Streaming Algorithms for Support-Aware Histograms

Justin Y. Chen [1] **Piotr Indyk** [1] **Tal Wagner** [2]

Abstract

Histograms, i.e., piece-wise constant approximations, are a popular tool used to represent data distributions. Traditionally, the difference between the histogram and the underlying distribution (i.e., the approximation error) is measured using the L_p norm, which sums the differences between the two functions over all items in the domain. Although useful in many applications, the drawback of this error measure is that it treats approximation errors of all items in the same way, irrespective of whether the mass of an item is important for the downstream application that uses the approximation. As a result, even relatively simple distributions cannot be approximated by succinct histograms without incurring large error.

In this paper, we address this issue by adapting the definition of approximation so that only the errors of the items that belong to the *support* of the distribution are considered. Under this definition, we develop efficient 1-pass and 2-pass streaming algorithms that compute near-optimal histograms in sub-linear space. We also present lower bounds on the space complexity of this problem. Surprisingly, under this notion of error, there is an exponential gap in the space complexity of 1-pass and 2-pass streaming algorithms. Finally, we demonstrate the utility of our algorithms on a collection of real and synthetic data sets.

1. Introduction

The exponential growth of massive data sets over the recent years necessitated the development of methods for computing succinct summaries of data. Such summaries should accurately preserve the desired data properties, while being succinct enough to be stored or communicated efficiently. A popular approach to representing data succinctly is to compute their *histograms*. For a data set whose elements come from the universe $[n] = \{1 \dots n\}$, a k-histogram approximation is a piece-wise constant function defined over $[n]$ consisting of k pieces (see Section 2 for a formal definition). Thanks to its compact representation (which requires storing only $O(k)$ numbers), such histograms are often used to approximate the distribution of data attributes, and have applications to a variety of tasks such as approximate query processing, data visualization and density estimation.

Traditionally, the problem of finding a good histogram approximation to a given data set is formalized as follows. The algorithm is given an empirical distribution P over $[n]$, i.e., a sequence $p_1 \dots p_n \geq 0$ such that $\sum_i p_i = 1$. The goal is to find a k-histogram f that minimizes the approximation error $\|P - f\|$, where $\|\cdot\|$ is some norm (typically ℓ_1 or ℓ_2). Multiple exact and approximate algorithms for computing such histograms given input P are known (Cormode et al., 2012). The problem has been also studied extensively in settings where the distribution P is given *implicitly*. In particular, there has been a significant body of work in the streaming setting, where the algorithm has limited storage, and is allowed only one (or a small number of) passes over the data. Two streaming models have been considered: (i) the *time-series model*, where the probabilities $p_1, p_2, \dots, p_n$ are given explicitly (Guha et al., 2001; 2004; Guha, 2005; Buragohain et al., 2007; Halim et al., 2009; Terzi & Tsaparas, 2006), or (ii) the *turnstile model*, where the algorithm is given a sequence of data elements $i \in [n]$, and each p_i is defined implicitly as the fraction of times i occurs in the input sequences (Gilbert et al., 2001; 2002; Guha et al., 2002; Muthukrishnan et al., 2005; Cormode et al., 2006; Hegde et al., 2016). Other implicit input models, e.g., allowing the algorithm to sample elements from the distribution P, has been studied as well, see e.g., (Indyk et al., 2012; Chan et al., 2014b;a; Acharya et al., 2015), or a recent survey (Canonne, 2020). The aforementioned streaming results typically offer multiplicative error guarantees, while the sampling algorithms offer an additive error guarantee.

Although useful, the histogram approximation problem as formulated above suffers from a fundamental issue, which is that the approximation error is measured over *all* elements i in $[n]$, regardless of whether the density of i is important for the downstream application that uses the estimation. This

[1]MIT, Cambridge, MA, USA [2]Microsoft Research, Redmond, WA, USA. Correspondence to: Justin Chen <justc@mit.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

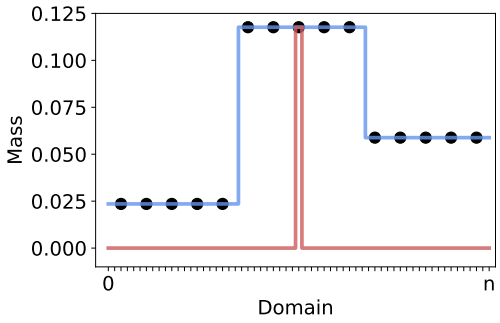

Figure 1. Optimal support-aware (blue) and support-oblivious (red) 3-piece histograms on sparse data (black points). The support-aware histogram achieves zero support-aware error while the support-oblivious histogram only gives a non-trivial approximation on a single data point.

means that even very simple distributions cannot be approximated by k-histograms with sublinear values of k without incurring essentially the maximum possible error. As an example, consider a distribution that is uniform over all *even* items $i \in [n]$. Such a distribution can be approximated by a 1-histogram over even numbers with 0 error, but any approximation by an $o(n)$-histogram over the whole domain $[n]$ incurs error of $1 - o(1)$, i.e., the maximum possible (Note that such error can be achieved by approximating all probabilities by 0). See Figure 1 for another example of this phenomenon.

In this paper, we propose to address this issue by making the following small but crucial modification to the definition of the problem: instead of measuring the approximating error over the whole domain $[n]$, we only measure it over the *support* of the distribution P. That is, for a given the distribution P over the domain $[n]$, the goal is to compute a k-histogram f over $[n]$ that (approximately) minimizes the error (see Section 2 for a detailed definition):

$$err_P(f) = \|(P - f)_{\text{supp}(P)}\|_s \tag{1}$$

In the rest of the paper we will focus on the case $s = 1$. We will refer to the error definition of Equation 1 as *support-aware L1 error*, and to histograms minimizing such error as *support-aware histograms*. By contrast, we will refer to the classic notion of measuring error over the entire domain as *support-oblivious* error.

The definition of support-aware error clearly avoids the simple example mentioned earlier. As shown in Section 6, it also improves the quality of histogram approximation for real data. Thus, in this paper we focus on developing efficient streaming algorithms for approximately computing best support-aware histograms.

1.1. Our results

In this paper we initiate the study of low-space streaming algorithms for support-aware histograms. Our main contributions are as follows. In all cases we consider randomized streaming algorithms with a constant probability of error, that operate in the so-called *strict turnstile* model. That is, the input stream can contain insertions as well as deletions of elements $i \in [n]$, as long as no element is deleted more often than inserted. For the multiset S of the end of the stream (after accounting for all insertions and deletions), the distribution P is defined by setting $p_i = m_i/m$, where m_i is the number of occurrences of i in S, and $m = \sum_i m_i$, where we assume that $m > 0$. We use H_k to denote the set of all k-histograms over $[n]$.

- We start from an observation that any streaming algorithm that simply detects whether P can be approximated by a *single line* with zero error, i.e., whether $\min_{f \in H_1} err_P(f) = 0$, requires $\Omega(n)$ bits of space, even if the algorithm is allowed a constant number of passes. This stands in contrast to the aforementioned *support-oblivious* streaming algorithms which provide multiplicative error guarantees. Because of this limitation, we focus on additive error guarantees in the reminder of this paper.

- We present two streaming algorithms that compute support-aware histograms with additive error guarantees. That is, the algorithms compute histograms f such that $err_P(f) \leq \min_{f^* \in H_k} err_P(f^*) + \epsilon$, for a parameter $\epsilon > 0$. The first algorithm performs a single pass and uses $\sqrt{n} \cdot \log(n) \cdot k/\epsilon^3 \cdot \text{polylog}(k \log(n/\epsilon))$ space, while the second algorithm performs two passes and uses $\log^2(n) \cdot k/\epsilon^3 \cdot \text{polylog}(k, \epsilon^{-1})$ space. We note that the reported histogram f might have more than k pieces, but the number of pieces in f does not exceed the space bound of the algorithm, i.e., the two algorithms have bi-criterion approximation guarantees.

- We complement these results by a lower bound showing that any single pass algorithm reporting a histogram f such that $err_P(f) \leq \min_{f^* \in H_k} err_P(f^*) + \epsilon$ must use $\Omega(\sqrt{n})$ bits of space, even when $k = 2$ and the algorithm is allowed to report a histogram with up to $o(\sqrt{n})$ pieces. This shows that the space bound of our 2-pass algorithm cannot be achieved in a single pass.

On the empirical side, we analyze the performance of our one-pass and two-pass streaming algorithms on several real-world datasets, demonstrating the practical application of support-aware histograms. In particular, both of our algorithms achieve much lower (up to 3x) support-aware error than natural baselines, for the same amount of allocated space.

1.2. Related work

As we mention in the introduction, streaming and sampling algorithms for histogram estimation have been studied extensively. In addition to the works listed earlier, the survey (Cormode et al., 2012), in particular section 3.7.1 "Histograms Over Streaming Data", provides a good (if somewhat dated) overview of the field. The vast majority of past work was focused on support-oblivious histograms. However, the following three papers are motivated by similar considerations as this paper, even though their technical development is quite different.

- (Muthukrishnan et al., 2005) considered streaming algorithms for general "workloads", where for each data item i the algorithm is given a weight w_i and the goal is to minimize $\sum_i (p_i - f_i)^2 w_i$. Unfortunately, their algorithms required storing essentially the whole weight vector $w_1 \dots w_n$, which required linear in n space (unless the weight vectors can be compressed losslessly into smaller space). Our formulation can be viewed as a special case of their definition where the weight vector w is the characteristic vector of the support of P. Since the support of P defined implicitly by the input stream, there is no need to specify it or store it explicitly, circumventing the lower bounds for the general weights.

- (Batu & Canonne, 2017; Diakonikolas et al., 2018) considered sampling algorithms for detecting whether the input distribution is uniform over its support, which essentially corresponds to checking whether it can be approximated with a 1-histogram. In their formulation, the optimal 1-histogram f was the minimizer of $\|(p - f_S)\|_1$, over all histograms f *and* sets $S \subset [n]$, where f_S is a vector of dimension n such that $(f_S)_i = f_i$ if $i \in S$ and $(f_S)_i = 0$ if $i \notin S$. Unfortunately, this definition is difficult to adopt in the streaming model, as storing the set S requires linear space in general. Therefore, in our definition we essentially let $S = \text{supp}(P)$ (note that $S \subset \text{supp}(P)$ without loss of generality).

- (Du et al., 2021) considered density estimators with respect to the error measure as in Equation 1, and used them in the context of learning-augmented streaming algorithms. However, they represented the density function using neural networks, not histograms. (Our paper was motivated in part by the goal of replacing neural networks with more computationally tractable density estimators that can be computed efficiently in the streaming model.)

