Strong Memory Lower Bounds for Learning Natural Models

Gavin Brown* GRBROWN@BU.EDU
Mark Bun† MBUN@BU.EDU
Adam Smith* ADS22@BU.EDU
Department of Computer Science, Boston University, Boston, USA.

Editors: Po-Ling Loh and Maxim Raginsky

Abstract

We give lower bounds on the amount of memory required by one-pass streaming algorithms for solving several natural learning problems. In a setting where examples lie in $\{0,1\}^d$ and the optimal classifier can be encoded using κ bits, we show that algorithms which learn to constant error using a near-minimal number of examples, $\tilde{O}(\kappa)$, must use $\tilde{\Omega}(d\kappa)$ bits of space. Our space bounds match the dimension of the ambient space of the problem's natural parametrization, even when it is quadratic in the size of examples and the final classifier. For instance, in the setting of d-sparse linear classifiers over degree-2 polynomial features, for which $\kappa = \Theta(d \log d)$, our space lower bound is $\tilde{\Omega}(d^2)$. Our bounds degrade gracefully with the stream length N, generally having the form $\tilde{\Omega}\big(d\kappa \cdot \frac{\kappa}{N}\big)$.

Bounds of the form $\Omega(d\kappa)$ were known for learning parity and other problems defined over finite fields. Bounds that apply in a narrow range of sample sizes are also known for linear regression. Ours are the first such bounds for problems of the type commonly seen in recent learning applications that apply for a large range of input sizes.

1. Introduction

The complex models that power much of machine learning's recent success are typically fit to large data sets using streaming algorithms that process examples one by one, updating a stored model as they go. Their performance is often limited by their memory footprint as much as it is by the complexity of their calculations (see e.g., Vaswani et al., 2017; Brown et al., 2020; Ramesh et al., 2021).

We give new lower bounds on the space required to solve natural learning problems in a streaming model, where each example can be processed only once. Consider a stream of data elements $Z = (Z_1, ..., Z_N)$ drawn from a distribution P on labeled examples in the set $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ denotes a set of possible feature vectors and $\mathcal{Y}$ a set of possible labels (e.g., $\{0,1\}$).

A learning algorithm's goal, given one pass over the stream Z, is to find a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ with small error on unseen examples from P. We focus on misclassification error, $\mathrm{err}_P(h) \stackrel{\text{def}}{=} \Pr_{(x,y)\sim P}(h(x) \neq y)$, though for real valued functions (e.g., when $\mathcal{Y} = [0,1]$) we consider the expected absolute error $\mathrm{err}_P(h) \stackrel{\text{def}}{=} \mathbb{E}_{(x,y)\sim P} |h(x) - y|$. The error of a learning algorithm A on P with input length N is defined as the expected error on a stream of N inputs drawn from P.

$$\mathrm{err}_{P^{\otimes N}}(A) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{Z\sim P^{\otimes N}} \big(\mathrm{err}_P(A(Z))\big) .$$

* Supported in part by NSF awards CCF-1763786 and CNS-2120667, a Sloan Foundation research award and gifts from Apple and Google.
† Supported in part by NSF awards CCF-1947889 and CNS-2046425.

We simply write $\text{err}_P(A)$ when N is clear from context.

Given a function class $\mathcal{H}$ of functions from $\mathcal{X} \to \mathcal{Y}$, we say that algorithm A *(agnostically) learns $\mathcal{H}$ to error ε* if, for every distribution P on $\mathcal{X} \times \mathcal{Y}$, the algorithm A finds a hypothesis with (expected) error at most ε more than that of the best hypothesis in $\mathcal{H}$, that is, $\text{err}_P(A) - \inf_{h \in \mathcal{H}} \left(\text{err}_P(h) \right) \leq \varepsilon$. For concreteness, we will consider an algorithm successful it if learns a particular class to error better by a constant than one gets by random guessing. For example, we might assume there is an $h^* \in \mathcal{H}$ for which $\text{err}_P(h^*) \leq 1/4$ and require our algorithm have expected error at most 0.49.

In this paper, we ask how much memory is needed for a streaming algorithm to agnostically learn $\mathcal{H}$ as a function of the size $\kappa = \log |\mathcal{H}|$ of the function space,[1] the data dimension $d = \log |\mathcal{X}|$ and the stream length N. A streaming algorithm receives each example once; the algorithm's memory size on a given execution is the maximum number of bits used to encode its state between processing examples. The output is a function of the final state.

If space is not a concern, observe that, for any class $\mathcal{H}$ and any distribution P, it suffices to receive $O(\kappa)$ examples from P in order to find a hypothesis with error within a small constant of the best hypothesis in $\mathcal{H}$. Thus, there is an algorithm that uses a stream of length $N = O(\kappa)$ and space $O(\kappa d)$ bits which simply stores its entire stream and attains low expected error.

Strong Bounds for Sparse Models We show simple, natural function classes (capturing, for example, sparse linear classification over a degree-2 polynomial feature space) for which this trivial memory bound is tight. Specifically, every algorithm that uses a stream of length $\tilde{O}(\kappa)$ and finds a hypothesis whose accuracy exceeds random guessing by a constant must, on average over executions, use $\tilde{\Omega}(d\kappa)$ bits of memory. Our lower bounds do not assume any particular form or running time of the algorithm or its output; they build on information complexity techniques developed by Braverman et al. (2020) for bounding the space complexity of statistical estimation.

In a breakthrough result, Raz (2018) proved such a bound for the class of parity functions over d-bit inputs. In his setting, the function class $\mathcal{H}$ is also the set of d-bit strings, so $d = \kappa$. Raz (2018) proved that any streaming algorithm solving parity learning requires either $\Omega(d^2) = \Omega(d\kappa)$ bits of memory or $2^{\Omega(\kappa)}$ examples. Subsequent papers extended and generalized these results, in particular to higher-degree polynomials and related classes (see Section 2).

These results are striking. However, they do not obviously imply lower bounds for the function classes (linear models, neural networks) that are the focus of much modern machine learning. Although several authors have studied memory bounds for problems of a more continuous flavor, initial lower bounds were generally limited to the form $\tilde{\Omega}(\max(\kappa, d))$—that is, the lower bounds are limited to either the size of the model or the size of a single example (Steinhardt and Duchi (2015); Garg et al. (2014); Braverman et al. (2016)). Several recent papers prove stronger lower bounds, but under significant restrictions on either parameter ranges (Sharan et al., 2019; Dagan et al., 2019; Dagan and Shamir, 2018) or the computational model (Marsden et al., 2022). We discuss these further in Related Work (Section 2).

We consider a simple distributional problem, described below, and show a memory lower bound of $\tilde{\Omega}(\kappa d \cdot \frac{\kappa}{N})$ for all $N \geq \kappa$. It implies memory lower bounds for learning a number of natural function classes. These include:

1. We focus on the cardinality of the function space for simplicity but, for continuous spaces, one should think of κ as the bit length of an appropriate discrete representation of functions in $\mathcal{H}$.

1. *Direct sums of k dictators:* Let $\mathcal{X} = [k] \times \{0,1\}^{d'}$ (for $d' = d - \log_2 k$, so inputs can be described with d bits) and consider classifiers $h_{i_1,\ldots,i_k}$ specified by k indices in $[d']$, where $h_{i_1,\ldots,i_k}(j, x) = x_{i_j}$ (that is, for each j there is a single bit of x that determines the label).
2. *Sparse linear classifiers over degree-2 polynomial features*: Let $\mathcal{X} = \{0,1\}^d$, and consider classifiers of the form $h(x) = \text{sign}(\langle w, \phi(x)\rangle)$ where $\phi(x)$ denotes the values of degree-2 monomials in the entries of x (so each entry of $\phi(x)$ equals $x_i x_j$ for two indices $i, j \in [d]$) and $w \in \{0,1\}^{\binom{d}{\leq 2}}$ has at most k nonzero entries. The classifiers we study may also be viewed as *k-term 2-DNFs*: Let $\mathcal{X} = \{0,1\}^d$ and consider functions given by the OR of k terms, each of which is the AND of two input bits.
3. *Multiclass sparse linear classifiers*: Let $\mathcal{X} = \{0,1\}^d$ and $\mathcal{Y} = [k]$ (so there are k distinct labels). Let $\mathcal{H}$ comprise all functions of the form $h(x) = \arg\max_{j \in [k]} \langle w_j, x\rangle$ where each $w_j \in \{0,1\}^d$ has $O(\log k)$ nonzero entries. The combining function can also be taken to be a "softmax" instead of the exact argmax.
4. *Real-valued regression*: Let $\mathcal{X} = \{0,1\}^d$ and $\mathcal{Y} = [0,1]$. Consider functions realizable by a sparse two-layer neural network with a single hidden layer of k ReLU nodes, each of which is connected to at most $O(\log k)$ input nodes. The weights on the wires in the first layer are either 0 or 1, and those in the second layer are in $\left\{0, \frac{1}{k-1}, \frac{1}{k-1}, ..., 1\right\}$.

For each of these settings, $\kappa = \tilde{\Theta}(k \log d)$ and we show a space lower bound of $\tilde{\Omega}(dk) = \tilde{\Omega}(d\kappa)$ when N is close to the minimal sample complexity of $\Theta(\kappa)$. For two of the classes above, sparse linear classifiers over the polynomial features and k-term 2-DNFs, our bounds apply for $1 \leq k \leq d/2$. Our bounds for the remaining problems apply for k anywhere from 1 to superpolynomial in d.

Our bounds degrade gracefully as the sample size increases. For $N \geq \kappa$, every learner that succeeds with a stream of N examples requires memory at least $\tilde{\Omega}(d\kappa^2/N)$. In this regard, our bounds behave similarly to the initial, weaker bounds for regression-like problems, but are not as strong as those for learning parity and other algebraic problems (as in Raz (2018)).

For Example 1 (dictators), our lower bounds are matched up to logarithmic factors in all parameter regimes by empirical risk minimization, which runs in l inear time; see Appendix G.2. For Examples 2 and 3 (sparse linear models), multiplicative weights—a widely used continuous optimization procedure—asymptotically matches our lower bounds on the particular input distributions that arise in our arguments; see Appendix G.3. (It is unclear whether the algorithm PAC learns these classes under arbitrary distributions, nor what provable guarantees are achievable for Example 4.)

A Simple, Distributional "Core" Problem Our bounds all derive from space lower bounds for the following simple problem, described more completely in Section 3 and parametrized by positive integers d, k, and $\rho \leq d$. The learner receives a stream of N inputs in $[k] \times \{0,1\}^d$, each of which consists of a "subpopulation identifier" in $[k]$ and a feature vector in $\{0,1\}^d$.

- The input stream is drawn i.i.d. from a distribution P which is a uniform mixture of k components. Each component j's distribution is specified by a set I_j of up to ρ indices in $[d]$ and bits $(b_{j,i})_{i \in I_j}$. An observation from component j is a pair (j, X) where $X \in \{0,1\}^d$ is uniform except for coordinates in I_j, which are set to their $b_{j,i}$ values. The entire distribution P is thus specified by k sets $I_1, ..., I_k$ and the associated bits $(b_{j,i})_{j \in [k], i \in I_j}$.

- To generate the parameters of P, each component's parameters are generated independently by sampling a number r uniformly in $\{0, \ldots, \rho\}$, and then selecting a size-r subset I_j of fixed features, with the bit values of the features selected uniformly.
- After receiving N examples drawn from P, the algorithm is presented with a test pair (j, x) which is either drawn from P (the "structured" case) or drawn uniformly at random from $[k] \times \{0,1\}^d$ ("uniform"); the algorithm must distinguish between these two cases.

One can reduce this distributional problem to agnostic learning of any of the classes mentioned above—see Appendix F.

For a distribution P in the class above, there is a simple optimal distinguisher: given a pair (j, x), it checks if x agrees with $b_{j,i}$ in each position $i \in I_j$; it outputs "structured" if all the checks pass and "uniform" otherwise. Even when $\rho = 1$ (so there are either 0 or 1 fixed bits in each subpopulation), this distinguisher has advantage $1/8$ over random guessing. More generally, it has advantage $\frac{1}{2} - O\left(\frac{1}{\rho}\right)$. It is not hard to learn such a distinguisher: for all $\rho \geq 1$, a distinguisher with constant advantage over random guessing can be learned from $\Theta(k \log(d))$ examples using space $O(dk)$. More generally, for $N = \Omega(k \log(d))$ and $1 \leq \rho \leq d$, a simple strategy (described for completeness in Appendix G.1) learns a distinguisher with constant advantage in space $O\left(dk \cdot \frac{k}{N} \cdot \frac{1}{\rho}\right)$. We show that these simple strategies are essentially optimal.

Theorem 1 (Informal, see Theorem 12) *Consider the above streaming problem with N examples, d dimensions, k components, and at most ρ fixed features, with $\rho = o(d^{1/4})$. Any algorithm solving this task to constant error less than $\frac{1}{2}$ requires space $\Omega\left(\frac{k^2 d}{N\rho^4}\right) = \Omega\left(dk \cdot \frac{k}{N} \cdot \frac{1}{\rho^4}\right)$.*

The ratio N/k is the expected number of examples from each subpopulation. For ρ that is at most logarithmic in d, the bounds have the form $\tilde{\Omega}\left(k \cdot \frac{d}{T}\right)$, where $T = N/k$. The bound may be viewed as incorporating two statements: the memory required to learn to distinguish a particular subpopulation using T examples scales as $\frac{d}{T}$, and there is no better way to solve the larger problem than to learn each subpopulation individually.

Discussion A widespread strategy in modern deep learning is to first train a large, dense network and then use it to find a smaller network by distillation or pruning. One common explanation for this approach is that optimization in the larger space is easier (see, e.g., Frankle et al. (2020); Bartlett et al. (2021)). Our work suggests a different explanation for this strategy's empirical success—namely, the larger parameter vector allows the training process to encode information whose relevance to the problem can only be understood later. In particular, we show that training algorithms for sparse models must sometimes use space proportional to the ambient dimension of the natural encoding, rather than with the size of the examples or the final classifier.

Our lower bounds hold only for one-pass streaming algorithms. This covers the common training strategy in settings where large amounts of data are available (e.g., Brown et al., 2020, who use partial epochs for some corpora). However, when data is not so abundant, machine learning models are often trained by taking multiple passes over the data sets. We conjecture that similar bounds hold for multi-pass algorithms. That is, it seems likely that a stream of $a \cdot n$ fresh examples is at least as useful as a passes over n examples; however, proving statements of that nature is challenging (Garg et al. (2019) and Dagan and Shamir (2018) provide notable successful examples), and we leave it as an open problem for future work.

Techniques We bound an algorithm's space usage using specific measures of information complexity. The core distributional problem we use is inspired by the clustering problem used by Brown et al. (2021) to prove that high-accuracy learning algorithms must sometimes store considerable information about individual examples in their final hypothesis. We change the problem in a few respects (fewer fixed bit positions; a focus on distinguishing instead of labeling). More importantly, the current paper employs a very different technical approach.

Our "single subpopulation" task, defined in Section 3, is closely related to the "hide-and-seek problem" of Shamir (2014). Both problems consider streams where examples are d-bit strings. In each example, all but a few (a priori unknown) indices are uniformly random. Beyond some technical details in the setting, the relevant difference is the notion of information cost used in our lower bound which, as we discuss below, allows us to lift the results to larger problems.

Our main technical tool is a notion of information cost recently introduced in Braverman et al. (2020), which we sometimes refer to as the "composable information cost" of algorithm M, denoted $\mathrm{CI}(M)$:

$$\mathrm{CI}(M) \stackrel{\text{def}}{=} \sum_{i=1}^{N} \sum_{t=i}^{N} I(M_t; X_i \mid M_{i-1}), \tag{1}$$

where X_i is the i-th example and M_t is the algorithm's state after processing X_t. We make no assumptions on the form of the memory state. $I(\cdot;\cdot|\cdot)$ denotes Shannon's (conditional) mutual information. This information cost is always measured relative to a specific distribution on the input X. Composable information cost is useful for streaming applications when proving direct-sum-type statements, turning a lower bound for a simpler problem into a bound for a more complex one. Furthermore, $\frac{1}{N} \cdot \mathrm{CI}(M)$ is a lower bound on the space used by the algorithm (Lemma 20).

A key first step, which illustrates the measure's utility, is a new lower bound we prove in Section 4 for the task of distinguishing whether a stream of bits is uniformly random or fixed (either to zero or one)—a special case of the coin problem (Braverman et al., 2020, 2021). In the coin problem, the learner is asked to distinguish whether a stream of flips resulted from a biased or unbiased coin. We prove that, when X is uniformly distributed, $\mathrm{CI}(M) = \Omega(1)$. By itself, this bound only implies (trivial) space usage of at least one bit. However, the bound is nontrivial: it shows that the information cost *is nonzero even conditioned on the answer* (in this case, "uniform"). Together with structure of composable information, it allows us to derive lower bounds for our core task.

Our argument that lifts a memory bound for a 1-bit stream to k interleaved d-bit streams is inspired by the arguments of Braverman et al. (2020) (namely, lower bounds for the Simultaneous k-Coins Problem and the random-order k-Coins Problem). Our applications require extension and modification of these techniques, which we now describe.

In some of our proofs, it is easier to work with a subset of the terms in (1): we define $\widetilde{\mathrm{CI}}$ as just the "diagonal terms" of CI:

$$\widetilde{\mathrm{CI}}(M) \stackrel{\text{def}}{=} \sum_{t=1}^{N} I(M_t; X_t \mid M_{t-1}). \tag{2}$$

Since mutual information is nonnegative, we have $\mathrm{CI}(M) \geq \widetilde{\mathrm{CI}}(M)$. Furthermore, $\widetilde{\mathrm{CI}}$ allows us to consider a "full-memory" version of M that observes all previous states. We discuss this more in Section 4.

