Estimation of Entropy in Constant Space with Improved Sample Complexity

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent work of Acharya et al. (NeurIPS 2019) showed how to estimate the entropy of a distribution $\mathcal{D}$ over an alphabet of size k up to $\pm\epsilon$ additive error by streaming over $(k/\epsilon^3) \cdot \mathrm{polylog}(1/\epsilon)$ i.i.d. samples and using only $O(1)$ words of memory. In this work, we give a new constant memory scheme that reduces the sample complexity to $(k/\epsilon^2) \cdot \mathrm{polylog}(1/\epsilon)$. We conjecture that this is optimal up to $\mathrm{polylog}(1/\epsilon)$ factors.

1 Introduction

In the field of *streaming algorithms*, an algorithm makes one pass (or few passes) over a database while using memory sublinear in the data it sees to then answer queries along the way or at the data stream's end. Researchers have developed various algorithms, as well as memory lower bounds, for such problems for over four decades [MP80, MG82, AMS99]. For the vast majority of research in the field, the database is assumed to be fixed, and algorithms are then analyzed through the lens of worst case analysis.

In this work, we look to further develop the relationship between streaming algorithms and statistics, specifically studying statistical inference through low-memory streaming algorithms. In this setup, rather than processing a worst-case instance of some fixed database, our input is instead a *distribution* $\mathcal{D}$, and our algorithm processes i.i.d. samples from $\mathcal{D}$ with the goal of inferring its properties. Natural questions then arise, such as understanding the tradeoffs between sample complexity, memory, accuracy, and confidence, or even understanding whether a low-memory algorithm exists at all for a particular inference problem even if we allow the streaming algorithm to draw an unlimited number of samples. Work on streaming algorithms for statistical inference problems began in [GM07], which studied nonparameteric distribution learning, followed by the work of [CLM10], studying low-memory streaming algorithms for use in robust statistics and distribution property testing. Interest in the area later exploded off after work of [SVW16], which explicitly raised the question of whether low memory might place fundamental limits on learning rates, with a flurry of works proving such limitations in response [MM17, KRT17, Raz17, GRT18, SSV19, GRT19, GKR20, GKLR21], starting with a work of [Raz19] on memory/sample tradeoff lower bounds for learning parities ($\mathcal{D}$ generates $(x, \langle w, x\rangle)$ for x uniform in the hypercube with w an unknown parameter, and the goal is to learn w).

In this work, following [ABIS19], we focus specifically on the problem of estimating the entropy of an unknown distribution $\mathcal{D}$ over $\{1, \ldots, k\}$ using a low-memory streaming algorithm over i.i.d. samples. It is known that to estimate the entropy up to ϵ additive error with large constant success probability, without memory constraints the optimal sample complexity is

$$n = \Theta\left(\max\left\{\frac{1}{\epsilon}\frac{k}{\log(k/\epsilon)}, \frac{\log^2 k}{\epsilon^2}\right\}\right)$$

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

[VV17, VV11, JVHW15, WY16]. Prior work by [BDKR05] also shows that a sublinear number of samples is possible for multiplicative approximation of entropy for distributions whose entropy is sufficiently large. The known optimal algorithms from prior work, however, must remember all samples and hence use $\Omega(n)$ words of memory[1]. The algorithm of [ABIS19] uses only $O(1)$ words of memory, though at the cost of requiring an increased sample complexity of $k \cdot \tilde{O}(1/\epsilon^3)$[2]. In this work, our goal is to address the question: *to what extent was the worsening of sample complexity in previous work necessary to achieve constant memory?*

Our Contribution. We show that using $O(1)$ words of memory[3], it is possible to obtain a sample complexity of $k \cdot \tilde{O}(1/\epsilon^2)$, which is an improvement over the previous memory-efficient sample complexity bound which had cubic dependence on $1/\epsilon$. The starting point of our algorithm revisits a simple estimator proposed by [ABIS19]. Their simple estimator uses $O(k \log^2(k/\epsilon)/\epsilon^3)$ samples to estimate the entropy in constant space. Our novel contribution is a modification which estimates a bias incurred by the estimator; this change allows us to use only $O(k \log^2 k \log^2(1/\epsilon)/\epsilon^2)$ samples. With the simple estimator with improved sampled complexity in hand, we show how an "interval-based" algorithm, similar to the one in [ABIS19], improves the dependence on k to $k \cdot \tilde{O}(1/\epsilon^2)$.

We remark that there has been other work on estimating entropy in the data streaming model [BG06, CCM10, HNO08], but those works are qualitatively different from our own current work and that of [ABIS19]. Specifically, they take the worst case point of view, where the stream items are not drawn i.i.d. from a distribution, but rather the stream itself is viewed as a worst-case input and the goal is to estimate its empirical entropy. In that model, $O(1)$ memory algorithms for $\pm\epsilon$ additive estimation to entropy provably do not exist, as there is a known memory lower bound of $\Omega(1/(\log^2(1/\epsilon)\epsilon^2))$ [CCM10].

Overview of Approach. We start by describing the basic algorithm of [ABIS19]. Their basic estimator takes a single random sample $\boldsymbol{i} \sim \mathcal{D}$, followed by N more i.i.d. samples. Then, they define $\boldsymbol{N}_x$ to be the number of these N samples equal to $\boldsymbol{i}$. The estimate $\hat{p}_{\boldsymbol{i}} := \boldsymbol{N}_x/N$ is an unbiased estimator of of the probability $p_{\boldsymbol{i}}$ of $\boldsymbol{i}$ according to $\mathcal{D}$, and for large N, $\log(1/\hat{p}_{\boldsymbol{i}})$ is a reasonable estimator for the entropy $H = H(\mathcal{D}) = \mathbf{E}[\log(1/p_{\boldsymbol{i}})]$ of $\mathcal{D}$. One can then average many such independent estimates. There is an additional technical detail, that $\hat{p}_{\boldsymbol{i}}$ may be zero (if $\boldsymbol{N}_x$ is zero), which is fixed via a "one-smoothing" trick of actually setting $\hat{p}_{\boldsymbol{i}} := (\boldsymbol{N}_x + 1)/N$ (which introduces an acceptably small amount of additional bias when N is sufficiently large).

Our improvement begins with the observation that $\log(1/\hat{p}_{\boldsymbol{i}})$ is *not* an unbiased estimator for H. We first propose a similar but different estimator to the previous simple estimator. We also begin by taking a random sample $\boldsymbol{i} \sim \mathcal{D}$; however, rather than letting $\boldsymbol{N}_x$ be sampled from the binomial distribution $\mathsf{Bin}(N, p_{\boldsymbol{i}})$, we sample a negative binomial random variable $\boldsymbol{X}$, which is the number of additional draws to see i exactly t more times (t is a parameter of the algorithm). Henceforth we let $\mathsf{NB}(t, p)$ denote such a negative binomial random variable, where the underlying Bernoulli experiment has success probability p. Then $\mathbf{E}[\boldsymbol{X}] = t/p_{\boldsymbol{i}}$, and we will use $\log(\boldsymbol{X}/t)$ as a reasonable estimate of $\log(1/p_{\boldsymbol{i}})$. This estimator is also biased, but we can correct for this bias using a few more samples.