2. Preliminaries

Setting: empirical distribution of a turnstile stream. As the input, we get a stream of insertions and deletions of elements in $[n] := \{1, \dots, n\}$, such that at any point in the stream, every item has been inserted at least as many times as it has been deleted (i.e., its count is non-negative). Let $m_i \geq 0$ denote the count of $i \in [n]$ at the end of the stream, and let $m = \sum_{i=1}^{n} m_i$ be the total count. The empirical distribution $P = (p_1, \dots, p_n)$ of the stream is given by $p_i = m_i/m$. Let $\text{supp}(P) = \{i : p_i > 0\}$ denote the support of P. For simplicity, we assume that the length of the input stream is polynomial in n, which in particular implies that each m_i can be represented using $O(\log n)$ bits.

Goal: support-aware histogram approximation. Our goal is to approximate P by a histogram with few pieces. A *k-piece histogram* is a function $f : [n] \to [0, 1]$ with $i_1 \leq \ldots \leq i_{k-1} \in [n]$ and $\gamma_1, \dots, \gamma_k \in [0, 1]$, such that, denoting $i_0 = 0$ and $i_k = n$, we have

$$f(i) = \gamma_j \quad \forall\, j \in [k] \;\text{ and }\; i \in \{i_{j-1} + 1, \dots, i_j\}.$$

In words, the delimiter indices $i_1 \leq \ldots \leq i_{k-1}$ partition $[n]$ into k pieces, and the histogram approximates each piece $j \in [k]$ with the fixed value γ_j. The *support-aware $L1$ error* of approximating P by f is given, as in eq. (1), by

$$err_P(f) = \sum_{i \in \text{supp}(P)} |p_i - f(i)|.$$

Let H_k be the family of all k-histograms over $[n]$. Given an input error parameter $\epsilon \in [0, 1]$ and number of pieces k, our goal is to find an approximately optimal histogram approximation for P, namely a histogram f such that

$$err_P(f) \leq \inf_{f^* \in H_k} err_P(f^*) + \epsilon.$$

In general, we allow a *bicriteria approximation guarantee*, which means that the output histogram f is allowed to have $k' \geq k$ pieces, with k' being as close to k as possible.

Sampling One of the subroutines that our streaming algorithms use is $L0$ *sampling*, which returns an item i selected uniformly at random from $\text{supp}(P)$, together with the value of p_i. To accomplish the first task we use the one-pass streaming algorithm of (Jowhari et al., 2011) which returns one such samples with probability $1 - \delta$ using $O(\log^2 n \log(1/\delta))$ space.

3. Lower bound for multiplicative approximation

In this section we provide a simple lower bound showing that any streaming algorithm that computes a support-aware

histogram with multiplicative error guarantees must use a linear amount of space, even when $k = 1$ (i.e., the histogram is just a line) and even if the algorithm is allowed a constant number of passes. In fact, the lower bound holds even if the goal is to determine whether the support-aware error is equal to 0 or not.

Theorem 3.1. *Any streaming algorithm (in the insertions-only model, randomized with a constant probability of error) that determines whether* $\min_{f \in H_1} err_P(f) = 0$ *requires* $\Omega(n)$ *bits of space, even if the algorithm is allowed to use* $O(1)$ *passes over the data.*

Proof. The proof follows via a reduction from Set Disjointness. Recall that Set Disjointness is a communication complexity problems involving two parties, Alice and Bob. The inputs of both parties are n-length bit vectors, denoted as $a_1 \dots a_n$ and $b_1 \dots b_n$, respectively. The goal of the problem is for both parties to determine whether there exists $i \in [n]$ such that $a_i = b_i = 1$. It is known that this task requires that Alice and Bob exchange $\Omega(n)$ bits (Kalyanasundaram & Schintger, 1992).

The reduction is as follows. For simplicity we consider one-pass algorithms; generalization to $O(1)$ passes is immediate. Suppose there exists an S-space streaming algorithm A as in the theorem statement. Then we can use it to solve Set Disjointness in S space as follows. First, Alice creates a stream S consisting of all elements i such that $a_i = 1$. The stream is input to A, and its state is transmitted to Bob. Then, Bob creates a stream S' consisting of elements j such that $b_j = 1$, which is input to the algorithm A. Let P be the distribution induced by the concatenation of streams S and S'. It can be seen that its non-zero coordinates are all equal (and thus the support-aware error of a 1-histogram is 0) if and only if the sets defined by vectors a and b are disjoint. This implies that S must be at least linear in n. □

4. One-Pass Algorithm

In this section we prove nearly matching upper and lower bounds for the space usage of one-pass streaming algorithm for support-aware histogram approximation, with additive error. The space usage is proportional to $\sqrt{n}$, where n is the domain size. In the next section we show that by allowing an additional pass, the dependence on n can be improved exponentially, to $\mathrm{polylog}(n)$.

We begin by presenting the one-pass algorithm. It uses an algorithm for $L1$ *heavy hitters* computation over the stream (Cormode & Muthukrishnan, 2005), defined as follows.

Fact 4.1 (L1-heavy hitters). Let $\ell \geq 1$ be an integer and $\epsilon \in (0, 1)$. There is a one-pass streaming algorithm in the turnstile model that uses $O(\epsilon^{-1} \ell \log n)$ words of space, and returns $Z \subset [n]$, such that $|Z| = O(\ell)$, and every $i \in [n]$ such that $p_i \geq 1/\ell$ (i.e., i is an $(1/\ell)$-heavy hitter) is included in Z. In addition, for every $i \in Z$, it outputs $z_i \in [0, 1]$ such that $|z_i - p_i| \leq \epsilon/\ell$.

Algorithm 1 One-Pass Algorithm

Input: stream S, parameters k, ε

Before the pass on the stream:

1: $s \leftarrow O(\sqrt{n} \cdot \log(n) \cdot k/\epsilon^3) \cdot \mathrm{polylog}(k \log(n/\epsilon))$
2: Draw s uniform i.i.d. samples from $[n]$, with replacement. Denote the set of samples S.

During the pass on the stream:

1: Run the $L1$ heavy hitter procedure from Fact 4.1, with $\ell = \sqrt{n}/\epsilon^2$, and let Z be the output.
2: Concurrently, maintain the exact counts of the elements in S.

After the pass on the stream:

1: Approximate each element in $i \in Z$ with its own histogram piece, with value z_i (notation from Fact 4.1).
2: Use dynamic programming (see (Cormode et al., 2012)) to compute the best k-piece histogram for the elements in $S \setminus Z$.

The $L1$ heavy hitters are elements on which the histogram approximation could accrue large error individually. Our one-pass algorithm computes them to ensure each one is approximated with high precision, and uses uniform sampling to approximate the non-heavy elements in the support. The algorithm is specified in Algorithm 1, and its guarantees are summarized in the next theorem, whose proof appears in Appendix A.

Theorem 4.2. *Algorithm 1 uses* $s = O(\sqrt{n} \cdot \log(n) \cdot k/\epsilon^3) \cdot \mathrm{polylog}(k \log(n/\epsilon))$ *space, and outputs a histogram* f *with* s *pieces such that* $err_P(f) \leq \min_{f^* \in H_k} err_P(f^*) + \epsilon$.

Remark. We remark that if we are given any upper bound n' on the support size (i.e., such that $\mathrm{supp}(P) \leq n'$), then the $\sqrt{n}$ term in Theorem 4.2 can be improved to $\sqrt{n'}$, by using $L0$ sampling (Jowhari et al., 2011) instead of the uniform samples in Algorithm 1 (with a logarthmic blowup).

Lower bound. The next theorem that the $\sqrt{n}$ term is *necessary* for one-pass algorithms, even for histograms with $k = 2$ pieces. Its proof is given in Appendix B.

Theorem 4.3. *Let* $\epsilon > 0$ *be a sufficiently small constant. Any one-pass streaming algorithm that outputs a histogram* f *such that* $err_P(f) \leq \min_{f^* \in H_2} err_P(f^*) + \epsilon$, *where* f *is allowed to have as many as* $O(\sqrt{n})$ *pieces, must use* $\Omega(\sqrt{n})$ *space.*

We emphasize that the space lower bound in the above theorem holds even when the output histogram f is allowed to

use as many as $O(\sqrt{n})$ pieces, while only having to approximately match the performance of the best 2-piece histogram f^*. Therefore the lower bound holds not only for proper histogram approximation (in which f would be restricted to using only 2 pieces), but also to the bicriteria guarantee of the upper bound in Theorem 4.2, thus complementing it directly up to low-order $\mathrm{poly}(k, \epsilon^{-1}, \log n)$ factors.

5. Two-Pass Algorithm

While $\sqrt{n}$ space complexity is necessary and sufficient for streaming algorithms that take one pass over the data, given a second pass over the data, a polylogarthmic dependence on n suffices.

Hierarchical Heavy Hitters Our two-pass algorithm uses prior work on hierarchical heavy hitters (Cormode et al., 2008). For the purposes of the algorithm, hierarchical heavy hitters are defined over a full binary tree with leaves $[n]$ (for simplicity, assume n is a power of two) s.t. the subtree rooted at the ith node at height ℓ has leaves $[i \cdot 2^\ell + 1, (i+1) \cdot 2^\ell]$ for $\ell \in [0, \lg n]$ and $i \in [0, n/2^\ell - 1]$.

For a node h in this tree, let $T(h)$ denote the subtree rooted at h and let $D(h)$ denote the set of leaves in $T(h)$, corresponding to the set of domain elements covered by h. For a given heaviness threshold ϕ, hierarchical heavy hitters are defined in a bottom up fashion. For a node h, let $V(h) \subseteq T(h)$ be the set of nodes in the subtree rooted at h (not including h) that are hierarchical heavy hitters. Then, h is itself a hierarchical heavy hitter if the domain elements $D(h) \setminus \left(\cup_{v \in V(h)} D(v)\right)$ have mass exceeding ϕ (e.g., the elements contained in h are heavy even after removing all the elements from h's heavy descendants).

We will use the following additional notation. Let $S(h) = D(h) \setminus \left(\cup_{v \in V(h)} D(v)\right)$ be the set of domain elements that count towards whether h is a hierarchical heavy hitter. Let $l(h)$ and $r(h)$ denote the left and right children of h, respectively. We will refer to hierarchical heavy hitters which are located at the leaves of the tree to be singleton heavy hitters (corresponding to the normal single-element notion of heavy hitters).

