Memory Bounds for the Experts Problem

Vaidehi Srinivas
vaidehi@u.northwestern.edu
Northwestern University
USA

David P. Woodruff
dwoodruf@cs.cmu.edu
Carnegie Mellon University
USA

Ziyu Xu
xzy@cmu.edu
Carnegie Mellon University
USA

Samson Zhou
samsonzhou@gmail.com
Carnegie Mellon University
USA

ABSTRACT

Online learning with expert advice is a fundamental problem of sequential prediction. In this problem, the algorithm has access to a set of n "experts" who make predictions on each day. The goal on each day is to process these predictions, and make a prediction with the minimum cost. After making a prediction, the algorithm sees the actual outcome on that day, updates its state, and then moves on to the next day. An algorithm is judged by how well it does compared to the best expert in the set.

The classical algorithm for this problem is the multiplicative weights algorithm, which has been well-studied in many fields since as early as the 1950s. Variations of this algorithm have been applied to and optimized for a broad range of problems, including boosting an ensemble of weak-learners in machine learning, and approximately solving linear and semi-definite programs. However, every application, to our knowledge, relies on storing weights for every expert, and uses $\Omega(n)$ memory. There is little work on understanding the memory required to solve the online learning with expert advice problem (or to run standard sequential prediction algorithms, such as multiplicative weights) in natural streaming models, which is especially important when the number of experts and number of days are both large.

We initiate the study of the learning with expert advice problem in the streaming setting, and show lower and upper bounds. Our lower bound for i.i.d., random order, and adversarial order streams uses a reduction to a custom-built problem with a novel masking technique, to show a smooth trade-off for regret versus memory. Our upper bounds show new ways to run standard sequential prediction algorithms in rounds on small "pools" of experts, thus reducing the necessary memory. For random-order streams, we show that our upper bound is tight up to low order terms. We hope that these results and techniques will have broad applications in online learning, and can inspire algorithms based on standard sequential prediction techniques, like multiplicative weights, for a wide range of other problems in the memory-constrained setting.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
STOC '22, June 20–24, 2022, Rome, Italy
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9264-8/22/06.
https://doi.org/10.1145/3519935.3520069

CCS CONCEPTS

• **Theory of computation** → **Streaming models**; **Streaming, sublinear and near linear time algorithms**; *Lower bounds and information complexity*.

KEYWORDS

online learning with experts, streaming algorithms, sequential prediction algorithms

ACM Reference Format:
Vaidehi Srinivas, David P. Woodruff, Ziyu Xu, and Samson Zhou. 2022. Memory Bounds for the Experts Problem. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3519935.3520069

1 INTRODUCTION

The online learning with experts problem forms a general framework for sequential forecasting. On each day, the algorithm must make a prediction about an outcome, based on the predictions of a set of "experts." In the *online learning with experts problem*, we let n be the number of experts, and the algorithm is tasked with making predictions for T days. For each day, the algorithm is provided with the predictions of each expert in $[n] = \{1, 2, \ldots, n\}$, and then produces its own prediction based on all the information it has received in the previous days and the experts' predictions from the current day. The algorithm is provided with feedback on its prediction in the form of the cost of its prediction and the costs of all expert predictions on the current day. This process repeats for each day the algorithm makes predictions. The costs are restricted to be in the range $[0, \rho]$ for some parameter $\rho > 0$.

The online learning with experts problem has been primarily studied with respect to achieving the optimal regret, i.e., the additional total cost the algorithm incurs over the best performing expert in hindsight (expert that incurs the least cumulative cost), divided by the number of days. The well-known weighted majority algorithm was derived in [46] for solving the discrete prediction with experts problem (where the set of possible predictions is restricted to a finite set, and the cost is 1 if the prediction is correct, and 0 otherwise) with $O(M + \log n)$ total mistakes, where M is the number of mistakes the best expert makes. Optimizations to the weighted majority algorithm, such as the randomized weighted majority algorithm, achieve regret $O\left(\sqrt{\frac{\log n}{T}}\right)$. There have been many improvements to the weighted and randomized weighted

majority algorithms in subsequent work. There has also been work designing other algorithms that achieve this regret bound, such as follow the perturbed leader [40], that are less computationally expensive. However, all of these algorithms rely on a paradigm of keeping track of the cumulative cost of every expert, which requires the algorithm to use $\Omega(n)$ bits of memory on each day.

In this paper, we approach the online learning with experts problem from the angle of memory-bounded learning in the data stream model, and analyze whether there exists a low-memory algorithm that still performs reasonably well when compared to the best expert. Specifically, we consider the memory, i.e., the space complexity, in the streaming model. We first note that in the discrete version of the prediction with experts problem, where there are two possible answers, 0 and 1, and the cost of each decision is in $\{0, 1\}$ (0 for picking the correct answer, and 1 otherwise), a trivial random guessing algorithm is correct on half of the days in expectation. At the other extreme, the regret bound of $O\left(\sqrt{\frac{\log n}{T}}\right)$ can be achieved by storing $O(n)$ words (i.e., the weight of each expert) in memory and implementing the weighted majority algorithm. Thus it is natural to ask:

> *What is the space/accuracy tradeoff of an algorithm for the online learning with experts problem in the streaming model?*

Prior Work on the Experts Problem. The experts problem has been studied in the discrete decision setting [46] and variants where costs are determined by a loss function [36, 58–61]. Consequently, many different problems can be reframed into the experts problem framework, including portfolio optimization [21, 22], boosting [27], and forecasting [12]. For a complete reference on these results, see [14]. More recent work has shown that the multiplicative weights algorithm, a generalization of the weighted majority algorithm, has a tight asymptotic regret bound of $\sqrt{\frac{\ln n}{4T}}$ in the general case [34]. Previous work has also addressed the computational efficiency of an experts algorithm from the perspective of time complexity. Efficient algorithms have been considered in instances where extra assumptions can be made on the expert, such as imposing a tree structure [39, 57], assuming the experts are threshold functions [47], or assuming the experts have a certain linear structure [40]. Although many different algorithms have been developed in this area for these problems, they all revolve around tracking the performance of every expert, which requires at least $\Omega(n)$ memory. To the best of our knowledge, our work is the first that examines the experts problem from a streaming perspective.