Our reductions use an algorithm for a "big" task to construct an algorithm for a "small" task. As is standard in information complexity arguments, this involves generating synthetic inputs to feed to the bigger algorithm alongside the real input. When these inputs are uniformly distributed, this introduces no overhead, but in our problems the inputs must be drawn from a specific distribution. This synthetic distribution is associated with a set of parameters F, which is itself a random variable and cannot be hard-wired into the algorithm. Our approach is simplified by conditioning on these parameters in the information cost itself, as

$$\mathrm{CI}(M \mid F) \stackrel{\text{def}}{=} \sum_{i=1}^{N} \sum_{t=i}^{N} I(M_t; X_i \mid M_{i-1}, F).$$

We define $\widetilde{\mathrm{CI}}(M \mid F)$ similarly. This means that our information complexity bounds are really measuring how an algorithm must store information about its input that is *not about the actual distribution*. In this sense, the bounds show that some form of memorization (in the spirit of Brown et al. (2021)) is needed at intermediate points in the computation.

Finally, we extend the treatment of random-order streams in Braverman et al. (2020). They apply a lower bound for the (single-coin) Coin Problem to prove a lower bound for the k-Coins Problem, where at each time step the learner receives an update from a randomly chosen coin. This theorem is then applied to data streaming problems (such as ℓ_2 Heavy Hitters) in the random-order model. This setting is similar to our core problem, and indeed our reduction is almost identical. We depart from Braverman et al. (2020) in the analysis: they define a notion of a "good" sequence of arrivals and, for any fixed good sequence, prove a lower bound for algorithms operating on that sequence. This suffices for their purposes but not for ours. For our learning task, an algorithm with advance knowledge of the sequence of arrivals can get by with lower information cost. Instead, in our analysis, we let the sequence $S \in [k]^N$ be a random variable, and prove a lower bound on the expression $\mathrm{CI}(M \mid S, F)$. Although we condition on S, we exploit the learner's uncertainty, at any given time, about the part of S it has not yet seen. This enables a more direct proof, which we believe may be useful in other random-order streaming applications.

2. Related Work

Lower Bounds Following Raz's Argument As mentioned in the introduction, one closely related line of work proves memory lower bounds for problems defined over finite fields, such as learning parities (Raz, 2018; Garg et al., 2018, 2019). The closest classes to the ones we consider are sparse parities. However, the techniques in the literature appear limited to analyzing parities that depend on at least $\log(d)$ variables. Lower bounds for these do not obviously imply bounds for the continuous function classes normally used in practice.

The algebraic bounds were generalized to rely on combinatorial conditions such as two-source extraction (Garg et al., 2018), mixing (Moshkovitz and Moshkovitz, 2017; Beame et al., 2018), and SQ dimension (Moshkovitz and Tishby (2017) and Gonen et al. (2020), following early work of Steinhardt et al. (2016)). These frameworks do not appear to yield nontrivial bounds for the our parameter settings, since the function classes we consider do not obviously satisfy the required combinatorial properties (in particular, they are very poor extractors and have low SQ dimension).

Lower Bounds for Single-Sample Distributed Regression The work on learning parity was inspired by a related line of work on learning in streaming and distributed models that focused

on regression-like problems (Steinhardt and Duchi, 2015; Braverman et al., 2016). The most directly relevant works consider k-sparse regression, showing lower bounds of $\tilde{\Omega}(d)$ on the memory required to learn with the minimal number of samples, $\tilde{\Theta}(k)$ (and lower bounds of $\tilde{\Omega}\big(d \cdot \frac{k}{N}\big)$ in general). Lower bounds such as ours, which exceed the data dimension, do not fit into these papers' distributed framework, since the protocol in which each player broadcasts their input always succeeds at learning and uses $O(d)$ bits of communication per player. That said, our bounds for the case $k = 1$ are essentially bounds on distributed regression. They require new proofs in order to bound an appropriately composable notion of information complexity.

Strong Lower Bounds for Restricted Parameter Ranges For linear regression in the streaming setting, Sharan, Sidford, and Valiant (2019) give a memory lower bound of $\Omega(d^2) = \Omega(d\kappa)$ as long as $N = o(d \log\log(1/\varepsilon))$, where ε is the accuracy parameter. This bound exceeds the size of a single example, but applies in a somewhat narrow range of sample complexity (or, seen as a lower bound on sample complexity, exceeds the unrestricted sample complexity of $\Theta(d)$ by a factor of $\log\log(1/\varepsilon)$). Independent work of Dagan et al. (2019) also gave a memory lower bound of $\Omega(d^2)$ for linear regression; although not stated as a distributional problem, it can be interpreted as applying to streams of length exactly $N = d - 1$.

Dagan and Shamir (2018) give memory-sample tradeoffs for statistical estimation tasks, building on prior work by Shamir (2014). Their key theorem gives lower bounds for distinguishing between certain families of distributions. An example is the set of distributions over $\{-1, +1\}^d$ where a single pair of indices i, j satisfies $\mathbb{E}[x_i x_j] = \varepsilon$ and all other pairs satisfy $\mathbb{E}[x_{i'} x_{j'}] = 0$. They prove that a streaming algorithm that solves this task with N samples requires memory size $\tilde{\Omega}(d^2 \cdot \frac{1/\epsilon^2}{N})$ for very low error, $\varepsilon = \tilde{O}(d^{-1/3})$. Their bound is matched (up to logarithmic factors) by upper bounds in the low-correlation regime they consider; it also degrades gracefully with the number of passes over the stream. While their techniques can be extended to supervised learning of some hypothesis classes (such as 1-sparse linear predictors that use degree-2 monomials), they do not extend to the constant-error regime we consider.

Streaming Lower Bounds for Other Statistical Problems The work whose techniques we use most directly is that of Braverman, Garg, and Woodruff (2020) (extended by Braverman, Garg, and Zamir (2021)). They considered the following *k-coin problem*: suppose there are k different coins. The algorithm receives a stream of examples of the form (j, b) where b is the result of a fresh toss of coin j. Its goal is to determine whether all the coins are fair or if some coin has bias greater than $\beta \approx \frac{1}{2} + \frac{1}{\sqrt{n}}$. We adopt the composable measure of information cost developed in that paper, and its use is critical in obtaining our final bounds.

Diakonikolas, Gouleakis, Kane, and Rao (2019) give a lower bound on testing uniformity of a distribution and related problems. They show that a streaming algorithm which, given N samples drawn i.i.d. from a distribution over $[2^d]$, reliably distinguishes the uniform distribution from a distribution that is $\Omega(1)$-far from uniform must use memory $\tilde{\Omega}\Big(\frac{2^d}{N}\Big)$ for $N \geq 2^{d/2}$. Although the lower bound far exceeds the size of any one sample and of the (one-bit) output, we are not aware of how to use it to derive a nontrivial bound for our setting.

Concurrent work of Marsden et al. (2022) proved memory lower bounds for convex optimization algorithms that use first-order queries, showing that any algorithm using significantly less than $d^{1.25}$ bits of memory requires a polynomial factor more queries than a memory-unconstrained algorithm in the low-error regime (i.e., $\varepsilon = o(d^{-6})$ for their lower bound to apply). The computational

setting of first-order queries is significantly different from the general learning setting we consider; understanding the models' relationship is an interesting topic for future work.

Memorization. In a batch setting, Brown et al. (2021) established learning tasks for which any high-accuracy learning algorithm, on receiving a data set X in $(\{0,1\}^d)^N$, must output a classifier M that satisfies $I(M;X \mid F) \geq \Omega(Nd)$, where F is the set of parameters of the data-generating distribution. As mentioned before, we rely on a modification of their hard distribution, but our techniques and results are significantly different. They lower bound the size of the learned hypothesis (this implies a lower bound on the space usage of the algorithm). Our results apply directly to the space needed by the algorithm, even when the hypothesis is itself small. The hard distribution in Brown et al. (2021) is tailored to the sample size, while our lower bounds fix the distribution and apply to a wide range of stream lengths. Finally, our lower bounds apply to any learner with a small constant advantage over random guessing instead of only learners with nearly Bayes-optimal accuracy. Identifying deeper connections between space lower bounds and memorization (Brown et al., 2021; Bassily et al., 2018; Livni and Moran, 2020) remains a fascinating open question.

3. Task Definitions

We define the core distributional task we study, as well as the simpler intermediate problems we analyze in order to understand it. We define and analyze an additional problem, Task B', in Appendix D. Here, and throughout the paper, we use the notation $\Delta(S)$ to refer to set of distributions over a set S and the notation $\mathcal{U}$ to refer to a uniform distribution (with the space implied by context). We use *"U"* and *"S"* to denote the outputs referring to "uniform" and "structured," respectively.

Definition 2 (Meta-Distributions) *Parameters: positive integers $k, d, \rho \leq d$. We sample distribution $\mathcal{P} \in \Delta(\{0,1\}^d)$ from the* structured subpopulation meta-distribution $\mathcal{Q}_{d,\rho} \in \Delta(\Delta(\{0,1\}^d))$ *as follows:*

- *Draw $r \in \{0,\ldots,\rho\}$ uniformly.*
- *Draw uniformly $\mathcal{J} = \{j_1,\ldots j_r\} \subseteq [d]$ (no replacement) and $\mathcal{B} = (b_1,\ldots,b_r) \in \{0,1\}^r$.*

To draw $x \in \{0,1\}^d$ from $\mathcal{P} = \mathcal{P}_{(\mathcal{J},\mathcal{B})}$, set $x_{j_i} \leftarrow b_i$ for each $j_i \in \mathcal{J}$. For $j \notin \mathcal{J}$, set $x_j \in \{0,1\}$ uniformly and independently.[2]

We sample distribution $\mathcal{P}_{\text{mix}} \in \Delta([k]\times\{0,1\}^d)$ from the structured population meta-distribution $\mathcal{Q}_{k,d,\rho} \in \Delta(\Delta([k] \times \{0,1\}^d))$ *as follows:*

- *For $j = 1,...,k$, draw $\mathcal{P}_j \sim \mathcal{Q}_{d,\rho}$ and let $F_j = (\mathcal{J}_j, \mathcal{B}_j)$ denote its parameters.*

Define $\mathcal{P}_{\text{mix}} \in \Delta([k] \times \{0,1\}^d)$ as a uniform mixture of k distributions of the form $\delta_j \otimes \mathcal{P}_j$, with δ_j denoting the point mass on j. Let F denote the pair $(\mathcal{J},\mathcal{B})$, the parameters of $\mathcal{P}$. Let $F_{\text{mix}} = (F_1,...,F_k)$ denote the parameters of $\mathcal{P}_{\text{mix}}$. We call both $\mathcal{P}$ and $\mathcal{P}_{\text{mix}}$ structured *distributions.*

2. That is, $\mathcal{P}$ is uniform over the $n-r$-dimensional hypercube specified by $\mathcal{J}$ and $\mathcal{B}$.

Definition 3 (Task Definitions) *N, T, k, d and $\rho \leq d$ are positive integer parameters.*

1. *(Task C, Core Problem) Learning algorithm M receives a stream of N i.i.d. samples from $\mathcal{P}_{\text{mix}}$. After N steps, the learner outputs a (possibly randomized) function $m : [k] \times \{0,1\}^d \to \{\text{“}U\text{”}, \text{“}S\text{”}\}$. The* advantage *of the learner is*

$$\mathop{\mathbb{E}}_{\substack{\mathcal{P}_{\text{mix}} \sim \mathcal{Q}_{k,d,\rho} \\ x=(x_1,\ldots,x_N) \sim_{\text{iid}} \mathcal{P}_{\text{mix}} \\ m \leftarrow M(x)}} \left[\Pr_{(j,y)\sim\mathcal{P}_{\text{mix}}}[m(j,y) = \text{“}S\text{”}] - \Pr_{(j,y)\sim\mathcal{U}}[m(j,y) = \text{“}S\text{”}] \right].$$

2. *(Task B, Single Subpopulation) Learner M receives a stream of T examples from $\mathcal{P}$, with each example in $\{0,1\}^d$, and outputs a value in $\{\text{“}U\text{”}, \text{“}S\text{”}\}$. The* advantage *of the learner is*

$$\Pr_{\substack{\mathcal{P}\sim\mathcal{Q}_{d,\rho} \\ x=(x_1,\ldots,x_N)\sim_{\text{iid}}\mathcal{P}}}[M(x) = \text{“}S\text{”}] - \Pr_{x=(x_1,\ldots,x_N)\sim_{\text{iid}}\mathcal{U}}[M(x) = \text{“}S\text{”}].$$

3. *(Task A, One-Bit Stream) The learner M receives a stream of T bits and outputs a value in $\{\text{“}U\text{”}, \text{“}S\text{”}\}$. The* advantage *of the learner is*

$$\Pr_{\substack{b\sim\mathcal{U} \\ x=(b,b,\ldots,b)}}[M(x) = \text{“}S\text{”}] - \Pr_{x=(x_1,\ldots,x_T)\sim_{\text{iid}}\mathcal{U}}[M(x) = \text{“}S\text{”}].$$

4. A Lower Bound for the One-Bit Stream Task

Theorem 4 *Consider a streaming algorithm M for Task A on T inputs that has advantage at least δ. Let $M_1, \ldots, M_T$ be the memory states of M when run on uniformly random inputs $X_1, \ldots, X_T \in \{0,1\}$. Then*

$$\widetilde{\text{CI}}(M) \stackrel{def}{=} \sum_{t=1}^{T} I(M_t; X_t \mid M_{t-1}) \geq \frac{\delta^4}{40}.$$

Recall that, although by itself this lower bound only implies a trivial space lower bound of $\Omega(1)$ bits, it is nontrivial: we measure information about the stream *when the stream is uniform*. In other words, the algorithm must contain information about the stream, even with respect to an observer who knows the answer. In later sections, we use this statement to prove lower bounds for the “larger” tasks. The proof has a simple but subtle setup. We now outline our approach; the full proof is in Appendix B.

By assumption, M has advantage δ. Since the structured data distribution chooses to fix the stream to 0 or 1 with equal probability, there exists a value $b^* \in \{0,1\}$ such that

$$\Pr[M(b^* \cdot 1^T) = \text{“}S\text{”}] - \Pr_{x_{\leq T}\sim\mathcal{U}}[M(x_{\leq T}) = \text{“}S\text{”}] \geq \delta, \tag{3}$$

where 1^T denotes the length-T vector of ones. For simplicity we assume $b^* = 1$; the calculations do not rely on this choice. Consider a stream generated from the following distribution, distinct from the structured distribution The distribution over streams is a mixture of two distributions, with the first mixture component being uniform over all streams (corresponding to i.i.d. bits). The second mixture component places all of its mass on the stream with every entry equal to $b^* = 1$. Let E

denote the event that this stream is drawn from the second component, and $\bar{E}$ the event that it is drawn from the first.

We now consider an observer that, at time t, sees $M_{\leq t}$, all the previous states of M.[3] At each time t, the observer computes a posterior belief about event E:

$$p_t \stackrel{\text{def}}{=} \Pr[E \mid M_{\leq t} = m_{\leq t}],$$

with prior $p_0 = \frac{1}{2}$. We analyze the random process $P_0, P_1, \ldots, P_T$ formed by these posteriors when the inputs $X_1, \ldots, X_T$ are uniformly random. This process depends on the inputs and any random choices of the algorithm. By construction the sequence begins at $\frac{1}{2}$ and, as we prove in Appendix B, on average decreases by about δ^2 over T time steps.

Lemma 5 *Assume algorithm M has advantage at least δ. Let random variable P_T be the final posterior of M', i.e., $P_T = \Pr[E \mid M_{\leq T}]$. When the inputs are uniform, we have $\mathbb{E}[P_T] \leq \frac{1}{2} - \frac{\delta^2}{4}$.*

We will looks at the "gaps" of the form $P_t - P_{t-1}$, the difference between consecutive time steps. Lemma 5 shows that the random process, on average, must have a few time steps with large gaps or many time steps with moderate gaps. We will show how these gaps reveal information about the input bits.

We restate the proof setup for emphasis. The algorithm M has δ advantage in the setup of Definition 3, where the stream may be fixed to 1's, fixed to 0's, or uniform. The observer is told that it sees M run on either the stream fixed to $b^* = 1$ or the uniform stream; it calculates a posterior belief about which case is true. In this proof, we analyze the sequence of these posteriors when the stream is actually uniform.

Let us zoom in on a single time step t. Fix the previous memory states $m_{<t}$: this also fixes the posterior p_{t-1} and the distribution over the next posterior P_t. This distribution is a uniform mixture of two components: let $\mathcal{D}_{1,m_{<t}}$ be the distribution over P_t conditioned on $X_t = 1$, and let $\mathcal{D}_{0,m_{<t}}$ be the distribution conditioned on $X_t = 0$. When X_t is uniform, P_t is thus drawn from the mixture $\frac{1}{2}(\mathcal{D}_{1,m_{<t}} + \mathcal{D}_{0,m_{<t}})$. Denote the means of these distributions $\mu_{1,m_{<t}}$ and $\mu_{0,m_{<t}}$. We have $\mathbb{E}[P_t \mid m_{<t}] = \frac{1}{2}(\mu_{1,m_{<t}} + \mu_{0,m_{<t}})$ and, furthermore, we prove that $p_{t-1} \leq \mu_{1,m_{<t}}$. This yields the following lemma, connecting the gaps in the posterior to the difference in the means of the mixture components.