Theorem 5.1. *There is a 2-pass streaming algorithm with constant failure probability that uses $O(k \log^2(n)/\varepsilon^3) \cdot \mathrm{polylog}(k, \varepsilon^{-1})$ space and outputs $f \in H_{ck/\varepsilon}$ for some constant c such that $err_P(f) \leq \min_{f' \in H_k} err_P(f') + \varepsilon$.*

In essence, the algorithm calculates hierarchical heavy hitters in the first pass in order to produce a partitioning of the domain into a small number of singleton heavy hitters and contiguous intervals s.t. each of the intervals has mass at most $\varepsilon/2k$. In the second pass, the algorithm exactly estimates each of the heavy hitters and approximates each interval by a sample median. As none of the intervals are too heavy, this median approximation suffices to produce an approximation almost as good as the optimal k-piece histogram.

Algorithm 2 Two-Pass Algorithm

Input: stream S, parameters k, ε

First pass:

$T \leftarrow$ Hierarchical Heavy Hitters of S with heaviness threshold $\varepsilon/2k$ using full binary tree over $[n]$ hierarchy via (Cormode et al., 2008), Section 5.2
$L \leftarrow \emptyset$ {Set of disjoint, maximal contiguous intervals with less than $\varepsilon/2k$ mass}
$H \leftarrow \emptyset$ {Set of singleton elements of $[n]$ with at least $\varepsilon/2k$ mass}
for $h \in T$ **do**
  if $|S(h)| = 1$ **then**
    $H \leftarrow H \cup \{S(h)\}$
  else
    for each maximal contiguous interval $[a, b] \in S(l(h))$ **do**
      $L \leftarrow L \cup \{[a, b]\}$
    end for
    for each maximal contiguous interval $[a', b'] \in S(r(h))$ **do**
      $L \leftarrow L \cup \{[a', b']\}$
    end for
  end if
  $L' \leftarrow$ all maximal contiguous intervals of $[n]$ not covered by $H \cup L$
  $L \leftarrow L \cup L'$
end for

Second pass:

Exactly count the frequency of each element in H
For each interval in L, use the algorithm of (Jowhari et al., 2011) to take $O(\varepsilon^{-2} \log(k/\varepsilon))$ i.i.d. samples selected uniformly at random from the support, together with their masses, and calculate the median of the sample masses
Output histogram approximation where each element in H is approximated by its true mass, and each element in an interval in L is approximated by the approximate median of its interval

In what follows we assume that the hierarchical heavy hitter algorithm (Cormode et al., 2008) succeeded, which occurs with constant failure probability. Furthermore, a set of items is called *heavy* if its total mass is at least $\varepsilon/2k$; it is called *light* otherwise.

Lemma 5.2. $|H| \leq \lceil \frac{2k}{\varepsilon} \rceil$

Proof. As no element belongs to multiple hierarchical heavy hitters, there can be at most $2k/\varepsilon$ hierarchical heavy hitters,

of which the elements of H are a subset. □

Lemma 5.3. $|L| = O\left(\frac{k}{\varepsilon}\right)$

Proof. By the same reasoning as the previous lemma, there are at most $2k/\varepsilon$ non-singleton heavy hitters. Let h be one such hierarchical heavy hitter. The number of intervals added to L from h will be the number of maximal contiguous intervals in $S(l(h))$ and in $S(r(h))$. Note that the subtrees $T(l(h))$ and $T(r(h))$ are disjoint. Without loss of generality, consider $S(l(h))$.

If $T(l(h)) \setminus \{h\}$ contains no hierarchical heavy hitters, it will contain a single maximal contiguous interval of all leaf nodes in the subtree rooted at $l(h)$. Let $u \in T(l(h)) \setminus \{h\}$ be a hierarchical heavy hitter s.t. there is no other hierarchical heavy hitter $v \in T(l(h))$ s.t. $u \in T(v)$ (i.e. u is not a descendant of another hierarchical heavy hitter in $T(l(h))$). Call such a hierarchical heavy hitter u a "direct descendant" of $l(h)$. The union of the leaves of the subtrees rooted at the direct descendants of $l(h)$ are exactly the leaves that are in the subtree $T(l(h))$ but not included in $S(l(h))$ (by the definition of hierarchical heavy hitters). Therefore, each direct descendant can add at most one additional maximal contiguous interval as each direct descendant removes a single contiguous subinterval, creating at most one new discontinuity.

Each hierarchical heavy hitter is the direct descendant of at most one other hierarchical heavy hitter. For the sake of contradiction, consider distinct hierarchical heavy hitters u, v, w s.t. u is a direct descendant of both v and w. This means that $u \in T(v)$ and $u \in T(w)$ but $v \notin T(w)$ and $w \notin T(v)$. By the binary tree structure of the hierarchy, this is impossible: for any two subtrees with nonempty intersection, one subtree must contain the other. Therefore, for each hierarchical heavy hitter, one maximal contiguous interval is added for $l(h)$, one for $r(h)$, and one is added for being a direct descendant of another hierarchical heavy hitter for a total of at most $\left\lceil \frac{6k}{\varepsilon} \right\rceil$ intervals.

It remains to count the number of elements added to L via the set L'. By the argument above and Lemma 5.2, there are $O(k/\varepsilon)$ elements/intervals in $H \cup L$ in step 15 of the algorithm. Therefore, L' can have size at most $O(k/\varepsilon)$ as the elements in $L \cup H$ can only partition $[n]$ into at most $|L| + |H| + 1$ pieces. So, at the end of the first pass, $|L| = O(k/\varepsilon)$, as required. □

Lemma 5.4. *Each interval in L has mass less than $\varepsilon/2k$.*

Proof. Consider any such non-singleton hierarchical heavy hitter h. Note that $S(l(h))$ and $S(r(h))$ must both have mass less than $\varepsilon/2k$. If both were heavy, $S(h)$ would be empty and if, without loss of generality, only $l(h)$ was heavy, then $S(h) = S(r(h))$ which must have mass less than $\varepsilon/2k$. Every interval added to L via a non-singleton heavy hitter is a subinterval of $S(l(h))$ or $S(r(h))$ and thus is light. The only other intervals added to L are those that were not included in any hierarchical heavy hitters and thus are also light. □

Lemma 5.5. *Let $x_1 \leq x_2 \leq \ldots \leq x_n$ be a sorted list of real numbers in $[0, 1]$, and let $\beta = \sum_{i=1}^{n} x_i$ be the total mass of the x_i's. Let M^* be the median of the list and let $\hat{M}_s$ be the median of a random sample (w/ replacement) of size s. Let $\ell(M) = \sum_{i=1}^{n} |x_i - M|$ be the ℓ_1 error of approximating the set by M. Then, for $\varepsilon \in [0, 1]$,*

$$\Pr(\ell(\hat{M}_s) - \ell(M^*) > \varepsilon\beta) \leq 2e^{-\varepsilon^2 s/8}.$$

The proof follows standard techniques and can be found in Appendix C. We are now ready to prove the main theorem.

Proof of Theorem 5.1. First, we will argue that the algorithm is correct. Using the algorithm of (Cormode et al., 2008), with constant failure probability, we will correctly find all hierarchical heavy hitters. By Lemma 5.3, after the first pass, we will have a partition of the domain $[n]$ into the sets H and L where each element in H is heavy and each interval in L is light. In the second pass, as we exactly count each element in H, our approximation has no error on these elements.

Assume that the median approximation for each interval in L is done exactly and call the resulting histogram approximation f'. Consider an optimal k-piece histogram $f^* \in H_k$ and any interval $\ell \in L$. Consider the case where f^* uses a single piece to cover the interval ℓ. Note that out of all constant approximations of the interval ℓ, the median mass of the nonzero elements will minimize the support-aware $L1$ error. Therefore, limited to the interval ℓ, the approximation error of f' will be at most that of f^*.

Now, consider the case where f^* uses multiple pieces to cover ℓ. Note that over any interval, f' will have approximation error no greater than that of the approximation that always predicts zero mass as the median is the optimal constant approximation. The zero approximation incurs error equal to the mass of the interval, which we know is at most $\varepsilon/2k$. Therefore, in this case, f' incurs additional error over f^* of at most $\varepsilon/2k$. As f^* has k total pieces, at most k distinct intervals in L can be covered by multiple pieces of f^*. Combining the two cases, $err_P(f') - err_P(f^*) \leq \varepsilon/2$.

It remains to account for the error in approximating the medians of the intervals in L. By Lemma 5.5, for each interval, by taking $O(\varepsilon^{-2} \log(k/\varepsilon))$ samples, the excess error on each interval due to sampling is at most $\varepsilon/2$ times the mass of the interval (even after union bounding over all $O(k/\varepsilon)$ intervals). As the total mass of all intervals is at

most 1, $err_P(f) - err_P(f') \leq \varepsilon/2$. In total,

$$err_P(f) - err_P(f^*) \leq \varepsilon,$$

the approximation has error at most ε more than the optimal.

Now, we will consider the space usage of the algorithm. Finding all $\varepsilon/2k$ hierarchical heavy hitters with constant failure probability requires $O(\frac{k \log n}{\varepsilon} \log \frac{k}{\varepsilon})$ space (while not shown here, the dependence on the failure probability is logarithmic). By Lemmas 5.2 and 5.3, storing H and L from the first pass takes $O(\frac{k \log n}{\varepsilon})$ space with the $\log n$ factor coming from the bits needed to store the indexes of the singleton heavy hitters or boundaries of the intervals.

In the second pass, exactly counting each singleton heavy hitter takes $O(\log n)$ bits per element in H for total space of $O(\frac{k \log n}{\varepsilon})$ space. As we need $O(\varepsilon^{-2} \log(k/\varepsilon))$ samples per interval in L, the total number of samples needed will be $O(\frac{k \log(k/\varepsilon)}{\varepsilon^3})$. Using the algorithm of (Jowhari et al., 2011), we can select a uniform sample from $\text{supp}(P)$ with failure probability δ in $O(\log^2 n \log(1/\delta))$ space. Setting δ s.t. we can union bound over all samples, the total space due to sampling is $O\left(\frac{k \log^2(n) \log(k/\varepsilon) \log(k/\varepsilon^3 \log(k/\varepsilon))}{\varepsilon^3}\right)$, completing the analysis. □

6. Experiments

DATASET	DOMAIN	SUPPORT	STREAM
TAXI	605K	418K	1.3M
CAIDA	17M	58K	30M
WAR & PEACE	18K	1.9K	53K
MCDONALDS	18K	2.2K	14K

Table 1. Domain size, support size, and stream length of each dataset used in experiments.