1.1 Setup of Online Learning with Experts

In the experts problem, there are n experts and T days. On each day $t \in [T] = \{1, \ldots, T\}$, the i-th expert makes a prediction $x_i^{(t)}$ from the answer set A, e.g., $A = \{0, 1\}$ in the discrete decision case. The algorithm then receives feedback in the form of costs $c_i^{(t)}$ for the i-th expert on day t, where we define $c_0^{(t)}$ to be the cost incurred by the prediction of the algorithm. In the discrete decision case, the algorithm does not receive the feedback directly, but rather receives the "correct answer" $y^{(t)} \in \{0, 1\}$ on day t. We can model this as a data stream that encompasses the following elements in the prescribed order: $x_1^{(t)}, \ldots, x_n^{(t)}, c_0^{(t)}, \ldots, c_n^{(t)}$, where we see this sequence first for $t = 1$, then for $t = 2$, and so on until $t = T$. In the discrete decision case, for each t we instead only see the sequence $x_1^{(t)}, \ldots, x_n^{(t)}, y^{(t)}$. Further, in the discrete decision case, the costs are determined by $y^{(t)}$, and our algorithms do not need to see the sequence of the $c_1^{(t)}, \ldots, c_n^{(t)}$, which is a useful property of our algorithms. Our lower bounds, however, still hold in the discrete decision case even if the algorithm is given $c_0^{(t)}, \ldots, c_n^{(t)}$.

For the majority of the paper, we will consider the setup where our algorithm picks an answer made by an expert (equivalent to picking an expert). We only consider the discrete decision case to show that our bounds also hold when the expert explicitly picks an answer. We use $M \in \mathbb{N}$ to denote an upper bound on the number of mistakes made by the best expert. Note that the streaming algorithm sees these sequences in order of increasing t, and cannot go back to see predictions or correct answers from previous days; the only information it has about such values is from what it stores in its memory. Further, our upper bounds hold in a setting where the expert predictions and feedback are considered distinct stream elements, i.e., streaming memory is needed to store information about the expert predictions of the *current* day (in addition to all of its information from the past days), $x_1^{(t)}, \ldots, x_n^{(t)}$, before the costs and correct answer of the current day are seen. Note that our lower bounds hold even in the setting where $x_1^{(t)}, \ldots, x_n^{(t)}$ and $y^{(t)}$ can be viewed at the same time.

Random-Order Streaming Model. We frequently consider the experts problem in the *random-order streaming model* [35], where we assume the days are permuted in a uniformly random order with respect to the order the days arrive in the stream. This is equivalent to saying that the input distribution over $(x^{(1)}, c^{(1)}), \ldots, (x^{(T)}, c^{(T)})$ is *exchangeable*, i.e., any permutation on $[T]$ of the data is equally likely in the input distribution. We note that this is slightly different from the traditional notion of a random-order stream where the permutation is over elements in the stream, whereas in the experts problem, the permutation is over the pairs of expert predictions and costs that are associated with a single day in the problem. Note that the permutation only applies across the ordering of days — the order of experts remains fixed and unchanged across the days. In addition, the exchangeability assumption in the random order model allows it to subsume the i.i.d. model, where $(x^{(t)}, c^{(t)}) \sim \mu$ for all $t \in [T]$. As a result, all upper bounds that hold in the random order model also hold in the i.i.d. model.

1.2 Our Contributions

We initiate the study of the online learning with experts problem in the streaming model. We consider streams where the order of days and corresponding expert choices and outcomes may be either (1) worst case streams, (2) random-order streams, where the order of the days is assumed to be in a random order, or (3) i.i.d. streams, where the expert choices and outcomes on each day are drawn from a fixed distribution. We prove upper and lower bounds for the experts problem in these different streaming models. We refer to a word of memory as $O(\log(nT))$ bits, and we use the $\widetilde{O}(\cdot)$ notation to suppress $\log^{O(1)}(nT)$ factors.

Lower Bound for I.I.D., Random Order, and Arbitrary Streams. We show that any algorithm which achieves a regret of δ with constant probability in the streaming model must use $\Omega\left(\frac{n}{\delta^2 T}\right)$ bits of memory. This lower bound holds for both arbitrary and random-order streams. In fact, the lower bound is valid even if the correct answers and the predictions of each expert are i.i.d., which implies validity of the lower bound in random order and arbitrary-order settings.

Theorem 1. *Let δ, $p < \frac{1}{2}$ be fixed constants, i.e., independent of other input parameters. Any algorithm that achieves δ regret for the experts problem with probability at least $1-p$ must use at least $\Omega\left(\frac{n}{\delta^2 T}\right)$ space. This lower bound holds even when the costs are binary and expert predictions, as well as the correct answer, are constrained to be i.i.d. across the days, albeit with different distributions across the experts.*

Note that for $\delta = \sqrt{\frac{1}{T}}$, Theorem 1 implies that $\Omega(n)$ space is necessary. Thus, we should not expect to obtain the same regret bounds as the best algorithms for online learning with experts, e.g., multiplicative weights, when the space is significantly smaller than n. In other words, Theorem 1 shows a separation for the online learning with experts problem between the classical centralized setting and the streaming model when only $o\left(\frac{n}{\log n}\right)$ space is permitted, in which case the $O\left(\sqrt{\frac{\log n}{T}}\right)$ regret that is obtainable for the classical centralized setting cannot be achieved by any streaming algorithm with $o\left(\frac{n}{\log n}\right)$ space. Moreover, this separation holds for random-order, i.i.d., and arbitrary-order streams.

Upper Bounds for Random-Order Streams. We next consider upper bounds for the online learning with experts problem in the random-order streaming model. We consider the case where the costs of the decisions of each expert have value in $[0, \rho]$ rather than in $\{0, 1\}$, where $\rho > 0$ is called the *width* of the problem.

Theorem 2 (Informal). *There exists an algorithm that takes a target parameter $\delta > \sqrt{\frac{16\log^2 n}{T}}$, uses $\tilde{O}\left(\frac{n}{\delta^2 T}\right)$ space, and achieves an expected regret of $\rho\delta$ in the random-order model, where ρ is the width of the problem.*

When the width ρ is normalized to 1, our space dependence on the regret δ in Theorem 2 is tight, given the lower bound in Theorem 1. We present the formal version of Theorem 2 as Theorem 9 in Section 4. Our algorithm shows that there are indeed natural tradeoffs between the memory required, the regret δ of the streaming algorithm, the total number T of days, and the total number n of experts.