Specifically, let $\boldsymbol{Y} = \boldsymbol{X}p_{\boldsymbol{i}}/t$ and consider the degree-r Taylor expansion of our estimate $\log(\boldsymbol{X}/t)$ and the ideal quantity $\log(1/p_{\boldsymbol{i}})$. As it will turn out, the expectation of the degree-r Taylor expansion of $\log(\boldsymbol{X}/t) - \log(1/p_{\boldsymbol{i}}) = \log \boldsymbol{Y}$ is a degree-r polynomial in $p_{\boldsymbol{i}}$. By drawing r additional samples, we may design an estimator for this polynomial, and subtract it from $\log(\boldsymbol{X}/t)$. Correcting some of the bias in this way gives us our improved estimate for $\log(1/p_{\boldsymbol{i}})$. Our analysis of this scheme shows that a sample complexity of $(k/\epsilon^2) \cdot \mathrm{polylog}(k/\epsilon)$ suffices. We then describe and analyze an improved algorithm in Section 3, which achieves $(k/\epsilon^2) \cdot \mathrm{polylog}(1/\epsilon)$ sample complexity by additionally incorporating a "bucketing" scheme, similar to one proposed in [ABIS19]. The idea is to

[1] As in prior work, we use a "word", or "machine word", to denote a unit of memory that can hold $\Theta(\log(k/\epsilon))$ bits. Essentially, a machine word is large enough to hold the name of an item in the alphabet, as well as the value of ϵ.

[2] We use $\tilde{O}(f)$ to denote a function which is $O(f \cdot \mathrm{poly}(\log f))$

[3] More precisely, we provide a uniform algorithm which given any k, ϵ generates a program with source code of size $O(\log\log(1/\epsilon))$ words, and that fixed program can then process any stream in $O(1)$ words of working memory; see Section B for details.

Subroutine `LogEstimator`$(\mathcal{D}, i)$

Input: Sample access to a distribution $\mathcal{D}$ supported on $[k]$, an index $i \in [k]$ where $p_i \neq 0$.
Output: A number $\boldsymbol{\eta} \in \mathbb{R}_{\geq 0}$, which is our bias estimate.

1. We draw enough samples from $\mathcal{D}$ so that i is sampled exactly t times, and let $\boldsymbol{X} \in \mathbb{N}$ denote the number of samples taken.
2. For $r \in \mathbb{N}$, let $f\colon \mathbb{R} \to \mathbb{R}$ denote the degree-r Taylor expansion of $\log z$ centered at 1, and $h_t\colon [0,1] \to \mathbb{R}$ be the degree-r polynomial satisfying
$$h_t(\rho) = \mathbf{E}_{\boldsymbol{Z} \sim \mathrm{NB}(t,\rho)}\left[f\left(\frac{\boldsymbol{Z} \cdot \rho}{t}\right)\right].$$
Finally, $g\colon [0,1]^r \to \mathbb{R}$ is the linear function with $g(\rho, \rho^2, \ldots, \rho^r) = h_t(\rho)$. We take r additional independent samples from $\mathcal{D}$, and for $j \in [r]$, we let $\boldsymbol{B}_j$ be the indicator random variable that the first j samples were all i. Note that $\{\boldsymbol{B}_j\}_{j \in [r]}$ can be encoded using a single counter requiring $\log r$ bits.
3. We return
$$\boldsymbol{\eta} \stackrel{\text{def}}{=} \log\left(\frac{\boldsymbol{X}}{t}\right) - g\left(\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_r\right).$$

Figure 1: Description of the estimator for $\log(1/p_i)$.

partition the possibilities for values of $\boldsymbol{X}$ into disjoint intervals $I_\ell = [b_{\ell-1}, b_\ell)$ for $\ell = 1, 2, \ldots, L$ and optimized choices of the breakpoints b_ℓ, then estimate both $\mathbf{Pr}[\boldsymbol{X} \in I_\ell]$ and the conditional contributions to entropy conditioned on $\boldsymbol{X} \in I_\ell$ for each ℓ. By estimating separately for each I_ℓ, one can show that the conditional variance is reduced to obtain an overall smaller sample complexity of $(k/\epsilon^2) \cdot \mathrm{polylog}(1/\epsilon)$, a strict improvement over that of [ABIS19]; details are in Section 3.

2 A Simple Algorithm and Analysis

Let $k \in \mathbb{N}$, and $\mathcal{D}$ be an unknown distribution supported on $[k]$. For any $i \in [k]$, we denote the probability that $i \in [k]$ is sampled by $\mathcal{D}$ as p_i. The goal is to design a low-space streaming algorithm which receives independent samples from $\mathcal{D}$ and outputs an estimate to the entropy:

$$H(\mathcal{D}) \stackrel{\text{def}}{=} \sum_{i=1}^{k} p_i \log\left(\frac{1}{p_i}\right) = \mathbf{E}_{\boldsymbol{i} \sim \mathcal{D}}\left[\log\left(\frac{1}{p_{\boldsymbol{i}}}\right)\right],$$

where logarithms above and throughout this paper are base-2, unless otherwise stated.

2.1 An Estimator for $\log(1/p_{\boldsymbol{i}})$

As mentioned in Section 1, similarly to [ABIS19] the algorithm aims to estimate $H(\mathcal{D})$ by taking a sample $\boldsymbol{i} \sim \mathcal{D}$ and estimating $\log(1/p_{\boldsymbol{i}})$. Then, averaging these estimates will give an estimator for $H(\mathcal{D})$ (albeit with a super-linear dependence on k, which we fix in Section 3). We describe the estimator in Figure 1.