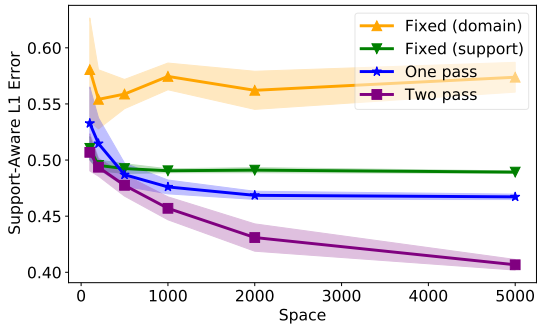

Figure 2. Comparison of support-aware $L1$ error with varying space usage with $k = 5$ on the Taxi dataset. Shading indicates one standard deviation over 10 trials.

We analyze the performance of our one-pass and two-pass streaming algorithms on a variety of real-world datasets,

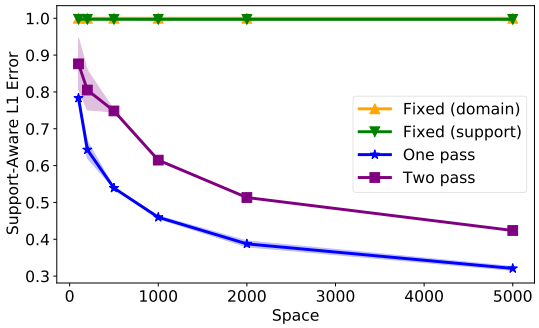

Figure 3. Comparison of support-aware $L1$ error with varying space usage with $k = 5$ on the CAIDA dataset. Shading indicates one standard deviation over 10 trials.

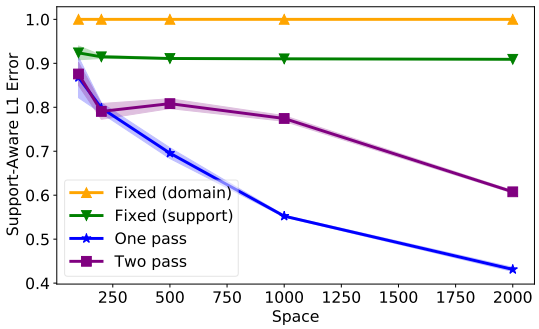

Figure 4. Comparison of support-aware $L1$ error with varying space usage with $k = 5$ on the War & Peace dataset. Shading indicates one standard deviation over 10 trials.

demonstrating the practical application of support-aware histograms.

Datasets We compare histogram algorithms on four datasets. The Taxi dataset (NYC Taxi and Limousine Commission, 2021) contains pickup times of yellow cabs in New York City in the month of January 2021. Stream elements come from (day of week, hour, minute, second) tuples with frequencies corresponding to the number of yellow cab pickups occurring at a given time throughout the month. The domain is in temporal order starting with midnight on Monday morning.

The CAIDA dataset (CAIDA, 2016) contains internet traffic data from a Tier1 ISP between Chicago and Seattle in 2016. Stream elements are the three most significant bytes of the destination IP addresses.

The War & Peace dataset (Tolstoy, 1869) contains 3-prefixes of all words in *War and Peace* by Leo Tolstoy. Stream elements are three letter strings with frequencies corresponding to the number of times the strings start words in the novel. The domain is in lexicographic order.

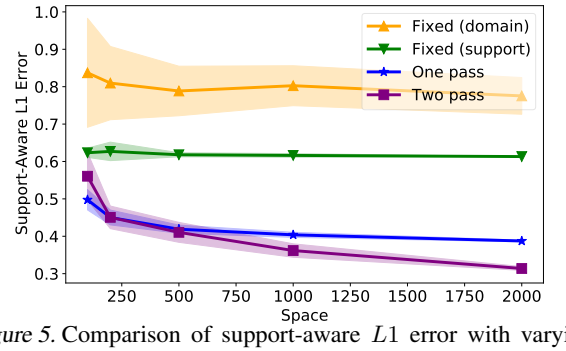

Figure 5. Comparison of support-aware $L1$ error with varying space usage with $k = 5$ on the McDonalds dataset. Shading indicates one standard deviation over 10 trials.

The McDonalds dataset (POI factory) contains the locations of McDonalds restaurants in the United States and Canada. Stream elements are the latitudes of the restaurants, rounded to the nearest 0.01.

The domain size, support size, and stream length of each dataset is displayed in Table 1.

Baselines and Implementation We compare our one-pass and two-pass algorithms against two simple baselines. For given parameters s, k, both baselines split the domain into k equal-sized intervals. For each interval, the *Fixed (support)* baseline takes s/k $L0$ samples from the stream constrained to that interval and uses the median mass of those samples to approximate the interval. The *Fixed (domain)* baseline does the same but takes s/k samples from the domain elements constrained to the interval rather than $L0$ sampling which only will sample non-zero domain elements. Hence, for sparse data, the Fixed (domain) baselines may often approximate intervals to have zero mass. The Fixed (support) and Fixed (domain) baselines are natural algorithms that optimize for support-aware $L1$ error and support-oblivious $L1$ error, respectively.

We compare these baselines to our one-pass and two-pass algorithms (Algorithm 1 and Algorithm 2). The one-pass algorithm uses $s/2$ space to compute heavy hitters with approximate counts via the Space Saving algorithm (Metwally et al., 2005). The rest of the $s/2$ space is used for $L0$ samples that form the set S on which we will compute the best k-piece histogram. Areas of the domain in between pieces of this histogram (as its endpoints are $L0$ samples, there are some parts of the domain not covered) are approximated by the median mass of all of the samples.

For the two-pass algorithm, as all of the streams in our experiments are insertion-only, we use the space-saving based implementation of hierarchical heavy hitters (Mitzenmacher et al., 2012). In the first pass, the heaviness threshold is set to $\frac{n \lg(n)}{s}$, and in the second pass, the $L0$ samples allowed by the budget s are evenly distributed over all light intervals.

All of the baselines and algorithms involve storing sampled elements from the stream along with their masses. The parameter s is varied to compare how the number of samples affects the performance of the algorithms (as shown in the x-axis of the figures). The k parameter determines the number of pieces used by the baselines and by the one-pass algorithm (after removing heavy hitters). We set $k = 5$ in the experiments in this section and display the same figures with $k = 10$ in Appendix E. In Appendix D, we experiment with baselines which scale k with the space budget to disambiguate the performance of our algorithms from the fact that they are improper and use more than k pieces. Each algorithm is run 10 times for each parameter setting. All figures display the mean support-aware $L1$ error along with one standard deviation shown in shading.

Results We start by just comparing the baselines. For all but the CAIDA dataset (in which neither baseline gets non-trivial error), Fixed (support) significantly outperforms Fixed (domain) across space usage. This is not surprising as Fixed (support) is optimized for support-aware error while Fixed (domain) is not. Still, these results show on real datasets that the sparsity of the support makes support-aware error a meaningfully different error metric to the classic support-oblivious notion of error. In particular, in these datasets, succinct histograms give better approximations for the non-zero elements rather than for the entire domain.

Compared to these baselines, our one-pass and two-pass algorithms achieve significantly smaller error, even with space usage in the hundreds (this is true for both the smaller War & Peace and McDonalds datasets as well as the large-scale Taxi and CAIDA datasets). As space usage increases, the algorithms tend to produce better approximations, beating the baselines by up to 3x on the CAIDA and War & Peace datasets.

Intriguingly, on the CAIDA and War & Peace datasets, the one-pass algorithm outperforms the two-pass algorithm though the two-pass algorithm has significantly smaller space complexity as indicated by our theoretical results. These datasets contain many heavy hitters which contribute significantly to the error of the two-pass algorithm (or indeed of any histogram with few pieces). While both the one-pass and two-pass algorithms compute heavy hitters from the stream and separately approximate them, the one-pass algorithm will store $s/2$ heavy hitters, while due to the higher heaviness threshold in the two-pass algorithm, only a fraction of that space will be used on singleton heavy hitters. The two-pass algorithm will use the remaining space to fit medians to the intervals defined by the non-singleton hierarchical heavy hitters, allowing it to outperform the one-pass algorithm on the other datasets which have fewer outliers.

Acknowledgements

This research was supported by the NSF TRIPODS program (award DMS-2022448), Simons Investigator Award, GIST-MIT Research Collaboration grant, MIT-IBM Watson collaboration, MathWorks Engineering Fellowship, and NSF Graduate Research Fellowship under Grant No. 1745302.

References

Acharya, J., Diakonikolas, I., Hegde, C., Li, J. Z., and Schmidt, L. Fast and near-optimal algorithms for approximating distributions by histograms. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pp. 249–263, 2015.

Batu, T. and Canonne, C. L. Generalized uniformity testing. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 880–889. IEEE, 2017.

Buragohain, C., Shrivastava, N., and Suri, S. Space efficient streaming algorithms for the maximum error histogram. In *2007 IEEE 23rd International Conference on Data Engineering*, pp. 1026–1035. IEEE, 2007.

CAIDA. CAIDA internet traces, 2016. URL http://www.caida.org/data/monitors/passive-equinix-chicago.xml.

Canonne, C. L. A survey on distribution testing: Your data is big. but is it blue? *Theory of Computing*, pp. 1–100, 2020.

Chan, S.-O., Diakonikolas, I., Servedio, R. A., and Sun, X. Efficient density estimation via piecewise polynomial approximation. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 604–613, 2014a.

Chan, S.-O., Diakonikolas, I., Valiant, P., and Valiant, G. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1193–1203. SIAM, 2014b.

Cormode, G. and Muthukrishnan, S. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

Cormode, G., Garofalakis, M., and Sacharidis, D. Fast approximate wavelet tracking on streams. In *International Conference on Extending Database Technology*, pp. 4–22. Springer, 2006.

Cormode, G., Korn, F., Muthukrishnan, S., and Srivastava, D. Finding hierarchical heavy hitters in streaming data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(4):1–48, 2008.

Cormode, G., Garofalakis, M., Haas, P. J., and Jermaine, C. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1–3):1–294, 2012.

Diakonikolas, I., Kane, D. M., and Stewart, A. Sharp bounds for generalized uniformity testing. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, 2018.

Du, E., Wang, F., and Mitzenmacher, M. Putting the "learning" into learning-augmented algorithms for frequency estimation. In *International Conference on Machine Learning*, 2021.

Gilbert, A. C., Kotidis, Y., Muthukrishnan, S., and Strauss, M. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Vldb*, volume 1, pp. 79–88, 2001.