Upper Bounds for Predictions on Arbitrary Streams. We next consider the online learning with experts problem in the more general adversarial streaming model. We propose an algorithm that is correct on a "large" fraction of days that allows for a space-accuracy tradeoff, even if the best expert makes a number of mistakes that is almost linear in T.

Theorem 3. *(Informal) Given an upper bound $M \in \left[0, \frac{\delta^2 T}{1280\log^2 n}\right]$ on the cost that the best expert incurs, and a target regret $\delta > \sqrt{\frac{128\log^2 n}{T}}$, there exists a streaming algorithm that uses space $\widetilde{O}\left(\frac{n}{\delta T}\right)$ and with probability at least $4/5$, has regret $\rho\delta$, where ρ is the width of the problem. The algorithm does not need to know M in advance.*

Theorem 3 is interesting across multiple regimes of parameters for arbitrary-order streams, i.e., worst-case streams. First, our algorithm provides space-accuracy tradeoffs that can achieve a sublinear number of mistakes. Specifically, the number of mistakes made by our algorithm nearly matches the asymptotic regret bound of the multiplicative weights algorithm [34] for corresponding values of $\delta = O\left(\sqrt{\frac{\log^2 n}{T}}\right)$. For constant $M = O(1)$, a natural algorithm can achieve δ regret simply by iteratively choosing "pools" of $\widetilde{O}\left(\frac{n}{T}\right)$ experts for "rounds" until the best expert is identified, where a round lasts until the pool becomes empty. Here, each expert is removed immediately after an incorrect prediction, and throughout a round, the algorithm makes a prediction by majority vote. We remark that the space bound of Theorem 3 matches this natural algorithm in the regime where $M = O(1)$, even when the algorithm is oblivious to the value of M, and generalizes to handle larger values of M. In contrast, the natural algorithm uses $\tilde{O}\left(\frac{Mn}{\delta T}\right)$ space to achieve δ regret for larger values of M. In particular for constant δ, our algorithm guarantees correctness on a constant $1-\delta$ fraction of days using $\widetilde{O}\left(\frac{n}{T}\right)$ space, even if the best expert is incorrect on $O\left(\frac{T}{\log^2 n}\right)$ days, i.e., the best expert makes almost a linear number of mistakes. Notably, this worst-case upper bound remains agnostic to the number of possible answers in the discrete decision setting, where the set of all possible answers is a finite set. Finally, we remark that Theorem 3 uses *less space* than the lower bound of Theorem 1 (recall that $\delta < 1$), revealing that the hardness of Theorem 1 stems from the best expert making a "large" number of mistakes.

On the other hand, Theorem 3 requires that the best expert incurs at most $O\left(\frac{T}{\log^2 n}\right)$ cost, even for a constant factor approximation. It is an interesting open question what regret bounds are achievable on arbitrary-order streams when the best expert is allowed to incur a constant fraction of errors, i.e., cost $O(T)$. We present the formal version of Theorem 3 as Theorem 7 in Section 3.

We defer missing proofs to the full version of the paper.

1.3 Technical Preliminaries: Standard Sequential Prediction Algorithms

At a high level, our memory-constrained algorithms for the experts problem sample subsets or "pools" of the n experts, and use standard sequential prediction techniques on these subsets. For the sake of modularity, we will formulate this as a black-box call to a sequential prediction algorithm (without memory constraints). In this section we define the properties that we require from such a sequential prediction algorithm, and suggest some candidate algorithms.

Definition 1 (Sequential prediction algorithm). *We say that an algorithm $\mathcal{A}$ is a valid* sequential prediction algorithm *if, given an instance of the online learning with experts problem such that the cost of expert i on day t, $c_i^{(t)} \in [0,1]$ for all $i \in [n]$ and $t \in [T]$,*

and a target parameter ε, we have that

$$\mathbf{E}[\textit{cost of } \mathcal{A}] \leq (1+\varepsilon)\left[\sum_{t=1}^{T} c_i^{(t)}\right] + \frac{\beta \ln n}{\varepsilon},$$

for some fixed constant β, and $\mathcal{A}$ maintains $O(n)$ words of memory.

Note that the constraint that $\mathcal{A}$ maintains $O(n)$ words of memory is not very restrictive, as this allows $\mathcal{A}$ to maintain the running total cost of each expert, which is all we need to run many of the best possible algorithms for this problem.

Perhaps the most well-studied such algorithm is the multiplicative weights algorithm. Here, we introduce this algorithm in the formulation of [3] (Algorithm 1).

Theorem 4 (Theorem 2.1 in [3]). *Suppose $c_j^{(t)} \in [0,1]$ for all $j \in [n], t \in [T]$ and $\varepsilon \leq \frac{1}{2}$. Then the multiplicative weights algorithm (Algorithm 1) satisfies for each $i \in [n]$,*

$$\sum_{t=1}^{T}\sum_{j=1}^{n} c_j^{(t)} p_j^{(t)} \leq (1+\varepsilon)\left[\sum_{t=1}^{T} c_i^{(t)}\right] + \frac{\ln n}{\varepsilon},$$

(where the left-hand side of this inequality is the expected cost of the multiplicative weights algorithm, and the right-hand side is in terms of the cost of some particular expert.)

Algorithm 1 The multiplicative weights algorithm.

Input: Number n of experts, number T of rounds, parameter ε

1: Initialize $w_i^{(1)} = 1$ for all $i \in [n]$.
2: **for** $t \in [T]$ **do**
3: $\quad p_i^{(t)} \leftarrow \frac{w_i^{(t)}}{\sum_{i\in[n]} w_i^{(t)}}$
4: $\quad$ Follow the advice of expert i with probability $p_i^{(t)}$.
5: $\quad$ Let $c_i^{(t)}$ be the cost for the decision of expert $i \in [n]$.
6: $\quad w_i^{(t+1)} \leftarrow w_i^{(t)}\left(1-\varepsilon c_i^{(t)}\right)$
7: **end for**

Another example of a sequential prediction algorithm is the follow the perturbed leader algorithm due to [40] (Algorithm 2), which maintains running totals of the cost incurred by each expert. On each day, the algorithm randomly "perturbs" the costs, and then follows the prediction of the expert with the lowest perturbed cost.

Theorem 5 (Theorem 1.1 in [40] applied to the experts problem). *Suppose $c_i^{(t)} \in [0,1]$ for all $i \in [n], t \in [T]$, and $\varepsilon \leq 1$. The FPL* (Algorithm 2) satisfies for each $i \in [n]$,*

$$\mathbf{E}\left[\textit{cost of FPL}^*(\varepsilon)\right] \leq (1+\varepsilon)\left[\sum_{t=1}^{T} c_i^{(t)}\right] + \frac{8(1+\ln n)}{\varepsilon}.$$

For $n \geq 3$, this satisfies the conditions of Definition 1 for $\beta = 16$.