Lemma 6 *For some time t, fix the memory states $m_{<t}$. Let $\mu_{1,m_{<t}} = \mathbb{E}[P_t \mid X_t = 1, M_{<t} = m_{<t}]$ and $\mu_{0,m_{<t}} = \mathbb{E}[P_t \mid X_t = 0, M_{<t} = m_{<t}]$. We have*

$$\mathbb{E}[P_t - P_{t-1} \mid M_{<t} = m_{<t}] \geq \frac{\mu_{0,m_{<t}} - \mu_{1,m_{<t}}}{2}.$$

Both sides of the inequality are negative. When the distributions $\mathcal{D}_{1,m_{<t}}$ and $\mathcal{D}_{0,m_{<t}}$ are distinct, the posterior P_t will contain some information about the input X_t; this lemma gives us a foothold to connect the gaps in the posterior with the difference in distributions.

We work directly with the information contained in the posterior. Lemma 19 allows us to, in the expression for $\widetilde{\mathrm{CI}}$, condition on all previous memory states. Since P_t is a function of these states,

3. As we make precise below, Lemma 19 states that moving to an observer with access to all previous states (instead of just the current state) does not affect our argument.

by the data processing inequality it contains no more information about X_t and we have

$$\widetilde{\mathrm{CI}}(M) = \sum_{t=1}^{T} I(M_t; X_t \mid M_{t-1}) = \sum_{t=1}^{T} I(M_t; X_t \mid M_{<t}) \geq \sum_{t=1}^{T} I(P_t; X_t \mid M_{<t}).$$

The precise notion of "difference in distributions" we need is the *Jensen-Shannon divergence*, denoted $\mathrm{JSD}(p \parallel q)$ and defined in Appendix A. For uniform random variable $U \in \{0,1\}$ and random variable A that is distributed according to p when $U = 0$ and distributed according to q when $U = 1$, Fact 18 states that we have $\mathrm{JSD}(p \parallel q) = I(U; A)$. Thus we have

$$\sum_{t=1}^{T} I(P_t; X_t \mid M_{<t}) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{m<t} \left[\mathrm{JSD}(\mathcal{D}_{1,m_{<t}} \parallel \mathcal{D}_{0,m_{<t}})\right], \tag{4}$$

Looking at (4), we might hope to prove a statement such as $\mathrm{JSD}(\mathcal{D}_{1,m_{<t}} \parallel \mathcal{D}_{0,m_{<t}}) \stackrel{?}{=} \Omega(|\mu_{1,m_{<t}} - \mu_{0,m_{<t}}|)$. If this were the case, we would be done: by Lemma 6, these mean-gaps on average are of size δ^2/T. However, this statement is false.[4] Instead, we prove the following lower bound on Jensen-Shannon divergence, which is tight up to the leading constant. Although the proof is elementary, we are not aware of the statement appearing previously.

Lemma 7 *Let $\mathcal{D}_0$ and $\mathcal{D}_1$ be distributions over $\mathbb{R}$ with means μ_0 and μ_1 respectively, and let $\mathcal{D} = \frac{\mathcal{D}_0 + \mathcal{D}_1}{2}$. Then*

$$\mathrm{JSD}(\mathcal{D}_0 \parallel \mathcal{D}_1) \geq \frac{1}{8} \cdot \frac{(\mu_0 - \mu_1)^2}{\mathrm{Var}(\mathcal{D})}.$$

Plugging this lower bound into Equation (4), we see that our final task is to show that variances are not too large on average. Formally, we show that an average choice of t and $m_{<t}$ yields a distribution over P_t with variance $O(1/T)$.

Lemma 8 *For every algorithm M solving Task A, we have $\mathbb{E} \sum_{t=1}^{T} \mathrm{Var}\left(P_t \mid M_{<t}\right) \leq \frac{5}{4}$. Here $\mathrm{Var}\left(P_t \mid m\right)$ denotes the conditional variance $\mathbb{E}_{P_t}\left[\left(P_t - \mathbb{E}[P_t \mid M_{<t} = m]\right)^2\right]$.*

We present the main idea in the proof of Lemma 8. Let $\Delta_t = P_t - P_{t-1}$, the increment in the posterior. Note that $\mathrm{Var}\left(\Delta_t\right) = \mathbb{E}_{M_{<t}}[\mathrm{Var}\left(P_t \mid M_{<t}\right)]$, since fixing $M_{<t} = m_{<t}$ also fixed $P_{t-1} = p_{t-1}$. If $\Delta_1, \ldots, \Delta_T$ were (pairwise) uncorrelated, then we would have $\mathrm{Var}\left(P_t\right) = \sum_{t=1}^{T} \mathrm{Var}\left(\Delta_t\right)$. Since $P_T \in [0,1]$, its variance is at most $\frac{1}{4}$, so the average time step would have $\mathrm{Var}\left(P_t \mid M_{<t}\right) = O(1/T)$. This argument does not hold for random processes in general: correlations between time steps may introduce "additional" variance that does not appear in P_T. To show that these correlations do not affect our result too much, we first show that the process we consider is decreasing on average (formally, that $P_0, P_1, \ldots, P_T$ form a *supermartingale*). This implies that the correlations between time steps all "point in the same direction" and only contribute a limited amount of additional variance.

4. As a counterexample, let p and q be the following distributions: $p(0) = q(1) = \frac{1+\alpha}{2}$, and $p(1) = q(0) = \frac{1-\alpha}{2}$, for some $\alpha > 0$. Then the difference in means is α, but a quick calculation shows that $\mathrm{JSD}(p \parallel q) = O(\alpha^2)$.

5. A Lower Bound for a Single-Subpopulation Task

We show how to turn the previous lower bound for the one-bit stream task into a lower bound for Task B, where each example is in $\{0,1\}^d$ and has some number of fixed bits, with the other bits uniformly random. The number of fixed bits is either 0 (w.p. $1/2$) or uniform within $\{0,\dots,\rho\}$ for some known ρ (otherwise). Let M be an algorithm for Task B. Define

$$\pi_r \stackrel{\text{def}}{=} \Pr[M = \text{“}S\text{”} \mid r \text{ fixed bits}].$$

Observe that this is well-defined even for $r > \rho$. We construct a learner M' for Task A, the single-stream problem, that takes advantage of the average gaps between π_{r+1} and π_r. M' randomly selects $r \in \{0,\dots,\rho\}$ and then uniformly picks r indices $\mathcal{J} \subseteq [d]$ (without replacement) and values $\mathcal{B} \in \{0,1\}^r$. At each time step t, M' provides M with an example z_t where the features $j_i \in \mathcal{J}$ are fixed to bit b_i. Furthermore, M' selects an index $j_0 \in [d]$ at which to insert the one-bit input stream. Note that if $j_0 \in \mathcal{J}$ the input stream is ignored; this is necessary for our information argument. At the end of the stream, M' uses the output of M and knowledge of π_r and π_{r+1} to produce a guess for whether the one-bit stream is fixed. See Algorithm 1 in Appendix C for a formal description of this reduction.

We argue in Appendix C that M' has advantage roughly δ/ρ when the advantage of M is δ. The proof, in effect, makes use of a hybrid argument: Algorithm 1 creates r fake fixed streams for a uniformly random $0 \le r \le \rho$. When the input stream is uniform, the distribution generated in the reduction will have r fixed streams, and when the input stream is fixed the distribution will (usually) have $r+1$ fixed streams. The minor caveat is that, with probability $\frac{r}{d}$, M' will select the special index j_0 to be fixed and “overwrite” its own input. When $r \ll d$ this does not significantly increase the error.

The above algorithm is carefully constructed so that, *when the input stream is uniform*, the distribution fed to M matches the distribution in Task B, even conditioned on the location of the input stream J_0. To highlight this crucial fact, we present it separately.

Fact 9 *The random variables $\mathcal{J}$ and $\mathcal{B}$ are independent of J_0. Since $\mathcal{J}$ and $\mathcal{B}$ determine the distributions of $M_{\le T}$ and $Z_{\le T}$ in the algorithm above, when the inputs $X_{\le T}$ are i.i.d. uniform bits we have $J_0 \perp (\mathcal{J}, \mathcal{B}, M_{\le T}, Z_{\le T})$.*

Lemma 10 (Information Cost) *For algorithm M solving Task B on a stream of length T, let M' solving Task A on a stream of length T be as defined above (see Algorithm 1 in Appendix C for a formal description). Let inputs $X_t \in \{0,1\}$ be uniform and let $Z_t \in \{0,1\}^d$ be structured with parameters F (see Definition 2). We have*

$$\widetilde{\mathrm{CI}}(M \mid F) \ge d \cdot \widetilde{\mathrm{CI}}(M').$$

Proof To generate inputs for M, M' stores random variable J_0, the location of the input stream, and the locations and values of the fake fixed streams: $\mathcal{J}$ and $\mathcal{B}$. Write $F = (\mathcal{J}, \mathcal{B})$, so the state of M' is $M'_t = (M_t, F, J_0)$ and thus

$$\begin{aligned}\sum_{t=1}^{T} I(M'_t; X_t \mid M'_{t-1}) &= \sum_{t=1}^{T} I(M_t; X_t \mid M_{t-1}, F, J_0) \\ &= \frac{1}{d}\sum_{t=1}^{T}\sum_{j=1}^{d} I(M_t; X_t \mid M_{t-1}, F, J_0 = j),\end{aligned}$$

writing out the expectation over $J_0 = j$. Next, note that M_t depends on X_t only through Z_t^j. Formally, conditioning on $J_0 = j$ and any values of M_{t-1} and F, we have the Markov chain M_t—Z_t^j—X_t. So we apply the data processing inequality and have

$$\widetilde{\mathrm{CI}}(M') \le \frac{1}{d}\sum_{t=1}^{T}\sum_{j=1}^{d} I(M_t; Z_t^j \mid M_{t-1}, F, J_0 = j).$$

Crucially, this expression involves only terms generated by M' as part of the reduction.

By Fact 9, we can remove the condition that $J_0 = j$, since for any j the tuple (M_t, Z_t^j, M_{t-1}, F) is independent of the location in which M' inserts the input stream. We have

$$\widetilde{\mathrm{CI}}(M') \le \frac{1}{d}\sum_{t=1}^{T}\sum_{j=1}^{d} I(M_t; Z_t^j \mid M_{t-1}, F).$$

When we condition on F, the collection of random variables $Z_t^{<j}$ is independent of Z_t^j. Via Fact 13, we can condition on $Z_t^{<j}$ and apply the chain rule over $Z_t = (Z_t^j)_{j\in[k]}$ to finish the proof:

$$\begin{aligned}\widetilde{\mathrm{CI}}(M') &\le \frac{1}{d}\sum_{t=1}^{T}\sum_{j=1}^{d} I(M_t; Z_t^j \mid M_{t-1}, F, Z_t^{<j}) \\ &= \frac{1}{d}\sum_{t=1}^{T} I(M_t; Z_t \mid M_{t-1}, F) = \frac{1}{d}\widetilde{\mathrm{CI}}(M \mid F).\end{aligned}$$

■

The following lower bound follows directly from the lower bound of Theorem 4 and the reduction presented in this section (Appendix C's accuracy argument in Lemma 29 and the information cost argument in Lemma 10).

Corollary 11 *Any algorithm M solving Task B on T examples with advantage at least δ satisfies*

$$\widetilde{\mathrm{CI}}(M \mid F) \stackrel{def}{=} \sum_{t=1}^{T} I(M_t; X_t \mid M_{t-1}, F) \ge \frac{d}{40}\left(\frac{\delta}{\rho+1} - \frac{\rho}{d}\right)^4.$$

6. A Lower Bound for the Core Problem

In Section 5, we saw that the lower bound for distinguishing uniform-versus-fixed one-bit streams implies a lower bound for distinguishing uniform-versus-structured d-bit strings, where the structured distribution is defined in Definition 2. A simple argument shows that this, in turn, implies a lower bound for the associated "test example" problem, where the learner is given a stream of structured inputs and then, at test time, is given an example that is either structured or uniform. We define this problem (Task B') and present the argument in Appendix D.

In this section we sketch the argument that this lower bound implies a lower bound on Task C. The proof, presented in Appendix E, is delicate, and we believe it may find applications in other random-order streaming problems.

Theorem 12 (Lower Bound for Task C) *Any algorithm M solving Task C with parameters $k, d,$ and ρ with advantage at least δ satisfies*

$$\mathrm{CI}(M \mid S, F) = \sum_{i=1}^{N} \sum_{t=i}^{N} I(M_t; X_i \mid M_{i-1}, S, F) \geq \frac{k^2 d}{320} \left(\frac{\delta}{\rho + 1} - \frac{\rho}{d} \right)^4 - O(k^2) - O(kd).$$

Here the inputs X_i are structured (Definition 3).

Recall that large composable information cost implies M uses a lot of space. Formally, Lemma 20 states that $\frac{1}{N} \cdot \mathrm{CI}(M \mid S, F) \leq \max_{t \in [N]} |M_t|$. Thus, for $\rho = o(d^{1/4})$ and constant δ we get a space lower bound of $\Omega\left(\frac{k^2 d}{N \rho^4}\right)$.

The proof of this theorem uses a similar reduction to that in Braverman et al. (2020)'s proof of the k-Coins Problem, but our analysis differs significantly. The reduction uses an algorithm M solving Task C to construct an algorithm M' solving Task B' (the single-subpopulation version of the problem). We pick a random index $j \in [k]$ and sequence of arrivals $s \in [k]^N$, insert the input stream whenever $s_t = j$, and generate the other examples from $k - 1$ "synthetic" subpopulations.

Informally, we show that $\mathrm{CI}(M \mid S, F)$ is k^2 times larger than the composable information cost of M'. The first factor of k arises because M does not "know" the location of the true input stream: it must solve all subpopulations at once.

The second factor of k is the crux of the proof, and we present it here.[5] Fix a sequence of arrivals up until time t_0, when we received an example from subpopulation j. Let T_1 denote the time of the next arrival from that subpopulation; it is a random variable. After some manipulation, we arrive at a sum of terms, each with the form (for some $t \geq t_0$):

$$\Pr[T_1 > t] \cdot I(M_t; X_{t_0} \mid M_{t_0 - 1}, T_1 > t). \tag{5}$$

Then we observe two facts. First, we have that $\Pr[T_1 > t] = k \cdot \Pr[T_1 = t + 1]$, since $T_1 - t_0$ is a geometric random variable; at every time step we see an arrival from subpopulation j with probability $\frac{1}{k}$. Second, since the algorithm cannot depend on events that happen *after* time t, it cannot distinguish the event $T_1 > t$ from the event $T_1 = t + 1$, so we have that (5) is equal to

$$k \cdot \Pr[T_1 = t + 1] \cdot I(M_t; X_{t_0} \mid M_{t_0 - 1}, T_1 = t + 1).$$

This equality introduces the second factor of k and facilitates the move from discussion information about M to information about M', which must account for the information M stores at time $T_1 - 1$.

Acknowledgements

We are grateful to Vitaly Feldman and Kunal Talwar for their insight and feedback throughout this project. In particular, they helped articulate conjectures that evolved into the theorems in this paper. The final product would have looked very different without their involvement.

5. We elide several details, including additional conditions in the information terms and technicalities in dealing with the end of the stream.

References

Peter L. Bartlett, Andrea Montanari, and Alexander Rakhlin. Deep learning: a statistical viewpoint. *Acta Numerica*, 30:87–201, 2021. doi: 10.1017/S0962492921000027.

Raef Bassily, Shay Moran, Ido Nachum, Jonathan Shafer, and Amir Yehudayoff. Learners that use little information. In *Algorithmic Learning Theory*, pages 25–55. PMLR, 2018.

Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-space tradeoffs for learning finite functions from random evaluations, with applications to polynomials. In *Conference On Learning Theory*, pages 843–856. PMLR, 2018.

Mark Braverman, Ankit Garg, Tengyu Ma, Huy L Nguyen, and David P Woodruff. Communication lower bounds for statistical estimation problems via a distributed data processing inequality. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 1011–1020, 2016.

Mark Braverman, Sumegha Garg, and David P Woodruff. The coin problem with applications to data streams. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 318–329. IEEE, 2020.

Mark Braverman, Sumegha Garg, and Or Zamir. Tight space complexity of the coin problem. In *IEEE Symposium on the Foundations of Comptuer Science (FOCS)*, 2021.

Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is memorization of irrelevant training data necessary for high-accuracy learning? In *ACM Symposium on the Theory of Computing (STOC)*, 2021.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Thomas M Cover and Joy A Thomas. Elements of information theory john wiley & sons. *New York*, 68:69–73, 1991.

Yuval Dagan and Ohad Shamir. Detecting correlations with little memory and communication. In *Conference On Learning Theory*, pages 1145–1198. PMLR, 2018.

Yuval Dagan, Gil Kur, and Ohad Shamir. Space lower bounds for linear prediction in the streaming model. In *Conference on Learning Theory*, pages 929–954. PMLR, 2019.

Ilias Diakonikolas, Themis Gouleakis, Daniel M Kane, and Sankeerth Rao. Communication and memory efficient testing of discrete distributions. In *Conference on Learning Theory*, pages 1070–1106. PMLR, 2019.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? *arXiv preprint arXiv:2009.08576*, 2020.

Ankit Garg, Tengyu Ma, and Huy Nguyen. On communication cost of distributed statistical estimation and dimensionality. *Advances in Neural Information Processing Systems*, 27:2726–2734, 2014.

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*, pages 990–1002, 2018.