Gilbert, A. C., Guha, S., Indyk, P., Kotidis, Y., Muthukrishnan, S., and Strauss, M. J. Fast, small-space algorithms for approximate histogram maintenance. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pp. 389–398, 2002.

Guha, S. Space efficiency in synopsis construction algorithms. In *Proceedings of the 31st international conference on Very large data bases*, pp. 409–420, 2005.

Guha, S., Koudas, N., and Shim, K. Data-streams and histograms. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pp. 471–475, 2001.

Guha, S., Indyk, P., Muthukrishnan, S., and Strauss, M. J. Histogramming data streams with fast per-item processing. In *International Colloquium on Automata, Languages, and Programming*, pp. 681–692. Springer, 2002.

Guha, S., Shim, K., and Woo, J. Rehist: Relative error histogram construction algorithms. In *VLDB*, volume 4, pp. 300–311, 2004.

Halim, F., Karras, P., and Yap, R. H. Fast and effective histogram construction. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 1167–1176, 2009.

Hegde, C., Indyk, P., and Schmidt, L. Fast recovery from a union of subspaces. *Advances in Neural Information Processing Systems*, 29:4394–4402, 2016.

Indyk, P., Levi, R., and Rubinfeld, R. Approximating and testing k-histogram distributions in sub-linear time. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pp. 15–22, 2012.

Jowhari, H., Sağlam, M., and Tardos, G. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 49–58, 2011.

Kalyanasundaram, B. and Schintger, G. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992.

Metwally, A., Agrawal, D., and El Abbadi, A. Efficient computation of frequent and top-k elements in data streams. ICDT'05, 2005.

Mitzenmacher, M., Steinke, T., and Thaler, J. Hierarchical heavy hitters with the space saving algorithm. In *2012 Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX)*, 2012.

Muthukrishnan, S., Strauss, M., and Zheng, X. Workload-optimal histograms on streams. In *European Symposium on Algorithms*, pp. 734–745. Springer, 2005.

NYC Taxi and Limousine Commission. Yellow Taxi Trip Records January, 2021. URL https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

POI factory. McDonalds USA/Canada locations. URL http://www.poi-factory.com/node/11154.

Terzi, E. and Tsaparas, P. Efficient algorithms for sequence segmentation. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pp. 316–327. SIAM, 2006.

Tolstoy, L. *War and Peace*. 1869. URL https://www.gutenberg.org/files/2600/2600-h/2600-h.html.

A. Proof of one-pass algorithm

In this section we prove Theorem 4.2. We recall that the algorithm (Algorithm 1) identifies the $L1$ heavy hitters and closely approximates them with their own histogram pieces, while approximating the rest of the elements using random sampling. Most of the proof is dedicated to dealing with the latter elements (the non-heavy hitters, whose mass is bounded by $\approx 1/\sqrt{n}$). It will therefore be convenient to encompass this part of the proof in the following auxiliary theorem.

Theorem A.1 (bounded masses)**.** *Let $\alpha \in (0,1)$ be constant. Suppose we are promised that $p_i \leq \epsilon^2/n^\alpha$ for all $i \in [n]$. Then, there is a 1-pass streaming algorithm that uses $\tilde{O}\left(n^{1-\alpha} \cdot k \cdot \log(n/\epsilon)\right)$ space, and outputs $f \in H_k$ such that $err_P(f) \leq \min_{f^* \in H_k} err_P(f^*) + \epsilon$.*

A.1. Proof of Theorem 4.2

We first show how to obtain Theorem 4.2 from Theorem A.1. Algorithm 1 finds $L1$ heavy hitters whose mass is at least $\epsilon^2/\sqrt{n}$. More precisely, by Fact 4.1, it uses $O(\sqrt{n}/\epsilon^3)$ space to find a subset $Z \subset [n]$ of size at most $O(\sqrt{n}/\epsilon^2)$, such that every $i \in [n]$ with $p_i \geq \epsilon^2/\sqrt{n}$ belongs to Z. Furthermore, for each such i it also returns z_i such that $|z_i - p_i| \leq \epsilon^3/\sqrt{n}$. Algorithm 1 allots each $i \in Z$ its own histogram piece whose value is z_i, and thus accrues error of at most $\epsilon^3/\sqrt{n}$ on each $i \in Z$, hence a total error of at most $|Z| \cdot \epsilon^3/\sqrt{n} \leq \epsilon$ on all the elements in Z. The remaining elements $[n] \setminus Z$ have masses bounded by $\epsilon^2/\sqrt{n}$, and can be approximated by Theorem A.1 with $\alpha = 1/2$, accruing an additional total error of ϵ of those elements. Theorem 4.2 follows by scaling ϵ by a constant. As for the space usage of Algorithm 1, Fact 4.1 uses $O(\sqrt{n}\log(n)/\epsilon^3)$ space, and exactly counting the masses of the samples in S takes $O(|S|)$ space, for a total of $O(\sqrt{n}\log(n)/\epsilon^3 + |S|) = O(\epsilon^{-3} \cdot k \cdot \sqrt{n}\log(n) \cdot \text{polylog}(k\log(n/\epsilon))$ space. □

A.2. Proof of Theorem A.1

We now prove Theorem A.1, thus completing the proof of Theorem 4.2. The algorithm described in the theorem is as follows. Before observing the stream, we draw $s = O\left(n^{1-\alpha} \cdot k \cdot \log(n/\epsilon)\right)$ uniform i.i.d. samples from $[n]$ (with replacement). Denote the sequence of samples by S. We use our pass over the stream to count the exact mass p_i of every i in S.

The space usage of the algorithm is clearly $O(s)$. The remainder of the proof will show that it achieves the guarantee of Theorem A.1, albeit with error $O(\epsilon \cdot k\log(n/\epsilon))$ instead of ϵ. Therefore, to get the requisite error guarantee, we scale ϵ down by $k\log(n/epsilon)$, bringing the total space usage to $O\left(n^{1-\alpha} \cdot k \cdot (\log(n/\epsilon) + \log\left(k\log(n/\epsilon)\right))\right) = \tilde{O}(\epsilon \cdot k\log(n/\epsilon))$, as stated in the theorem.

In order to choose our output f in the end of the pass on the stream, we need some more notation. Let $h \in H_k$ be a histogram. Recall it is defined by $i_1 \leq \ldots \leq i_{i-1} \in [n]$ and $\gamma_1, \ldots, \gamma_k \in [0,1]$, with the convention $i_0 = 0$ and $i_k = n$. We split the items in each piece in h into exponential intervals according to the masses. For $j = 1, \ldots, k$ and $z = \lfloor \log_{1+\epsilon}(m) \rfloor, \ldots, \lceil \log_{1+\epsilon}(n^\alpha/\epsilon^2) \rceil$, define:

$$I_{j,z}^{(h)} = \{i_{j-1}+1, \ldots, i_j\} \cap \{i : p_i \in \left((\tfrac{1}{1+\epsilon})^{z+1}, (\tfrac{1}{1+\epsilon})^z\right]\},$$

and

$$S_{j,z}^{(h)} = S \cap I_{j,z}^{(h)}.$$

Now define our cost estimate for h according to our samples S,

$$est_S(h) := \sum_{j=1}^{k} \sum_{z} \frac{n}{s} \cdot |S_{j,z}^{(h)}| \cdot \left|\gamma_j - (\tfrac{1}{1+\epsilon})^z\right|.$$

As a small remark, note that S here is treated as a sequence or multiset, so if the same element comes up twice in the sample, it is counted twice toward $|S_{j,z}^{(h)}|$.

Finally, for $\Gamma > 0$, let $H_k^{[\Gamma]}$ denote the subset of k-histograms such that each of their values γ_j is at most Γ, and such that each of their values γ_j is an integer multiple of ϵ/n. We will show that the histogram $f \in H_k^{[\epsilon^2/n^\alpha]}$ that minimizes the estimated error $est_S(f)$ is an approximately optimal solution to the original problem (i.e., it also minimizes the true error $err_P(\cdot)$ up to an additive error of at most ϵ), and can be found by dynamic programming.

We proceed to proving correctness, i.e., the approximate optimality of the returned histogram. The idea is that the random sample estimates the size of every sufficiently large $I_{j,z}^{(h)}$, while the smaller ones do not contribute much error anyway. Using those size estimates, and the fact that the histogram error on all points in a given interval is roughly the same, we can get a good estimate for its true cost.

More formally, fix $h \in H_k^{[\epsilon^2/n^\alpha]}$. As above, h defines $I_{j,z}^{(h)}$, and and together with the sample it also defines $S_{j,z}^{(h)}$. Note that the contribution of an interval $I_{j,z}^{(h)}$ to the true error $err_P(h)$ is $\sum_{i \in I_{j,z}^{(h)}} |p_i - \gamma_j|$, while its contribution to the estimated error $est_S(h)$ is $\frac{n}{s} \cdot |S_{j,z}^{(h)}| \cdot \left|(\frac{1}{1+\epsilon})^z - \gamma_j\right|$. To argue about the relation between those quantities, we classify the intervals $I_{j,z}^{(h)}$ into "heavy" and "light" ones, where an interval is heavy if $|I_{j,z}^{(h)}| \geq 10n^\alpha$, and light otherwise. Let $\mathcal{I}_H^{(h)}$ denote the set of heavy intervals, and $\mathcal{I}_L^{(h)}$ the light ones.

For a heavy interval, we can show that its contributions to the true and estimated errors are both roughly the same, with probability high enough for a union bound over all candidate solutions.