There are three main steps in the analysis. In the first, we show that the estimator has small bias. The second is showing that the above estimator has low variance. Finally, we show that the estimator may be computed with few bits. In Figure 1, we set $r = \Theta(\log(1/\epsilon))$ and $t = \Theta(\log^2(1/\epsilon))$ to obtain an estimator whose bias is at most ϵ and variance is at most $O(\log^2 k)$. It then follows that repeating the estimate of $\log(1/p_{\boldsymbol{i}})$ for $O(\log^2 k/\epsilon^2)$ i.i.d. chosen $\boldsymbol{i} \sim \mathcal{D}$ gives the desired estimate with probability at least $2/3$. These parameter settings establish the following theorem:

Theorem 1. *There exists a single-pass data stream algorithm using $O(1)$ words of working memory that processes a stream of $O(k\epsilon^{-2}\log^2 k \log^2(1/\epsilon))$ i.i.d. samples from an unknown distribution $\mathcal{D}$ on $[k]$ and returns an additive ϵ approximation of $H(\mathcal{D})$ with probability $2/3$.*

The space complexity in the theorem above follows since computing the estimator just requires maintaining integers in the sets $[k]$, $[t]$, and $[r]$, as well as computing a low-degree polynomial (whose coefficients we bound in Appendix B). To compute the average of multiple estimators in small space it suffices to compute the sum of the estimates where each estimator is computed in series. The sample complexity bound (given the specified parameters) in the above theorem follows directly from the sample complexity of `LogEstimator`. By virtue of the fact our estimators are based on negative binomial distributions ($\boldsymbol{X}$ in Figure 1 is the number of Bernoulli trials until t successes), this in turn follows directly from the expectation of negative binomial distributions:

Fact 2.1 (Expected Sample Complexity of `LogEstimator`)**.** *Suppose we draw $\boldsymbol{i} \sim \mathcal{D}$ and execute* `LogEstimator`$(\mathcal{D}, \boldsymbol{i})$*. Then, the expected sample complexity is*

$$\sum_{i=1}^{k} p_i \left(r + \frac{t}{p_i} \right) = r + tk.$$

Although the number of samples we draw is a random variable that is only bounded in expectation, note that it implies the existence of a good algorithm that always has a bounded sample complexity: namely, we can simply terminate the algorithm early and output Fail if it draws a large constant factor times more samples than we expect, which happens with low probability by Markov's inequality.

Before moving on to the showing the properties of the estimator, we verify that $h_t(\rho)$ is a degree-r polynomial.

Lemma 2.2. *For any $r \in \mathbb{N}$, let $f \colon \mathbb{R} \to \mathbb{R}$ denote the degree-r Taylor expansion of $\log(z)$ centered at 1. Then, for any $\rho > 0$ and $t \in \mathbb{N}$,*

$$h_t(\rho) = \mathbf{E}_{\boldsymbol{Z} \sim NB(t,\rho)} \left[f\left(\frac{\boldsymbol{Z} \cdot \rho}{t} \right) \right]$$

is a polynomial of degree at most r.

Proof. Recall that the random variable $\boldsymbol{Z} \sim \mathrm{NB}(t, \rho)$ is the number of independent trials from a $\mathrm{Ber}(\rho)$ distribution before one sees t successes. Furthermore, f is the degree-r Taylor expansion of $\log z$ centered at 1, and

$$f(z) = \sum_{i=1}^{r} \frac{(-1)^{i+1}}{i} \cdot (z-1)^r.$$

By linearity of expectation, it suffices to show that for every $j \in \{1, \ldots, r\}$, $\mathbf{E}_{\boldsymbol{Z}}\left[(\boldsymbol{Z}\rho/t - 1)^j \right]$ is a degree-j polynomial in ρ. Note that $\boldsymbol{Z}$ is a sum of t independent $\mathrm{Geo}(\rho)$ random variables, so by expanding $(\frac{1}{t}\sum_{i=1}^{t} \boldsymbol{G}_i \rho - 1)^j$ and applying linearity of expectation once more, it suffices to show that

$$\mathbf{E}_{\boldsymbol{G} \sim \mathrm{Geo}(\rho)} \left[(\boldsymbol{G} \cdot \rho)^j \right] = \rho^j \mathbf{E}_{\boldsymbol{G} \sim \mathrm{Geo}(\rho)} \left[\boldsymbol{G}^j \right] = \rho^j \sum_{k=1}^{\infty} \rho (1-\rho)^{k-1} k^j$$

is a degree-j polynomial in ρ. We note that this latter term, $\mathbf{E}_{\boldsymbol{G}}\left[\boldsymbol{G}^j \right]$ may be expressed as $\rho \cdot \mathrm{Li}_{-j}(1-\rho)$, where $\mathrm{Li}_{-j}(\cdot)$ is the polylogarithm function (see [Wei]). $\mathrm{Li}_{-j}(1-\rho)$ happens to be a rational function, where the denominator is exactly ρ^{j+1}, which cancels the ρ^{j+1} term . In addition, the numerator of $\mathrm{Li}_{-j}(1-\rho)$ is a degree-j polynomial in ρ, which gives the desired polynomial representation. □

Finally, it will be useful for the variance calculation to show that the correction term is always bounded, which we show here.

Lemma 2.3. *There exists a universal constant $c > 0$ such that, for any $r, t \in \mathbb{N}$, if we let $g \colon [0,1]^r \to \mathbb{R}$ be the linear function where $g(\rho, \rho^2, \ldots, \rho^r) = h_t(\rho)$, then $g(b) \in [-c, c]$ for all $b \in \{0,1\}^r$.*

Proof. Recall $g \colon [0,1]^r \to \mathbb{R}$ is the linear function where $g(\rho, \rho^2, \ldots, \rho^r) = h_t(\rho)$. Hence, in order to show that $g \colon \{0,1\}^r \to \mathbb{R}$ is bounded, it suffices to show that the sum-of-magnitudes of the $r+1$ coefficients of h_t is bounded. Since we have

$$h_t(\rho) = \mathbf{E}_{\boldsymbol{Z} \sim \mathrm{NB}(t,\rho)} \left[f\left(\frac{\boldsymbol{Z} \cdot \rho}{t} \right) \right] = \sum_{i=1}^{r} \frac{(-1)^{i+1}}{i} \cdot \mathbf{E}_{\boldsymbol{Z} \sim \mathrm{NB}(t,\rho)} \left[\left(\frac{\boldsymbol{Z} \cdot \rho}{t} - 1 \right)^i \right].$$

Notice that in Lemma 2.2, we showed that each $\mathbf{E}_{\boldsymbol{Z}}\left[(\boldsymbol{Z}\rho/t-1)^i\right]$ is a degree-i polynomial in ρ, and the bound (2) implies that, for each $i \in \{1,\ldots,r\}$ these polynomials are at most $\left(O(i/\sqrt{t})\right)^i$ in magnitude. Furthermore, since these are degree-i polynomials bounded in $[0,1]$, we conclude (by Lemma 4.1 in [She13]), that the coefficients in $\mathbf{E}_{\boldsymbol{Z}}\left[(\boldsymbol{Z}\rho/t-1)^i\right]$ are at most $\left(O(i/\sqrt{t})\right)^i$. In particular, we have that the r coefficients of $h_t(\rho)$ are at most

$$\sum_{i=1}^{r} \frac{1}{i} \cdot \left(O(i/\sqrt{t})\right)^i \leq \sum_{i=1}^{r} \left(O(i/\sqrt{t})\right)^i = O(1/\sqrt{t})$$

because $r \ll \sqrt{t}$. To show that $g\colon \{0,1\}^r \to \mathbb{R}$ is bounded, we add the magnitudes of the r coefficients, which is $O(r/\sqrt{t}) = O(1)$ when $r = O(\log(1/\epsilon))$ and $t = O(\log^2(1/\epsilon))$. □