Sumegha Garg, Ran Raz, and Avishay Tal. Time-Space Lower Bounds for Two-Pass Learning. In *34th Computational Complexity Conference (CCC 2019)*, volume 137, pages 22:1–22:39, 2019.

Alon Gonen, Shachar Lovett, and Michal Moshkovitz. Towards a combinatorial characterization of bounded-memory learning. *Advances in Neural Information Processing Systems*, 33:9028–9038, 2020.

Roi Livni and Shay Moran. A limitation of the PAC-Bayes framework. *Advances in Neural Information Processing Systems*, 33, 2020.

Annie Marsden, Vatsal Sharan, Aaron Sidford, and Gregory Valiant. Efficient convex optimization requires superlinear memory. *arXiv preprint arXiv:2203.15260*, 2022.

Dana Moshkovitz and Michal Moshkovitz. Mixing implies lower bounds for space bounded learning. In *Conference on Learning Theory*, pages 1516–1566. PMLR, 2017.

Michal Moshkovitz and Naftali Tishby. A general memory-bounded learning algorithm. *arXiv preprint arXiv:1712.03524*, 2017.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. *Journal of the ACM (JACM)*, 66(1):1–18, 2018.

Ohad Shamir. Fundamental limits of online and distributed algorithms for statistical learning and estimation. *Advances in Neural Information Processing Systems*, 27:163–171, 2014.

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*, pages 890–901, 2019.

Jacob Steinhardt and John Duchi. Minimax rates for memory-bounded sparse linear regression. In *Conference on Learning Theory*, pages 1564–1587. PMLR, 2015.

Jacob Steinhardt, Gregory Valiant, and Stefan Wager. Memory, communication, and statistical queries. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 1490–1516. JMLR.org, 2016. URL `http://proceedings.mlr.press/v49/steinhardt16.html`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Appendix A. Preliminaries

In this section, we present basic facts and tools used in the paper. For additional definitions and background, see a reference such as Cover and Thomas (1991). Lemma 19 is new to this paper and Lemma 20 is slight modification of a statement in Braverman et al. (2020).

Fact 13 *Let A, B, and C be random variables. Suppose A and C are independent. Then we have that $I(A;B) \le I(A;B \mid C)$.*

Fact 14 *Let A, B, and C be random variables. Then $I(A;B \mid C) \le I(A;B) + I(A;C \mid B)$.*

These facts are easily proved by writing out the chain rule two different ways and using the nonnegativity of mutual information:

$$\begin{aligned} I(A;B,C) &= I(A;B) + I(A;C \mid B) \\ &= I(A;C) + I(A;B \mid C). \end{aligned}$$

A standard property of mutual information is the "data processing inequality," which says that if random variables X—Y—Z form a Markov chain, then $I(X;Z) \le I(Y;Z)$. Streaming algorithms are a special case of this, so we have the following.

Proposition 15 (DPI for Streaming) *For any streaming algorithm M (using only private randomness at each time step) run on i.i.d. inputs $\{X_t\}$ and any time steps $t_1 \le t_2 \le t_3$, we have*

$$I(M_{t_3}; X_{t_1} \mid M_{t_1-1}) \le I(M_{t_2}; X_{t_1} \mid M_{t_1-1}).$$

We require several ways to measure differences in probability distributions.

Definition 16 (Distances and Divergences) *Let p and q be probability distributions over the same space $\mathcal{X}$. Define the following:*

- Kullback-Leibler divergence*:* $\mathrm{KL}(p \parallel q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$.
- Jensen-Shannon divergence*:* $\mathrm{JSD}(p \parallel q) = \frac{1}{2}\mathrm{KL}\left(p \parallel \frac{p+q}{2}\right) + \frac{1}{2}\mathrm{KL}\left(q \parallel \frac{p+q}{2}\right)$.
- Total variation distance*:* $\mathrm{TV}(p,q) = \frac{1}{2}\sum_{x \in \mathcal{X}} |p(x) - q(x)|$.
- Squared Hellinger distance*:* $H^2(p,q) = 1 - \sum_{x \in \mathcal{X}} \sqrt{p(x)q(x)}$.

Fact 17 *For any distributions p and q, we have $\mathrm{TV}(p,q) \le \sqrt{2}H(p,q)$.*

We use the fact that two of our measures relate directly to the problem of distinguishing distributions.

Fact 18 *Let $B \in \{0,1\}$ be a uniform random variable. For distributions p and q, let random variable X satisfy $X \sim p$ if $B = 0$ and $X \sim q$ otherwise.*

(i) $I(X;B) = \mathrm{JSD}(p \parallel q)$.

(ii) Let $\mathcal{F}$ be the space of functions $f : \mathcal{X} \to \{0,1\}$. We have

$$\max_{f \in \mathcal{F}} \Pr[f(X) = B] = \frac{1 + \mathrm{TV}(p,q)}{2}.$$

$\widetilde{\mathrm{CI}}(M)$, as defined in Equation (2), contains information terms that condition on the previous memory state. Our proof of Theorem 4 uses the following lemma, which says that we may replace this term with one that conditions on all previous memory states.

Lemma 19 *Consider any streaming task where the inputs $X_1, \ldots, X_T$ are independent. For any streaming algorithm M and any time step $t \in [T]$, we have*

$$I(M_t; X_t \mid M_{t-1}) = I(M_t; X_t \mid M_{<t}).$$

Proof Observe that we have the Markov chain $M_{<t-1}$—M_{t-1}—M_t—X_t. Using the chain rule, we have

$$\begin{aligned} I(M_{\le t}; X_t) &= I(M_{t-1}; X_t) + I(M_t; X_t \mid M_{t-1}) + I(M_{<t-1}; X_t \mid M_{t-1}, M_t) \\ &= 0 + I(M_t; X_t \mid M_{t-1}) + 0. \end{aligned}$$

The first term is zero because the stream is i.i.d., and the third term is zero because of the Markov chain. Applying the chain rule again in a different order, and using the fact that $M_{<t} \perp X_t$, we have

$$\begin{aligned} I(M_{\le t}; X_t) &= I(M_{<t}; X_t) + I(M_t; X_t \mid M_{<t}) \\ &= 0 + I(M_t; X_t \mid M_{<t}). \end{aligned}$$

So the two information terms are equal. ■

Lemma 20 *In Task C, the core task, let random variables S and F be the sequence of subpopulation arrivals and subpopulation parameters, respectively. For any algorithm M and time step $t \in [N]$ we have*

$$\sum_{i=1}^{t} I(M_t; X_i \mid M_{i-1}, S, F) \le |M_t| \tag{6}$$

and thus

$$\frac{1}{N} \cdot \mathrm{CI}(M \mid S, F) \le \max_{t \in [N]} |M_t|. \tag{7}$$

Proof First we show that Equation (6) immediately implies Equation (7). Reordering the terms in $\mathrm{CI}(M \mid S, F)$, we see that

$$\begin{aligned}\frac{1}{N} \cdot \mathrm{CI}(M \mid S, F) &= \frac{1}{N} \sum_{i=1}^{N} \sum_{t=i}^{N} I(M_t; X_i \mid M_{i-1}, S, F) \\ &= \frac{1}{N} \sum_{t=1}^{N} \sum_{i=1}^{t} I(M_t; X_i \mid M_{i-1}, S, F) \\ &\leq \mathop{\mathbb{E}}_{t}[|M_t|] \leq \max_{t \in [N]} |M_t|.\end{aligned}$$

It remains to prove Equation (6). We have $I(X_i; M_{<i-1}, X_{<i} \mid M_{i-1}, S, F) = 0$ (since X_i depends only on F and S). We apply Fact 13 and have the inequality

$$\sum_{i=1}^{t} I(M_t; X_i \mid M_{i-1}, S, F) \leq \sum_{i=1}^{t} I(M_t; X_i \mid M_{i-1}, M_{<i-1}, X_{<i} S, F).$$

Next we add another (nonnegative) mutual information term and apply the chain rule twice:

$$\begin{aligned}\sum_{i=1}^{t} I(M_t; X_i \mid M_{i-1}, S, F) &\leq \sum_{i=1}^{t} \Big[I(M_t; X_i \mid M_{i-1}, M_{<i-1}, X_{<i}, S, F) \\ &\qquad + I(M_t; M_{i-1} \mid M_{<i-1}, X_{<i}, S, F) \Big] \\ &= \sum_{i=1}^{t} I(M_t; X_i, M_{i-1} \mid M_{<i-1}, X_{<i}, S, F) \\ &= I(M_t; X_{\leq i}, M_{\leq i-1} \mid S, F).\end{aligned}$$

This term is at most $H(M_t \mid S, F)$, which is at most $H(M_t) \leq |M_t|$. ■

Appendix B. Proofs for Section 4

Recall that, for algorithm M with advantage δ, our analysis considers an observer who, at time t, sees all of M's previous memory states $m_{\leq t}$. We consider running M on either the 1's stream or the uniform stream, and analyze the observer's posterior belief $\{p_t\}$ about E, the event that the stream is fixed to all 1's. Restricting our focus to the 1's stream (as opposed to 0's) versus uniform is without loss of generality, as we argued above.

We first prove that, on average, the final posterior P_T has decreased significantly. Recall that we fix prior $P_0 = \frac{1}{2}$.

Lemma 21 (Lemma 5 Restated) *Assume algorithm M has advantage at least δ. Let random variable P_T be the final posterior of M', i.e., $P_T = \Pr[E \mid M_{\leq T}]$. When the inputs are uniform, we have $\mathbb{E}[P_T] \leq \frac{1}{2} - \frac{\delta^2}{4}$.*

Proof Let f_1 denote the joint distribution over memory states $m_{\leq T}$ when the input is fixed to 1's, and let f_u denote the distribution when the input is uniform. By Bayes rule, for any fixed $m_{\leq T}$ we have

$$p_t = \Pr[E \mid M_{\leq T} = m_{\leq T}] = \frac{f_1(m_{\leq T}) \cdot \frac{1}{2}}{f_u(m_{\leq T}) \cdot \frac{1}{2} + f_1(m_{\leq T}) \cdot \frac{1}{2}}$$

and the expectation under the uniform inputs is

$$\mathbb{E}[P_T] = \frac{1}{2} \sum_m \frac{f_u(m) f_1(m)}{\frac{1}{2}(f_1(m) + f_u(m))}.$$

Writing $f_u(m)f_1(m) = \sqrt{f_u(m)f_1(m)}\sqrt{f_u(m)f_1(m)}$, we apply the arithmetic mean/geometric mean inequality and have

$$\mathbb{E}[P_t] \leq \frac{1}{2} \sum_m \sqrt{f_u(m) f_1(m)} = \frac{1}{2} \left(1 - H^2(f_u, f_1)\right),$$

where we have introduced the squared Hellinger distance (Definition 16). By Fact 17, the Hellinger-total variation inequality, we arrive at

$$\mathbb{E}[P_T] \leq \frac{1}{2} - \frac{1}{4}\mathrm{TV}^2(f_u, f_1).$$

This suffices to finish the proof: the TV distance between f_u and f_1 is at least δ. This follows from the δ-advantage assumption on M (stated in Equation (3)) and Fact 18 part (ii), which implies that access to the states $m_{<T}$ only increases the space of distinguishing functions and thus can only increase the TV distance. ■

For any time t and fixed set of previous memory states $m_{<t}$, the distribution over P_t is a mixture of two distributions: $\mathcal{D}_{1,m_{<t}}$, arising when $X_t = 1$, and $\mathcal{D}_{0,m_{<t}}$, otherwise. We show that, when P_t has a large change in expectation from p_{t-1}, the means of these two distributions are far apart. The core of the proof is a convexity argument showing that, when we have input $X_t = 1$, the posterior belief in the stream being fixed increases on average.

Lemma 22 (Lemma 6 Restated) *For some time t, fix the memory states $m_{<t}$. Let $\mu_{1,m_{<t}} = \mathbb{E}[P_t \mid X_t = 1, M_{<t} = m_{<t}]$ and $\mu_{0,m_{<t}} = \mathbb{E}[P_t \mid X_t = 0, M_{<t} = m_{<t}]$. We have*

$$\mathbb{E}[P_t - P_{t-1} \mid M_{<t} = m_{<t}] \geq \frac{\mu_{0,m_{<t}} - \mu_{1,m_{<t}}}{2}.$$

Proof We have $\mathbb{E}[P_t \mid M_{<t} = m_{<t}] = \frac{1}{2}(\mu_{0,m_{<t}} + \mu_{1,m_{<t}})$. Fixing $M_{<t} = m_{<t}$ also fixes $P_{t-1} = p_{t-1}$ for some value p_{t-1}, so it suffices to prove

$$p_{t-1} \leq \frac{\mu_{0,m_{<t}} - \mu_{1,m_{<t}}}{2} - \frac{\mu_{0,m_{<t}} + \mu_{1,m_{<t}}}{2} = \mu_{1,m_{<t}}.$$

Let f_0 be the distribution over memory states at time t when the input is $X_t = 0$, and define f_1 similarly. With this notation, and recalling that event E means the input stream is fixed to 1's, we

can write the posterior as

$$\begin{aligned} p_t = \Pr[E \mid M_{\le t} = m_{\le t}] &= \frac{f_1(m_t) \cdot p_{t-1}}{f_1(m_t) \cdot p_{t-1} + \frac{f_1(m_t)+f_0(m_t)}{2} \cdot (1 - p_{t-1})} \\ &= \frac{2 \cdot f_1(m_t) \cdot p_{t-1}}{(1 + p_{t-1}) f_1(m_t) + (1 - p_{t-1}) f_0(m_t)} \\ &= \frac{2 \cdot f_1(m_t) \cdot p_{t-1}}{(f_1(m_t) + f_0(m_t)) + p_{t-1}(f_1(m_t) - f_0(m_t))}. \end{aligned}$$

Considering the expectation of the posterior conditioned on $X_t = 1$, we have

$$\begin{aligned} \mu_t^{(1)} = \mathop{\mathbb{E}}_{f_1}\left[\frac{2p_{t-1}}{1 + p_{t-1} + (1 - p_{t-1}) \cdot \frac{f_0(m_t)}{f_1(m_t)}}\right] &\ge \frac{2p_{t-1}}{1 + p_{t-1} + (1 - p_{t-1}) \cdot \mathbb{E}_{f_1}\left[\frac{f_0(m_t)}{f_1(m_t)}\right]} \\ &= \frac{2p_{t-1}}{1 + p_{t-1} + (1 - p_{t-1}) \cdot 1} \\ &= p_{t-1}, \end{aligned}$$

The inequality use Jensen's inequality: for any value $p \in [0, 1]$, $\frac{1}{1+p+(1-p)\cdot\alpha}$ is convex in α. To see this, let $g(\alpha) = (1 + p + \alpha - p\alpha)^{-1}$. Then $g'(\alpha) = \frac{p-1}{(1+p+\alpha-p\alpha)^2}$ and $g''(\alpha) = \frac{2(p-1)^2}{(1+p+\alpha-p\alpha)^3}$. The numerator is nonnegative. Since $\alpha \ge p\alpha$, the denominator of g'' is positive. ■

Next we lower bound the Jensen-Shannon divergence of two distributions in terms of the difference of their means and the variance of their (uniform) mixture. For simplicity we state the proof for discrete distributions. Since both distributions are absolutely continuous with respect to their mixture, the continuous case is analogous.

Lemma 23 (Lemma 7 Restated) *Let $\mathcal{D}_0$ and $\mathcal{D}_1$ be distributions over $\mathbb{R}$ with means μ_0 and μ_1 respectively, and let $\mathcal{D} = \frac{\mathcal{D}_0+\mathcal{D}_1}{2}$. Then*

$$\mathrm{JSD}(\mathcal{D}_0 \parallel \mathcal{D}_1) \ge \frac{1}{8} \cdot \frac{(\mu_0 - \mu_1)^2}{\mathrm{Var}(\mathcal{D})}.$$

Proof Define $\gamma(x) = \frac{\mathcal{D}_1(x)-\mathcal{D}(x)}{\mathcal{D}(x)} = \frac{\mathcal{D}_1(x)-\mathcal{D}_0(x)}{2\mathcal{D}(x)}$. We have $\mathcal{D}_0 = (1 - \gamma)\mathcal{D}$ and $\mathcal{D}_1 = (1 + \gamma)\mathcal{D}$.

$$\begin{aligned} \mathrm{JSD}(\mathcal{D}_0 \parallel \mathcal{D}_1) &= \frac{1}{2}\sum_x \mathcal{D}_0(x) \ln \frac{\mathcal{D}_0(x)}{\mathcal{D}(x)} + \mathcal{D}_1(x) \ln \frac{\mathcal{D}_1(x)}{\mathcal{D}(x)} \\ &= \frac{1}{2}\sum_x \mathcal{D}(x)\big((1 - \gamma(x)) \ln (1 - \gamma(x)) + (1 + \gamma(x)) \ln (1 + \gamma(x))\big) \\ &\ge \frac{1}{2}\sum_x \mathcal{D}(x)\gamma(x)^2 = \frac{1}{2}\sum_x \mathcal{D}(x)\left(\frac{\mathcal{D}_1(x) - \mathcal{D}_0(x)}{2 \cdot \mathcal{D}(x)}\right)^2. \end{aligned}$$

The inequality applies the fact[6] that, for $\gamma \in (-1,1)$, $(1-\gamma)\ln(1-\gamma)+(1+\gamma)\ln(1+\gamma) \geq \gamma^2$. Define $\mu = \frac{1}{2}(\mu_0+\mu_1)$, the mean of $\mathcal{D}$. We have

$$\begin{aligned}
\mu_1 - \mu_0 &= \mathop{\mathbb{E}}_{X\sim\mathcal{D}_1}[X-\mu] - \mathop{\mathbb{E}}_{X\sim\mathcal{D}_0}[X-\mu] \\
&= \sum_x (\mathcal{D}_1(x)-\mathcal{D}_0(x))(x-\mu) \\
&= \mathop{\mathbb{E}}_{X\sim\mathcal{D}}\left[(X-\mu)\cdot\frac{\mathcal{D}_1(X)-\mathcal{D}_0(X)}{\mathcal{D}(X)}\right] \\
&\leq \sqrt{\mathop{\mathbb{E}}_{X\sim\mathcal{D}}\left[(X-\mu)^2\right]}\sqrt{\mathop{\mathbb{E}}_{X\sim\mathcal{D}}\left[\left(\frac{\mathcal{D}_1(X)-\mathcal{D}_0(X)}{\mathcal{D}(X)}\right)^2\right]},
\end{aligned}$$

applying Cauchy-Schwarz. By definition, $\mathbb{E}_{X\sim\mathcal{D}}\left[(X-\mu)^2\right] = \mathrm{Var}(\mathcal{D})$. Plugging in the bound from Equation B, we have

$$\mu_1 - \mu_0 \leq \sqrt{\mathrm{Var}(\mathcal{D})}\sqrt{8\cdot \mathrm{JSD}(\mathcal{D}_0 \parallel \mathcal{D}_1)}.$$

Taking squares and rearranging finishes the proof. ■

Before proving Theorem 4, we show that the average variances are not too large. The first step in this direction is to show that, when the inputs are uniform, the sequence of posteriors forms a supermartingale. The proof uses a convexity argument similar to that of Lemma 6.

Lemma 24 *Let random process $P_0, P_1, \ldots, P_T$ be the sequence of posteriors when the inputs X_t are i.i.d. uniform. Then, for any t and past memory states $m_{<t}$,*

$$\mathbb{E}[P_t \mid M_{<t} = m_{<t}] \leq p_{t-1}.$$

Proof Throughout this proof, leave the conditioning on $m_{<t}$ implicit. Let f_0 be the distribution over memory states at time t when the input is $X_t = 0$, and define f_1 similarly. With this notation, and recalling that event E means the input stream is fixed to 1's, we can write the posterior as

$$\begin{aligned}
p_t = \Pr[E \mid M_{\leq t} = m_{\leq t}] &= \frac{f_1(m_t)\cdot p_{t-1}}{f_1(m_t)\cdot p_{t-1} + \frac{f_1(m_t)+f_0(m_t)}{2}\cdot(1-p_{t-1})} \\
&= \frac{2\cdot f_1(m_t)\cdot p_{t-1}}{(1+p_{t-1})f_1(m_t) + (1-p_{t-1})f_0(m_t)} \\
&= \frac{2\cdot f_1(m_t)\cdot p_{t-1}}{(f_1(m_t)+f_0(m_t)) + p_{t-1}(f_1(m_t)-f_0(m_t))}.
\end{aligned}$$

6. To see this, take derivatives. Let $f(\gamma) = (1-\gamma)\ln(1-\gamma)+(1+\gamma)\ln(1+\gamma)$. We have $f'(\gamma) = \ln\frac{1+\gamma}{1-\gamma}$ and $f''(\gamma) = \frac{2}{1-x^2} \geq 2$, while $\frac{d^2}{dx^2}x^2 = 2$, and $f(0) = 0^2 = 0$.

We consider the expectation over uniform inputs, rewriting it as an expectation over f_1:

$$\begin{aligned}\frac{\mu_t^{(0)}+\mu_t^{(1)}}{2} &= \mathop{\mathbb{E}}_{\frac{f_0+f_1}{2}}\left[\frac{2f_1(m_t)p_{t-1}}{(f_1(m_t)+f_0(m_t))+p_{t-1}(f_1(m_t)-f_0(m_t))}\right] \\ &= \sum_{m_t}\left(\frac{f_1(m_t)+f_0(m_t)}{2}\right)\left(\frac{2f_1(m_t)p_{t-1}}{(f_1(m_t)+f_0(m_t))+p_{t-1}(f_1(m_t)-f_0(m_t))}\right) \\ &= \mathop{\mathbb{E}}_{f_1}\left[\frac{p_{t-1}}{1+p_{t-1}\cdot\frac{f_1(m_t)-f_0(m_t)}{f_1(m_t)+f_0(m_t)}}\right] \\ &= \mathop{\mathbb{E}}_{f_1}\left[\frac{p_{t-1}}{1+p_{t-1}\cdot\frac{1-f_0(m_t)/f_1(m_t)}{1+f_0(m_t)/f_1(m_t)}}\right].\end{aligned}$$

For any fixed $p\in[0,1]$, $\frac{1}{1+p\cdot\frac{1-\alpha}{1+\alpha}}$ is concave for $\alpha\in[0,1]$. To see this, let $h(\alpha)=\frac{1+\alpha}{1+\alpha+p-p\alpha}$. Then $h'(\alpha)=\frac{2p}{(1+\alpha+p-p\alpha)^2}$ and $h''(\alpha)=\frac{-4(1-p)p}{(1+\alpha+p-p\alpha)^3}$. The denominator is positive and the numerator is nonpositive.

So we have via Jensen's inequality that

$$\frac{\mu_t^{(0)}+\mu_t^{(1)}}{2}\le\frac{p_{t-1}}{1+p_{t-1}\cdot\mathbb{E}_{f_1}\left[\frac{1-f_0(m_t)/f_1(m_t)}{1+f_0(m_t)/f_1(m_t)}\right]}\le\frac{p_{t-1}}{1+0}.$$

To see the second inequality, note that the denominator is always at most some constant (that depends on f_1) and that $\mathbb{E}_{f_1}[1-f_0(m_t)/f_1(m_t)]=0$. ■

We now upper bound the average "expected variance" of P_t, i.e., the variance of P_t on average when $M_{<t}$ is fixed.

Lemma 25 (Lemma 8 Restated) *For every algorithm M solving Task A, we have*

$$\mathbb{E}\sum_{t=1}^{T}\mathrm{Var}\left(P_t\ \mid\ M_{<t}\right)\le\tfrac{5}{4}.$$

Here $\mathrm{Var}\left(P_t\ \mid\ m\right)$ *denotes the conditional variance* $\mathbb{E}_{P_t}\left[\left(P_t-\mathbb{E}[P_t\mid M_{<t}=m]\right)^2\right]$.

Proof Define random variable $\Delta_t\stackrel{\text{def}}{=}P_t-P_{t-1}$, the difference in posteriors, and write $P_T=P_0+\sum_{t=1}^{T}\Delta_t$. We will recursively apply the following elementary variance decomposition.

Claim 26 *Let A, B, and C be random variables. We have*

$$\mathrm{Var}\left(A+B\right)=\mathrm{Var}\left(A\right)+\mathop{\mathbb{E}}_{B,C}\left[\left(B-\mathbb{E}[B\mid C]\right)^2\right]+\mathrm{Var}\left(\mathbb{E}[B\mid C]\right)+2\cdot\mathrm{Cov}(A,\mathbb{E}[B\mid C]).$$

Proof [Proof of Claim 26] First, $\mathrm{Var}\,(A+B) = \mathrm{Var}\,(A) + \mathrm{Var}\,(B) + 2\cdot\mathrm{Cov}(A,B)$. Next, we add and subtract $\mathbb{E}[B \mid C]$ and expand:

$$\begin{aligned}\mathrm{Var}\,(B) = \mathop{\mathbb{E}}_{C}[\mathrm{Var}\,(B)] &= \mathop{\mathbb{E}}_{B,C}[(B-\mathbb{E}[B])^2] \\ &= \mathop{\mathbb{E}}_{B,C}[(B-\mathbb{E}[B\mid C]+\mathbb{E}[B\mid C]-\mathbb{E}[B])^2] \\ &= \mathop{\mathbb{E}}_{B,C}\big[(B-\mathbb{E}[B\mid C])^2+(\mathbb{E}[B\mid C]-\mathbb{E}[B])^2\big] \\ &\quad + \mathop{\mathbb{E}}_{B,C}[2(B-\mathbb{E}[B\mid C])(\mathbb{E}[B\mid C]-\mathbb{E}[B])] \\ &= \mathop{\mathbb{E}}_{C,B}[(B-\mathbb{E}[B\mid C])^2] + \mathrm{Var}\,(\mathbb{E}[B\mid C]) + 0,\end{aligned}$$

noting that we have $\mathbb{E}[\mathbb{E}[B \mid C] - \mathbb{E}[B]] = 0$ for any $B = b$. To finish, we again add and subtract $\mathbb{E}[B \mid C]$ and expand:

$$\begin{aligned}\mathrm{Cov}(A,B) &= \mathop{\mathbb{E}}_{A,B,C}[(A-\mathbb{E}[A])(B-\mathbb{E}[B])] \\ &= \mathop{\mathbb{E}}_{A}\left[(A-\mathbb{E}[A])\mathop{\mathbb{E}}_{B,C}[B-\mathbb{E}[B\mid C]+\mathbb{E}[B\mid C]-\mathbb{E}[B]]\right] \\ &= \mathop{\mathbb{E}}_{A}\left[(A-\mathbb{E}[A])\mathop{\mathbb{E}}_{B,C}[0+\mathbb{E}[B\mid C]-\mathbb{E}[B]]\right] = \mathrm{Cov}(A,\mathbb{E}[B\mid C]),\end{aligned}$$

again using the fact that $\mathbb{E}[B \mid C] = \mathbb{E}[B]$. ■

Repeatedly applying the claim to the sum $P_0 + \sum_{t=1}^{T}\Delta_t$, we get

$$\begin{aligned}\mathrm{Var}\,(P_T) &= \mathrm{Var}\,(P_0) + \sum_{t=1}^{T}\Big(\mathop{\mathbb{E}}_{M_{<t},\Delta_t}[(\Delta_t-\mathbb{E}[\Delta_t\mid M_{<t})^2] + \mathrm{Var}\,(\mathbb{E}[\Delta_t\mid M_{<t}) \\ &\quad + 2\cdot\mathrm{Cov}(P_{t-1},\mathbb{E}[\Delta_t\mid M_{<t}])\Big) \\ &\geq \sum_{t=1}^{T}\mathop{\mathbb{E}}_{M_{<t},\Delta_t}[(\Delta_t-\mathbb{E}[\Delta_t\mid M_{<t}])^2] + 2\cdot\sum_{t=1}^{T}\mathrm{Cov}(P_{t-1},\mathbb{E}[\Delta_t\mid M_{<t}]),\end{aligned}$$

since the variance terms are nonnegative. Recall that conditioning on $m_{<t}$ fixes p_{t-1}, so

$$\mathop{\mathbb{E}}_{M_{<t},\Delta_t}[(\Delta_t-\mathbb{E}[\Delta_t\mid M_{<t}])^2] = \mathop{\mathbb{E}}_{M_{<t},P_t}[(P_t-\mathbb{E}[P_t\mid M_{<t}])^2].$$

Thus it remains to lower bound the sum of the covariance terms (which may be negative).

We begin with the fact that, for any random variables A and B, $\mathrm{Cov}(A,B) = \mathbb{E}[(A-\mathbb{E}[A])B]$. We then apply two facts about the sequence of posteriors: first, that $\mathbb{E}[\Delta_t \mid M_{<t}]$ is nonpositive, and second, that $(P_{t-1} - \mathbb{E}[P_{t-1}])$ has 1 as an upper bound. (The former fact we proved in Lemma 24.)

$$\begin{aligned}\sum_{t=1}^{T}\mathrm{Cov}(P_{t-1},\mathbb{E}[\Delta_t\mid M_{<t}]) &= \sum_{t=1}^{T}\mathbb{E}\left[(P_{t-1}-\mathbb{E}[P_{t-1}])\,\mathbb{E}[\Delta_t\mid M_{<t}]\right] \\ &\geq \sum_{t=1}^{T}\mathbb{E}\left[\mathbb{E}[\Delta_t\mid M_{<t}]\right] = \sum_{t=1}^{T}\mathbb{E}\left[\Delta_t\right].\end{aligned}$$

By linearity of expectation, this sum is exactly $\mathbb{E}\left[\sum_t \Delta_t\right] = \mathbb{E}[P_T - P_0]$. Since $P_0 = 1/2$ this expectation is at least $-1/2$, so we have, recalling the factor of 2 in front of the covariance sum,

$$\frac{1}{4} \geq \mathrm{Var}\left(P_T\right) \geq \sum_{t=1}^{T} \mathop{\mathbb{E}}_{M_{<t},P_t}[(P_t - \mathbb{E}[P_t \mid M_{<t}])^2] - 1.$$

Adding 1 to both sides finishes the proof. ■

Theorem 27 (Theorem 4 Restated) *Consider a streaming algorithm M for Task A on T inputs that has advantage at least δ. Let $M_1, \ldots, M_T$ be the memory states of M when run on uniformly random inputs $X_1, \ldots, X_T \in \{0,1\}$. Then*

$$\widetilde{\mathrm{CI}}(M) \stackrel{def}{=} \sum_{t=1}^{T} I(M_t; X_t \mid M_{t-1}) \geq \frac{\delta^4}{40}.$$

Proof Recall from Equation 4 that

$$\sum_{t=1}^{T} I(M_t; X_t \mid M_{<t}) \geq \sum_{t=1}^{T} \mathop{\mathbb{E}}_{M_{<t}}\left[\mathrm{JSD}(\mathcal{D}_{0,m_{<t}} \parallel \mathcal{D}_{1,m_{<t}})\right].$$

For clarity, denote $\mathrm{JSD}(\mathcal{D}_{0,m_{<t}} \parallel \mathcal{D}_{1,m_{<t}}) \stackrel{\text{def}}{=} \mathrm{JSD}_t$ and $\mathbb{E}_{P_t}\left[(P_t - \mathbb{E}[P_t \mid M_{<t}])^2\right] \stackrel{\text{def}}{=} \mathrm{Var}_t$. Both of these are random variables that depend on $M_{<t}$; the latter is exactly the variance of the mixture $\frac{1}{2}(\mathcal{D}_{0,m_{<t}} + \mathcal{D}_{1,m_{<t}})$.

Taking expectations over time steps and then over $M_{<t}$, we apply Cauchy-Schwarz and then Lemma 7, our lower bound on the Jensen-Shannon divergence:

$$\begin{aligned}\left(\mathop{\mathbb{E}}_{t}\mathop{\mathbb{E}}_{M_{<t}}[\,\mathrm{JSD}_t]\right)\left(\mathop{\mathbb{E}}_{t}\mathop{\mathbb{E}}_{M_{<t}}[\,\mathrm{Var}_t]\right) &\geq \left(\mathop{\mathbb{E}}_{t}\mathop{\mathbb{E}}_{M_{<t}}\left[\sqrt{\mathrm{JSD}_t \cdot \mathrm{Var}_t}\right]\right)^2 \\ &\geq \frac{1}{8}\left(\mathop{\mathbb{E}}_{t}\mathop{\mathbb{E}}_{M_{<t}}\left[|\mu_t^{(0)} - \mu_t^{(1)}|\right]\right)^2.\end{aligned}$$

On average, the difference between $\mu_{0,m_{<t}}$ and $\mu_{1,m_{<t}}$ is appreciable: we apply Jensen's inequality to move the absolute value outside the expectation and apply Lemma 6:

$$\begin{aligned}\left(\mathop{\mathbb{E}}_{t}\mathop{\mathbb{E}}_{M_{<t}}[\,\mathrm{JSD}_t]\right)\left(\mathop{\mathbb{E}}_{t}\mathop{\mathbb{E}}_{M_{<t}}[\,\mathrm{Var}_t]\right) &\geq \frac{1}{8}\left(\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{M_{<t}}[\mu_{0,m_{<t}} - \mu_{1,m_{<t}}]\right)^2 \\ &\geq \frac{1}{2}\left(\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{M_{<t}}[\mathbb{E}[P_t - P_{t-1} \mid M_{<t}]]\right)^2.\end{aligned}$$

(Note that this produces a factor of 2 inside the square.) We use the fact that $\mathbb{E}[\mathbb{E}[X \mid Y]] = \mathbb{E}[X]$ for random variables X and Y and cancel out the intermediate terms in the sum, arriving at

$$\left(\mathop{\mathbb{E}}_{t}\mathop{\mathbb{E}}_{M_{<t}}[\,\mathrm{JSD}_t]\right)\left(\mathop{\mathbb{E}}_{t}\mathop{\mathbb{E}}_{M_{<t}}[\,\mathrm{Var}_t]\right) \geq \frac{1}{2}\left(\frac{1}{T}\mathbb{E}[P_T - P_0]\right)^2 \geq \frac{1}{32} \cdot \frac{\delta^4}{T}.$$

The final inequality uses Lemma 5: the algorithm has advantage δ, so $\mathbb{E}[P_T - P_0] \le -\frac{\delta^2}{4}$.

Rewriting the expectations as sums over t and multiplying by T^2 on both sides, we have

$$\left(\sum_{t=1}^{T} \mathbb{E}\left[\mathrm{JSD}_t\right]\right)\left(\sum_{t=1}^{T} \mathbb{E}\left[\mathrm{Var}_t\right]\right) \ge \frac{\delta^4}{32}.$$

Lemma 8 says the sum of variances is at most $\frac{5}{4}$, so we have $\widetilde{\mathrm{CI}}(M) = \sum_{t=1}^{T} \mathrm{JSD}_t \ge \frac{\delta^4}{40}$. ■

Appendix C. Proofs for Section 5

We state the formal reduction in Algorithm 1. The algorithm assumes knowledge of the exact values of π_r for all $r \in \{0, \ldots, \rho+1\}$. These values can in principle be computed to arbitrary accuracy given access to M; since our argument is information-theoretic we can ignore the computational concerns.