Algorithm 2 The follow the perturbed leader algorithm (FPL*) from [40], instantiated for the experts problem.

Input: Number n of experts, number T of rounds, parameter ε

1: **for** $t \in [T]$ **do**
2: $\quad$ **for** $i \in [n]$ **do**
3: $\quad\quad$ Draw r from a standard exponential distribution, and set $p_i^{(t)} = 2r/\varepsilon$ with probability $\frac{1}{2}$, and $p_i^{(t)} = -2r/\varepsilon$ otherwise
4: $\quad$ **end for**
5: $\quad$ Follow the expert i for whom the sum of their total cost so far and $p_i^{(t)}$ is the lowest
6: **end for**

1.4 Related Work on the Experts Problem

While the space complexity of the experts problem in the models described above has not been previously studied, many related problems have been studied in the streaming model, and there are many results for the problem when space is not constrained. We discuss how the hardness, proof techniques, and algorithms for these problems relate to the space-constrained experts problem.

Identifying an Approximately Good Expert is Harder. A closely related problem is the expert identification problem, where the algorithm must output the index of an expert that does approximately as well as the best expert at the end of the stream. A natural strategy might be to use a heavy-hitter algorithm to identify the best expert. The *heavy hitters* problem is a classical problem that has been well-studied in the streaming model. In the ε-heavy hitters problem, the algorithm sees a stream of elements from $[n] = \{1, 2, \ldots, n\}$. At the end of the stream, it must output any item that accounts for at least an ε -fraction of the ℓ_p norm, for some real value $p \geq 1$. One can consider the experts problem as a data stream of insertions to a vector $V \in \mathbb{N}^n$, where the value at index i is the number of days the i-th expert has been correct. Running an ℓ_p ε-heavy hitters algorithm over this stream would return the experts that are correct on at least $\varepsilon \|V\|_p$ days (see, e.g., [6, 10, 11, 17, 19, 44] and the references therein). Another variant of the heavy hitters problem is the ε-maximum problem, which outputs a value, rather than the the index, that is within $\varepsilon \|V\|_1$ of the true maximum element (see, e.g., [6]). If we were to select an input where the best expert makes no mistakes, while all other experts are correct on $T/2+1$ days, ε would need to be on the order of $O(1/n)$ to find the best expert, and any heavy hitters algorithm for finding any good expert would require $\Omega(n)$ space, given that their memory usages depends at least linearly on $1/\varepsilon$.

More generally, we note that there is a reduction from the well-studied two player set disjointness communication problem (see, e.g., [5, 8, 16, 62]), in which Alice is given a set $X \in \{0,1\}^n$ and Bob is given a set $Y \in \{0,1\}^n$ and their goal is to distinguish whether $|X \cap Y| = 0$ or $|X \cap Y| = 1$. Observe that Alice can create a stream of n days so only expert i can be correct on day i and furthermore, expert i is correct on day i if and only if $X_i = 1$. Bob similarly creates a stream of n days so that the best expert on their combined stream is $X \cap Y$ if $|X \cap Y| = 1$, thus allowing Bob and Alice to solve the set disjointness problem, by using another round

of communication to check whether the best expert is indeed in both X and Y. Note that we can extend this to T days by copying Alice $T/2$ times and copying Bob $T/2$ times, and placing all copies of Alice before all copies of Bob. Since set disjointness requires total communication $\Omega(n)$ even for randomized protocols, this reduction immediately implies there is an $\Omega(n)$ lower bound for identifying the best expert, even for randomized streaming algorithms and even if the best expert makes no mistakes while any other expert is correct on at most half of the total number of days. These results show a distinction between the hardness of finding an expert that does well, relative to the best expert, and the hardness of predicting well relative to the best expert. We study the latter problem, and break this $\Omega(n)$ lower bound by obtaining an $\widetilde{O}(n/T)$ upper bound for the setting of parameters above.

Multiplicative Weights. Multiplicative weights is a meta-algorithm that maintains weights over a set of objects. On iteration $i+1$, the algorithm sets the weight of item x according to the update rule $w_x^{(i+1)} \leftarrow w_x^{(i)}\left(1-\varepsilon P_x^{(i+1)}\right)$, where ε is the learning rate, and $P_x^{(i+1)}$ is some penalty applied to x. Forms of the multiplicative weights algorithm have been independently discovered for problems in many fields from as early as the 1950s [13]. Many of these problems generalize to the discrete prediction with expert advice problem, first analyzed by Littlestone and Warmuth [46], or the continuous online learning with expert advice problem, and for both problems, the multiplicative weights algorithm achieves asymptotically optimal regret [20, 48]. Notable applications of the multiplicative weights algorithms include AdaBoost [27] and approximately solving zero-sum games [28]. Multiplicative weights can also be used to efficiently approximate a wide class of linear programs and semi-definite programs, which have given fast approximations for a broad range of NP-complete problems, including the traveling salesperson problem, some scheduling problems, and multi-commodity flow [30, 49]. Recent work has analyzed the multiplicative weights algorithm for stochastic experts [1, 51, 54], and bounded the regret in terms of the variance of the best expert [15, 38]. There has also been much recent work in adaptively optimizing the learning rate [18, 25, 43]. For a more complete overview of the history and applications of multiplicative weights, the reader is referred to the surveys [3, 7, 14, 26].

Follow the Perturbed Leader. The follow the perturbed leader (FPL) algorithm (Algorithm 2), due to Kalai and Vempala [40], achieves similar guarantees to multiplicative weights for the experts problem, and can be efficiently generalized to a large set of online problems. They define a linear generalization of these online problems. Consider a set of possible decisions $\mathcal{D} \subset \mathbb{R}^n$ and a set of possible events $\mathcal{S} \subset \mathbb{R}^n$. On each day, the algorithm chooses a decision $d_t \in \mathcal{D}$. Then the event of that day $s_t \in \mathcal{S}$ is revealed, and the algorithm incurs cost $d_t \cdot s_t$. The total cost of the algorithm, $\sum_t d_t \cdot s_t$, is evaluated against the best static decision in hindsight, $\min_{d\in\mathcal{D}} d \cdot \sum_t s_t$. We can model the standard experts problem as an instance of this problem where n is the number of experts, $\mathcal{S}$ is made up of vectors with entries between 0 and 1, and $\mathcal{D}$ is the set of vectors for which one index is 1, and all others are 0. For each day t, the FPL algorithm then calculates a random perturbation vector $p_t \in \mathbb{R}^n$. Then it follows the decision that optimizes $\min_{d\in\mathcal{D}} \quad d \cdot \left(p_t + \sum_{i\le t} s_i\right)$. In the case of the standard experts problem, this is generating a perturbation for each expert, and then following the expert for whom the sum of their cost and their perturbation is the smallest.

This linear generalization means that for some structured problems, this algorithm is computationally more efficient than multiplicative weights. However, like multiplicative weights, it requires access to the running cost $\sum_t s_t$. For the general experts problem, this is an n dimensional vector, so this still requires $\Omega(n)$ memory.

Online Convex Optimization. A common setting in online convex optimization is to minimize the regret, defined by $\sum_{t=1}^{T} f_t(x_t) - \min_{x\in X} \sum_{t=1}^{T} f_t(x)$, where X is some convex set and the functions $f_1, \ldots, f_T : X \to \mathbb{R}$ are convex cost functions. A special case of online convex optimization is when the goal is to minimize a convex function f over a convex domain X. If the algorithm is given oracle access to a noisy gradient, i.e., an oracle that outputs an unbiased estimate to the gradient with "small" variance, then stochastic gradient descent is known to have expected regret at most $O(1/\sqrt{T})$. For a more precise statement, see [37, 55, 56].

Multi-Armed Bandits. Space complexity has been considered in the related problem of *multi-armed bandits*. The multi-armed bandits problem is a classic problem in reinforcement learning, in which there are some number of *arms* and each arm has a fixed reward distribution that can be sampled from at each time step. [45] has shown that only constant space is required to achieve regret that is within an $O(1/\Delta)$ factor of the optimal regret, where Δ is the difference between the mean reward of the best and second best arms. This space efficient algorithm for the bandits problem, however, is not applicable to the experts problem, since it does not capture the adversarial nature of sequential prediction, i.e., the performance of an expert changes on a daily basis, while the reward distribution of each arm is static in the standard streaming model. Nor does it leverage the ability of the algorithm to view the results of all experts on each day. Hence, expert algorithms are not comparable to bandit algorithms, since an expert algorithm has information about all "arms" on each day. [4] analyze the problem of finding the best arm with optimal sample complexity in the streaming arm model, where the algorithm must save an arm to sample from it. They prove a tight bound of requiring $\Theta(k)$ arms of space to find the top k arms. These results show that solving the experts problem is fundamentally harder than the multi-armed bandits problem, since solving the bandits problem with low regret can be done in constant space.

Learning in Streams. There has been a substantial amount of work that analyzes the tradeoff between space and sample complexity for statistical learning and estimation problems in the streaming model, where the stream elements are assumed to be samples drawn i.i.d. from a fixed distribution. A series of work has studied the problem of inferring the index of a row sampled from a matrix [32, 33, 52], and the parity learning problem [42, 53]. More recent work has also analyzed distribution testing relevant to cryptographic settings, such as lower bounds in the streaming model for testing against Goldreich's pseudorandom generator [31]. Other lines of work examine more specific learning problems in the streaming model, such as finding correlations in multivariate

data [24], collision probability estimation, finding the connectivity of an undirected graph, and rank estimation [23]. In the distributed setting, communication lower bounds have been analyzed for convex optimization [2]. Our study of the experts problem, however, makes no assumptions on the distributions each element in the stream is drawn from, and is a prediction rather than an inference problem. Space usage was considered by [41] in their analysis of pairwise losses in online learning, but not in the general sense of space complexity. They consider a modified form of regret under the pairwise loss with respect to a finite buffer of previous stream items rather than all previous items in the stream. This problem, however, is very different from the experts setting of proving space complexity bounds for any algorithm, while still comparing the performance of the algorithm to that of the best expert.

1.5 Overview of our Techniques

1.5.1 Lower Bound by Reducing to Distributed Detection. We first give an overview of the lower bounds that we present in Section 2. We create a new problem that combines n instances of the distributed detection problem [9], with each instance corresponding to an expert — we call this new problem DiffDist. In essence, DiffDist is the problem of distinguishing between:

(1) Every expert flips an independent fair coin to determine its prediction, i.e., each expert predicts correctly with probability $\frac{1}{2}$.
(2) A single expert predicts correctly with probability $\frac{1}{2} + O(\delta)$ on each day, and every other expert predicts correctly with probability $\frac{1}{2}$.

We note that the predictions of each expert form a separate instance of the distributed detection problem, each of which has a randomized communication lower bound of $\Omega(\frac{1}{\delta^2})$. We then use a careful combination of existing techniques, e.g., [9, 29, 50, 63] to show an $\Omega\left(\frac{n}{\delta^2}\right)$ randomized communication lower bound for the DiffDist problem.

After having shown a randomized communication lower bound for the DiffDist problem, one of our key ideas is then to introduce a randomized reduction from the DiffDist problem, so that each player corresponds to a day, and each expert prediction is the corresponding bit in the DiffDist problem. We would like to say that the single expert with higher probability of correctness translates to a separation in the cost of the algorithm for online learning with expert advice. However, even if the experts are incorrect, it seems possible that an algorithm could ignore the experts and still have high accuracy. For example, if we let the the correct answer be 1 for every day, an experts algorithm could be perfectly accurate in case (1) simply by predicting 1 everyday. Thus, we create a mask for each day by setting it to be a random fair coin flip and we XOR both the outcome of each day, as well as the output of each expert, by the mask. Hence in case (1), the algorithm cannot have accuracy significantly higher than $\frac{1}{2}$ with constant probability, regardless of its output sequence, since the expert predictions defined via our masking, and the correct answer, all remain independent fair coin flips. On the other hand, in case (2), the algorithm must be correct on a good fraction of days to keep up with the best expert. This results in a separation in performance between cases (1) and (2). Our hard instance is inherently distributional, and the choice of hard distribution is crucial to ensure that the algorithm has no information in case (1) about the correct answer on a future day, regardless of the past.

1.5.2 Upper Bound for Random-Order Streams. We first consider upper bounds for the online learning with experts problem in random-order streams, corresponding to the results in Section 4. For simplicity, we will describe an overview of our algorithms and proof techniques in the setting where the costs are restricted to $\{0, 1\}$, i.e., an expert either makes a mistake or does not. Due to space constraints on the algorithm, we can only afford to sample a small number of experts in each round. Thus our algorithm (in Algorithm 6) initializes a pool of $k = O\left(\frac{n \log n}{\delta^2 T}\right)$ different experts from $[n]$ at the beginning of each round. We then run a standard sequential prediction algorithm on this restricted pool of experts. If this pool of experts makes too many total mistakes *in expectation*, then we resample a new pool of experts and run the sequential prediction algorithm on this new pool. Note that we can explicitly compute the expectation of the pool, since we have access to the expert predictions in the pool, their corresponding weights, and the outcomes across each day over the duration of the round.

In the random-order model, we can show that if the best expert does not make too many mistakes, then the sequential prediction algorithm will perform well upon sampling the best expert. Thus, to prove our algorithm has low expected regret, we must demonstrate that one of two cases must be true: (1) the algorithm does not sample the best expert because the algorithm has already been performing well or (2) the algorithm samples the best expert "early" enough in the stream and the best expert is not subsequently discarded. To handle the first case, we observe that we only delete the pool of experts if the sequential prediction algorithm is performing poorly, so if the total number of rounds is low, there must be rounds with significantly long duration and also sufficiently high expectation. Thus the algorithm will perform well in expectation.

Analysis of Pool Selection Times. However, the second case foreshadows an issue in the analysis: if the best expert is never discarded from the pool, then the best expert can only be added to the pool in the last round. Thus, even if we condition on the entire algorithm using R rounds for some integer $R > 0$, the probability that the best expert is not added to the pool in round R may be significantly larger than the probability that the best expert is added to the pool in round R. This issue is further compounded by the fact that once a pool is selected, the day on which the next round begins is completely deterministic and possibly adversarial, so it does not suffice to, for example, consider the probability the best expert is added to the pool on a random day.

We overcome this challenge by "decoupling" the number of rounds from the sampling of the best expert. What we mean by this is as follows. We consider the distribution of all days on which new rounds begin. We can simulate the sampling process with a sequence of times $t_1, t_2, \ldots$ so that each t_i is drawn from the distribution of possible times that round i can end, conditioned on the entire history of the process up to time t_{i-1}. Observe that this is a well-defined sequential process for defining each term t_i, in the

sense that to obtain t_i, we can simply draw from a distribution of possible durations for a round and then add the duration to t_{i-1}.

We observe that due to the distributional properties of the random-order model, once the algorithm samples the best expert, then with high probability the sequence of rounds will terminate. Moreover, since the algorithm performs sampling with replacement between rounds, the probability that the best expert is added to the pool on each drawing of k experts is the same across all rounds. Thus, the probability distribution for the total number of rounds can be related to a geometric distribution — if the algorithm uses R rounds, then the best expert cannot be sampled in the first $R-1$ rounds with high probability. This allows us to show that if the algorithm samples the best expert, there was likely a small number of rounds and therefore the total cost of the algorithm is not too high.

Unknown Error for the Best Expert. It remains to remove the assumption of knowing the error rate for the best expert, for which we again use the promise of the random order streaming model, which allows us to use short prefixes of days in order to obtain an estimate of the error rate.

To that end, we note that it suffices to acquire a $(1+O(\delta))$-approximation to the number of mistakes made by the best expert, since the regret will only be increased by $O(\delta)$ if we have such an estimate. To find a $(1+O(\delta))$-approximation to the number of mistakes of the best expert, we initialize our guess γ for the mistakes to be $\frac{T}{2}$. We then split the stream into epochs of length $O\left(\frac{\delta T}{\log \frac{1}{\delta}}\right)$ and perform a binary search by repeatedly updating γ depending on whether the current guess is too high or too low based on the performance of the best expert in the epoch. Thus by epoch k, our guess γ is within a $\left(1+\frac{1}{2^k}\right)$-factor of the actual number of mistakes made by the best expert. Hence it suffices to use $O\left(\log \frac{1}{\delta}\right)$ epochs to update γ, which can only increase the total regret additively by δ, since each epoch has length $O\left(\frac{\delta T}{\log \frac{1}{\delta}}\right)$.

1.5.3 Upper Bound for Arbitrary-Order Streams. We consider arbitrary order streams in Section 3. Unfortunately, when the stream no longer arrives in a random order, then we again have no guarantees on how the best expert will perform if it is sampled at any given time. We observe that if the costs are $\{0,1\}$ for each day, then we can attempt to emulate the simpler majority elimination algorithm by removing all incorrect experts on a day on which the algorithm is incorrect. Thus, our starting point is an algorithm that initializes a pool of $k = O\left(\frac{n \log n}{T\delta}\right)$ different experts from $[n]$ at the beginning of each round and removes incorrect experts on incorrect days until the pool is depleted, at which point the next round begins and a new pool of k different experts from $[n]$ is initialized. On each day, the algorithm outputs the majority vote of the experts in the pool.

However, removing all incorrect experts would significantly increase the chance that the best expert is removed from the pool, even over multiple rounds. For example, if the best expert makes a constant number of mistakes, then it is possible that it only survives a constant number of days before it is removed from the pool. If all other experts perform poorly, then the algorithm could only be correct on a constant number of days in every group of $O\left(\frac{T\delta}{\log n}\right)$ days, which is subconstant even if δ is a constant. Therefore, we should relax the conditions for removal of experts; a natural choice is to only remove experts that have been incorrect for $\frac{\delta}{4}$ fraction of the time since the pool has been initialized, *regardless of the outcome of each day*. The intuition is that all experts make errors on at most a $\frac{\delta}{4}$ fraction of the days in the pool, so the algorithm should make errors on at most an $O(\delta)$ fraction of the days over the pool.

Accumulation of Errors. However, this surprisingly fails because it allows experts to "build-up" future errors by having good accuracy on previous days. For example, suppose we have a pool of 100 experts and we choose to eliminate experts that are wrong on half of the days since the pool has been initialized. Suppose all experts are correct on the first 50 days but then from day 51 to day 100, exactly half the experts are wrong on every single day. On day 100, half of the experts are eliminated and the algorithm has made 50 mistakes, but the remaining experts have not made any mistakes. Thus, even if half of the remaining 50 experts are wrong on *every single day* from day 101 to day 200, they will not be eliminated until day 200, which causes the algorithm to err on every single day during that interval. We can continue this geometric approach by allowing half of the experts to be wrong on an interval with double the length, e.g., 13 of the remaining 25 experts are wrong every single day from day 200 to day 400, so that the algorithm will always be incorrect after the first 50 days, which clearly contradicts the desired claim.

