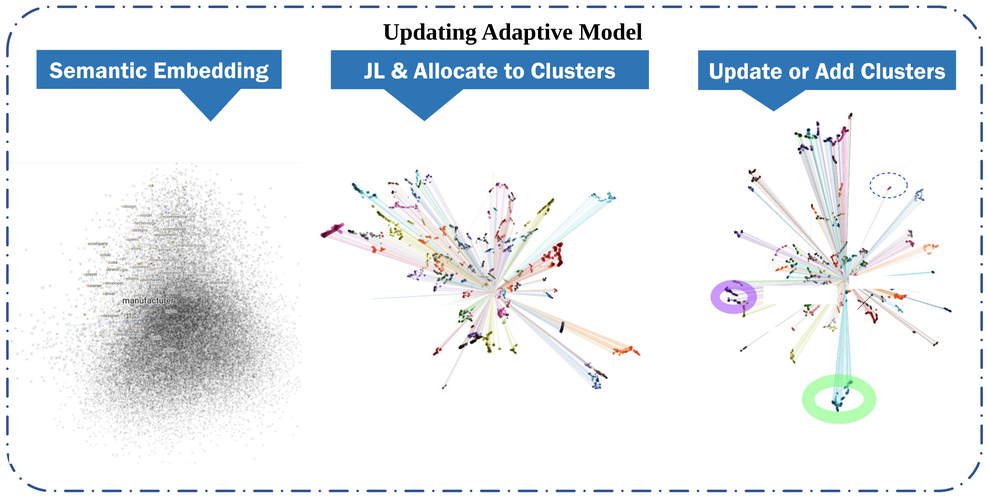
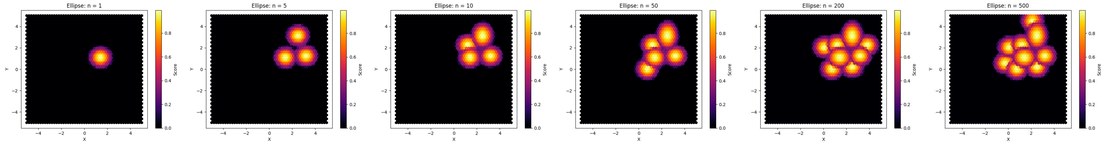


Figure 1: Algorithm Overview

We provide an algorithm that carefully learns a continuous function E(x) (or E(x) – p(x) to account for priors) via joint energy-based modeling. Inspired by Gaussian Splatting [3] and Gaussian Mixture Models, we learn monotonically by adding and enlarging Gaussian fields, providing monotonic function updates to maintain 0% FNR. For each Gaussian/cluster, we perform an exponential search to optimize covariance without pre-existing distribution information and maintain O(log(n)) clusters, adapting quickly to O(n) distribution shifts.

$$\mathcal{L}(f) = -\frac{1}{|S|}\sum_{x\in S}\mathbb{1}[f(x) > \tau] + \alpha\,\mathbb{E}_{x\sim \mathrm{Unif}(\mathcal{D})}[\mathbb{1}[f(x) > \tau]]$$

Figure 2: The formation and reinforcement of a new pattern across n = 1 to 500 inserts



Eagerness

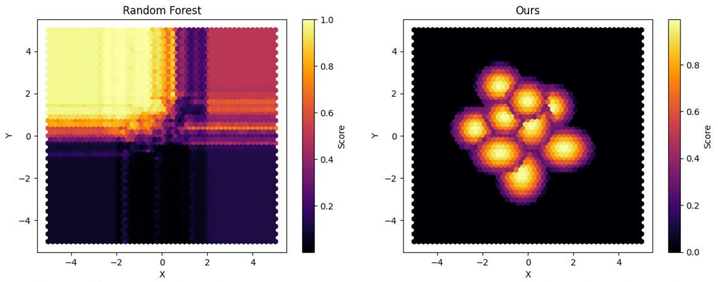


Figure 3: Many algorithms vastly over-represent the key-space based on limited training and challenging evals., leading to higher risk during distribution shifts.

Evaluations

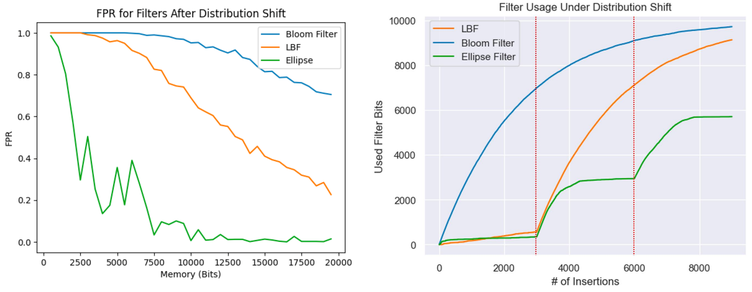


Figure 4a: FPR varying with total bloom filter memory for different bloom filter models under a workload involving insertion of a shifted key distribution with k = 4

Figure 4b: Under a simulated workload with mixture of Gaussian key patterns, modeling the heavy-insertion of a few topics, our filter handles dist. shifts and even pre-shift matches the performance of other techniques that leverage dist. knowledge.

References

[1] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. The Case for Learned Index Structures. In CoRR, volume abs/1712.01208, 2017. URL http://arxiv.org/abs/1712.01208.

[2] Zhenwei Dai and Anshumali Shrivastava. Adaptive Learned Bloom Filter (Ada-BF): Efficient Utilization of the Classifier. In CoRR, volume abs/1910.09131, 2019. URL: http://arxiv.org/abs/1910.09131.

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. In ACM Transactions on Graphics, volume 42, number 4, July 2023. URL: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting.