Tight bounds for membership filters with two-sided errors

October 21, 2025

Abstract

We study two-sided membership filters, a natural generalization of classical Bloom-like filters that allows both false positive and false negative errors. Two-sided filters are motivated by a number of applications, and they can potentially use less space by tolerating both types of errors.

We establish the fundamental space lower bound for two-sided filters: any filter achieving false positive rate ε_P and false negative rate ε_N must use at least $D_{KL}(1-\varepsilon_N \| \varepsilon_P)$ bits per key, the binary Kullback-Leibler divergence between $\text{Bern}(1-\varepsilon_N)$ and $\text{Bern}(\varepsilon_P)$. This bound generalizes the classical $\log(1/\varepsilon_P)$ lower bound for one-sided filters and provides a natural and complete characterization of the space-error trade-off.

We also present a simple but inefficient construction that achieves this lower bound, demonstrating its tightness.

1 Introduction

Approximate membership query filters (or simply *filters*), such as Bloom filters [6], have achieved tremendous success in many applications including networking [9], databases, distributed systems [31], biology [24], and others [23]. These data structures can "store" a set of *keys* S in some universe U, using space linear in $|S|$, regardless of the size of U. Information-theoretically, storing such a set of size n takes $\log \binom{|U|}{n}$[1] bits, which is prohibitively large as filters are typically used when $|U| \gg n$. Hence this compactness, while impressive, must come with some inaccuracies. Typically, this inaccuracy is characterized by a small *false positive rate* (FPR) ε of the user's choice, such that an ε fraction of the *non-keys* $U \setminus S$ will be recognized as keys by the filter. Meanwhile, the false negative rate (FNR) is usually guaranteed at zero.

Since Bloom's proposal of the first filter in 1970, various works in data structure design have been devoted to designing filters that are space-optimal, computationally cheap, and feature-rich (allowing for deletion, for instance). Many filter designs since then, including variants of the quotient [3, 2, 28] and cuckoo [16, 14, 8, 32] filters, as well as other structures for storing (multi)set of *fingerprints* of the keys [26, 5, 15, 4], can achieve space usage $O(n)$-close to the $n \log \frac{1}{\varepsilon}$ bits lower bound, shown by Carter et al. [10]. Some other designs can even achieve this lower bound up to $o(n)$ extra bits [12, 29] by considering the closely related *retrieval* data structures (defined in section 2.1).

It seems clear then that $\log \frac{1}{\varepsilon}$ characterizes the fundamental trade-off between per-element space and error for filters. However, as the term "approximate membership" suggests, these filters with one-sided errors are special cases of a more general class of data structures: those which store S with *both false negatives and false positives*, which can potentially use even less space than the classical filters. Despite being a natural generalization motivated by a number of practical applications, there has been no theoretical characterization of the trade-off between space, FPR, and FNR for two-sided filters [2]. This leads to the following natural question:

What is the minimum space required for a two-sided filter that stores a set of size n with FPR of ε_P and FNR of ε_N, independent of the universe size?

1.1 Our results

Our first result is a space lower bound for data structures storing n keys in a universe of size u, up to FPR of ε_P and FNR of ε_N, for the case where key density n/u is fixed. This lower bound applies not only to filters (which operate regardless of the universe size), but also for other approximate membership structures tailor made for known universes:

Theorem 1. *For any fixed $p = \frac{n}{u}$, as $n \to \infty$, any data structure storing key set of size n with FPR of $\varepsilon_P \in (0, 1)$ and FNR of $\varepsilon_N \in (0, 1 - \varepsilon_P)$ must use at least*

$$D_{KL}(1 - \varepsilon_N \| \varepsilon_P) - \frac{(1 - \varepsilon_N - \varepsilon_P)^2}{2\varepsilon_P(1 - \varepsilon_P)\ln 2} \cdot p + O(p^2)$$

bits of space per key. Here, $D_{KL}(q_1 \| q_2)$ denotes the binary Kullback-Leibler divergence of a $\mathrm{Bern}(q_1)$ *distribution from a* $\mathrm{Bern}(q_2)$ *distribution:*

$$D_{KL}(q_1 \| q_2) = q_1 \log \frac{q_1}{q_2} + (1 - q_1) \log \frac{1 - q_1}{1 - q_2}.$$

For two-sided filters which are *universe-independent*, by considering the usual setting where $u \gg n$, we immediately obtain the following corollary:

[1] All logarithms are base-2 in this paper.

[2] From now on, we use *two-sided* filters to denote such structures with two-sided errors, and the classical filters will be referred to as *one-sided* filters.

Corollary 2. *Any two-sided filter storing n keys with FPR of $\varepsilon_P \in (0,1)$ and FNR of $\varepsilon_N \in (0, 1-\varepsilon_P)$ must use at least $D_{KL}(1-\varepsilon_N \| \varepsilon_P)$ bits of space per key.*

This lower bound generalizes the one-sided lower bound $\log \frac{1}{\varepsilon_P}$, as $D_{KL}(1\|\varepsilon_P) = \log \frac{1}{\varepsilon_P}$ for any $\varepsilon_P \in (0,1)$. Moreover, this value is intuitive from a statistical perspective: by drawing n samples from a large population with positive rate ε_P, the probability of obtaining positive rate at least $1-\varepsilon_N$ on the samples is $2^{-n(D_{KL}(1-\varepsilon_N\|\varepsilon_P)+o(1))}$. The "amount of information" needed to describe such a sample is therefore around $n \cdot D_{KL}(1-\varepsilon_N\|\varepsilon_P)$ bits.

The proof for Theorem 1 is based on lossy data compression, formalizing the above intuition and extending the idea of [19]. Some technicalities arise from the fact that the membership information (whether $x \in S$ or not) for each $x \in U$ is dependent on the membership information of the rest of U. To obtain a rigorous lower bound, we apply the law of large numbers and prove a generalized version of the rate-distortion theorem for two error metrics.

It's worth noting that our result also refines an early result [27] for "lossy dictionaries", a combination of two-sided filter and retrieval structure. In particular, for two-sided filters, the authors gave a lower bound of

$$\begin{aligned}&(1-\varepsilon_N)\log\frac{1}{\varepsilon_P+p}-\Theta(1)\\ =&(1-\varepsilon_N)\log\frac{1}{\varepsilon_P}-\Theta(1)-\frac{1-\varepsilon_N}{\varepsilon_P\ln 2}\cdot p+O\left(p^2\right)\end{aligned}$$

bits per element. Theorem 1 characterizes the $\Theta(n)$ term and refines the dependency on p. Their result was based on counting arguments similar to [10], which could explain the discrepancy.

Finally, to complement the lower bound, we construct a conceptually simple two-sided filter that achieves the space lower bound, thereby demonstrating its tightness:

Theorem 3. *There exists a two-sided filter with space usage $n \cdot D_{KL}(1-\varepsilon_N\|\varepsilon_P) + o(n)$ bits per key, for any $\varepsilon_P \in (0,1)$ and $\varepsilon_N \in (0, 1-\varepsilon_P)$.*

We note that this construction is not practical. In fact, it is unlikely that we can construct this filter in polynomial time due to a reduction to the learning parity with noise (LPN) problem [7]. It is open whether we can construct such a filter in polynomial time, or if there is some inherent hardness in the problem.

1.2 Why introduce false negatives?

Despite their success, one-sided filters sometimes have limited performance and versatility due to the stringent no-false-negative requirement. As pointed out by Hurley and Waldvogel [19], certain filter applications in networking [9] would tolerate or even *prefer* false negatives. An example is *set difference reconciliation*, where Bob sends a filter of his local storage S_B to Alice, such that Alice can compute $S_A \setminus S_B$ and update Bob on what he does not have. Here, a false negative in Bob's filter only causes a minor overhead, yet a false positive will lead to synchronization errors. For more examples, see table 1.

Moreover, even in traditionally FPR-heavy tasks, the idea of introducing FNR to further reduce FPR is often employed, as exemplified by designs such as the retouched Bloom filter [13], generalized Bloom filter [21], and autoscaling Bloom filter [20]. In *the Bloom paradox* [30], the authors also discussed *selective insertion* and *selective query* as a means to reduce FPR and introduce FNR, in order to minimize the overall cost. All these approaches can be viewed as heuristics for the FPR-FNR trade-off for a given space usage, and it is therefore of great interest to study the fundamental limit of this approach, as well as methods that can achieve this limit.

The connections to retrieval and lossy dictionary structures further motivate our investigation into two-sided filters, as these related fields have demonstrated the value of allowing controlled information loss for improved space efficiency and performance. Understanding the fundamental

Application area	Necessity of one-sided error
Distributed caching	No
Object location in P2P systems	No
Approximate set reconciliation	False negative preferred
Resource routing	No
Loop detection	False negative preferred
Flow detection	Yes
Multicast	Yes
Hyphenation Exceptions	Yes
Set intersection	Yes
Differential files	Yes

Table 1: A summary of network applications of filters with the role of false negatives highlighted. This table is taken from Hurley and Waldvogel [19].

limits of two-sided filters thus contributes not only to filter design but also to the broader understanding of space-efficient data structures with approximate semantics.

We discuss other related works in section 2.4.

2 Preliminaries

2.1 Approximate Membership, Filters, and Retrieval

In this paper, we use the generic term "approximate membership structure" to denote any data structure that stores a set of keys with both false positive and false negative which operates for a fixed, known universe U. One can think of U as the set $[2^w]$, where w is the word length in a RAM model, a standard assumption in the literature [26]. This structure behaves significantly different from the *filters* we define below. Our main result is in the context of two-sided filters defined later, but our methods in theorem 1 come from the study of such structures [19].

Definition 4 (Approximate Membership Structure)**.** A ***approximate membership structure*** $\mathcal{M}$ is a probabilistic data structure that stores a set of keys with both false positive and false negative errors. $\mathcal{M}$ implements the following two operations:

1. On input ε_P, ε_N, key set S, and universe U, $\mathcal{M}$ is initialized by storing an internal state of some $L_{\mathcal{M}}(|S|, |U|, \varepsilon_P, \varepsilon_N)$ bits.

2. On query $x \in U$, (by an abuse of notation) $\mathcal{M}$ outputs some $\mathcal{M}(x) \in \{0, 1\}$. The query answer should satisfy:
$$\begin{cases} \mathbb{P}_{x \sim \text{Unif}(U \setminus S)}[\mathcal{M}(x) = 1] & \leq \varepsilon_P, \\ \mathbb{P}_{x \sim \text{Unif}(S)}[\mathcal{M}(x) = 0] & \leq \varepsilon_N, \end{cases}$$
where the probabilities are taken over the random oracle functions used in both initialization and query.

Now we define filters. To capture the notion that filters operate regardless of the universe size, we require the space usage to be independent of the universe size, assuming access to random oracles functions on the universe U. This is an idealized version of hash functions, and most filter structures (including Bloom, Cuckoo, Quotient, etc.) can indeed be implemented without access to U under this assumption. This also allows us to separate the information-theoretic core of the problem from practical hash function designs.

Definition 5 (Two-Sided Filter)**.** A ***two-sided filter*** $\mathcal{F}$ is a probabilistic data structure with access to binary random oracle functions $O_1, O_2, \ldots : U \to \{0, 1\}$. $\mathcal{F}$ implements the following two operations using these random oracle functions:

1. On input ε_P, ε_N, and key set S, $\mathcal{F}$ is initialized by storing an internal state of some $L_{\mathcal{F}}(|S|, \varepsilon_P, \varepsilon_N)$ bits.

2. On query $x \in U$, (by an abuse of notation) $\mathcal{F}$ outputs some $\mathcal{F}(x) \in \{0, 1\}$. The query answer should satisfy:
$$\begin{cases} \mathbb{P}_{x \sim \text{Unif}(U \setminus S)}[\mathcal{F}(x) = 1] & \leq \varepsilon_P, \\ \mathbb{P}_{x \sim \text{Unif}(S)}[\mathcal{F}(x) = 0] & \leq \varepsilon_N, \end{cases}$$
where the probabilities are taken over the random oracle functions used in both initialization and query.

We call $\mathcal{F}$ ***one-sided*** if it has $\varepsilon_N = 0$. We sometimes use "error rates $(\varepsilon_P, \varepsilon_N)$" to denote an FPR of ε_P and FNR of ε_N.

For completeness, we also define the closely related retrieval data structure [12, 29]:

Definition 6 (Retrieval)**.** A ***retrieval*** data structure $\mathcal{R}$ is a compact dictionary-like structure that associates a k-bit value $v_1, \ldots, v_n$ with each of the n keys in $\{x_1, \ldots, x_n\} = S \subseteq U$. Upon a query $x \in U$,

- If $x = x_i \in S$ is a key, then $\mathcal{R}$ must return the correct value v_i.
- If $x \in U \setminus S$ is a non-key, then $\mathcal{R}$ can return an arbitrary value in $\{0, 1\}^k$.

This data structure has space lower bound nk, since it can easily implement a one-sided filter with FPR of 2^{-k} via the following procedure [12][29]:

1. Set the value of key $x_i \in S$ to be $v_i = f(x_i)$, for random oracle "fingerprint" function $f : U \to \{0, 1\}^k$.
2. Construct retrieval structure $\mathcal{R}$ which stores value v_i for each key x_i.
3. When queried a key $x \in U$, the filter queries $\mathcal{R}$ with the same x and obtains value v. Filter returns 1 iff $f(x) = v$.

2.2 Weak vs strong space optimality

Given the space lower bound, we want to distinguish two types of filters that are often referred to as "optimal" in the literature:

Definition 7. Membership filter $\mathcal{F}$ is ***strongly optimal*** if for any pair of fixed $\varepsilon_P, \varepsilon_N$,
$$\frac{1}{n} L_{\mathcal{F}}(n, \varepsilon_P, \varepsilon_N) = D_{KL}(1 - \varepsilon_N \| \varepsilon_P) + o(1), \text{ as } n \to \infty.$$
Meanwhile, $\mathcal{F}$ is ***weakly optimal*** if for any $\varepsilon_P, \varepsilon_N$,
$$\frac{1}{n} L_{\mathcal{F}}(n, \varepsilon_P, \varepsilon_N) \leq D_{KL}(1 - \varepsilon_N \| \varepsilon_P) + O(1), \text{ as } n \to \infty.$$

Most practical filters claiming to achieve optimality (see **??**) are only *weakly* optimal, even as their load approaches 1 and becomes practically infeasible. This is for good reason: any one-sided filter supporting insertion must have an $O(n)$ multiplicative overhead, and thus *cannot be* strongly optimal [22]. Moreover, ε_P is often set to be small in practice, and by taking $\varepsilon_P \to 0$ and ε_N constant, the $D_{KL}(1 - \varepsilon_N \| \varepsilon_P)$ term dominates, and weakly optimal filters will have a negligible multiplicative overhead in this limit. It's when $\varepsilon_P + \varepsilon_N \approx 1$ that the strong optimality becomes significantly better.

In this paper, we focus solely on *strongly* optimal two-sided membership filters, as we are interested in the fundamental space-error trade-off. Another reason is the following simple baseline is, in fact, already weakly optimal.

2.3 Selective insertion: a baseline

Selective insertion is a naturally arising two-sided filter proposed as a solution to the "Bloom paradox" [30]. Essentially, we use a one-sided filter with FPR of ε_P to store only $1-\varepsilon_N$ portion of the keys S. This structure clearly achieves the desired error levels, and its space function is:

$$(1-\varepsilon_N)n\log\frac{1}{\varepsilon_P} = n(D_{KL}(1-\varepsilon_N\|\varepsilon_P) + \underbrace{H(\varepsilon_N)+\varepsilon_N\log(1-\varepsilon_P)}_{\text{overhead.}})$$

For $\varepsilon_P+\varepsilon_N<1$, the overhead is between 0 and 1 bit per key, where the value 1 is taken when $\varepsilon_N=\frac{1}{2}$ and $\varepsilon_P\to 0$. We thus conclude that this structure is weakly optimal. Furthermore, this is the best possible for any structure that *fixes* an $1-\varepsilon_N$ portion of keys to keep, so a strongly optimal filter must randomize on which exact keys to store.

However, this baseline has *unbounded* multiplicative overhead when $\varepsilon_P+\varepsilon_N\approx 1$: the lower bound approaches zero at this limit, yet the selective insertion method still requires $\Omega(1-\varepsilon_N)$ space per key, even when $\varepsilon_P\to 1-\varepsilon_N$.

2.4 Related works

In this subsection, we focus mainly on space-optimal (one-sided) membership filters in the literature. Some filter designs choose to give up (even weak) space optimality for simplicity and practical performance [17][18], but they are less relevant for our purpose. Additionally, we also consider previously proposed space lower bound techniques for similar problems and compare their techniques with ours.

2.4.1 Weakly optimal filters

Following the first weakly optimal filter by Carter et al. [10], many later designs choose to store the *fingerprints* $\{f(x_1)\}_{i=1}^n$ of the keys $S=\{x_i\}_{i=1}^n$ exactly, often in a hash table-like structure. Here, $f:U\to\{0,1\}^k$ is a hash function, often modeled as a random oracle. Suppose we have perfect hash function $h:U\to[n]$ that is bijective on S, then we can simply use a hash table to store $f(x_i)$ in the $h(x_i)$th entry, which achieves FPR of 2^{-k} with the best possible nk bits. The benefit of the hash-table-like structure is the allowance for insertions (even deletions), which also explains why they cannot be strongly optimal. Nevertheless, we can still analyze the inherent limitation of the hash-based approach.

It's been observed that a perfect hash function takes $\log\frac{n^n}{n!}\approx n\log e$ bits to store [1]. In a sense, this $\Omega(n)$ overhead is intrinsic to the hash-table-like structure, as the filters would store a unique position for each x_i in the table, in addition to the fingerprint value. In this view, the various popular filter designs can be seen as different ways to resolve hash collisions. Variants of quotient filter [3, 2, 28] uses linear probing and storing bits of meta information to resolve collisions, while Cuckoo filter [16, 14, 8, 32] uses cuckoo hashing to resolve collisions. Other proposed structures for storing (multi)set of fingerprints [10, 26, 5, 15, 4] also suffer from this overhead due to the same reason of storing some notion of position.

2.4.2 The retrieval problem

In contrast, another class of filters based on the retrieval problem (see section 2.1) do not suffer from the same overhead. Instead of hashing, they use algebraic methods to associate keys $x\in S$ with random values $f(x)$, avoiding the need for storing extra information. Specifically, they would *learn a function* from S to $\{0,1\}^k$ that satisfies some properties. The cost is that S must be a static set that never changes.

The association of filters with retrieval is proposed independently by Dietzfelbinger and Pagh [12] and Porat [29]. Both works apply the same technique of solving linear equations over

a finite field, essentially fitting a linear function from the vector valued representation of keys to the scalar fingerprint values. Our optimal two-sided filter construction in section 4 is inspired by this idea. Instead of a perfect linear function, we solve for the linear function that minimizes the number of errors, a notoriously hard optimization problem related to decoding random linear codes and learning parity with noise (LPN) problem [7].

2.4.3 Space lower bounds and related problems

The counting argument. The classical space lower bound of $n \log \frac{1}{\varepsilon}$ is proven in 1978 by Carter et al. [10]. They applied a counting argument, calculating how many sets of size $n + \varepsilon(u - n)$ (the set each filter actually recognizes) is needed, in order to cover all $\binom{u}{n}$ possible sets of size n. They then take $u \to \infty$ while fixing n, to account for the universe-independent property. While inspiring, this particular reasoning restricts to filters with a fixed FPR of ε, whereas most filters can only achieve ε error *in expectation*[3]. In this case, each configuration of the filter could potentially recognize sets of various sizes. Pagh and Rodler [27] applied the same counting argument for their lower bound of lossy dictionaries.

Rate distortion argument. In our work, we incorporate the rate-distortion theory viewpoint, which sees the filter as a means of compressing the membership information in U. This method naturally takes into account the randomness both during construction and query, as well as the fact that the error rates ε_P and ε_N are measured in expectation. This idea was first introduced by Hurley and Waldvogel [19], who used rate-distortion theory to study approximate membership structures in a fixed universe U, with i.i.d. membership (defined in section 3.1), and a predetermined weight on the two types of error ε_P and ε_N. We adopt this view and modified all three aspects, in order to obtain a lower bound explicitly in both ε_P and ε_N, which applies to filters for a set of fixed size n, in a universe of unknown size.

Incremental filters. This is a more restrictive setting where the (one-sided) filters must support insertion from empty to at most n keys, where n is known before-hand. For this problem, Lovett and Porat [22] showed that such a filter must use $C(\varepsilon)n \log \frac{1}{\varepsilon}$ bits to achieve an FPR of ε for some $C(\varepsilon) > 1$. A corollary is that incremental filters (and dynamic filters which supports deletion) can never hope to be strongly optimal in the generic membership filter sense. The lower bound in this setting has yet to be exactly characterized, and there is a dynamic filter that uses $(1 + o(1))(n \log \frac{1}{\varepsilon} + n \log_2 e)$ bits for fixed ε [4].

2.4.4 Other data structures with two-sided errors

As mentioned, several Bloom-filter-based heuristics trade off FPR and FNR. Retouched Bloom filters [13] reset bits randomly or selectively to reduce FPR (the random variant is weaker than selective insertion). Generalized Bloom filters [21] set some hashed bits to 1 and others to 0 on each insertion, yielding an FPR bound independent of the number of inserted keys. Compacted Bloom filters [25] compress blocks of the bit vector to save space and reduce FPR, incurring nonzero FNR. Autoscaling Bloom filters [20] keep per-bit counts and drop low-load bits to optimize overall accuracy by trading FPR for FNR. We can view these heuristics as different ways to trade-off FPR, FNR, and space, despite being far from optimal.

[3] In Bloom filter's case, there could even be an extreme case where all bits are set to 1 and $\varepsilon = 1$, albeit unlikely.

3 Space Lower Bound for Two-Sided Filters

In this section, we prove the main lower-bound result Theorem 1. As before, all logarithms are base 2. Let $D_{KL}(p\|q)$ be the binary KL-divergence of a Bern(p) distribution from a Bern(q) distribution:

$$D_{KL}(p\|q) = p\log\frac{p}{q} + (1-p)\log\frac{1-p}{1-q}.$$

We first give a high-level overview of the proof, which follows from the observation that a filter is a *lossy code* that encodes the membership information (whether $x \in S$ or not) for each $x \in U$. Prior work [19] considered the most natural setting for applying information-theoretic tools, which is somewhat different from the filter setting:

1. Every $x \in U$ has a probability p of being in S a priori.
2. Alice and Bob agree on a filter construction. Alice will use the filter to store S and send the filter to Bob, who will query every element to decode all the membership information.
3. The error, or distortion, is measured by the percentage of membership information that Bob incorrectly decodes.

To connect this problem to the filter setting we need three steps. In section 3.1, we give an overview of the lossy compression for the i.i.d. setting, while proving a generalized version of the rate-distortion theorem for two error metrics. Then in section 3.2, we reduce the setting where S is a set of fixed size n to the case where S has i.i.d. membership with probability $p = \frac{n}{u}$, in the sense that both settings share similar lower bounds. Finally, in section 3.3, we study the rate-distortion function for sufficiently small p and obtain the desired lower bound.

3.1 Lossy compression of i.i.d. membership information

Consider the i.i.d. membership setting for the moment. Namely, each $x \in U$ has a $p = \frac{n}{u}$ probability of being contained in S, independent from all other elements. Fixing p and taking $u \to \infty$, we can formulate the construction of a filter into the following *lossy compression* problem (see, e.g., Chapter 10 of [11]):

1. Alice observes i.i.d. random variables $X_1, \ldots, X_u \sim \text{Bern}(p)$. In our context, $X_i = \mathbb{1}\{x_i \in S\}$.
2. Alice compresses the outcome $\{X_i\}_{i=1}^u$ using an R-bit (random) encoding function $f_u : \{0,1\}^u \to \{0,1\}^{uR}$. Here R is the **rate**, the number of bits transmitted per element.
3. Bob uses (random) decoding function $g_u : \{0,1\}^{uR} \to \{0,1\}^u$ to recover approximations of the membership information $\{\hat{X}_i\}_{i=1}^u$, by taking

$$(\hat{X}_1, \ldots, \hat{X}_u) = g_u(f_u(X_1, \ldots, X_u)).$$

The **distortion** is defined by a custom measure of distance between the $(X_i, \hat{X}_i)$ pairs. In contrast to the textbook setting, we use two (instead of one) distortion functions to measure the false positive and false negative rates simultaneously:

$$d_P(x,\hat{x}) = \begin{cases} \frac{1}{1-p} & \text{if } (x,\hat{x}) = (0,1) \\ 0 & \text{otherwise.} \end{cases} \qquad d_N(x,\hat{x}) = \begin{cases} \frac{1}{p} & \text{if } (x,\hat{x}) = (1,0) \\ 0 & \text{otherwise.} \end{cases}$$

Given the distortion measures, the rate-distortion functions are defined as follows:

Definition 8. For any $u \in \mathbb{N}$ and $p \in (0,1)$, the **rate-distortion function** $R_{p,u}(\varepsilon_P, \varepsilon_N)$ is the minimum per-element rate for codes with expected distortion at most $(\varepsilon_P, \varepsilon_N)$:

$$R_{p,u}(\varepsilon_P, \varepsilon_N) := \min \Bigg\{ R : \exists R\text{-bit functions } f_u, g_u, \text{ such that } \mathbb{E}\left[\frac{1}{u}\sum_{i=1}^{u} d_P(X_i, \hat{X}_i)\right] \le F_P, \\ \mathbb{E}\left[\frac{1}{u}\sum_{i=1}^{u} d_N(X_i, \hat{X}_i)\right] \le F_N \Bigg\}.$$

Meanwhile, for each p, the **information rate-distortion function** $R_p^{(I)}(\varepsilon_P, \varepsilon_N)$ is:

$$R_p^{(I)}(\varepsilon_P, \varepsilon_N) := \min\{I(X;\hat{X}) : X \sim \text{Bern}(p), \mathbb{E}[d_P(X,\hat{X})] \le \varepsilon_P, \mathbb{E}[d_N(X,\hat{X})] \le \varepsilon_N\},$$

where the minimum is taken over all joint distributions of $(X, \hat{X})$.

We prove the following two lemmas to characterize the rate-distortion functions in our setting; proofs are in appendix A.1. The first explicitly characterizes $R_p^{(I)}(\varepsilon_P, \varepsilon_N)$, and the second generalizes a central result in rate distortion theory.

Lemma 9. *Let $p \in (0,1)$. If $\varepsilon_P + \varepsilon_N \ge 1$, then $R_p^{(I)}(\varepsilon_P, \varepsilon_N) = 0$. Otherwise, we have:*

$$R_p^{(I)}(\varepsilon_P, \varepsilon_N) = H(p(1-\varepsilon_N) + (1-p)\varepsilon_P) - pH(1-\varepsilon_N) - (1-p)H(\varepsilon_P),$$

where $H(p)$ is the binary entropy of a Bern(p) *random variable.*

Proof. Because $X \in \{0,1\}$, we can characterize the conditional distribution $\hat{X} \mid X$ with two parameters:

$$\begin{cases} \lambda & := \mathbb{P}[\hat{X} = 1 \mid X = 0], \\ \mu & := \mathbb{P}[\hat{X} = 1 \mid X = 1]. \end{cases}$$

Let $\hat{X}(\lambda, \mu)$ denote the random variable whose conditional distribution is as defined above. Then, by standard results in information theory,

$$\begin{aligned} I(X; \hat{X}(\lambda,\mu)) &= H(\hat{X}(\lambda,\mu)) - H(\hat{X}(\lambda,\mu) \mid X) \\ &= H(p\mu + (1-p)\lambda) - pH(\mu) - (1-p)H(\lambda). \end{aligned}$$

The last step is to take the minimum mutual information over all (λ, μ) satisfying the distortion constraints, namely:

$$\begin{cases} \mathbb{E}[d_P(X, \hat{X}(\lambda,\mu))] = \lambda & \le \varepsilon_P, \text{ and} \\ \mathbb{E}[d_N(X, \hat{X}(\lambda,\mu))] = 1-\mu & \le \varepsilon_N. \end{cases}$$

Hence, we just need to solve the optimization problem:

$$\begin{aligned} \min \quad & I(X;\hat{X}(\lambda,\mu)) = H(p\mu + (1-p)\lambda) - pH(\mu) - (1-p)H(\lambda), \\ \text{s.t.} \quad & \begin{cases} 0 \le \lambda \le \varepsilon_P, \\ 1 - \varepsilon_N \le \mu \le 1, \end{cases} \end{aligned}$$

Taking derivatives shows that the objective function is always decreasing in λ and increasing in μ, so we can just take $\lambda = \varepsilon_P$ and $\mu = 1 - \varepsilon_N$. □

Lemma 10. *For all $p \in (0,1)$ and $u \in \mathbb{N}$, we have:*

$$R_{p,u}(\varepsilon_P, \varepsilon_N) \ge R_p^{(I)}(\varepsilon_P, \varepsilon_N).$$

Proof. See appendix A.1. □

3.2 From fixed key density to i.i.d. membership

When the element set S is generated in the i.i.d. fashion above, $|S|$ follows a $\mathrm{Binom}(u,p)$ distribution, which is close to a $\mathrm{Poisson}(n)$ distribution when n is fixed and u is large. However, we can show that this difference is negligible if $n \to \infty$ and p is fixed.

Lemma 11. *For all $\varepsilon_P, \varepsilon_N \in (0,1)$ such that $\varepsilon_P + \varepsilon_N < 1$, and for all fixed rational $p = \frac{n}{u} \in (0,1)$, any approximate membership structure $\mathcal{M}$ (Definition 4) must satisfy:*

$$\liminf_{n\to\infty} \frac{1}{n} L_{\mathcal{M}}(n, u, \varepsilon_P, \varepsilon_N) \geq \frac{1}{p} R_p^{(I)}(\varepsilon_P, \varepsilon_N).$$

Proof. Let n_1, u_1 be coprime integers such that $\frac{n_1}{u_1} = p$, and define sequences $\{n_j\}, \{u_j\}$ to be $n_j = jn_1$ and $u_j = ju_1$, for all $j \in \mathbb{N}$. Let S_j be the set of keys generated by the i.i.d. membership process, where each element in universe $[u_j]$ is independently included in S_j with probability p. Clearly, $|S_j| \sim \mathrm{Binom}(u_j, p)$, with $\mathrm{var}(|S_j|/n_j) = O(1/n_j)$. By Chebyshev's inequality, there is a decreasing sequence of real numbers $\delta_j = \Theta(n_j^{-1/3})$ such that for each j we have

$$\mathbb{P}\big[|S_j - n_j| \geq \delta_j n_j\big] \leq \frac{\mathrm{var}(|S_j|/n_j)}{\delta_j^2 n_j^2} = O(n_j^{-1/3}) = O(\delta_j).$$

Then, using a filter of appropriate size, we can construct a protocol for transmitting the membership information of the universe $[u_j]$:

1. If $|S_j - n_j| \leq \delta_j n_j$, then we build an approximate membership structure of n_j elements with expected error rates ε_P and ε_N, using at most $L(n_j, u_j, \varepsilon_P, \varepsilon_N)$ bits. Since it's likely that $|S_j| \neq n_j$, we also use $O(\delta_j n_j \log u_j)$ bits to explicitly specify some of the elements we dropped, or the dummy elements we added.
2. If $|S_j| > (1+\delta_j)n_j$, then we send a dummy message indicating $\hat{X}_i = 1$ for all $i \in [u_j]$.
3. If $|S_j| < (1-\delta_j)n_j$, then we send a dummy message indicating $\hat{X}_i = 0$ for all $i \in [u_j]$.

Note that cases 2 and 3 incur expected d_P distortion $\leq \frac{1}{1-p}$ and expected d_N distortion $\leq \frac{1}{p}$ respectively. For case 1, the expected distortions are at most ε_P and ε_N, since the explicit encoding can only decrease the error rates. Thus, we can bound the expected distortions of this protocol by

$$\begin{cases} \mathbb{E}\left[\frac{1}{u_j}\sum_{i=1}^{u_j} d_P(X_i, \hat{X}_i)\right] \leq (1-\delta_j)\varepsilon_P + \frac{\delta_j}{1-p} & =: a_j, \\ \mathbb{E}\left[\frac{1}{u_j}\sum_{i=1}^{u_j} d_N(X_i, \hat{X}_i)\right] \leq (1-\delta_j)\varepsilon_N + \frac{\delta_j}{p} & =: b_j, \end{cases}$$

where $a_j \to \varepsilon_P$ and $b_j \to \varepsilon_N$ as $j \to \infty$. Now, this protocol uses at most $L(n_j, u_j, \varepsilon_P, \varepsilon_N) + O(\delta_j n_j \log u_j)$ bits. By lemma 10, we have:

$$\frac{1}{u_j} L(n_j, u_j, \varepsilon_P, \varepsilon_N) \geq R_p^{(I)}(a_j, b_j) - \frac{1}{u_j} O(\delta_j n_j \log u_j), \text{ for all } j \in \mathbb{N}.$$

As $j \to \infty$, we have $\delta_j n_j \log u_j = o(u_j)$, so the claim follows from the continuity of $R_p^{(I)}$. □

3.3 Putting it all together

Now we are ready to prove the space lower bound for approximate membership structures:

Proof for theorem 1. It suffices to show how the expression for $R_p^{(I)}$ converges to $D_{KL}(1-\varepsilon_N\|\varepsilon_P)$ as $p \to 0$. Plugging in the expression for $R_p^{(I)}$ from lemma 9, the right hand side of the inequality in lemma 11 equals:

$$\begin{aligned}\frac{1}{p}R_p^{(I)}(\varepsilon_P,\varepsilon_N) &= \frac{1}{p}\Big[H\big(p(1-\varepsilon_N)+(1-p)\varepsilon_P\big) - pH(1-\varepsilon_N) - (1-p)H(\varepsilon_P)\Big] \\ &= H(\varepsilon_P) - H(1-\varepsilon_N) + \frac{1}{p}\Big[H\big(\varepsilon_P + p(1-\varepsilon_N-\varepsilon_P)\big) - H(\varepsilon_P)\Big]\end{aligned}$$

We can approximate the last term by a second-order Taylor expansion:

$$= H(\varepsilon_P) - H(1-\varepsilon_N) + (1-\varepsilon_N-\varepsilon_P)H'(\varepsilon_P) + \frac{(1-\varepsilon_N-\varepsilon_P)^2 H''(\varepsilon_P)}{2}\cdot p + O(p^2).$$

Plugging in $H(p) = -p\log p - (1-p)\log(1-p)$, $H'(p) = \log\frac{1-p}{p}$, and $H''(p) = \frac{1}{p} + \frac{1}{1-p}$, we have:

$$\begin{aligned}\frac{1}{p}R_p^{(I)}(\varepsilon_P,\varepsilon_N) &= -\varepsilon_P\log\varepsilon_P - (1-\varepsilon_P)\log(1-\varepsilon_P) + \varepsilon_N\log\varepsilon_N \\ &\quad + (1-\varepsilon_N)\log(1-\varepsilon_N) + (1-\varepsilon_P-\varepsilon_N)\log\frac{1-\varepsilon_P}{\varepsilon_P} \\ &\quad + \frac{(1-\varepsilon_N-\varepsilon_P)^2}{2\varepsilon_P(1-\varepsilon_P)}\cdot p + O(p^2) \\ &= \varepsilon_N\log\frac{\varepsilon_N}{1-\varepsilon_P} + (1-\varepsilon_N)\log\frac{1-\varepsilon_N}{\varepsilon_P} + \frac{(1-\varepsilon_N-\varepsilon_P)^2}{2\varepsilon_P(1-\varepsilon_P)}\cdot p + O(p^2) \\ &= D_{KL}(1-\varepsilon_N\|\varepsilon_P) + \frac{(1-\varepsilon_N-\varepsilon_P)^2}{2\varepsilon_P(1-\varepsilon_P)}\cdot p + O(p^2)\end{aligned}$$

□

Finally, for two-sided filters, the lower bound follows from taking $p \to 0$.

Proof for corollary 2. Assuming the appropriate random oracles on the universe, a filter can implement any approximate membership structure of the same error rates. Therefore, by lemma 11, we have:

$$\liminf_{n\to\infty}\frac{L(n,\varepsilon_P,\varepsilon_N)}{n} \geq \lim_{p\to 0, p\in\mathbb{Q}}\frac{1}{p}R_p^{(I)}(\varepsilon_P,\varepsilon_N),$$

which is just $D_{KL}(1-\varepsilon_N\|\varepsilon_P)$. □

4 A Strongly Optimal Filter Construction

In this section, we present a filter design that achieves the space lower bound of $D_{KL}(1-\varepsilon_N\|\varepsilon_P) + o(1)$ per element. Our construction is inspired by Porat [29]. Although impractical, this filter is conceptually simple and serves to complement the lower bound result.

Theorem 12. *Assuming access to appropriate random oracle functions, when $\varepsilon_P = \frac{1}{q}$ for some prime power q, there is a strongly optimal two-sided filter that achieves error rates $(\varepsilon_P,\varepsilon_N)$ with probability at least* 0.9.

Proof. Let $n = |S|$ and let $\mathbb{F}_q$ be the finite field with $q = 1/\varepsilon_P$ elements. Assume we have a vector-valued random oracle function $h : U \to \mathbb{F}_q^m$. For each $x \in U$, and for each entry $j \in [m]$,

$h(x)_j$ is i.i.d. uniform on $\mathbb{F}_q$. WLOG, let $S = \{x_1, \ldots, x_n\}$. We consider the following linear system of equations, where y is not the all-zero vector:

$$\begin{cases} \langle h(x_1), y\rangle = 0, \\ \vdots \\ \langle h(x_n), y\rangle = 0. \end{cases}$$

where $\langle u, v\rangle$ denotes $\sum_{j=1}^{m} u_j v_j$ in $\mathbb{F}_q$, and y is an unknown vector in $\mathbb{F}_q^m$. Define m to be the smallest number such that the non-zero solution y satisfying the most equations will, with probability at least 0.9, satisfy at least a $1 - \varepsilon_N$ fraction of the equations.

We use this system to construct a strongly optimal filter $\mathcal{F}$. We first describe the initialization and query process, and then prove its optimality.

Initialization. On input ε_P, ε_N, and key set $S = \{x_1, \ldots, x_n\}$, $\mathcal{F}$ stores the binary encoding of an optimal solution y to the above system of equations. This takes $m \log q + O(1)$ bits. The construction succeeds if the stored y satisfies at least a $1 - \varepsilon_N$ fraction of the equations.

Query. On query $x \in U$, $\mathcal{F}$ queries the oracle to get $h(x)$ and outputs 1 iff $\langle h(x), y\rangle = 0$. The probability that $\mathcal{F}(x) = 1$ is exactly $\frac{1}{q}$ for non-keys, and for a uniform random query over the keys S, $\mathcal{F}(x) = 0$ with probability at least $1 - \varepsilon_N$.

Analysis. The filter satisfies the target FNR of ε_N over the randomness of the initialization by design. For FPR, we have $\mathbb{P}_{x\sim\mathrm{Unif}(U\setminus S)}[\mathcal{F}(x) = 1] = \frac{1}{q}$ by the following lemma:

Lemma 13. *For any fixed $y_1, \ldots, y_m \in \mathbb{F}_q$ not all zero, and for i.i.d. $h_1, \ldots, h_m$ uniform on $\mathbb{F}_q$, we have:*

$$\mathbb{P}\left[\sum_{j=1}^{m} y_j h_j = 0\right] = \frac{1}{q}.$$

Proof. WLOG suppose $y_1 \neq 0$. Then, for any $c \in \mathbb{F}_q$, we have:

$$\mathbb{P}\left[\sum_{j=1}^{m} y_j h_j = c\right] = \mathbb{P}\left[h_1 = y_1^{-1}\left(c - \sum_{j=2}^{m} y_j h_j\right)\right] = \frac{1}{q}.$$

□

We now bound its space usage by showing that, for any fixed δ, for sufficiently large n, it always suffices to take m to be:

$$m = \left\lceil n\left(\frac{D_{KL}(1 - \varepsilon_N \| \varepsilon_P)}{\log q} + \delta\right)\right\rceil$$

to ensure that the filter is strongly optimal with FNR at most ε_N.

We will use the second moment method to show that, $\mathbb{P}[Z \geq 1] = 1 - o(1)$ as $n \to \infty$. Let Y be the set of all possible non-zero values of y with $|Y| = q^m - 1$, and let Z be the number of solutions y satisfying the FNR requirement. Let X_y be the indicator that a fixed non-zero y satisfies $(1 - \varepsilon_N)n$ equations. Then, we have:

$$\begin{aligned} \mathbb{E}[Z] &= \sum_{y\in Y} \mathbb{E}[X_y] \\ &= \sum_{y\in Y} \mathbb{P}[\langle h(x_1), y\rangle = 0, \ldots, \langle h(x_n), y\rangle = 0] \\ &= (q^m - 1) \cdot 2^{-n \cdot D_{KL}(1-\varepsilon_N \| \varepsilon_P) - o(n)}, \end{aligned}$$

where the last step is by Chernoff-Hoeffding theorem. We denote this probability $\mathbb{P}[X_y = 1]$ by p_n.

To bound $\mathbb{E}[Z^2]$, write

$$\mathbb{E}[Z^2] = \sum_{y \in Y} \mathbb{E}[X_y] + \sum_{y \neq y'} \mathbb{E}[X_y X_{y'}].$$

Fix distinct y, y'. For each $i \in [n]$, the pair of events $\{\langle h(x_i), y\rangle = 0\}$ and $\{\langle h(x_i), y'\rangle = 0\}$ are independent whenever y and y' are linearly independent over $\mathbb{F}_q$; indeed, the mapping from $h(x_i)$ to $\big(\langle h(x_i), y\rangle, \langle h(x_i), y'\rangle\big)$ is a linear surjection from $\mathbb{F}_q^m$ to $\mathbb{F}_q^2$. The probability of each pair of values is just $\frac{1}{q^2}$, so $\mathbb{E}[X_y X_{y'}] = \frac{1}{q^2}$.

If instead y' is a non-zero scalar multiple of y (there are exactly $q-2$ such y' for each fixed y when excluding y itself), then $\langle h(x_i), y'\rangle = 0$ iff $\langle h(x_i), y\rangle = 0$ for all i, so $X_y = X_{y'}$ and $\mathbb{E}[X_y X_{y'}] = \mathbb{E}[X_y] = p_n$. Hence the second moment equals:

$$\begin{aligned}
\mathbb{E}[Z^2] &= \sum_{y \in Y} \mathbb{E}[X_y] + \sum_{\substack{y \neq y' \\ y' \in \langle y\rangle \setminus \{y\}}} \mathbb{E}[X_y X_{y'}] + \sum_{\substack{y \neq y' \\ y, y' \text{ lin. indep.}}} \mathbb{E}[X_y X_{y'}] \\
&\leq (q^m - 1)\, p_n + (q^m - 1)(q-2)\, p_n + (q^m - 1)(q^m - q)\, p_n^2 \\
&= (q^m - 1)\Big((q-1)\, p_n + (q^m - q)\, p_n^2\Big).
\end{aligned}$$

By Paley–Zygmund inequality,

$$\begin{aligned}
\mathbb{P}[Z \geq 1] &\geq \frac{\mathbb{E}[Z]^2}{\mathbb{E}[Z^2]} \\
&\geq \frac{(q^m - 1)^2 p_n^2}{(q^m - 1)\Big((q-1)\, p_n + (q^m - q)\, p_n^2\Big)} \\
&= \frac{(q^m - 1)\, p_n}{(q-1) + (q^m - q)\, p_n}.
\end{aligned}$$

With our choice of m, we have $q^m p_n = 2^{\delta n - o(n)}$ and all other terms are negligible. Hence this probability tends to 1 as $n \to \infty$. □

References

[1] Djamal Belazzougui, Fabiano C. Botelho, and Martin Dietzfelbinger. Hash, displace, and compress. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 682–693. Springer, 2009.

[2] Michael A. Bender, Martin Farach-Colton, Mayank Goswami, Rob Johnson, Samuel McCauley, and Shikha Singh. Bloom filters, adaptivity, and the dictionary problem. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 182–193. IEEE Computer Society, 2018.

[3] Michael A. Bender, Martin Farach-Colton, Rob Johnson, Russell Kraner, Bradley C. Kuszmaul, Dzejla Medjedovic, Pablo Montes, Pradeep Shetty, Richard P. Spillane, and Erez Zadok. Don't thrash: How to cache your hash on flash. *Proc. VLDB Endow.*, 5(11):1627–1637, 2012.

[4] Michael A. Bender, Martin Farach-Colton, John Kuszmaul, William Kuszmaul, and Mingmou Liu. On the optimal time/space tradeoff for hash tables. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1284–1297. ACM, 2022.

[5] Ioana O. Bercea and Guy Even. A dynamic space-efficient filter with constant time operations. In Susanne Albers, editor, *17th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2020, June 22-24, 2020, Tórshavn, Faroe Islands*, volume 162 of *LIPIcs*, pages 11:1–11:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[6] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[7] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 435–440. ACM, 2000.

[8] Alexander Dodd Breslow and Nuwan Jayasena. Morton filters: Faster, space-efficient cuckoo filters via biasing, compression, and decoupled logical sparsity. *Proc. VLDB Endow.*, 11(9):1041–1055, 2018.

[9] Andrei Z. Broder and Michael Mitzenmacher. Survey: Network applications of bloom filters: A survey. *Internet Math.*, 1(4):485–509, 2003.

[10] Larry Carter, Robert W. Floyd, John Gill, George Markowsky, and Mark N. Wegman. Exact and approximate membership testers. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 59–65. ACM, 1978.

[11] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2001.

[12] Martin Dietzfelbinger and Rasmus Pagh. Succinct data structures for retrieval and approximate membership (extended abstract). In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, volume 5125 of *Lecture Notes in Computer Science*, pages 385–396. Springer, 2008.

[13] Benoit Donnet, Bruno Baynat, and Timur Friedman. Retouched bloom filters: allowing networked applications to trade off selected false positives against false negatives. In Christophe Diot and Mostafa H. Ammar, editors, *Proceedings of the 2006 ACM Conference on Emerging Network Experiment and Technology, CoNEXT 2006, Lisboa, Portugal, December 4-7, 2006*, page 13. ACM, 2006.

[14] David Eppstein. Cuckoo filter: Simplification and analysis. In Rasmus Pagh, editor, *15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016, June 22-24, 2016, Reykjavik, Iceland*, volume 53 of *LIPIcs*, pages 8:1–8:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[15] Tomer Even, Guy Even, and Adam Morrison. Prefix filter: Practically and theoretically better than bloom. *Proc. VLDB Endow.*, 15(7):1311–1323, 2022.

[16] Bin Fan, David G. Andersen, Michael Kaminsky, and Michael Mitzenmacher. Cuckoo filter: Practically better than bloom. In Aruna Seneviratne, Christophe Diot, Jim Kurose, Augustin Chaintreau, and Luigi Rizzo, editors, *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, CoNEXT 2014, Sydney, Australia, December 2-5, 2014*, pages 75–88. ACM, 2014.

[17] Thomas Mueller Graf and Daniel Lemire. Xor filters. *ACM J. Exp. Algorithmics*, 25:1–16, 2020.

[18] Thomas Mueller Graf and Daniel Lemire. Binary fuse filters: Fast and smaller than xor filters. *ACM J. Exp. Algorithmics*, 27:1.5:1–1.5:15, 2022.

[19] Paul Hurley and Marcel Waldvogel. Bloom filters: One size fits all? In *32nd Annual IEEE Conference on Local Computer Networks (LCN 2007), 15-18 October 2007, Clontarf Castle, Dublin, Ireland, Proceedings*, pages 183–190. IEEE Computer Society, 2007.

[20] Denis Kleyko, Abbas Rahimi, Ross W. Gayler, and Evgeny Osipov. Autoscaling bloom filter: controlling trade-off between true and false positives. *Neural Comput. Appl.*, 32(8):3675–3684, 2020.

[21] Rafael P. Laufer, Pedro B. Velloso, and Otto Carlos Muniz Bandeira Duarte. A generalized bloom filter to secure distributed network applications. *Comput. Networks*, 55(8):1804–1819, 2011.

[22] Shachar Lovett and Ely Porat. A lower bound for dynamic approximate membership data structures. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 797–804. IEEE Computer Society, 2010.