Algorithm 1: M' for Task A

input: stream $x_1, \ldots, x_T \in \{0,1\}$; algorithm M for Task B; parameters d and $\rho \le d$.

Draw $r \sim \{0, \ldots, \rho\}$ uniformly; `/* number of "fake" fixed streams */`
Draw $\mathcal{J} = \{j_1, \ldots, j_r\} \subseteq [d]$ w/o replacement; `/* indices of fixed streams */`
Draw $\mathcal{B} = (b_1, \ldots, b_r) \in \{0,1\}^r$ uniformly; `/* values of fixed streams */`
Draw $j_0 \in [d]$ uniformly; `/* location of input stream */`
for $t = 1, \ldots, T$ **do**
  Receive sample $x_t \in \{0,1\}$;
  Draw $z_t \in \{0,1\}^d$ uniformly;
  Set $z^t_{j_0} \leftarrow x^t$;
  $\forall j_\ell \in \mathcal{J}$, set $z^t_{j_\ell} \leftarrow b_{j_\ell}$; `/* if` $j_0 \in \mathcal{J}$`, overwrites` x_t `*/`
  Execute $M(z^t)$;
end
Receive output of M, `OUT` $\in$ {*"S"*, *"U"*};
if $\pi_{r+1} \ge \pi_r$ **then**
  return `OUT`;
else
  return $\neg$`OUT`; `/* other entry in {"S", "U"} */`
end

Lemma 28 *Suppose M for Task B has advantage at least δ. Let $R \sim \mathrm{Uniform}(\{0, 1 \ldots, \rho\})$. Then $\mathbb{E}[|\pi_{R+1} - \pi_R|] \ge \frac{\delta}{\rho+1}$.*

Proof Let r^* be the number with the highest probability of returning *"S"*, i.e., $r^* = \mathrm{argmax}_{r \in [\rho]} \pi_r$.[7] By the advantage assumption, $\pi_{r^*} - \pi_0 \ge \delta$. We discard the terms beyond r^* and apply the triangle

7. With advantage $\delta > 0$, we know $r^* \ge 1$.

inequality:

$$\begin{aligned}\mathbb{E}[|\pi_{R+1} - \pi_R|] &= \frac{1}{\rho+1}\sum_{r=0}^{\rho}|\pi_{r+1} - \pi_r| \\ &\geq \frac{1}{\rho+1}\sum_{r=0}^{r^*-1}|\pi_{r+1} - \pi_r| \\ &\geq \frac{1}{\rho+1}\left|\sum_{r=0}^{r^*-1}\pi_{r+1} - \pi_r\right|.\end{aligned}$$

The intermediate terms in the sum cancel out and we are left with $|\pi_{r^*} - \pi_0| \geq \delta$. ■

Lemma 29 (Accuracy of Algorithm 1) *Assume M has advantage at least δ. Then M' has advantage at least*

$$\frac{\delta}{\rho+1} + \frac{\rho}{d}$$

Proof Recall the definition of Task A: the distribution $\mathcal{U}$ generates uniform bits and the structured distribution selects a random bit $b \in \{0,1\}$ and fixes all values to b We wish to show that

$$\Pr_{x_{\leq T}\sim\mathcal{P}_{\text{bit}}}[M'(x_{\leq T}) = \text{“}S\text{”}] - \Pr_{x_{\leq T}\sim\mathcal{U}}[M'(x_{\leq T}) = \text{“}S\text{”}] \geq \frac{\delta}{\rho+1} - \frac{\rho}{d}.$$

To do this, we show that, on both the structured distribution and $\mathcal{U}$, M' is correct with probability at least $\frac{1}{2} + \frac{1}{2}\left(\frac{\delta}{\rho+1} - \frac{\rho}{d}\right)$.

M' randomly selects $r \in \{0,\ldots,\rho\}$ as the number of synthetic fixed streams to feed to M. Fix some r: we lower bound the total variation distance between the two output distributions of M (corresponding to structured and uniform inputs to M'). When the input stream is uniform, M receives synthetic inputs z^t with exactly r fixed streams, so outputs *“S”* with probability π_r. When the input stream is structured, M receives synthetic inputs z^t with $r+1$ fixed streams unless $j_0 \in \mathcal{J}$, i.e. the input stream is overwritten. This only happens with probability $\frac{r}{d}$, so when the input stream is structured M outputs *“S”* with probability $\left(1 - \frac{r}{d}\right)\pi_{r+1} + \frac{r}{d}\cdot\pi_r$. Thus, for any r, the total variation distance is at least

$$\left|\left(1 - \frac{r}{d}\right)\pi_{r+1} + \frac{r}{d}\cdot\pi_r - \pi_r\right| \geq |\pi_{r+1} - \pi_r| - \frac{\rho}{d},$$

using the fact that $r \leq \rho$ and $\pi_{r+1}, \pi_r \in [0,1]$.

By Fact 18, this lower bound on total variation distance implies a lower bound on the accuracy of M'. Since the accuracy of M' is an average over r, we have by linearity of expectation that

$$\Pr[M' \text{ correct}] = \frac{1}{2} + \frac{1}{2}\left(\mathop{\mathbb{E}}_{R}[|\pi_{R+1} - \pi_R|] - \frac{\rho}{d}\right).$$

By Lemma 28, the expectation is at least $\frac{\delta}{\rho+1}$. ■

Appendix D. Proofs: From Distinguishing Inputs to Distinguishing a Test Example

In this section, we move between two versions of the "single subpopulation" task: in Task B' the learner receives T structured inputs and must distinguish a structured test example from a uniform one, while in Task B the learner must determine whether its inputs are all structured or all uniform. Recall that $\mathcal{U}$ denotes the uniform distribution.

Definition 30 (Task B') *Parameters: positive integers $T, d, \rho \leq d$. The learning algorithm M receives a stream of T i.i.d. samples from $\mathcal{P}$. After T time steps, the learner outputs a (possibly randomized) function $m : \{0,1\}^d \to \{$"U", "S"$\}$. The* advantage *of the learner is*

$$\mathop{\mathbb{E}}_{\substack{\mathcal{P}\sim\mathcal{Q}_{d,\rho} \\ x=(x_1,\ldots,x_T)\sim_{\text{iid}}\mathcal{P} \\ m\leftarrow M(x)}} \left[\Pr_{y\sim\mathcal{P}}[m(y) = \text{"S"}] - \Pr_{y\sim\mathcal{U}}[m(y) = \text{"S"}] \right].$$

Lemma 31 *For any algorithm M solving Task B' on T examples with advantage at least δ, there is an algorithm M' solving Task B on $T+1$ examples with advantage at least $\delta/2$ and satisfying*

$$\sum_{t=1}^{T+1} I(M'_t; X_t \mid M'_{t-1}, F) \leq \sum_{t=1}^{T} I(M_t; X_t \mid M_{t-1}, F) + 1.$$

Here the inputs X_t are drawn from a structured distribution with parameters F (see Definition 2).

This information inequality also holds when the X_t are i.i.d. uniform, but we only require the result for structured distributions.

Proof In this proof, let X_i denote the structured inputs drawn from $\mathcal{P}$ (see Definition 2) and U_i denote i.i.d. uniform inputs. Task B' on T examples asks the learner to distinguish a test example, i.e.,

$$(X_{\leq T}, X_{T+1}) \quad \text{from} \quad (X_{\leq T}, U_{T+1}),$$

and Task B on $T+1$ samples asks the learner to distinguish its inputs:

$$X_{\leq T+1} = (X_{\leq T}, X_{T+1}) \quad \text{from} \quad U_{\leq T+1} = (U_{\leq T}, U_{T+1}).$$

Consider an algorithm M solving Task B' and write it $M(\cdot,\cdot)$, as it takes as input a stream of length T and an additional test example. Define

$$\Pr[M(X_{\leq T}, X_{T+1}) = \text{"S"}] = p_{x,x}$$
$$\Pr[M(X_{\leq T}, U_{T+1}) = \text{"S"}] = p_{x,u}.$$

Since M has advantage at least δ, by definition we have $p_{x,x} - p_{x,u} \geq \delta$. Now, there is some probability $p_{u,u}$ such that

$$\Pr[M(U_{\leq T}, U_{T+1}) = \text{"S"}] = p_{u,u}.$$

Using the fact that $p_{u,u}$ cannot be close to both $p_{x,x}$ and $p_{x,u}$, we design the algorithm M' for Task B depending on which it is closer to.

Case 1: If $p_{x,x} - p_{u,u} \geq \delta/2$, M' simply runs M and outputs M's answer. This has advantage $p_{x,x} - p_{u,u} \geq \delta/2$.

Case 2: If $p_{x,x} - p_{u,u} < \delta/2$, then we know that $p_{u,u} - p_{x,u} \geq \delta/2$, which again gives us a way to distinguish the two inputs. In this case, M' runs M through time T and then generates a fresh uniform example U_{T+1} to use as the final input. In this case, M' has advantage $p_{u,u} - p_{x,u} \geq \delta/2$.

To prove the information complexity claim, observe that, until time T, M' just executes M, so we have

$$\sum_{t=1}^{T} I(M_t; X_t \mid M_{t-1}, F) = \sum_{t=1}^{T} I(M'_t; X_t \mid M'_{t-1}, F).$$

M' has one final step, but without loss of generality we can assume the final state of M is a single bit (i.e., its answer) and thus we have, in case 1, that

$$\begin{aligned}\sum_{t=1}^{T+1} I(M'_t; X_t \mid M'_{t-1}, F) &= \sum_{t=1}^{T} I(M_t; X_t \mid M_{t-1}) + I(M'_{T+1}; X_{T+1} \mid M_T, F) \\ &\leq \sum_{t=1}^{T} I(M_t; X_t \mid M_{t-1}, F) + 1\end{aligned}$$

In case 2, M' discards its final input and thus the same inequality holds. ■

Combining this argument with Corollary 11, we get the following lower bound.

Corollary 32 *Any algorithm M solving Task B' on T examples with advantage at least δ satisifies*

$$\sum_{t=1}^{T} I(M_t; X_t \mid M_{t-1}, F) \geq \frac{d}{160} \cdot \left(\frac{\delta}{\rho+1} - \frac{\rho}{d}\right)^4 - 1.$$

Appendix E. Proofs for Section 6

Theorem 33 (Theorem 12 Restated) *Any algorithm M solving Task C with parameters k, d, and ρ with advantage at least δ satisfies*

$$\mathrm{CI}(M \mid S, F) = \sum_{i=1}^{N} \sum_{t=i}^{N} I(M_t; X_i \mid M_{i-1}, S, F) \geq \frac{k^2 d}{320} \left(\frac{\delta}{\rho+1} - \frac{\rho}{d}\right)^4 - O(k^2) - O(kd).$$

Here the inputs X_i are structured (Definition 3).

Proof We first set up some notation. Denote the sequence of arrivals' subpopulations by $s \in [k]^N$. For sequence s, let ℓ_j be the number of arrivals from j, i.e., the number of time steps t where $s_t = j$. For the a-th arrival from subpopulation j, let $q_j(a) \in [N]$ denote the time of that arrival. For $a > \ell_j$, define $q_j(a) = N+1$. In this proof we sometimes abbreviate mutual information expressions of the form $I(A; B \mid C = c)$ to $I(A; B \mid c)$, leaving the random variable in the conditioning implicit. We will also denote the length-t prefix of a sequence s via the notation $s[:t]$. Similarly, $s[t:]$ denotes the suffix, all the terms from $t+1$ until the end. Note that, contrary to programming language conventions, the start index is *exclusive*.

We rewrite the composable information cost of M, identifying terms by the subpopulation of their input X_i and their arrival number a:

$$\begin{aligned}\mathrm{CI}(M \mid S, F) &= \mathop{\mathbb{E}}_{s} \sum_{i=1}^{N} \left(\sum_{t=i}^{N} I(M_t; X_i \mid M_{i-1}, F, s) \right) \\ &= \mathop{\mathbb{E}}_{s} \sum_{j=1}^{k} \sum_{a=1}^{\ell_j} \left(\sum_{t=q_j(a)}^{N} I(M_t; X_{q_j(a)} \mid M_{q_j(a)-1}, F, s) \right)\end{aligned}$$

(Note that L_j is a random variable that depends on S.) We introduce an indicator random variable and write the arrival sum over $a = 1, \ldots, N$, since there are at most that many arrivals:

$$\begin{aligned}\mathrm{CI}(M \mid S, F) &= \mathop{\mathbb{E}}_{s} \sum_{j=1}^{k} \sum_{a=1}^{N} \mathbb{1}_{\{\ell_j \geq a\}} \sum_{t=q_j(a)}^{N} I(M_t; X_{q_j(a)} \mid M_{q_j(a)-1}, F, s) \\ &= \sum_{j,a} \mathop{\mathbb{E}}_{s} \left[\mathbb{1}_{\{\ell_j \geq a\}} \sum_{t=q_j(a)}^{N} I(M_t; X_{q_j(a)} \mid M_{q_j(a)-1}, F, s) \right] \\ &= \sum_{j,a} \Pr[L_j \geq a] \mathop{\mathbb{E}}_{s \mid L_j \geq a} \left[\sum_{t=q_j(a)}^{N} I(M_t; X_{q_j(a)} \mid M_{q_j(a)-1}, F, s) \right].\end{aligned}$$

We now consider the expectation above and rewrite it as an expectation over 4 terms: first the choice of $q_j(a)$, then the choice[8] of $s[: q_j(a)]$, then the choice of $q_j(a+1)$, and finally the choice of $s[q_j(a) :]$. For brevity define $t_0 = q_j(a)$ and $t_1 = q_j(a+1)$. Thus we have

$$\begin{aligned}\mathrm{CI}(M \mid S, F) &= \sum_{j,a} \Pr[L_j \geq a] \mathop{\mathbb{E}}_{\substack{t_0, s[:t_0] \\ \mid L_j \geq a}} \mathop{\mathbb{E}}_{\substack{t_1 \\ s[t_0:]}} \left[\sum_{t=t_0}^{N} I(M_t; X_{t_0} \mid M_{t_0-1}, F, s) \right] \\ &= \sum_{j,a} \Pr[L_j \geq a] \mathop{\mathbb{E}}_{\substack{t_0, s[:t_0] \\ \mid L_j \geq a}} \mathop{\mathbb{E}}_{\substack{t_1 \\ s[t_0:]}} \left[Z^{j,a}_{s[:t_0]} \right],\end{aligned}$$

using $Z^{j,a}_{s[:t_0]}$ as shorthand for the sum it replaces. (Note that the inner expectation need not condition on $L_j \geq a$, since each entry in s is drawn independently.)

We now analyze $\mathbb{E}_{t_1, s[t_0:]} \left[Z^{j,a}_{s[:t_0]} \right]$. We are free to condition on the event $t_1 = q_j(a+1)$ in the mutual information terms since it is a function of $s[t_0 :]$; we do so and push the expectation over $s[t_0 :]$ back inside the mutual information notation:

$$\mathop{\mathbb{E}}_{t_1, s[t_0:]} Z^{j,a}_{s[:t_0]} = \mathop{\mathbb{E}}_{t_1} \sum_{t=t_0}^{N} I(M_t; X_{t_0} \mid M_{t_0-1}, F, s[: t_0], S[t_0 :], t_1).$$

8. Because we condition on $L_j \geq a$ and consider the prefix up until the a-th arrival, this prefix has j as its last element and contains $a - 1$ other occurrences of j.

At this point we focus on the terms between arrivals from subpopulation j, introducing an indicator random variable to discard the information terms corresponding to M_{t_1} and beyond. So we arrive at

$$\begin{aligned}\mathop{\mathbb{E}}_{t_1,s[t_0:]} Z^{j,a}_{s[:t_0]} &\geq \mathop{\mathbb{E}}_{t_1} \sum_{t=t_0}^{N} \mathbb{1}_{\{t_1>t\}} I(M_t; X_{t_0} \mid M_{t_0-1}, F, s[:t_0], S[t_0:], t_1) \\ &= \sum_{t=t_0}^{N} \Pr[T_1 > t] \mathop{\mathbb{E}}_{t_1|T_1>t} I(M_t; X_{t_0} \mid M_{t_0-1}, F, s[:t_0], S[t_0:], t_1) \\ &= \sum_{t=t_0}^{N} \Pr[T_1 > t] \cdot I(M_t; X_{t_0} \mid M_{t_0-1}, F, s[:t_0], S[t_0:], T_1, T_1 > t),\end{aligned}$$

pushing the expectation over T_1 into the information terms and adding the condition that $T_1 > t$.

We now execute the fundamental operations in the proof. The first is that the mutual information terms do not change when conditioning on $T_1 = t+1$ in place of $T_1 > t$, since these events depend on the sequence *after* time t. We align the two expressions to highlight the switch:

$$\begin{aligned}&I(M_t; X_{t_0} \mid M_{t_0-1}, F, s[:t_0], S[t_0:], T_1, T_1 > t) \\ =\ &I(M_t; X_{t_0} \mid M_{t_0-1}, F, s[:t_0], S[t_0:], T_1 = t+1).\end{aligned}$$

The second fundamental operation is to observe that, for any fixed t_0, $T_1 - t_0$ is a truncated geometric random variable, since at each time step we observe an example from subpopulation j with probability $\frac{1}{k}$, under the restriction that $T_1 \leq N$. Thus, for any $t_0 \leq t < N$,

$$\Pr[T_1 > t] = k \cdot \Pr[T_1 = t+1].$$

When $t = N$ we have $\Pr[T_1 > N] = \Pr[T_1 = N+1]$. Thus we have

$$\begin{aligned}\mathop{\mathbb{E}}_{t_1,s[t_0:]} Z^{j,a}_{s[:t_0]} &\geq \sum_{t=t_0}^{N} \Pr[T_1 > t] \cdot I(M_t; X_{t_0} \mid M_{t_0-1}, F, s[:t_0], S[t_0:], T_1, T_1 = t+1) \\ &= \sum_{t=t_0}^{N-1} k \cdot \Pr[T_1 = t+1] \cdot I(M_t; X_{t_0} \mid M_{t_0-1}, F, s[:t_0], S[t_0:], T_1 = t+1) \\ &\quad + \Pr[T_1 = N+1] \cdot I(M_N; X_{t_0} \mid M_{t_0-1}, F, s[:t_0], S[t_0:], T_1 = N+1) \\ &= \sum_{t=t_0}^{N-1} k \cdot f(t) + f(N). \qquad (8)\end{aligned}$$

defining the shorthand $f(t)$:

$$f(t) \stackrel{\text{def}}{=} \Pr[T_1 = t+1] \cdot I(M_t; X_{t_0} \mid M_{t_0-1}, F, s[:t_0], S[t_0:], T_1 = t+1).$$

To proceed, we need the following observation about this function $f(\cdot)$.

Claim 34 *$f(t)$ is nonincreasing in t.*

Proof $f(t)$ is a product of two terms, both of which are nonincreasing in t. This holds for the mutual information term by Proposition 15 (the DPI for streaming). It holds for the probability term because $T_1 - t_0$ is truncated geometric. ■

Continuing from Equation (8), we have the following lower bound:

$$\mathop{\mathbb{E}}_{t_1, s[:t_0]} Z^{j,a}_{s[:t_0]} \geq k \sum_{t=t_0}^{N-1} f(t) + f(N) \geq \frac{k}{2} \cdot \mathbb{1}_{\{t_0 \neq N\}} \sum_{t=t_0}^{N} f(t).$$

To see this, first note that we can discard $f(N)$, as it is nonnegative. If $t_0 = N$ the lower bound is vacuous, so assume otherwise and use Claim 34 to write

$$k \cdot f(N-1) = \frac{k}{2} f(N-1) + \frac{k}{2} f(N-1) \geq \frac{k}{2} f(N-1) + \frac{k}{2} f(N).$$

Hit the earlier terms in the sum with a factor of $\frac{1}{2}$.

We can rewrite the sum $\sum_t f(t)$ as an expectation over $T_1 = Q_j(a+1)$, recalling that T_1 is a function of $S[t_0 :]$ and we need not condition on both.

$$\begin{aligned}
\sum_{t=t_0}^{N} f(t) &= \sum_{t=t_0}^{N} \Pr[T_1 = t+1] \cdot I(M_t; X_{t_0} \mid M_{t_0-1}, F, s[: t_0], S[t_0 :], T_1 = t+1) \\
&= I(M_{T_1-1}; X_{t_0} \mid M_{t_0-1}, F, s[: t_0], S[t_0 :]) \\
&= I(M_{Q_j(a+1)-1}; X_{q_j(a)} \mid M_{q_j(a)-1}, F, s[: q_j(a)], S[q_j(a) :]).
\end{aligned}$$

Thus we reach the following lower bound on $\mathrm{CI}(M \mid S, F)$, pulling the expectation over s back out to the front.

$$\begin{aligned}
\mathrm{CI}(M \mid S, F) &= \sum_{j,a} \Pr[L_j \geq a] \mathop{\mathbb{E}}_{\substack{t_0, s[:t_0] \\ |L_j \geq a}} \mathop{\mathbb{E}}_{\substack{t_1 \\ s[t_0:]}} \left[Z^{j,a}_{s[:t_0]} \right] \\
&\geq \sum_{j,a} \Pr[L_j \geq a] \mathop{\mathbb{E}}_{\substack{t_0, s[:t_0] \\ |L_j \geq a}} \left[\frac{k}{2} \cdot \mathbb{1}_{\{t_0 \neq N\}} \sum_{t=t_0}^{N} f(t) \right] \\
&= \frac{k}{2} \mathop{\mathbb{E}}_{s} \sum_{j=1}^{k} \sum_{a=1}^{\ell_j} \mathbb{1}_{\{q_j(a) \neq N\}} I(M_{q_j(a+1)-1}; X_{q_j(a)} \mid M_{q_j(a)-1}, s, F),
\end{aligned}$$

For any sequence s, the indicator random variable $\mathbb{1}_{\{q_j(a) \neq N\}}$ will be zero exactly once, corresponding to the final arrival. Since all of these information terms are bounded above by $H(X_t) \leq d$, we have

$$\mathrm{CI}(M \mid S, F) \geq \frac{k}{2} \mathop{\mathbb{E}}_{s} \sum_{j=1}^{k} \sum_{a=1}^{\ell_j} I(M_{q_j(a+1)-1}; X_{q_j(a)} \mid M_{q_j(a)-1}, s, F) - \frac{kd}{2}. \tag{9}$$

For any subpopulation j and sequence s (which fixes ℓ_j), Algorithm 2 defines an algorithm solving Task B' on ℓ_j examples. Denote the algorithm $M_{j,s}$ and let $\delta_{j,s}$ be its advantage. By

Algorithm 2: M' for Task B'

input: structured stream $x_1, \ldots, x_N \in \{0,1\}^d$; test example x_{test}; algorithm M; index $j \in [k]$; sequence $s \in [k]^N$; parameters d, k, ρ, N

For $i \in [k] \setminus \{j\}$, sample subpopulation parameters F_i;
for $\ell = 1, \ldots, N$ **do**
  if $s_\ell = j$ **then**
    Execute $M(x_t, j)$;
    $t \leftarrow t + 1$;
  else
    Draw $z_\ell \sim F_{s_\ell}$; `/* generate synthetic input */`
    Execute $M(z_\ell, s_\ell)$;
  end
end
Receive trained classifier M_N;
return $M_N(x_{\text{test}})$

construction, $\mathbb{E}_{j,s}[\delta_{j,s}] = \delta$, where δ is the advantage of M. By Corollary 32, we have a lower bound on the composable information cost of $M_{j,s}$:

$$\sum_{a=1}^{\ell_j} I(M_{q_j(a+1)-1}; X_{q_j(a)} \mid M_{q_j(a)-1}, s, F) \geq \frac{d}{160} \cdot \left(\frac{\delta_{j,s}}{\rho+1} - \frac{\rho}{d}\right)^4 - 1.$$

Note that this expression is convex in $\delta_{j,s}$, so we plug the lower bound into Equation (9), rewrite the sum over j as an expectation, and apply Jensen's inequality:

$$\begin{aligned}
\mathrm{CI}(M \mid S, F) &\geq \frac{k}{2} \mathop{\mathbb{E}}_{s} \sum_{j=1}^{k} \left(\frac{d}{160} \cdot \left(\frac{\delta_{j,s}}{\rho+1} - \frac{\rho}{d}\right)^4 - 1 \right) - \frac{kd}{2} \\
&= \frac{k^2}{2} \mathop{\mathbb{E}}_{s,j} \left(\frac{d}{160} \cdot \left(\frac{\delta_{j,s}}{\rho+1} - \frac{\rho}{d}\right)^4 - 1 \right) - \frac{kd}{2} \\
&\geq \frac{k^2 d}{320} \cdot \left(\frac{\mathbb{E}_{s,j}[\delta_{j,s}]}{\rho+1} - \frac{\rho}{d}\right)^4 - \frac{k^2}{2} - \frac{kd}{2}.
\end{aligned}$$

Since $\mathbb{E}_{j,s}[\delta_{j,s}] = \delta$, we are done. ■

Appendix F. Reductions from Core Problem to Agnostic Learning

In this section we show how agnostic learning algorithms for several natural functions classes can be turned into constant-advantage learning algorithms for Task C. Throughout, "agnostic learning" implies learning to sufficiently small constant accuracy with sufficiently high constant probability.

For each hypothesis class below, we restate the definition and then show (i) how to turn a labeled example from Task C into a labeled example for the given hypothesis class and (ii) how to use an

agnostically learned hypothesis h^* and a test example from Task C to output an answer (either *"S"*or *"U"*, for "uniform" and "structured") with constant advantage. None of these reductions require additional space or examples.

Direct Sums of k Dictators Let $\mathcal{X} = [k] \times \{0,1\}^{d'}$ (for $d' = d - \log_2 k$, so inputs can be described with d bits) and consider classifiers $h_{i_1,\ldots,i_k}$ specified by k indices in $[d']$, where $h_{i_1,\ldots,i_k}(j,x) = x_{i_j}$ (that is, for each j there is a single bit of x that determines the label).

Let $\mathcal{H}_{DS}$ denote the hypothesis class described above. We reduce from Task C with $\rho = 1$ and data dimension d', creating an algorithm M for Task C that uses an agnostic learning algorithm for the $\mathcal{H}_{DS}$. Given an example (x, j) from Task C, M constructs a labeled example $((j, x'), y)$ by drawing $y \in \{0,1\}$ randomly and setting $x' \leftarrow (y \cdot 1^d) \oplus x$, where $\oplus$ denotes bitwise XOR. Given a learned hypothesis h^* and test example (j, x_{test}), M constructs $((j, x'_{\text{test}}), y)$ in the same manner and outputs *"S"*if $h^*((j', x'_{\text{test}})) = y$ and *"U"*otherwise.

The XOR operation ensures that, when a subpopulation j has a feature $i \in [d']$ fixed to 0, the label y of (j, x) is 1 iff $x_i = 1$. For these subpopulations, then, there is a dictator that labels them exactly. For the other subpopulations (those with a feature fixed to 1, or with no fixed features), there is no dictator that labels examples with accuracy better than $\frac{1}{2}$. With $\rho = 1$, subpopulations have a 1-in-4 chance of getting a fixed feature with value 0; this choice is independent across subpopulations, so in expectation (over the choice of distribution) the best hypothesis in $\mathcal{H}_{DS}$ will have accuracy $\frac{1}{2}\left(1 - \frac{1}{4}\right) + 1 \cdot \frac{1}{4} = \frac{1}{2} + \frac{1}{8}$. With high probability an agnostic learning algorithm will return a hypothesis h^* with accuracy within a small constant of that, so in expectation h^* will have accuracy at least $\frac{1}{2} + c'$ for some positive constant c'.

We now show that the algorithm M for Task C described above has constant advantage. Recall the definition of advantage:

$$\mathop{\mathbb{E}}_{\substack{\mathcal{P}_{\text{mix}} \sim \mathcal{Q}_{k,d,\rho} \\ x=(x_1,\ldots,x_N) \sim_{\text{iid}} \mathcal{P}_{\text{mix}} \\ m \leftarrow M(x)}} \left[\Pr_{(j,y)\sim\mathcal{P}_{\text{mix}}}[m(j,y) = \text{"}S\text{"}] - \Pr_{(j,y)\sim\mathcal{U}}[m(j,y) = \text{"}S\text{"}] \right].$$

Here $\mathcal{P}_{\text{mix}}$ denotes the structured distribution and $\mathcal{U}$ denotes the uniform distribution over $[k] \times \{0,1\}^d$. Since M outputs *"S"* exactly when h^* is correct, the first probability is at least $\frac{1}{2} + c'$ in expectation. When the input is drawn from $\mathcal{U}$, the label y is uniform and independent of the pair (j, x'), so h^* is correct with probability exactly $\frac{1}{2}$.

We have established that an agnostic learning algorithm for $\mathcal{H}_{DS}$ yields a constant-advantage learner for Task Cwith the same space and sample efficiency, so any agnostic learner for $\mathcal{H}_{DS}$ requires $\Omega\left(kd' \cdot \frac{k}{N}\right)$ bits of memory. When $d = \omega(\log k)$, this is $\Omega\left(kd \cdot \frac{k}{N}\right)$.

Sparse Linear Classifiers over the Degree-2 Polynomial Features Let $\mathcal{X} = \{0,1\}^d$, and consider classifiers of the form $h(x) = \text{sign}(\langle w, \phi(x)\rangle)$ where $\phi(x)$ denotes the values of degree-2 monomials in the entries of x (so each entry of $\phi(x)$ equals $x_i x_j$ for two indices $i, j \in [d]$) and $w \in \{0,1\}^{\binom{d}{\leq 2}}$ has at most k nonzero entries.

Let $\mathcal{H}_{SL}$ denote the hypothesis class described above. We reduce from Task C by reducing from the Direct Sums of k Dictators class: any algorithm for agnostically learning $\mathcal{H}_{SL}$ on d dimensions can be used to agnostically learn $\mathcal{H}_{DS}$. Let $f : [k] \to \{0,1\}^k$ represent one-hot encoding and let $d' = d - k$. Given a labeled example $((j, x), y) \in [k] \times \{0,1\}^{d'} \times \{0,1\}$ for the Direct Sum of k

Dictators class, construct a labeled example $(f(j) \circ x, y)$, where $\circ$ denotes concatenation. Note that this construction requires $d \geq k$.

For any function $h_{i_1,\ldots,i_k}$ in the Direct Sums of k Dictators class, there is a function $h(z) = \text{sign}(\langle w, k(z)\rangle)$ with k-sparse $w \in \{0,1\}^{\binom{d}{\leq 2}}$ such that for all (j,x) we have $h_{i_1,\ldots,i_k}(j,x) = h(f(j) \circ x)$. Explicitly, indexing into w with pairs (i,j), we construct

$$w_{(i,j)} = \begin{cases} 1 & \text{if } j \leq k \text{ and } i = i_j + k \\ 0 & \text{otherwise} \end{cases}.$$

Thus, under this encoding, $\mathcal{H}_{DS} \subseteq \mathcal{H}_{SL}$ and any agnostic learning algorithm for $\mathcal{H}_{SL}$ is also one for $\mathcal{H}_{DS}$, since for a classifier h^* and constant c we have

$$\text{err}(h^*) \leq \min_{h \in \mathcal{H}_{SL}} \text{err}(h) + c \leq \min_{h \in \mathcal{H}_{DS}} \text{err}(h) + c.$$

This implies that agnostically learning $\mathcal{H}_{SL}$ for $d \geq k$ requires $\Omega\left(k(d-k) \cdot \frac{k}{N}\right)$ bits of memory. For $d \geq 2k$, this is $\Omega\left(kd \cdot \frac{k}{N}\right)$.

k-term 2-DNFs Let $\mathcal{X} = \{0,1\}^d$ and consider functions given by the OR of k terms, each of which is the AND of two input bits.

Let $\mathcal{H}_{DNF}$ denote the hypothesis class described above. We use the same reduction as we used in $\mathcal{H}_{SL}$: any algorithm for agnostically learning k-term 2-DNFs can be used to learn $\mathcal{H}_{DS}$. We use the same encoding: given a labeled example $((j,x),y)$, construct a labeled example $(f(j) \circ x, y)$. For every function $h_{i_1,\ldots,i_k} \in \mathcal{H}_{DS}$, there is an equivalent function $h \in \mathcal{H}_{DNF}$, namely:

$$h(x) = \bigvee_{j=1}^{k} \left(x_j \wedge x_{k+i_j}\right).$$

Thus the lower bound for $\mathcal{H}_{SL}$ also applies here, and for any $d \geq 2k$ agnostically learning $\mathcal{H}_{DNF}$ requires $\Omega\left(kd \cdot \frac{k}{N}\right)$ bits of memory.

Multiclass Sparse Linear Classifiers Let $\mathcal{X} = \{0,1\}^d$ and $\mathcal{Y} = [k]$ (so there are k distinct labels). Let $\mathcal{H}$ comprise all functions of the form $h(x) = \arg\max_{j \in [k]} \langle w_j, x\rangle$ where each $w_j \in \{0,1\}^d$ has $O(\log k)$ nonzero entries.

We reduce from Task C with $\rho = O(\log k)$. Given an example (j,x) from Task C, we construct a labeled example (x,j), with the subpopulation identifier as the label. Given a hypothesis h^* and test example (j, x_{test}), we output *"S"* if $h^*(x_{\text{test}}) = j$ and *"U"* otherwise.

Since with $\rho = O(\log k)$ the optimal linear multiclass classifier of this form has at most low constant error, this output will be correct with high constant probability and we have a space lower bound of $\Omega\left(\frac{k^2 d}{N\rho^4}\right) = \Omega\left(\frac{k^2 d}{N \log^4 k}\right)$.

Real-Valued Regression Let $\mathcal{X} = \{0,1\}^d$ and $\mathcal{Y} = [0,1]$. Consider functions realizable by a sparse two-layer neural network with a single hidden layer of k ReLU nodes, each of which is connected to at most $O(\log k)$ input nodes. The weights on the wires in the first layer are either 0 or 1, and those in the second layer are in $\left\{0, \frac{1}{k-1}, \frac{1}{k-1}, \ldots, 1\right\}$.

We reduce from Task C with $\rho = O(\log k)$. Note that we can express the set of target values as $\left\{0, \frac{1}{k-1}, \frac{2}{k-1}, \ldots, 1\right\} = \left\{\frac{j-1}{k-1}\right\}_{j=1}^{k}$. We modify the previous reduction: given an example (j, x) from Task C, we construct a labeled example $(x, (j-1)/(k-1))$. Given a hypothesis h^* and test example (j, x_{test}), we compute $\beta = \left|h^*(x_{\text{test}}) - \frac{j-1}{k-1}\right|$ and flip a coin that comes up heads with probability β. If the coins is heads output *"U"* and if tails output *"S"*.