The key to the above counterexample is that experts that are incorrect on later days can cause a larger number of incorrect outputs by the algorithm because these experts were correct on previous intervals. At a first glance, it seems we can avoid this issue by instead resetting a timer for the remaining experts each time the size of the pool roughly halves. Namely, suppose we define a timer u to first demarcate the beginning of the round. Any expert that is inaccurate for at least a $\frac{\delta}{4}$ fraction of the days since time u is deleted. Each time the size of P decreases by roughly half, the variable u is updated to the new time. As before, the current round ends when the pool is completely depleted of experts, at which point the next round begins and a new pool is chosen. However, this *still* does not work because now the timer can be set adversarially to always cause the best expert to be deleted.

Surprisingly, the issue is alleviated if we instead require an even more stringent demand from the experts in the pool. Instead of asking for experts to make errors on at most $\frac{\delta}{4}$ fraction of the days in the pool, we instead ask experts to make errors on at most $O\left(\frac{\delta}{\log n}\right)$ fraction of the days in the pool. Since the timers are no longer reset, it is once again possible for the best expert to not be deleted. Moreover, the extra $O(\log n)$ factor allows us to overcome to build-up of errors in the previous counterexample, because the errors can only accumulate over $O(\log n)$ rounds.

The intuition for the algorithm is that one of two cases should hold. Either there is a small number of rounds, which indicates that the experts in some pool performed well over a large period of time, or there is a large number of rounds, in which case it is likely that the best expert is added to the pool in some round. We would like to show that in the latter case, the best expert being added to the pool compels the algorithm to perform well overall. The idea is that

if we add the best expert to the pool on a *random* day, it is unlikely that the best expert will ever be deleted from the pool and thus the algorithm will have good accuracy.

Decoupling for Arbitrary-Order Streams. Whereas the random-order model analysis crucially used the fact that the best expert would not be deleted if it was sampled to the pool, this property no longer holds for arbitrary-order streams. For instance, there could be $\Omega(M)$ consecutive days in which the best expert makes a mistake, so that our algorithm will likely delete the best expert during that time. Thus, we define "bad" times as days on which the best expert would be deleted from the pool if it were sampled on that day. Then we can upper bound the total number of rounds by the sum of the number of rounds starting on bad times and the number of rounds starting on good times. The number of rounds beginning on bad times is upper bounded simply by the number of bad times, which in turn cannot be too large by an averaging argument. We then look to upper bounding the number of rounds starting on good days. Note that if the best expert is sampled on a good day, then it will never be deleted, which terminates the sequence of resamplings. Thus, the number of rounds initiated on good days follows a geometric distribution. Hence, we can show that the number of rounds initiated on good days and thus the *total* number of rounds is "low" with good probability. It follows that with good probability, the total number of mistakes by the algorithm must therefore also be low since the algorithm only resamples if its accuracy is poor. Thus, although these techniques may not be as black box as those for random-order streams, our algorithm can achieve high-probability bounds for arbitrary-order streams, while we do not know if this is possible with random order streams.

2 LOWER BOUND FOR ALL STREAMING MODELS

We will provide our lower bound in terms of δ, the regret the algorithm incurs. We note that our lower bound is valid in the i.i.d. setting for discrete prediction, and consequently is a lower bound in all (adversarial, random order, i.i.d., and continuous costs) settings we consider in the paper. Moreover, the lower bound holds even if the algorithm still has access to all $\Omega(n)$ predictions of the experts when the outcome of the day is revealed (and loses access to the predictions only when it receives the next day's predictions).

Our lower bound is achieved by reducing the problem of discrete prediction with expert advice to an n-fold version of the distributed detection problem we call DiffDist.

Definition 2 (The ε-DiffDist Problem). *We have T players, each of whom holds n bits, indexed from 1 to n. We must distinguish between two cases, which we refer to as "$V = 0$" and "$V = 1$". Let μ_0 be a Bernoulli distribution with parameter $\frac{1}{2}$, i.e., a fair coin, and let μ_1 be a Bernoulli distribution with parameter $\frac{1}{2} + \varepsilon$.*

- *(NO Case, "$V = 0$") Every index for every player is drawn i.i.d. from a fair coin, i.e., μ_0.*
- *(YES Case, "$V = 1$") An index $L \in [n]$ is selected arbitrarily – the L-th bit of each player is chosen i.i.d. from μ_1. All other bits for every player are chosen i.i.d. from μ_0.*

Intuitively, each player in the ε-DiffDist problem corresponds to a different day in the learning with experts problem. The n bits held by each player correspond to the n expert predictions for each day. Thus, in the NO case for the ε-DiffDist problem, each expert is correct on half of the days in expectation (and with high probability) while in the YES case, there exists a single expert that is correct on a $\frac{1}{2} + \varepsilon$ fraction of the days in expectation (and with high probability). We note that we consider an algorithm that solves the ϵ-DiffDistproblem with probability $1 - p$, for some fixed constant $p \in [0, 1]$, where this probability is over both the randomness of the input distribution, as well as the private randomness of the algorithm.

2.1 Communication Lower Bound of the ϵ-DiffDist Problem

We prove our lower bounds using the *blackboard model*, where each element of the stream is treated as a party (for us, each party will correspond to a day of predictions and corresponding outcome), and an algorithm for computing on the stream is seen as a multiparty communication protocol. Each of T parties has private randomness and communicates by posting a message to the blackboard, and we denote the transcript of all communication in a protocol by $\Pi \in \{0, 1\}^*$. The communication cost of a protocol is the maximum bit length of the transcript, where the maximum is taken over all inputs and all coin tosses of the protocol. The communication complexity is the minimum communication cost of a correct protocol, where we will consider *distributional* correctness, meaning that a protocol is correct with failure probability γ if it fails with probability at most γ, where the probability is taken over the joint distribution of the inputs and the protocol's private coins. We will take γ to be a constant throughout, and will specify the input distributions we consider. We will also allow the protocol to have its own private coins, as we will need this when proving a direct sum theorem for information cost, described below. In the streaming model, the space complexity of an algorithm is the maximum amount of space in bits used by the algorithm. Note that any lower bound S on the randomized communication complexity in the blackboard model implies an S/T lower bound on the space complexity of a 1-pass randomized streaming algorithm for solving the communication problem, since one player must communicate at least S/T bits.