Lemma A.2 (heavy intervals)**.** *If $|I_{j,z}^{(h)}| \geq 10n^\alpha$, then with probability at least $1 - (\epsilon/n)^{O(k)}$,*

$$\left| \sum_{i \in I_{j,z}^{(h)}} |p_i - \gamma_j| - \frac{n}{s} \cdot |S_{j,z}^{(h)}| \cdot \left|(\tfrac{1}{1+\epsilon})^z - \gamma_j\right| \right| \leq \epsilon^2 + \epsilon \sum_{i \in I_{j,z}^{(h)}} p_i.$$

Proof. Since $|I_{j,z}^{(h)}| \geq 10n^\alpha$, by the Chernoff bound, our $s = O(n^{1-\alpha} \cdot k \log(n/\epsilon)$ samples suffice to have $||I_{j,z}^{(h)}| - \frac{n}{s}|S_{j,z}^{(h)}|| \leq n^\alpha$ with probability $1 - (\epsilon/n)^{O(k)}$. Furthermore, since for every $i \in I_{j,z}^{(h)}$ we have $p_i \in \left[(\frac{1}{1+\epsilon})^{z+1}, (\frac{1}{1+\epsilon})^z\right)$, then

$$|p_i - (\tfrac{1}{1+\epsilon})^z| \leq |(\tfrac{1}{1+\epsilon})^z - (\tfrac{1}{1+\epsilon})^{z+1}| = (1 - \tfrac{1}{1+\epsilon})(\tfrac{1}{1+\epsilon})^z \leq (1 - \tfrac{1}{1+\epsilon})(1+\epsilon)p_i = \epsilon \cdot p_i,$$

and therefore,

$$|p_i - \gamma_j| = |p_i - (\tfrac{1}{1+\epsilon})^z| \pm |(\tfrac{1}{1+\epsilon})^z - \gamma_j| = |(\tfrac{1}{1+\epsilon})^z - \gamma_j| \pm \epsilon \cdot p_i.$$

Together,

$$\begin{aligned}
\sum_{i \in I_{j,z}^{(h)}} |p_i - \gamma_j| &= \sum_{i \in I_{j,z}^{(h)}} \left(|(\tfrac{1}{1+\epsilon})^z - \gamma_j| \pm \epsilon \cdot p_i \right) \\
&= |I_{j,z}^{(h)}| \cdot |(\tfrac{1}{1+\epsilon})^z - \gamma_j| \pm \epsilon \sum_{i \in I_{j,z}^{(h)}} p_i \\
&= \left(\tfrac{n}{s}|S_{j,z}^{(h)}| \pm n^\alpha \right) \cdot |(\tfrac{1}{1+\epsilon})^z - \gamma_j| \pm \epsilon \sum_{i \in I_{j,z}^{(h)}} p_i.
\end{aligned}$$

As a result, the quantity from the lemma statement that we are trying to upper-bound is at most

$$n^\alpha \cdot |(\tfrac{1}{1+\epsilon})^z - \gamma_j| + \epsilon \sum_{i \in I_{j,z}^{(h)}} p_i.$$

For the first term, we recall that $\gamma_j \leq \epsilon^2/n^\alpha$ and $(\frac{1}{1+\epsilon})^z \leq \epsilon^2/n^\alpha$, thus $|(\frac{1}{1+\epsilon})^z - \gamma_j| \leq \epsilon^2/n^\alpha$ and the term is at most ϵ^2. □

This implies the following corollary, that bounds the difference between the true and estimated errors, as long as they are measured only on the heavy intervals.

Corollary A.3. *With probability at least 0.999, for all candidate solutions $h \in H_k^{[\epsilon^2/n^\alpha]}$ simultaneously, we have*

$$\left| \sum_{I_{j,z}^{(h)} \in \mathcal{I}_H^{(h)}} \sum_{i \in I_{j,z}^{(h)}} |p_i - \gamma_j| - \sum_{I_{j,z}^{(h)} \in \mathcal{I}_H^{(h)}} \sum_{i \in I_{j,z}^{(h)}} \frac{n}{s} \cdot |S_{j,z}^{(h)}| \cdot \left|(\tfrac{1}{1+\epsilon})^z - \gamma_j\right| \right| \leq O(\epsilon \cdot k \log(n/\epsilon)).$$

Proof. Recall that $H_k^{[\epsilon^2/n^\alpha]}$ is the set of k-histograms with values γ_j which are multiple integers of ϵ/n in the range $[0, \epsilon^2/n^\alpha]$. Thus, $\left|H_k^{[\epsilon^2/n^\alpha]}\right| = \binom{n}{k} \cdot (\frac{n}{\epsilon})^k \leq (n/\epsilon)^{2k}$. So in Lemma A.2 we can take a union bound over all heavy intervals induced by all $h \in H_k^{[\epsilon^2/n^\alpha]}$. As a result, the absolute difference between the two terms in the statement of the corollary is upper-bounded by

$$\epsilon^2 |\mathcal{I}_H^{(h)}| + \epsilon \sum_{I_{j,z}^{(h)} \in \mathcal{I}_H^{(h)}} \sum_{i \in I_{j,z}^{(h)}} p_i.$$

The first term is at most $O(\epsilon k \log(n/\epsilon))$ since the number of intervals is $O(\epsilon^{-1} k \log(n/\epsilon))$. For the second term, note that the sum is over all i that reside in heavy intervals, and we can upper-bound it by the sum over all items, $\epsilon \sum_{i=1}^n p_i = \epsilon$. □

Next, light intervals. Their total *true* contribution is small, just by being light. Their total *estimated* contribution is also small, with more modest probability. Recall that $\mathcal{I}_L^{(h)}$ denotes the set of light intervals.

Lemma A.4 (light intervals)**.** *Fix any single $h \in H_k^{[\epsilon^2/n^\alpha]}$. With probability at least 0.999,*

$$\sum_{I_{j,z}^{(h)} \in \mathcal{I}_L^{(h)}} \left| \sum_{i \in I_{j,z}^{(h)}} |p_i - \gamma_j| - \frac{n}{s} \cdot |S_{j,z}^{(h)}| \cdot \left|(\tfrac{1}{1+\epsilon})^z - \gamma_j\right| \right| \leq O(\epsilon \cdot k \log(n/\epsilon)).$$

Proof. Since a light interval $I_{j,z}^{(h)} \in \mathcal{I}_L^{(h)}$ satisfies $|I_{j,z}^{(h)}| \leq 10n^\alpha$, and since $|p_i - \gamma_j| \leq \epsilon^2/n^\alpha$ (regardless of the lightness of the interval), then

$$\sum_{I_{j,z}^{(h)} \in \mathcal{I}_L^{(h)}} \sum_{i \in I_{j,z}^{(h)}} |p_i - \gamma_j| \leq 10\epsilon^2 \cdot |\mathcal{I}_L^{(h)}|.$$

Furthermore, since $\mathbb{E}[\frac{n}{s}|S_{j,z}^{(h)}|] = |I_{j,z}^{(h)}| \leq 10n^\alpha$ and (deterministically) $\left|(\frac{1}{1+\epsilon})^z - \gamma_j\right| \leq \epsilon^2/n^\alpha$,

$$\mathbb{E}\left[\sum_{I_{j,z}^{(h)} \in \mathcal{I}_L^{(h)}} \frac{n}{s} \cdot |S_{j,z}^{(h)}| \cdot \left|(\tfrac{1}{1+\epsilon})^z - \gamma_j\right|\right] \leq 10\epsilon^2 \cdot |\mathcal{I}_L^{(h)}|.$$

By Markov's inequality, the probability the latter random variable does not exceed $10000\epsilon^2 \cdot |\mathcal{I}_L^{(h)}|$ is at least 0.999. The lemma is implied by noticing that there are at most $k \log_{(1+\epsilon)}(\epsilon^2/n^\alpha) = O(k\epsilon^{-1} \log(n/\epsilon))$ intervals, so $|\mathcal{I}_L^{(h)}| \leq O(k\epsilon^{-1} \log(n/\epsilon))$. □

Now we can prove the theorem with the following two claims.

Claim A.5. *With probability at least 0.999, for all candidate solutions $h \in H_k^{[\epsilon^2/n^\alpha]}$ simultaneously, we have*

$$est_S(h) \geq err_P(h) - O(\epsilon \cdot k \log(n/\epsilon)).$$

Proof. On one hand,

$$\begin{aligned}
est_S(h) &= \sum_{I_{j,z}^{(h)}} \frac{n}{s} \cdot |S_{j,z}^{(h)}| \cdot \left|(\tfrac{1}{1+\epsilon})^z - \gamma_j\right| \\
&\geq \sum_{I_{j,z}^{(h)} \in \mathcal{I}_H^{(h)}} \sum_{i \in I_{j,z}^{(h)}} \frac{n}{s} \cdot |S_{j,z}^{(h)}| \cdot \left|(\tfrac{1}{1+\epsilon})^z - \gamma_j\right| && \text{restricting to heavy intervals} \\
&\geq \sum_{I_{j,z}^{(h)} \in \mathcal{I}_H^{(h)}} \sum_{i \in I_{j,z}^{(h)}} |p_i - \gamma_j| - O(\epsilon \cdot k \log(n/\epsilon)) && \text{Corollary A.3.}
\end{aligned}$$

On the other hand,

$$err_P(h) = \sum_{I_{j,z}^{(h)} \in \mathcal{I}_H^{(h)}} \sum_{i \in I_{j,z}^{(h)}} |p_i - \gamma_j| + \sum_{I_{j,z}^{(h)} \in \mathcal{I}_L^{(h)}} \sum_{i \in I_{j,z}^{(h)}} |p_i - \gamma_j|,$$

and the second term (contribution of light intervals) was already upper-bounded by $10\epsilon^2|\mathcal{I}_L^{(h)}| = O(\epsilon k \log(n/\epsilon))$ in Lemma A.4, thus

$$err_P(h) \le \sum_{I_{j,z}^{(h)} \in \mathcal{I}_H^{(h)}} \sum_{i \in I_{j,z}^{(h)}} |p_i - \gamma_j| + O(\epsilon k \log(n/\epsilon)),$$

and the claim follows. □

Claim A.6. *For the optimal solution $h^* \in H_k^{[\epsilon^2/n^\alpha]}$,*

$$est_S(h^*) \le err_P(h^*) + O(\epsilon \cdot k \log(n/\epsilon)).$$

Proof. We break up the error contribution into heavy and light intervals as usual. Corollary A.3 tells us that the heavy contribution is the same in the true and estimated errors up to $\pm O(\epsilon \cdot k \log(n/\epsilon))$, for every $h \in H_k^{[\epsilon^2/n^\alpha]}$. Lemma A.4 tells us that the light contribution is also the same up to $\pm O(\epsilon \cdot k \log(n/\epsilon))$ (the difference is that in the light lemma we only have enough probability to ensure this for any single $h \in h \in H_k^{[\epsilon^2/n^\alpha]}$, so we use it for the optimum h^*). Together we have

$$|est_S(h^*) - err_P(h^*)| \le O(\epsilon \cdot k \log(n/\epsilon)),$$

which is stronger than the claim. □

The two claims together show that returning $\hat{h} \in H_k^{[\epsilon^2/n^\alpha]}$ that minimizes the estimated error is almost as good as returning the one that minimizes the true error:

$$\begin{aligned} err_P(\hat{h}) &\le est_S(\hat{h}) + O(\epsilon \cdot k \log(n/\epsilon)) && \text{Claim A.5} \\ &\le est_S(h^*) + O(\epsilon \cdot k \log(n/\epsilon)) && \text{optimality of } \hat{h} \text{ w.r.t. estimated error} \\ &\le err_P(h^*) + O(\epsilon \cdot k \log(n/\epsilon)) && \text{Claim A.6.} \end{aligned}$$

So we return a solution with optimal value (in the discretized set of histograms $H_k^{[\epsilon^2/n^\alpha]}$) up to an additive loss of $O(\epsilon \cdot k \log(n/\epsilon))$. We can scale ϵ down by $O(k \log(n/\epsilon))$ as mentioned in the beginning of the proof. Finally it remains to observe that the discretization of $H_k^{[\epsilon^2/n^\alpha]}$ into integer multiples of ϵ/n doesn't matter since we lose only ϵ/n per $i \in [n]$ compared to non-discretized optimum, so only an additional ϵ. Theorem A.1 is proven. □

B. Proof of one-pass lower bound

In this section we prove Theorem 4.3. For clarity of presentation, we begin by proving the weaker version where the output histogram f is allowed to use only 2 pieces (Theorem B.1). Afterwards, in Appendix B.1, we will show how to extend the proof and obtain the same lower bound for f that can use as many as $O(\sqrt{n})$ pieces.

Theorem B.1. *Let $\epsilon > 0$ be a sufficiently small constant. Any one-pass streaming algorithm that outputs a 2-piece histogram f such that $err_P(f) \le \min_{f^* \in H_2} err_P(f^*) + \epsilon$, must use $\Omega(\sqrt{n})$ space.*

Proof. The proof is by reduction from the Augmented Indexing problem. We recall that Indexing is a one-way communication problem where Alice's input is a bitstring $a_1, ..., a_t$, Bob's input is $j \in [t]$, Alice sends one message to Bob, and Bob needs to report a_j with probability better than $1/2$. In the Augmented Indexing variant, Bob also gets $a_1, \ldots, a_{j-1}$ as part of his input. Both variants are known to require $\Omega(t)$ communication. We set $t = \sqrt{n}$.

The reduction Our universe is $1, \ldots, 3n$ where for simplicity n is an integer square. The counts of all items $i = 2n+1, \ldots, 3n$ are always set to 1. We partition $1, ..., 2n$ into $2\sqrt{n}$ equal-size consecutive intervals of length $\sqrt{n}$, and denote them as $A_1, B_1, A_2, B_2, \ldots, A_{\sqrt{n}}, B_{\sqrt{n}}$. We think of the A_j's as Alice's cells and of the B_j's as Bob's cells.

Alice's reduction: Given her input, for every $j = 1, \ldots, \sqrt{n}$, Alice sets all elements in A_j to a_j. So, every A_j contains either no supported elements (if $a_j = 0$) or $\sqrt{n}$ mice (if $a_j = 1$). She does it by streaming the appropriate updates into the streaming algorithm, and then sends the memory state to Bob.

Bob's reduction: Bob streams into the memory state the following updates:

1. He sets all elements in every $A_1, ..., A_{j-1}$ to zeros (he knows where the nonzeros are by the Augmented part of Augmented Indexing).
2. In each $B_1, \ldots, B_{j-1}$, he sets $\frac{\sqrt{n}}{j-1}$ of the elements to have mass $\sqrt{n}$ (elephants).
3. In B_j, he sets $\gamma\sqrt{n}$ of the elements to have mass $\sqrt{n}$ (elephants), where $\gamma > \epsilon$ is a small constant.

This concludes the stream. Note that the total count of masses is $O(n)$, the algorithm is guaranteed to return an optimal 2-histogram up to an additive error of ϵn. Now Bob lets the algorithm find an approximately best 2-histogram. Wlog, the values of the histogram are either $\sqrt{n}$ or 1, since these are the only frequencies in the input. If at least half of the elements in A_j are given histogram value 1, Bob reports 1, and reports 0 otherwise.

To show the correctness of the communication protocol, first note that the algorithm has to put one piece with value $\sqrt{n}$ on the prefix (in order to cover the elephants in $B_1, \ldots, B_{j-1}$) and another piece with value 1 on the suffix (to cover the mice in $2n + 1 \ldots, 3n$), as otherwise the error is at least $\approx n$, which as we will see momentarily is much larger than the optimum in either case. So the question is where it places the breakpoint between the two pieces.

Consider the case $a_j = 0$. In this case, the optimal solution has zero error: we can cover $A_1, B_1, \ldots, A_j, B_j$ with the $\sqrt{n}$-piece, and the rest with the 1-piece. So, the algorithm must move from the $\sqrt{n}$-piece to the 1-piece **after** B_j (and in particular A_j needs to be covered by the $\sqrt{n}$-piece), since otherwise, it incurs error $\approx \gamma n$ on B_j, which is more than ϵn.

Consider the case $a_j = 1$. In this case, the optimal solution has error $\approx \gamma n$: we can cover $A_1, B_1, \ldots, A_{j-1}, B_{j-1}$ with the $\sqrt{n}$-piece, and the rest with the 1-piece, so that the only error we incur is $\approx \gamma n$ on B_j. So, the algorithm must move from the $\sqrt{n}$-piece to the 1-piece **before** at least half of A_j (and in particular, at least half of A_j needs to be covered by the 1-piece), since otherwise, it incurs error $\approx \frac{1}{2}n$ on A_j, and the gap from the optimal error $\approx \gamma n$ is more than $\approx \gamma n$ if $\gamma < \frac{1}{4}$.

So, as long as the algorithm succeeds with probability more than 0.5, the theorem is proven. □

B.1. Proof of Theorem 4.3

Proof. The proof is patterned by the above proof of Theorem B.1, with certain modifications to accommodate the much larger of pieces in the output histogram f.

The reduction Assume that we are given an algorithm that satisfies the guarantees in the theorem statement. Given an instance of Augmented Indexing over $\sqrt{n}$ bits, we will show how to solve the problem via the streaming algorithm on a stream over domain size n.

Assume for simplicity that n is a perfect square. We will split the domain of the stream into $\sqrt{n}$ contiguous chunks of size $\sqrt{n}$ called $S_1, \ldots, S_{\sqrt{n}}$. For some constant $b \in [0, 1]$ which we will later define, we will split each chunk S_i into $b\sqrt{n}$ equal size subintervals $S_i^1, \ldots, S_i^{b\sqrt{n}}$. The first index of each subinterval will be reserved for Bob and the rest will be reserved for Alice.

Alice's reduction Alice will go through her $\sqrt{n}$ bits of her string x and perform the following stream operations:

- If $x_i = 0$, do nothing.
- If $x_i = 1$, add $a\sqrt{n}$ mice to chunk S_i for some constant $a \in [0, 1]$ in the following way. For each subinterval of S_i, pick $\frac{a}{b}\sqrt{n}$ indices other than the first index of the subinterval. Add a singleton element to the stream corresponding to each of these indices.

The result on the frequency distribution will be that for all of Alice's bits which equal 1, there will be $a\sqrt{n}$ elements with mass $1/m$ if m is the total mass of the stream.

Bob's reduction Bob has a single index i of interest as well as knowledge of $x_1, \ldots, x_{i-1}$. Bob will execute the following stream updates.

- For $j = 1, \ldots, i-1$, Bob will use his knowledge of Alice's bits to delete any elements Alice added in the chunks $S_1, \ldots S_{i-1}$.
- For each subinterval of S_i, Bob will add $\sqrt{n}$ copies of the first index of the subinterval.

The result on the frequency distribution will be that there will be zero mass on $S_1, \ldots S_{i-1}$ and there will be $b\sqrt{n}$ elements with mass $\sqrt{n}/m$ in chunk S_i.

After running the streaming algorithm, we will get some histogram approximation of the frequency distribution. In order to use this to solve the indexing problem, we will do the following post-processing step. Let $c \in [0, 1]$ be some constant with $c < b$ and let the first indices of each of the subintervals of S_i be refered to as as Bob's indices.

- If at least $c\sqrt{n}$ of Bob's indices are approximated to have mass at least $\frac{1}{2\sqrt{n}}$, then report that $x_i = 1$.
- If at least $c\sqrt{n}$ of Bob's indices are approximated to have mass less than $\frac{1}{2\sqrt{n}}$, then report that $x_i = 0$.

Note that the total mass of the stream $m = \theta(n)$. We will proceed by cases to show that if the streaming algorithm has the $\varepsilon + opt_2(S)$ error guarantee in the theorem statement, then Bob will correctly recover Alice's ith bit. For now, we will parameterize the number of pieces the streaming algorithm produces as $k\sqrt{n}$ for some constant k.

Case 1: $x_i = 0$**.** In this case, $opt_2(S) = 0$ as we can simply have the first piece of the histogram predict mass $\frac{1}{\sqrt{n}}$ for chunks $S_1, \ldots, S_i$ and predict mass $\frac{1}{n}$ for chunks $S_{i+1}, \ldots, S_{\sqrt{n}}$. As Alice and Bob's elements do not overlap, two pieces suffice to perfectly approximate the frequency distribution.

Assume that at least $c\sqrt{n}$ of Bob's indices are approximated to have mass less than $\frac{1}{2\sqrt{n}}$. we will show that this implies that ε must be large. In this case, the error incurred by the streaming algorithm will be

$$\left(\frac{1}{\sqrt{n}} - \frac{1}{2\sqrt{n}}\right) c\sqrt{n} = \frac{c}{2}.$$

So, if $\varepsilon < \frac{c}{2}$, then the reduction will correctly identify when $x_i = 0$ as many of Bob's indices must be predicted to have large mass.