2.2 Bounding Bias of Estimator

Lemma 2.4. *Let $\mathcal{D}$ be any distribution and consider any $i \in [k]$. If, for $\epsilon \in (0,1)$, we instantiate* $\texttt{LogEstimator}(\mathcal{D}, i)$ *with $r = \Theta(\log(1/\epsilon))$ and $t = \Theta(\log^2(1/\epsilon))$, which produces the random variable $\boldsymbol{\eta}$, then*

$$\left|\mathbf{E}[\boldsymbol{\eta}] - \log\left(\frac{1}{p_i}\right)\right| \leq \epsilon.$$

The remainder of the section constitutes the proof of Lemma 2.4, which will follow from a sequence of claims.

Claim 2.5. *In an execution of* $\texttt{LogEstimator}(\mathcal{D}, i)$*, let $\boldsymbol{X}$ and $\boldsymbol{\eta}$ be defined as in Line 1 and Line 3 of Figure 1, and let $\boldsymbol{Y} = \boldsymbol{X} \cdot p_i/t$. Then, $\mathbf{E}[\boldsymbol{\eta}] - \log(1/p_i) = \mathbf{E}_{\boldsymbol{X}}[h(\boldsymbol{Y})]$, where $h(z)$ is the error in the degree-r Taylor expansion of $\log z$ at 1.*

Proof of Claim 2.5. Notice that $\boldsymbol{X}$ is the number of trials from $\mathrm{Ber}(p_i)$ until we see t successes. We now have the following string of equalities:

$$\begin{aligned}\mathbf{E}_{\boldsymbol{X},\boldsymbol{B}_1,\ldots,\boldsymbol{B}_r}\left[\boldsymbol{\eta} - \log\left(\frac{1}{p_i}\right)\right] &= \mathbf{E}_{\boldsymbol{X}}[\log \boldsymbol{Y}] - \mathbf{E}_{\boldsymbol{B}_1,\ldots,\boldsymbol{B}_r}[g(\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_r)] \\ &= \mathbf{E}_{\boldsymbol{X}}[f(\boldsymbol{Y}) + h(\boldsymbol{Y})] - g(p_i, p_i^2, \ldots, p_i^r) = \mathbf{E}_{\boldsymbol{X}}[h(\boldsymbol{Y})],\end{aligned}$$

where we used the fact that g is a linear function, and that $\mathbf{E}[\boldsymbol{B}_\ell] = p_i^\ell$ in order to substitute

$$\mathbf{E}_{\boldsymbol{B}_1,\ldots,\boldsymbol{B}_r}[g(\boldsymbol{B}_1, \ldots, \boldsymbol{B}_r)] = g(p_i, p_i^2, \ldots, p_i^r).$$

Furthermore, we divide $\log \boldsymbol{Y} = f(\boldsymbol{Y}) + h(\boldsymbol{Y})$, where $f(z)$ is the degree-r Taylor expansion of $\log z$ at 1, and $h(z) = \log z - f(z)$ is the error in the degree-r Taylor expansion of $\log(z)$, i.e.,

$$h(z) = \log(z) - f(z) = \sum_{\ell=r+1}^{\infty} (-1)^{\ell+1} \cdot \frac{(z-1)^\ell}{\ell}.$$

Finally, by construction of g, $\mathbf{E}[f(\boldsymbol{Y})] = g(p_i, p_i^2, \ldots, p_i^r)$, which gives the desired equality. □

Lemma 2.6. *For any $\epsilon \in (0,1)$, letting $r = \Theta(\log(1/\epsilon)$ and $t = \Theta(\log^2(1/\epsilon))$, we have that for $p_i > 0$, $|\mathbf{E}_{\boldsymbol{X}}[h(\boldsymbol{Y})]| \leq \epsilon$.*

Proof of Lemma 2.6. We note that, for any $z \in \mathbb{R}_{>0}$,

$$\begin{aligned}|h(z)| &= \left|(z-1)^r \sum_{\ell=1}^{\infty} (-1)^\ell \cdot \frac{(z-1)^\ell}{r+\ell}\right| \leq |z-1|^r \cdot \left|\sum_{\ell=1}^{\infty} (-1)^\ell \cdot \frac{(z-1)^\ell}{\ell}\right| \\ &= |z-1|^r \cdot |\log z| \lesssim |z-1|^{r+1} + \left(\frac{9}{10}\right)^r + \mathbb{1}\{z \leq 1/10\} \cdot \log(1/z). \qquad (1)\end{aligned}$$

Hence, we have

$$|\mathbf{E}_{\boldsymbol{X}}[h(\boldsymbol{Y})]| \leq \mathbf{E}_{\boldsymbol{X}}[|h(\boldsymbol{Y})|] \leq \mathbf{E}_{\boldsymbol{X}}\left[|\boldsymbol{Y}-1|^{r+1}\right] + \epsilon/2 + \mathbf{E}_{\boldsymbol{X}}[\mathbb{1}\{\boldsymbol{Y} \leq 1/10\}\log(1/\boldsymbol{Y})],$$

where we used the fact that $\boldsymbol{Y} > 0$ and $r = \Theta(\log(1/\epsilon))$ to say $(9/10)^r < \epsilon/2$. In order to bound the above two quantities, we use the fact that the random variable $\boldsymbol{Y}$ is a subgamma random variable and thus has good concentration around its mean (which is 1 for the case of $\boldsymbol{Y}$), giving the desired inequality.