Let E be the event that the coin comes up heads. On a structured input, the probability the reduction described above yields the incorrect answer is

$$\Pr[E] = \mathop{\mathbb{E}}_{j} \sum_{x} \Pr[X_{\text{test}} = x] \cdot \left|h^*(x_{\text{test}}) - (j-1)/(k-1)\right|,$$

exactly the expected error of the regression model on the structured distribution.

We now show that, with $\rho = O(\log k)$, the optimal regression model of this form has at most small constant error, say $\frac{1}{10}$. Each hidden node has a wire to the output node with a weight of the form $\frac{j-1}{k-1}$; call this "hidden node j." The activation function for this node is ReLU, with a fixed offset (or *bias*) b_j: $f(x) = \max\{0, x - b_j\}$. Call a subpopulation *good* if it has at least $2\log k$ indices fixed to 1. We now construct a specific neural network. If subpopulation j is not good, we set to 0 the weights of all wires incoming to hidden node j. If subpopulation j is good, we (i) take $2\log k$ wires running from indices fixed to 1 to hidden node j and set their weights to 1, and (ii) set $b_j = (2\log k) - 1$ as the bias. We claim that, with high probability over the choice of distribution, this construction has low constant error (in particular, with high probability over the choice of example it produces the exact correct label). To see this, first note that for $\rho \gg 2\log k$, with high probability a large constant fraction of the subpopulations are good. If this occurs, the neural network will have low constant error. Let random variable Z_j be the input to hidden node j on a random input X (drawn from the structured distribution, as in the reduction above). The label for X is $\frac{j-1}{k-1}$; the neural network outputs this value exactly if (i) j is a good subpopulation and (ii) for every other subpopulation $\ell \neq j$, $Z_\ell < 2\log k$. Condition (i) happens with high constant probability by assumption. Condition (ii) happens with probability at most $\frac{1}{k}$, since $Z_\ell = 2\log k$ only when all the wires leading to hidden node ℓ receive input 1, which happens with probability exactly $\frac{1}{k^2}$. A union bound over the (at most) $k-1$ other good subpopulations concludes the argument.

On uniform inputs, the output $h^*(X_{\text{test}})$ generated in the reduction has some distribution that is independent of j. Letting $\mu = \mathbb{E}_{x\sim\mathcal{U}}[h(x)]$, we have by Jensen's inequality that

$$\begin{aligned}\Pr[E] &= \mathop{\mathbb{E}}_{j} \mathop{\mathbb{E}}_{x_{\text{test}}\sim\mathcal{U}} \left[\left|h^*(x_{\text{test}}) - (j-1)/(k-1)\right|\right] \\ &\geq \mathop{\mathbb{E}}_{j} \left[\left|\mu - (j-1)/(k-1)\right|\right].\end{aligned}$$

This is at least $\frac{1}{8} - O(1/k)$, with the lower-order term coming from the fact that j is discrete. Informally, with probability $\frac{1}{2}$ we must have $|\mu - j/k| \geq \frac{1}{4}$.

Because $\Pr[E]$ has constant separation between the two cases, we have a learner for Task C with constant advantage and a space lower bound of $\Omega\left(\frac{k^2 d}{N \log^4 k}\right)$ for this real-valued regression task.

Appendix G. Upper Bounds

In this section, we present algorithms for some of the learning tasks considered in this paper. Appendix G.1 contains an algorithm for solving Task C, the core distributional task. Appendix G.2

presents an algorithm for agnostic learning of Direct Sums of k Dictators. Both of these algorithms are based on explicitly counting bits in each feature. Appendix G.3, in contrast, presents an analysis of the standard multiplicative weights algorithm for learning the class of Sparse Linear Classifiers over Degree-2 Polynomial Features. We analyze the algorithm for the distribution generated in the reduction to Task C. A similar analysis applies to learning Multiclass Sparse Linear Classifiers.

G.1. An Upper Bound for the Core Problem

In this section, we present a time-and-space efficient algorithm for Task C and sketch its analysis. On streams of length N, it uses $\tilde{O}\left(\frac{k^2 d}{N\rho}\right)$ space. When $N \gtrsim k \log k \log d$, the algorithm will have constant error. Recall that, when $\rho = o(d^{1/4})$, Theorem 12 gives a space lower bound of $\Omega\left(\frac{k^2 d}{N\rho^4}\right)$. Importantly, we do *not* prove that this algorithm is an agnostic learner that works for any distribution: we only prove that it has a constant advantage (see Definition 3).

We present M in Algorithm 3. It proceeds over τ epochs, within each epoch attending to only a subset of the k subpopulations.[9]

Space Analysis Storing the list of partitions $S_1, \ldots, S_\tau$ requires $O(k \log k)$ bits. During any epoch, M works with $O(k/\tau)$ subpopulations and stores a reference string (d' bits) and tracks candidate indices via another array of d' bits. It also tracks the B_j lists after each epoch: there are at most k of these and they require at most $O(\rho \log d)$ bits to specify. Thus M requires space

$$O\left(k \log k + \frac{k}{\tau} \cdot d' + k\rho \log d\right) = O\left(\frac{k^2 d \log k \log d}{\rho N}\right),$$

which matches our lower bound up to a factor of $\frac{1}{\rho^3} \log k \log d$.

Error Analysis We show that this algorithm has constant advantage. Setting $N_0 = ck \log k \log d$ for sufficiently large constant c ensures that, with high constant probability, within each of the τ epochs all k subpopulations receive at least $c' \log d$ examples for some constant c' (via the m-copy coupon collector problem). When M sees $c' \log d$ examples from a subpopulation j (during an epoch t in which $j \in S_t$), it will with high probability discard all unfixed features from the set $[d']$, and be left with a list B_j that is either (i) empty or (ii) contains only indices of fixed features. Since there are no more than ρ fixed features, this implies that with high probability M does not return `FAIL`.

We have established that, with high probability, M will identify all of the fixed features in the first $d' = d/\rho$ indices. This setting of d' ensures that, with constant probability over the choice of $r \in \{0, \ldots, \rho\}$ and $\{j_1, \ldots, j_r\} \in [d]$, at least one fixed feature will land in $[d']$. When M has correctly identified all the fixed indices in the first d' indices (and there is at least one fixed feature), it will always output *"S"* on structured inputs and will output *"S"* w.p. at most $\frac{1}{2}$ in the uniform case.

G.2. Agnostic Learning of Direct Sums of k Dictators

We show how to time-and-space efficiently agnostically learn the Direct Sum of k Dictators class described in the introduction. Recall the definition: Let $\mathcal{X} = [k] \times \{0,1\}^{d'}$ (for $d' = d - \log_2 k$,

9. The space usage can be lowered further by working with only a subset of indices. For simplicity, we ignore this regime.

Algorithm 3: M' for Task B'

input: stream $(x_1, j_1), \ldots, (x_N, j_N)$; test example x_{test}; parameters d, k, ρ, N, c

Set $N_0 \leftarrow ck \log k \log d$, $\tau \leftarrow \lfloor N/N_0 \rfloor$, $d' \leftarrow \lceil d/\rho \rceil$;
Create partition $S_1, \ldots, S_\tau \subseteq [k]$; `/* each of size k/τ */`
for $t = 1, \ldots, \tau$ **do**
  For all $\ell \in S_t$, set $x_\ell^{\text{ref}} \leftarrow$ NULL; `/* Leave uninitialized */`
  For all $\ell \in S_t$, set $C_\ell \leftarrow 1^{d'}$; `/* Candidate indices */`
  for $i = 1, \ldots, N_0$ **do**
    Receive next example (x, j);
    if $j \in S_t$ **then**
      if $x_j^{\text{ref}} = \mathit{NULL}$ **then**
        Set $x_j^{\text{ref}} \leftarrow x[1 : d']$; `/* Set reference string */`
      else
        Set $C_j \leftarrow C_j \wedge (x = x_j^{\text{ref}})$; `/* bitwise AND, EQUAL */`
      end
    end
  end
  for $\ell \in S_t$ **do**
    Set $B_\ell \leftarrow \{(a, x_\ell^{\text{ref}}[a]) : a \in [d'], C_\ell[a] = 1\}$; `/* Fixed bits, values */`
    if $|B_\ell| \geq \rho$ **then**
      return FAIL;
    end
  end
end
Receive (x_{test}, j);
for $(a, b) \in B_j$ **do**
  if $x_{\text{test}}[a] \neq b$ **then**
    return *"U"*;
  end
end
return *"S"*;

so inputs can be described with d bits) and consider classifiers $h_{i_1,\ldots,i_k}$ specified by k indices in $[d']$, where $h_{i_1,\ldots,i_k}(j,x) = x_{i_j}$ (that is, for each j there is a single bit of x that determines the label). Note that this class has size $(d')^k \le d^k$, so (by the standard uniform convergence argument for finite hypothesis classes, i.e., a Chernoff bound and union bound) $O(k \log d)$ samples suffice to guarantee that with constant probability ERM returns a classifier within constant accuracy of the best possible. Let $\kappa = k \log d$.

We first show how to implement ERM using $O(\kappa)$ samples and $O(d\kappa)$ space and time. By definition, the probability that a classifier $h_{i_1,\ldots,i_k}$ misclassifies a point $(j,x) \in \mathcal{X}$ is the probability that the label y differs from the bit x_{i_j}. To track the empirical error, then, it suffices to store a matrix $A \in \mathbb{R}^{k \times d}$, initialized to all zeros, and update it upon receiving a labeled example $((j,x),y) \in \mathcal{X} \times \{0,1\}$ in the following way:

$$\forall i \in [d], \quad A_{j,i} = \begin{cases} A_{j,i} + 1 & \text{if } x_i \neq y \\ A_{j,i} & \text{otherwise} \end{cases}.$$

After κ examples, we select the classifier $h_{i_1,\ldots,i_k}$ with the smallest empirical error: for each row $j \in [k]$ we select $i_j^* = \operatorname{argmin}_i A_{j,i}$, the index that minimizes the error.

This correctly implements ERM, so it is an agnostic learner. For each bit of our input we execute $O(1)$ operations, so the time used is $O(d\kappa)$. The space usage is just the matrix A, which has kd integer entries, each of which is between 0 and κ, so the algorithm uses $O(kd \log \kappa)$ space (this is $O(d\kappa)$ as long as $\log d \le k^{O(1)}$).

This algorithm naturally extends to longer streams of length $N = O(\tau\kappa)$ for $\tau \ge \Omega(1)$. We run a similar procedure over τ epochs, in each epoch receiving κ examples and working with a $\frac{1}{\tau}$ fraction of the rows of A, i.e., a subset of the subpopulations. At the end of the epoch, we store the minimum-error indices i_j^* for the rows we consider. At the end of the stream we assemble these indices to pick a single classifier. This algorithm implements ERM, runs in linear time $O(\tau\kappa d) = O(Nd)$, and uses space $O\left(\frac{d\kappa \log \kappa}{\tau}\right) = O\left(\frac{d\kappa^2 \log \kappa}{N}\right)$, matching our lower bound up to logarithmic factors.

G.3. Multiplicative Weights for Sparse Linear Classifiers

In this subsection we present an analysis (specific to our meta-distribution) of the standard multiplicative weights learning algorithm for the hypothesis class of Sparse Linear Classifiers over the Degree-2 Polynomial Features. Recall the definition of the class, which we denote $\mathcal{H}_{SL}$: Let $\mathcal{X} = \{0,1\}^d$, and consider classifiers of the form $h(x) = \text{sign}(\langle w, \phi(x) \rangle)$ where $\phi(x)$ denotes the values of degree-2 monomials in the entries of x (so each entry of $\phi(x)$ equals $x_i x_j$ for two indices $i,j \in [d]$) and $w \in \{0,1\}^{\binom{d}{\le 2}}$ has at most k nonzero entries. For simplicity, we will actually analyze learning monomials of *exactly* degree 2; this does not materially alter the algorithm.

The standard analysis of multiplicative weights yields an average regret bound that decreases with $\tilde{O}(1/\sqrt{N})$. Informally, in our setting the classifiers with constant advantage have $\sum_{i,j} w_{i,j} = \Theta(k)$. When scaled down so the weights represent a probability distribution, this corresponds to an advantage of $O(1/k)$. To guarantee this regret via the standard multiplicative weights analysis thus requires $N = \tilde{\Omega}(k^2)$. Our analysis shows that $N = \tilde{\Omega}(k)$ examples suffice for the distributions arising in our reduction.

Recall the distribution generated by our reduction from the core task (via Direct Sums of k Dictators). Each example is associated with a subpopulation $j \in [k]$. Because of our XOR construction,

for each subpopulation j with an index i fixed to 0, when the example comes from subpopulation j we have $x_i = y$. Call a pair (i, j) *good* if subpopulation j has index i fixed to 0. For any distribution generated in our reduction, then, the optimal classifier $w \in \{0,1\}^{\binom{d}{2}}$ sets $w_{i,j} = 1$ if (i, j) is good and to $w_{i,j} = 0$ otherwise. Since each subpopulation (independently) has such an index with probability $1/4$, with high probability the distribution generated in our reduction will have at least $k/5$ good pairs.

Consider the following instantiation of the multiplicative weights algorithm: we keep a weight vector $w^{(t)} \in [0,1]^{\binom{d}{2}}$, initialized to all ones. Upon receiving example $x^{(t)}$, the learner constructs loss vector $\ell^{(t)} \in \{0,1\}^{\binom{d}{2}}$ as follows:

$$\ell_{i,j}^{(t)} = \begin{cases} 1 & \text{if } \phi_{i,j}(x^{(t)}) = 1 \text{ and } y^{(t)} = 0 \\ 0 & \text{otherwise} \end{cases}.$$

The weight vector is then updated via $w_{i,j}^{(t)} \leftarrow \exp\left\{-\eta \ell_{i,j}^{(t)}\right\} w_{i,j}^{(t-1)}$. After processing N examples, the learner computes $p_{i,j} = \frac{w_{i,j}^{(N)}}{\sum_{i',j'} w_{i',j'}^{(N)}}$ and returns a final classifier of the form $\operatorname{sign}(\langle p, \phi(x)\rangle - \beta)$. The algorithm has η and β as parameters.

Observe that good indices will have $w_{i,j}^{(t)} = 1$ for all t. Pick a non-good index (i', j'). Using cases, we lower bound the probability it experiences loss. In the first case, suppose exactly one of i' or j' indexes into the one-hot encoding generated in the direct sum reduction. When the example $x^{(t)}$ is not from subpopulation j' we have $\phi_{i',j'}(x^{(t)}) = x_{i'}^{(t)} x_{j'}^{(t)} = 0$ and thus $\ell_{i',j'}^{(t)} = 0$. When the example is from subpopulation j' then the bit $\phi_{i',j'}(x^{(t)})$ and y are independent, so we have $\ell_{i',j'}^{(t)} = 1$ with probability $\frac{1}{4}$. In the second case, suppose both i' and j' index into the non-one-hot encoding section. Furthermore suppose there is a subpopulation for which neither index is fixed.[10] Then $\phi_{i',j'}(x) = x_{i'} x_{j'} = 1$ and $y = 0$ with probability at least $\frac{1}{8k}$. We need not analyze the third case (when i' and j' both index into the one-hot encoding section), since in our distributions this will always yield $x_{i'} x_{j'} = 0$. In the remainder of this analysis we will ignore indices that fall into this case.

Suppose η is a small constant, $\beta \leq \frac{1}{c_1 k}$ and, for every non-good index (i', j'), we have $w_{i',j'}^{(N)} \leq \frac{1}{c_2 d^2}$ (for sufficiently large constants c_1, c_2). These conditions suffice to ensure that the classifier returned above has constant advantage above random guessing. To see this, first note that the total mass assigned to *all* non-good indices is at most $\frac{1}{c_2} \ll 1$, which ensures that $p_{i,j} = \Omega(1/k) \geq \beta$ for every good index (i, j). Thus, when the example has label 1 and is from a subpopulation with a good index, the learned classifier will be correct. When the example has label 0, we have $\langle p, \phi(x)\rangle \leq \frac{k+d}{c_2 d^2} \leq \beta$, since at most $k + d$ indices in $\phi(x)$ can be nonzero and $d \geq k$.

It now remains to show that the algorithm learns such a classifier with high probability. Fix a data distribution containing at least $\frac{k}{5}$ subpopulations with good indices and fix a non-good index (i, j). Let random variable $Z_{i,j}$ denote $\sum_t \ell_{i,j}^{(t)}$. Then $Z_{i,j}$ stochastically dominates $\operatorname{Bin}(N, 1/8k)$, since each example is independent and receives loss with probability at least $\frac{1}{8k}$. Furthermore, $w_{i,j}^{(N)} = \exp\{-\eta Z_{i,j}\}$. To ensure $w_{i,j}^{(N)} \leq \frac{1}{c_2 d^2}$, then, we require $Z_{i,j} \geq \frac{1}{\eta} \log c_2 d^2 \geq c_3 \log d$ for some constant c_3. By a standard Chernoff bound, $\Pr[Z_{i,j} \leq c_3 \log d] \ll \frac{1}{d^2}$ whenever $N \gg k \log d$.

10. This happens with high probability over the choice of data distribution.

By a union bound over the $\leq d^2$ non-good indices, this ensures that the algorithm returns a constant-advantage classifier with high constant probability.

Contents