Case 2: $x_i = 1$**.** In this case, consider the two piece histogram that simply uses 1 piece and predicts mass $1/n$ everywhere. This gives an upper bound on $opt_2(S)$ (and for our setting of a, b should actually be optimal).

$$opt_2(S) \leq \left(\frac{1}{\sqrt{n}} - \frac{1}{n}\right) b\sqrt{n} \leq b$$

Assume that at least $c\sqrt{n}$ of Bob's indices are approximated to have mass at least $\frac{1}{2\sqrt{n}}$. We will show that this implies ε must be large. Note that in this case, at least $(c-k)\sqrt{n}$ of Bob's indices must be covered by a histogram piece with height at least $\frac{1}{2\sqrt{n}}$ that also covers some other of Bob's indices (assuming $c > k$). Thus, there must be $(c-k)\sqrt{n} \cdot \frac{a}{b}$ of Alice's elements which are predicted to have mass at least $\frac{1}{2\sqrt{n}}$. So, the error of the streaming algorithm's approximation is at least

$$\left(\frac{1}{2\sqrt{n}} - \frac{1}{n}\right)\left(\frac{a}{b}\right)(c-k)\sqrt{n} = \frac{a(c-k)}{2b} - o(1).$$

As we are guaranteed that the stream error is at most $opt_2(S) + \varepsilon$, the reduction will correctly identify $x_i = 1$ as long as

$$\varepsilon < \frac{a(c-k)}{2b} - b.$$

It remains to give reasonable settings for the constants $a, b, c, k \in [0, 1]$ under the following constraints

- $a + b \leq 1$
- $a/b \in \mathbb{Z}$
- $b > c$
- $c > k$
- $\varepsilon < \min\{\frac{c}{2}, \frac{a(c-k)}{2b} - b\}$

Setting $a = \frac{1}{2}, b = \frac{1}{8}, c = \frac{1}{10}, k = \frac{1}{40}$ satisfies all of the conditions. Then, the bound on ε becomes

$$\varepsilon < \min\{\frac{1}{20}, \frac{1}{40}\} = \frac{1}{40}.$$

For this parameter regime, any streaming algorithm on domain size n that outputs a histogram approximation with at most $\sqrt{n}/40$ pieces with error $opt_2(S) + \varepsilon$ for $\varepsilon < 40$ can be used to solve augmenting indexing on $\sqrt{n}$ bits and thus must use $\Omega(\sqrt{n})$ space.

□

C. Proof of Lemma 5.5

Proof of Lemma 5.5. Without loss of generality, assume n is even. Let $X_1, \ldots, X_s$ be random variables s.t. X_i corresponds to the event that the ith random sample is less than $x_{n/2-\delta n}$ for some $\delta \in [0, 1/2]$. Let $S = \sum_{i=1}^{s} X_i$ be the number of samples that are less than $x_{n/2-\delta n}$. $\hat{M}_s \leq x_{n/2-\delta n}$ if and only if $S \geq s/2$. By Hoeffding bound,

$$\begin{aligned} \Pr(\hat{M}_s \leq x_{n/2-\delta n}) &= \Pr(S \geq s/2) \\ &= \Pr(S \geq \mathbb{E}[S] + \delta s) \leq e^{-2\delta^2 s}. \end{aligned}$$

The same bound holds for bounding the probability that $\hat{M}_s \geq x_{n/2-\delta n}$. So, with probability at least $1 - 2e^{-2\delta^2 s}$, the sample median is within δn of the rank of the true median.

Assume that $\hat{M}_s \in [x_{n/2-\delta n}, x_{n/2+\delta n}]$. Then, the loss of M^* and $\hat{M}_s$ is equivalent for $i \in [1, n/2 - \delta n] \cup [n/2 + \delta n, n]$:

$$\begin{aligned} &\sum_{i=1}^{n/2-\delta n} |x_i - \hat{M}_s| + \sum_{i=n/2+\delta n}^{n} |x_i - \hat{M}_s| \\ &= \sum_{i=1}^{n/2-\delta n} (\hat{M}_s - x_i) + \sum_{i=n/2+\delta n}^{n} (x_i - \hat{M}_s) \\ &= \sum_{i=1}^{n/2-\delta n} (M^* - x_i - M^* + \hat{M}_s) \\ &\quad + \sum_{i=n/2+\delta n}^{n} (x_i - M^* + M^* - \hat{M}_s) \\ &= (n/2 - \delta n)(-M^* + \hat{M}_s + M^* - \hat{M}_s) \\ &\quad + \sum_{i=1}^{n/2-\delta n} (M^* - x_i) + \sum_{i=n/2+\delta n}^{n} (x_i - M^*) \\ &= \sum_{i=1}^{n/2-\delta n} |x_i - M^*| + \sum_{i=n/2+\delta n}^{n} |x_i - M^*|. \end{aligned}$$

The additional loss of $\hat{M}_s$ over M^* on $i \in (n/2 - \delta n, n/2 + \delta n)$ is at most the mass of the elements in this range. The mass of these elements is at most a $\frac{2\delta n}{n/2+\delta n}$ fraction of the total mass. Therefore,

$$\ell(\hat{M}_s) - \ell(M^*) \leq \frac{2\delta n}{n/2 + \delta n}\beta \leq 4\delta\beta.$$

Setting $\delta = \varepsilon/4$ completes the proof. □

D. Additional Experiments: Varying k with Space

Both of our algorithms are improper–while their guarantees are in terms of k-piece histograms, they in fact use more than k pieces. While this is standard in histogram approximation, from an empirical standpoint, it is important to understand whether the gains of our algorithms could be achieved by baselines which simply use more pieces. To that end, we present the same experiments as in Section 6 but now with the "Fixed" baselines also using $k = Space/3$ or $k = Space/20$. Our algorithms as well as the solid line baselines use $k = 5$.

For the Taxi dataset, these many-piece fixed algorithms indeed perform very well, but for the CAIDA, War & Peace, and McDonalds datasets, our algorithms still outperform these baselines (for McDonalds, the baselines match the performance of our one pass algorithm). The Taxi dataset has no heavy hitters and is thus an easy case for the fixed baselines. On datasets with heavy hitters or less uniform structure, these baselines do not perform as well as our algorithms.

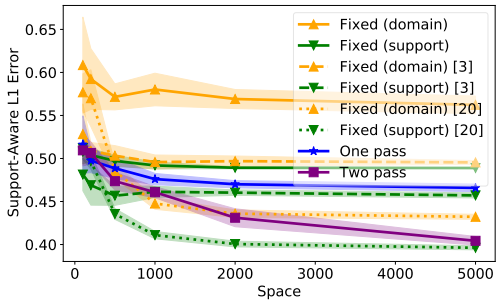

Figure 6. Comparison of support-aware $L1$ error with varying space usage with various k on the Taxi dataset. Shading indicates one standard deviation over 10 trials.

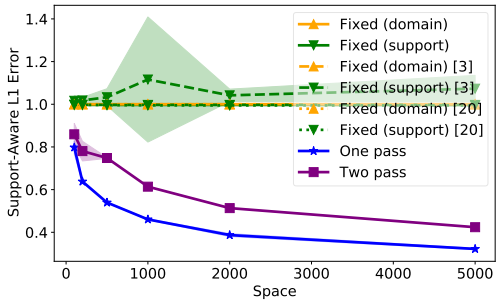

Figure 7. Comparison of support-aware $L1$ error with varying space usage with various k on the CAIDA dataset. Shading indicates one standard deviation over 10 trials.

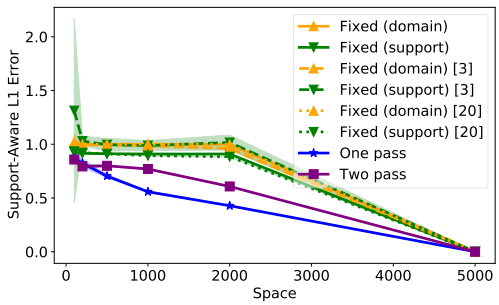

Figure 8. Comparison of support-aware $L1$ error with varying space usage with various k on the War and Peace dataset. Shading indicates one standard deviation over 10 trials.

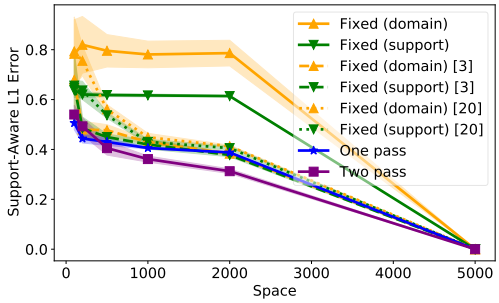

Figure 9. Comparison of support-aware $L1$ error with varying space usage with various k on the McDonalds dataset. Shading indicates one standard deviation over 10 trials.

E. Additional Experiments: $k = 10$

In this section, we display additional experimental results. In Section 6, we compare our algorithms against several baselines with $k = 5$. Here, we present the same experiments with $k = 10$. There is qualitatively little difference between the performance of any of the algorithms or baselines with $k = 5$ compared to $k = 10$.

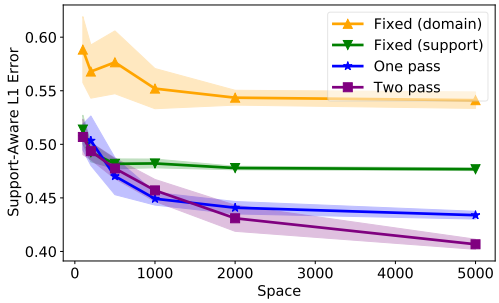

Figure 10. Comparison of support-aware $L1$ error with varying space usage with $k = 5$ on the Taxi dataset. Shading indicates one standard deviation over 10 trials.

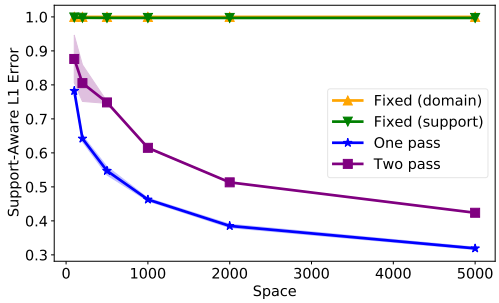

Figure 11. Comparison of support-aware $L1$ error with varying space usage with $k = 5$ on the CAIDA dataset. Shading indicates one standard deviation over 10 trials.

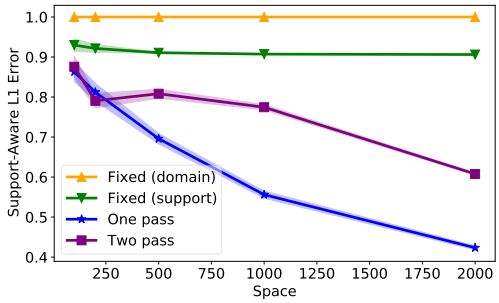

Figure 12. Comparison of support-aware $L1$ error with varying space usage with $k = 5$ on the War and Peace dataset. Shading indicates one standard deviation over 10 trials.

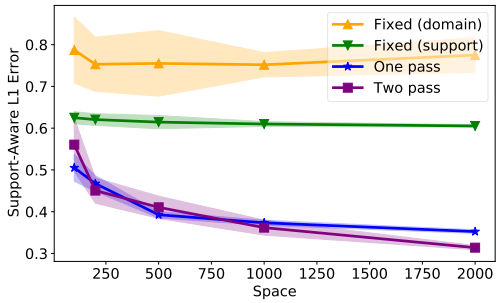

Figure 13. Comparison of support-aware $L1$ error with varying space usage with $k = 5$ on the McDonalds dataset. Shading indicates one standard deviation over 10 trials.