Definition 2.7 (Subgamma Random Variable)**.** *For $\sigma, B \in \mathbb{R}$, a random variable $\boldsymbol{Z}$ with expectation μ is (σ, B)-subgamma if for all $\lambda \in \mathbb{R}$ with $|\lambda| < 1/|B|$,*

$$\psi_{\boldsymbol{Z}}(\lambda) \stackrel{\text{def}}{=} \ln\left(\mathbf{E}\left[e^{\lambda(\boldsymbol{Z}-\mu)}\right]\right) \leq \frac{\lambda^2\sigma^2}{2(1-\lambda|B|)}.$$

It is not hard to verify (see Section C) that the random variable $\boldsymbol{Y}$ is centered at 1, and that there are constants $\alpha, \beta \in \mathbb{R}_{\geq 0}$ so $\boldsymbol{Y}$ is $(\alpha/\sqrt{t}, \beta/t)$-subgamma. Then, by taking the Taylor expansion of $\mathbf{E}\left[e^{\lambda(Y-1)}\right]$, we have that for any $|\lambda| < t/\beta$, and any $j \in \mathbb{N}$,

$$\mathbf{E}_{\boldsymbol{X}}\left[|\boldsymbol{Y}-1|^j\right] \leq \frac{j!}{\lambda^j} \cdot \exp\left(\frac{\alpha^2\lambda^2}{2t(1-\lambda\beta/t)}\right) \leq \frac{\alpha^j j!}{t^{j/2}} \cdot e^3, \tag{2}$$

by picking $\lambda = \sqrt{t}/\alpha$, which is less than t/β for large enough t. Letting $j = r+1$ and setting $t = O(r^2)$, we get the desired bound of $o(\epsilon)$. In order to bound $\mathbf{E}_{\boldsymbol{X}}[\mathbb{1}\{\boldsymbol{Y} \leq 1/10\}\log(1/\boldsymbol{Y})]$, we compute it explicitly, and recall that $\boldsymbol{X} \geq t$, so that the above event is satisfied only if $p_i \leq 1/10$.

$$\begin{aligned}
&\mathbf{E}_{\boldsymbol{X}}[\mathbb{1}\{\boldsymbol{Y} \leq 1/10\}\log(1/\boldsymbol{Y})] \\
&\leq \mathbf{E}_{\boldsymbol{X}}\left[\frac{\mathbb{1}\{\boldsymbol{Y} \leq 1/10\}}{\boldsymbol{Y}}\right] \qquad (3)\\
&= \sum_{\ell=t}^{t/(10p_i)} \binom{\ell-1}{t-1} p_i^t (1-p_i)^{\ell-t} \cdot \frac{t}{\ell p_i} \leq \frac{t}{10p_i} \max_{\ell \in [t, t/(10p_i)]} \left(\frac{e(\ell-1)}{t-1}\right)^{t-1} p_i^{t-1} \cdot \frac{t}{\ell} \\
&\leq \frac{t}{10} \max_{\ell \in [t, t/(10p_i)]} \left(\frac{e^2(\ell-1)}{t-1}\right)^{t-2} p_i^{t-2} = \exp(-\Omega(t)).
\end{aligned}$$

□

3 Improving Sample Complexity via Bucketing

In this section, we focus on estimating the expected value of $\log(\boldsymbol{X}/t)$ with error at most ϵ. Our goal here is to remove the $\mathrm{poly}(\log k)$ dependencies in the sample complexity of estimation. In particular, we prove the following theorem, which improves on the dependence on k in Theorem 1.

Theorem 2. *There exists a single-pass data stream algorithm using $O(1)$ words of working memory that processes a stream of $O(k\log^4(1/\epsilon)/\epsilon^2)$ i.i.d. samples from an unknown distribution $\mathcal{D}$ on $[k]$ and returns an additive ϵ approximation of $H(\mathcal{D})$ with probability at least $2/3$.*

Given the work done in Section 2, it will suffice to estimate the quantity H (we give the explicit reduction in Lemma 3.1 shortly):

$$H \coloneqq \mathbf{E}_{\boldsymbol{i}\sim\mathcal{D}, \boldsymbol{X}\sim\mathrm{NB}(t,p_{\boldsymbol{i}})}\left[\log\left(\boldsymbol{X}/t\right)\right], \tag{4}$$

where t is set to $\Theta(\log^2(1/\epsilon))$, such that we can then apply the correction term of Section 2. Recall that the randomness in the above expectation is taken over the random choice of $\boldsymbol{i} \sim \mathcal{D}$, and $\boldsymbol{X}$ is a negative binomial random variable drawn from $\mathrm{NB}(t, p_{\boldsymbol{i}})$. First, we show that it suffices to estimate (4) in order to estimate the entropy, given our tools from Section 2.

Lemma 3.1. *Consider a fixed distribution $\mathcal{D}$, and for $\epsilon > 0$ suppose $\hat{H} \in \mathbb{R}$ is such that $|H - \hat{H}| \leq \epsilon$. Then, there exists a $O(1)$ word streaming algorithm which given $\hat{H}$ and using an additional $O(\log(1/\epsilon)/\epsilon^2)$ independent samples from $\mathcal{D}$, outputs an estimate to the entropy of $\mathcal{D}$ up to error $\pm 2\epsilon$ with probability at least 0.9.*

It thus suffices to design an algorithm to estimate (4). Our approach is to use a bucketing scheme. At a high level, we partition the range of $\boldsymbol{X}$ into L intervals: $I_1, I_2, \ldots, I_L$. We compute the conditional expectation of $\log(\boldsymbol{X}/t)$ in each interval separately. Then, we take the weighted average of these conditional expectations, where the weights are determined by the probability of the intervals.

Unbounded $\boldsymbol{X}$: As specified above, the random variable $\boldsymbol{X}$ is a mixture of negative binomial random variables, so $\boldsymbol{X}$ may be unbounded. In addition, if we had sampled $\boldsymbol{i} \sim \mathcal{D}$ where $p_{\boldsymbol{i}}$ was very small, $\boldsymbol{X}$'s value will tend to be very large. It will be convenient to introduce a parameter $X_{\max} \in \mathbb{N}$ and consider the random variable $\boldsymbol{X}' := \min(\boldsymbol{X}, X_{\max})$. Let $\tilde{H}$ denotes the expected value of $\boldsymbol{X}'$:

$$\tilde{H} := \mathbf{E}_{\boldsymbol{i},\boldsymbol{X}}\left[\log(\boldsymbol{X}'/t)\right].$$

For the rest of the section, we will seek to approximate $\tilde{H}$, and the fact that this is a good estimate for H follows from the following lemma.

Lemma 3.2. *Let $\boldsymbol{i} \sim \mathcal{D}$, and let $\boldsymbol{X}$ and be a negative binomial random variable from NB$(t, p_{\boldsymbol{i}})$. Let $\boldsymbol{X}'$ be the bounded version of $\boldsymbol{X}$: $\boldsymbol{X}' := \min(\boldsymbol{X}, X_{\max})$. Let $t \in \mathbb{N}$ and $\epsilon \in (0,1)$. If we set $X_{\max} = tk\ln(2)/\epsilon$, then*

$$\left|H - \tilde{H}\right| = \mathbf{E}_{\boldsymbol{i},\boldsymbol{X}}\left[\log(\boldsymbol{X}/t) - \log(\boldsymbol{X}'/t)\right] \leq \epsilon\,.$$

Comparison to related work: It is worth noting that the proofs in this section are inspired by the work of [ABIS19]. The authors used a similar bucketing technique to estimate entropy. While the structure of our proof is similar, there are subtle differences between our work and what they did. First, we are focusing on estimating different quantities. In particular, we work with an unbounded random variable while their estimator is bounded. Moreover, they have a two-step bucketing system where they draw a sample $\boldsymbol{i}$ and two estimates for $p_{\boldsymbol{i}}$; they use one estimate for detecting which bucket falls into and the second one to estimate entropy in that bucket. One of the complications of this approach is that the second estimator may fall into a different bucket; Thus, they have to "clip" the second estimator to make sure it is close to the bucket of the first estimator. We have circumvented these hurdles by using the same estimate for detecting which bucket we are in and estimating $\log(\boldsymbol{X}/t)$ in that bucket.