[23] Lailong Luo, Deke Guo, Richard T. B. Ma, Ori Rottenstreich, and Xueshan Luo. Optimizing bloom filter: Challenges, solutions, and comparisons. *IEEE Commun. Surv. Tutorials*, 21(2):1912–1949, 2019.

[24] Páll Melsted and Jonathan K. Pritchard. Efficient counting of k-mers in DNA sequences using a bloom filter. *BMC Bioinform.*, 12:333, 2011.

[25] Negar Mosharraf, Anura P. Jayasumana, and Indrakshi Ray. Compacted bloom filter. In *2nd IEEE International Conference on Collaboration and Internet Computing, CIC 2016, Pittsburgh, PA, USA, November 1-3, 2016*, pages 304–311. IEEE Computer Society, 2016.

[26] Anna Pagh, Rasmus Pagh, and S. Srinivasa Rao. An optimal bloom filter replacement. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 823–829. SIAM, 2005.

[27] Rasmus Pagh and Flemming Friche Rodler. Lossy dictionaries. In Friedhelm Meyer auf der Heide, editor, *Algorithms - ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*, volume 2161 of *Lecture Notes in Computer Science*, pages 300–311. Springer, 2001.

[28] Prashant Pandey, Alex Conway, Joe Durie, Michael A. Bender, Martin Farach-Colton, and Rob Johnson. Vector quotient filters: Overcoming the time/space trade-off in filter design. In Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava, editors, *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 1386–1399. ACM, 2021.

[29] Ely Porat. An optimal bloom filter replacement based on matrix solving. In Anna E. Frid, Andrey Morozov, Andrey Rybalchenko, and Klaus W. Wagner, editors, *Computer Science - Theory and Applications, Fourth International Computer Science Symposium in Russia, CSR 2009, Novosibirsk, Russia, August 18-23, 2009. Proceedings*, volume 5675 of *Lecture Notes in Computer Science*, pages 263–273. Springer, 2009.

[30] Ori Rottenstreich and Isaac Keslassy. The bloom paradox: When not to use a bloom filter. *IEEE/ACM Trans. Netw.*, 23(3):703–716, 2015.

[31] Sasu Tarkoma, Christian Esteve Rothenberg, and Eemil Lagerspetz. Theory and practice of bloom filters for distributed systems. *IEEE Commun. Surv. Tutorials*, 14(1):131–155, 2012.

[32] Minmei Wang, Mingxun Zhou, Shouqian Shi, and Chen Qian. Vacuum filters: More space-efficient and faster replacement for bloom and cuckoo filters. *Proc. VLDB Endow.*, 13(2):197–210, 2019.

A Appendix

A.1 Proof for lemma 10

Lemma 14 (Same as lemma 10)**.** *For all $p \in (0,1)$ and $u \in \mathbb{N}$, we have*

$$R_{p,u}(\varepsilon_P, \varepsilon_N) \geq R_p^{(I)}(\varepsilon_P, \varepsilon_N).$$

Before the proof itself, we first show the useful fact that $R_{p,u}(\varepsilon_P, \varepsilon_N)$ is jointly convex in both inputs.

Lemma 15. *For all $u \in \mathbb{N}, p \in (0,1)$, inputs $\varepsilon_P, \varepsilon'_P, \varepsilon_N, \varepsilon'_N \in [0,1]$, and $\lambda \in (0,1)$, we have:*

$$R_{p,u}(\lambda\varepsilon_P + (1-\lambda)\varepsilon'_P, \lambda\varepsilon_N + (1-\lambda)\varepsilon'_N) \leq \lambda R_{p,u}(\varepsilon_P, \varepsilon_N) + (1-\lambda) R_{p,u}(\varepsilon'_P, \varepsilon'_N)$$

Proof. Fixing X, let $\hat{X}$ and $\hat{X}'$ be random variables achieving the optimal rates:

$$\begin{cases} I(X;\hat{X}) & = R_{p,u}(\varepsilon_P, \varepsilon_N), \\ I(X;\hat{X}') & = R_{p,u}(\varepsilon'_P, \varepsilon'_N). \end{cases}$$

Now consider random variable $\hat{X}_\lambda$ defined by:

$$\hat{X}_\lambda = \begin{cases} \hat{X} \text{ with probability } \lambda, \\ \hat{X}' \text{ with probability } 1-\lambda, \end{cases}$$

independently from $X, \hat{X}, \hat{X}'$. From linearity of expectation, we have:

$$\begin{cases} \mathbb{E}[d_P(X, \hat{X}_\lambda)] & \leq \lambda\varepsilon_P + (1-\lambda)\varepsilon'_P, \\ \mathbb{E}[d_N(X, \hat{X}_\lambda)] & \leq \lambda\varepsilon_N + (1-\lambda)\varepsilon'_N. \end{cases}$$

Since $I(X;\hat{X})$ is convex in the conditional distribution $\hat{X} \mid X$, the desired statement follows from:

$$\begin{aligned} R_{p,u}(\lambda\varepsilon_P + (1-\lambda)\varepsilon'_P, \lambda\varepsilon_N + (1-\lambda)\varepsilon'_N) &\leq I(X;\hat{X}_\lambda) \\ &\leq \lambda I(X;\hat{X}) + (1-\lambda) I(X;\hat{X}') \\ &= \lambda R_{p,u}(\varepsilon_P, \varepsilon_N) + (1-\lambda) R_{p,u}(\varepsilon'_P, \varepsilon'_N). \end{aligned}$$

□

Now we are ready for the proof. We use the following shorthand for the tuples of random variables: $X^u = (X_1, \ldots, X_u)$ and $\hat{X}^u = (\hat{X}_1, \ldots, \hat{X}_u)$.

Proof for Lemma 10. For all R such that there exists pair of (random) functions $f : \{0,1\}^u \to \{0,1\}^{uR}$ and $g : \{0,1\}^{uR} \to \{0,1\}^u$ satisfying distortion requirements $\varepsilon_P, \varepsilon_N$, we must have:

$$\begin{aligned}
uR &\geq H(f(X^u)) \\
&\geq I(X^u; f(X^u)) \\
&\geq I(X^u; \hat{X}^u) \\
&= H(X^u) - H(X^u \mid \hat{X}^u) \\
&= \sum_{i=1}^{u} H(X_i) - \sum_{i=1}^{u} H(X_i \mid \hat{X}^u, X_1, \ldots, X_{i-1}) \\
&\geq \sum_{i=1}^{u} H(X_i) - \sum_{i=1}^{u} H(X_i \mid \hat{X}_i) \\
&= \sum_{i=1}^{u} I(X_i; \hat{X}_i) \\
&\geq \sum_{i=1}^{u} R_{p,u}(\mathbb{E}[d_P(X_i, \hat{X}_i)], \mathbb{E}[d_N(X_i, \hat{X}_i)]) \\
&\geq uR_p^u \left(\mathbb{E}\left[\frac{1}{u} \sum_{i}^{u} d_P(X_i, \widehat{X}_i) \right], \mathbb{E}\left[\frac{1}{u} \sum_{i}^{u} d_N(X_i, \hat{X}_i) \right] \right) \text{ by convexity,} \\
&\leq uR_{p,u}\left(\varepsilon_P, \varepsilon_N\right).
\end{aligned}$$

□