The algorithm: We write $\tilde{H}$ in terms of conditional expectation in the intervals.

$$\tilde{H} = \sum_{\ell=1}^{L} \underbrace{\mathbf{Pr}_{\boldsymbol{i}\sim\mathcal{D},\boldsymbol{X}\sim\mathrm{NB}(t,p_{\boldsymbol{i}})}\left[\boldsymbol{X}' \in I_\ell\right]}_{q_\ell :=} \cdot \underbrace{\mathbf{E}_{\boldsymbol{i}\sim\mathcal{D},\boldsymbol{X}\sim\mathrm{NB}(t,p_{\boldsymbol{i}})}\left[\log(\boldsymbol{X}'/t) \mid \boldsymbol{X}' \in I_\ell\right]}_{H_\ell :=}.$$

Let q_ℓ denote the probability of $\boldsymbol{X}'$ being in I_ℓ, and H_ℓ denote the conditional expectation in I_ℓ. Our algorithm estimate q_ℓ and H_ℓ for each interval to find an estimate for $\tilde{H}$. Below we give a brief description of our algorithm, and the pseudocode can be found in Algorithm 1.

Below, we define $b_0 = t < b_1 < \cdots < b_L = X_{\max}$to be $L+1$ parameters (which we will set shortly) that denote the boundary points of the intervals:

$$I_\ell = [b_{\ell-1}, b_\ell) \qquad \forall i \in [L-1]\,, \qquad\qquad I_L = [b_{L-1}, b_L]\,.$$

For each interval I_ℓ, we draw r_ℓ samples from $\mathcal{D}$, namely $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_{r_\ell} \sim \mathcal{D}$. For each $\boldsymbol{i}_j$, we start drawing samples from $\mathcal{D}$ in the process of drawing a negative binomial random variable $\boldsymbol{X}_j \sim \mathrm{NB}(t, p_{\boldsymbol{i}_j})$; then, we will set $\boldsymbol{X}'_j = \min(\boldsymbol{X}_j, X_{\max})$. Furthermore, we will only consider $\boldsymbol{X}'_j$'s that fall in I_ℓ, which means that we can stop early if we already know $\boldsymbol{X}'_j$ will be too large. In particular, if we draw b_ℓ samples and have not observed t instances of $\boldsymbol{i}_j$, we can already conclude $\boldsymbol{X}'_j$ is not in I_ℓ and stop sampling. Among these r_ℓ samples $\{\boldsymbol{i}_1, \ldots, \boldsymbol{i}_{r_\ell}\}$, let $\boldsymbol{c}_\ell$ denote the number of $\boldsymbol{X}'_j$'s that fall into I_ℓ. We estimate the weight of each bucket by $\hat{\boldsymbol{q}} := \boldsymbol{c}_\ell / r_\ell$. For the last bucket, we set $\hat{\boldsymbol{q}}_\ell$ in a way that the sum of the weight is one:

$$\hat{\boldsymbol{q}}_\ell = \frac{\boldsymbol{c}_\ell}{r_\ell}\,, \qquad \forall j = 1, \ldots, L-1\,, \qquad \hat{\boldsymbol{q}}_L := 1 - \sum_{j=1}^{L-1} \hat{\boldsymbol{q}}_L\,.$$

Also, we compute an average of $\log(\boldsymbol{X}'_j/t)$ of such $\boldsymbol{X}'_j$'s and denote it by $\hat{\boldsymbol{H}}_\ell$:

$$\hat{\boldsymbol{H}}_\ell = \frac{\sum_{j=1}^{r_\ell} \mathbb{1}\{\boldsymbol{X}'_j \in I_\ell\} \cdot \log(\boldsymbol{X}'_j/t)}{\boldsymbol{c}_\ell} \qquad \forall \ell = 1, \ldots, L\,.$$

In these definitions, we take $\hat{\boldsymbol{H}}_\ell = \log(b_\ell/t)$ if $\boldsymbol{c}_\ell = 0$. Our estimate for $\tilde{H}$ is the weighted sum of $\hat{\boldsymbol{H}}_\ell$:

$$\hat{\boldsymbol{H}} = \sum_{\ell=1}^{L} \hat{\boldsymbol{q}}_\ell \cdot \hat{\boldsymbol{H}}_\ell\,.$$

Algorithm 1 Estimating $\mathbf{E}[\log \boldsymbol{X}/t]$ via Bucketing

procedure LOGESTIMATOR(k, ϵ, sample access to $\mathcal{D}$)
  $\hat{\boldsymbol{H}} \leftarrow 0$
  for $\ell = 1, 2, \ldots, L$ **do**
    $\boldsymbol{c}_\ell \leftarrow 0$, $\hat{\boldsymbol{H}}_\ell \leftarrow 0$
    for r_ℓ times **do**
      Draw $\boldsymbol{i} \sim \mathcal{D}$
      Draw b_ℓ samples from $\mathcal{D}$ but terminate early if t occurrences of $\boldsymbol{i}$ are observed.
      $\boldsymbol{X} \leftarrow$ number of samples drawn
      $\hat{\boldsymbol{H}}_\ell \leftarrow \hat{\boldsymbol{H}}_\ell + \log(\boldsymbol{X}/t) \cdot \mathbb{1}\{\boldsymbol{X} \in I_\ell\}$ and $\boldsymbol{c}_\ell \leftarrow \boldsymbol{c}_\ell + \mathbb{1}\{\boldsymbol{X} \in I_\ell\}$
    $\hat{\boldsymbol{H}}_\ell \leftarrow \hat{\boldsymbol{H}}_\ell / c_\ell$, $\hat{\boldsymbol{q}}_\ell \leftarrow c_\ell / r_\ell$
    if $\ell = L$ **then**
      $\hat{\boldsymbol{q}}_L \leftarrow 1 - \sum_{\ell=1}^{L-1} \hat{\boldsymbol{q}}_\ell$
    $\hat{\boldsymbol{H}} \leftarrow \hat{\boldsymbol{H}} + \hat{q}_\ell \hat{\boldsymbol{H}}_\ell$

It is fairly straightforward to show that Algorithm 1 uses a constant number of words. Note that r_ℓ (similarly b_ℓ's) can be computed from $r_{\ell-1}$, so we do not need to calculate and store all the r_ℓ's beforehand. Also, to compute $\hat{\boldsymbol{q}}_L$, we do not need all the $\hat{\boldsymbol{q}}_\ell$'s. We only need to keep a running sum of $\hat{\boldsymbol{q}}_\ell$. Thus, we only need a constant number of words of memory. We analyze correctness simply by bounding the variance. Namely, the remainder of the section will be devoted to proving the following lemma, which will imply that our estimator will be within $\pm\epsilon$ of $\tilde{H}$ with constant probability.

Lemma 3.3. *For any $k \in \mathbb{N}$ and $\epsilon > 0$, there exists a setting of $L > 0$, parameters $b_0 = t < b_1 < \cdots < b_L = X_{\max}$ and $r_1, \ldots, r_\ell$ for $\ell \in [L]$ such that*

$$\mathbf{E}\left[\left(\sum_{\ell=1}^{L} \hat{\boldsymbol{q}}_\ell \cdot \hat{\boldsymbol{H}}_\ell - \sum_{\ell=1}^{L} q_\ell \cdot H_\ell\right)^2\right] = O(\epsilon^2). \tag{5}$$

We note that, once we prove (3.3), we guarantee that our estimator is within $\pm O(\epsilon)$ of $\tilde{H}$ with probability 0.9 by Chebyshev's inequality. Before delving into the proof, we note that we may re-write the left-hand side of (5) as:

$$\begin{aligned}
\sum_{\ell=1}^{L} \hat{\boldsymbol{q}}_\ell \cdot \hat{\boldsymbol{H}}_\ell - \sum_{\ell=1}^{L} q_\ell \cdot H_\ell &= \sum_{\ell=1}^{L} (\hat{\boldsymbol{q}}_\ell - q_\ell)\, \hat{\boldsymbol{H}}_\ell + \sum_{\ell=1}^{L} q_\ell \left(\hat{\boldsymbol{H}}_\ell - H_\ell\right) \\
&= \sum_{\ell=1}^{L-1} (\hat{\boldsymbol{q}}_\ell - q_\ell)\, \hat{\boldsymbol{H}}_\ell + \left(1 - \sum_{\ell=1}^{L-1} \hat{\boldsymbol{q}}_\ell - 1 + \sum_{\ell=1}^{L-1} q_\ell\right) \hat{\boldsymbol{H}}_L + \sum_{\ell=1}^{L} q_\ell \left(\hat{\boldsymbol{H}}_\ell - H_\ell\right) \\
&= \sum_{\ell=1}^{L-1} (\hat{\boldsymbol{q}}_\ell - q_\ell) \left(\hat{\boldsymbol{H}}_\ell - \hat{\boldsymbol{H}}_L\right) + \sum_{\ell=1}^{L} q_\ell \left(\hat{\boldsymbol{H}}_\ell - H_\ell\right). \qquad (6)
\end{aligned}$$

Therefore, the upper bound on (5) follows from the following two lemmas.

Lemma 3.4. *For any setting of $t = b_0 < \cdots < b_L = X_{\max}$ and $\{r_\ell \in \mathbb{N} : \ell \in [L]\}$, we have*

$$\mathbf{E}\left[\left(\sum_{\ell=1}^{L} q_\ell \left(\hat{\boldsymbol{H}}_\ell - H_\ell\right)\right)^2\right] \le \sum_{\ell=1}^{L} q_\ell \left(1 - q_\ell\right)^{r_\ell} \cdot \log^2(b_\ell/b_{\ell-1}) + 2 \sum_{\ell=1}^{L} \frac{\log^2(b_\ell/b_{\ell-1})}{r_\ell + 1}.$$

Lemma 3.5. *For any setting of $t = b_0 < \cdots < b_L = X_{\max}$ and $\{r_\ell \in \mathbb{N} : \ell \in [L]\}$, we have*

$$\mathbf{E}\left[\left(\sum_{\ell=1}^{L-1}(\hat{\boldsymbol{q}}_\ell - q_\ell)(\hat{\boldsymbol{H}}_\ell - \hat{\boldsymbol{H}}_L)\right)^2\right] \leq O(1) \cdot \sum_{\ell=1}^{L-1} \log^2(b_L/b_{\ell-1}) \cdot \log^{(\ell)} k \cdot \frac{q_\ell(1-q_\ell)}{r_\ell}.$$

Hence, we consider the following setting of parameters, where we let $L = \log^* k$,[4] such that we have $b_0 = t$, and

$$\forall \ell \in \{1, \ldots, L-1\}, \quad b_\ell \stackrel{\text{def}}{=} \frac{tk}{(\log^{(\ell)} k)^4} \quad , \quad b_L \stackrel{\text{def}}{=} \frac{tk}{\epsilon} \quad \text{and let} \quad r_\ell \stackrel{\text{def}}{=} \frac{\log^2(b_L/b_{\ell-1}) \log^{(\ell)} k}{\epsilon^2}. \tag{7}$$

We give a bound on the expected sample complexity.

Lemma 3.6. *The expected sample complexity of the algorithm is $O(k \log^4(1/\epsilon)/\epsilon^2)$.*

Proof. For the intervals $\ell = \{1, \ldots, L-1\}$, we always spend r_ℓ tries to determine whether a sample falls within a particular interval. Note that we take one sample to determine $\boldsymbol{i} \sim \mathcal{D}$, and then we take at most b_ℓ samples. Therefore, the sample complexity for these is

$$\sum_{\ell=1}^{L-1} r_\ell \cdot b_\ell = \frac{tk \log^2(1/\epsilon)}{\epsilon^2} \sum_{\ell=1}^{L-1} \frac{1}{\log^{(\ell)} k} \leq kt \cdot O(\log^2(1/\epsilon)/\epsilon^2),$$

where we used (and will continue to use) the fact that for any positive $\delta > 0$, we have

$$\sum_{\ell=1}^{L} \frac{1}{(\log^{(\ell)} k)^\delta} = \sum_{h=0}^{L} \frac{1}{(\log^{(L-h)} k)^\delta} = O(1). \tag{8}$$

In particular, for any $z > 1$ where $L = \log^* z$,

$$\log^{(L-h)} z > \underbrace{\exp\left(\exp\left(\ldots \exp(1) \ldots\right)\right)}_{h \text{ times}}.$$

In particular, the series in (8) has denominators growing faster than geometric growth, so the sum is dominated by the largest value which is a constant. Finally, it remains to bound the expected sample complexity of the bucket L. Here, we note

$$r_L = \frac{O(1)}{\epsilon^2} \cdot \log^2\left(\frac{\log^{(L-1)} k}{\epsilon}\right) \leq O\left(\frac{\log^2(1/\epsilon)}{\epsilon^2}\right).$$

Therefore, the expected sample complexity incurred in the interval L is

$$r_L \cdot \sum_{i=1}^{k} q_i \cdot \frac{t}{q_i} = O(k \log^4(1/\epsilon)/\epsilon^2).$$

□

4 Conclusions

We presented an algorithm for returning an additive ϵ approximation of the Shannon entropy of a distribution over $[k]$. The algorithm required $O(k\,\epsilon^{-2} \log^4(1/\epsilon))$ i.i.d. samples from the unknown distribution and a constant number of words of memory. In terms of the ϵ dependence, this improves over the state-of-the-art [ABIS19] by a factor $1/\epsilon$ in the sample complexity. More generally, we expect that the technique used, that of correcting the bias via low-degree polynomials will be useful in the context of the other inference problems in the data stream setting. The main open problem is determining whether the sample complexity of our result is optimal. We conjecture that this is the case, up to the $\mathrm{poly}(\log(1/\epsilon))$ factors. See Section F for a discussion of this conjecture.

[4] Recall that $\log^* z$ is the number of iterated logarithms (base 2) before the result is less than or equal to 1.

References

[ABIS19] Jayadev Acharya, Sourbh Bhadane, Piotr Indyk, and Ziteng Sun. Estimating entropy of distributions in constant space. In *Proceedings of the 32nd Annual Conference on Neural Information Processing (NeurIPS)*, pages 5163–5174, 2019.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[BDKR05] Tugkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. The complexity of approximating the entropy. *SIAM J. Comput.*, 35(1):132–150, 2005.

[BG06] Lakshminath Bhuvanagiri and Sumit Ganguly. Estimating entropy over data streams. In *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*, pages 148–159, 2006.

[CCM10] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. A near-optimal algorithm for estimating the entropy of a stream. *ACM Trans. Algorithms*, 6(3):51:1–51:21, 2010.

[CLM10] Steve Chien, Katrina Ligett, and Andrew McGregor. Space-efficient estimation of robust statistics and distribution testing. In *Proceedings of the 1st Annual Conference Innovations in Computer Science (ICS)*, pages 251–265, 2010.

[GKLR21] Sumegha Garg, Pravesh K. Kothari, Pengda Liu, and Ran Raz. Memory-sample lower bounds for learning parity with noise. In *Proceedings of the 25th International Workshop on Randomization and Computation (RANDOM)*, pages 60:1–60:19, 2021.

[GKR20] Sumegha Garg, Pravesh K. Kothari, and Ran Raz. Time-space tradeoffs for distinguishing distributions and applications to security of goldreich's PRG. In *Proceedings of the 24th International Workshop on Randomization and Computation (RANDOM)*, pages 21:1–21:18, 2020.

[GM07] Sudipto Guha and Andrew McGregor. Space-efficient sampling. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 171–178, 2007.

[GRT18] Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 990–1002, 2018.

[GRT19] Sumegha Garg, Ran Raz, and Avishay Tal. Time-space lower bounds for two-pass learning. In *Proceedings of the 34th Computational Complexity Conference (CCC)*, pages 22:1–22:39, 2019.

[HNO08] Nicholas J. A. Harvey, Jelani Nelson, and Krzysztof Onak. Sketching and streaming entropy via approximation theory. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (IEEE)*, pages 489–498, 2008.

[JVHW15] Jiantao Jiao, Kartik Venkat, Yanjun Han, and Tsachy Weissman. Minimax estimation of functionals of discrete distributions. *IEEE Trans. Inf. Theory*, 61(5):2835–2885, 2015.

[KRT17] Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1067–1080, 2017.

[MG82] Jayadev Misra and David Gries. Finding repeated elements. *Sci. Comput. Program.*, 2(2):143–152, 1982.

[MM17] Dana Moshkovitz and Michal Moshkovitz. Mixing implies lower bounds for space bounded learning. In *Proceedings of the 30th Conference on Learning Theory (COLT)*, pages 1516–1566, 2017.

[MP80] J. Ian Munro and Mike Paterson. Selection and sorting with limited storage. *Theor. Comput. Sci.*, 12:315–323, 1980.

[Raz17] Ran Raz. A time-space lower bound for a large class of learning problems. In *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 732–742, 2017.

[Raz19] Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. *J. ACM*, 66(1):3:1–3:18, 2019.

[She13] Alexander A. Sherstov. Making polynomials robust to noise. *Theory Comput.*, 9:593–615, 2013.

[SSV19] Vatsal Sharan, Aaron Sidford, and Gregory Valiant. Memory-sample tradeoffs for linear regression with small error. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 890–901, 2019.

[SVW16] Jacob Steinhardt, Gregory Valiant, and Stefan Wager. Memory, communication, and statistical queries. In *Proceedings of the 29th Conference on Learning Theory (COLT)*, pages 1490–1516, 2016.

[VV11] Gregory Valiant and Paul Valiant. The power of linear estimators. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 403–412, 2011.

[VV17] Gregory Valiant and Paul Valiant. Estimating the unseen: Improved estimators for entropy and other properties. *J. ACM*, 64(6):37:1–37:41, 2017.

[Wei] Eric W. Weisstein. Polylogarithm. from MathWorld–A Wolfram Web resource. Last accessed Feb. 9 2022.

[WY16] Yihong Wu and Pengkun Yang. Minimax rates of entropy estimation on large alphabets via best polynomial approximation. *IEEE Trans. Inf. Theory*, 62(6):3702–3720, 2016.

Checklist

1. For all authors...
   - 1.1 Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   - 1.2 Did you describe the limitations of your work? [N/A] The model assumes that the stream consists of iid samples. This is discussed in the introduction.
   - 1.3 Did you discuss any potential negative societal impacts of your work? [Yes]
   - 1.4 Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   - 2.1 Did you state the full set of assumptions of all theoretical results? [Yes]
   - 2.2 Did you include complete proofs of all theoretical results? [Yes] Proofs not included in the body of the paper are included in the supplementary material/appendix.
3. If you ran experiments...
   - 3.1 Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   - 3.2 Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   - 3.3 Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   - 3.4 Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   - 4.1 If your work uses existing assets, did you cite the creators? [N/A]
   - 4.2 Did you mention the license of the assets? [N/A]
   - 4.3 Did you include any new assets either in the supplemental material or as a URL? [N/A]
   - 4.4 Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   - 4.5 Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   - 5.1 Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   - 5.2 Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   - 5.3 Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]