To prove a communication lower bound on the ϵ-DiffDist problem, we prove an analogue of the direct sum theorem [5] that applies to the ϵ-distributed detection problem. The classic direct sum theorem from [5] cannot be directly applied, since it is only applicable to decision problems, where the correct answer can be solely determined from the inputs. In our case the goal is to correctly infer a latent bit — the correct answer is not a deterministic function of the input bits, but rather we are in a hypothesis testing scenario where we must infer the latent bit correctly with good probability under its respective posterior distribution.

Hence, we will use a technique that is an analogue of the direct sum theorem in [5], but instead we directly show a lower bound on the mutual information in the case $V = 0$. The mutual information $I(X; Y)$ between two variables X and Y is equal to $H(X) - H(X|Y)$, or equivalently, $H(Y) - H(Y|X)$, where for a random variable Z, $H(Z)$ is the Shannon entropy of the distribution of Z. We refer the reader to [5] for more background on information theory and the information complexity that we use.

Lemma 1 (Decomposable lemma). *Consider the distribution* $\mathbf{X} \sim \mu_0^n$ *under the NO case of the* ϵ*-DiffDist problem. For any protocol* Π *that solves the* ϵ*-DiffDist problem with constant probability, the following inequality holds:* $I(\mathbf{X};\Pi) \geq \sum_{i=1}^{n} I(\mathbf{X}_i;\Pi \mid V=0)$ *under* μ_0^n.

To lower bound individual summands, we use the following

Lemma 2 (Reduction lemma). *For a protocol* Π *that solves the* ϵ*-DiffDist problem with probability at least* $1-p$*, where* $p \in [0, 0.5)$ *is a fixed constant, the following inequality holds in the NO case of the* ϵ*-DiffDist problem: for every* $i \in [n]$, $I(\mathbf{X}_i;\Pi \mid V=0) \geq \Omega(\epsilon^{-2})$.

Note that the ϵ-DiffDist problem is not solvable with $o(\epsilon^{-2})$ players (samples). We obtain the following randomized communication complexity lower bound for ε-DiffDist, where recall correctness is distributional, as described at the beginning of this section.

Lemma 3. *The communication complexity of the* ε*-DiffDist problem with a constant* $1-p$ *probability, for any fixed constant* $p \in [0, 0.5)$*, is* $\Omega\left(\frac{n}{\varepsilon^2}\right)$.

In our proof of this fact, we appeal to a lower bound on the so-called Strong Data Processing Inequality (SDPI) constant for the ϵ-distributed detection problem, which was shown in [63]. This then allows us to use a theorem of [9] to lower bound the mutual information even conditioned on $V=0$, which is referred to as the min-information cost in that paper. This additional conditioning on $V=0$ in the mutual information lower bound now allows us to prove a direct sum theorem on the mutual information for the OR of n copies of the ϵ-distirbuted detection problem by following the framework of [5]. Finally, we use that randomized communication complexity is lower bounded by this mutual information.

2.2 Reduction from DiffDist to the Experts Problem

We can now show a lower bound for the discrete prediction experts problem by reducing to it from the ε-DiffDist problem. Define an oracle algorithm, $\mathcal{A}$, that achieves δ regret on the expert prediction problem with constant probability more than $\frac{1}{2}$. Our goal is to show that we can solve the ε-DiffDist problem with constant probability more than $\frac{1}{2}$ by using $\mathcal{A}$. At a high level, we will treat each player i's bit string as the predictions that a set of n "experts" made on day i. To provide intuition for our reduction, we first describe a simpler reduction from ε-DiffDist to the experts problem. The instance of the experts problem we construct has 1 as the correct answer on every day. We let each bit index correspond to an expert, and consequently, the predictions of the experts on a day i are the bits of player i.

In the YES case of ε-DiffDist, there is an index, i.e. an expert, which is correct on approximately $\frac{1}{2}+O(\delta)$ of the days. Thus, $\mathcal{A}$ should also be correct $\frac{1}{2}+O(\delta)$ of the time with high probability. On the other hand, in the NO case, the experts are predicting randomly. Hence, the best expert does no better than a fair coin for its prediction. Our goal is to have $\mathcal{A}$ have accuracy at least $\frac{1}{2}+\Theta(\delta)$ in the YES case and an $\Theta(\delta)$ less fraction of days in the NO case.

However, while $\mathcal{A}$ ensures an upper bound on δ, it makes no guarantees about the maximum accuracy $\mathcal{A}$ can achieve. For example, an algorithm that simply predicts 1 on each day will achieve 100% accuracy in both cases. Thus, this reduction cannot "force" $\mathcal{A}$ to be sufficiently inaccurate in the NO case.

Masking in the Reduction. To remedy this issue, we introduce a notion of "masking", i.e., obfuscating the the correct answer of each day in our construction so we can ensure an upper bound on the accuracy of $\mathcal{A}$ when in the NO case of the ε-DiffDist problem. In our actual reduction, formulated in Algorithm 3, we compute a "mask" for each day by sampling a random bit from an independent fair coin. The mask XOR'ed with 1 will be the correct answer to the experts problem on that day. In addition, we also XOR the mask with the player's bits corresponding to that day to produce the experts predictions. This masking procedure ensures that all expert predictions and true outcomes are mutually independent in the NO case. That is, since the mask is drawn i.i.d. from a fair coin on each day, and the expert predictions are also drawn i.i.d. from a fair coin, the masked expert predictions remain distributed according to i.i.d. fair coins. So, the true outcome on each day is distributed according to a fair coin that is completely independent of the expert predictions and past information provided to $\mathcal{A}$. Thus, $\mathcal{A}$ can do nothing to increase (or decrease) its probability of success on each day from $\frac{1}{2}$. On the other hand, in the YES case, there still remains an expert that is correct on a $\frac{1}{2}+\Theta(\delta)$ fraction of days, so $\mathcal{A}$ will still get a $\frac{1}{2}+\Omega(\delta)$ fraction of days correct.

Algorithm 3 The following algorithm is a reduction from ε-DiffDist to the experts problem where $\mathcal{A}$ is an oracle algorithm that solves the experts problem with δ regret and probability $\frac{1}{2}$. Let $c=\sqrt{2\ln(24)}$ and set $\varepsilon=\delta(c+1)$, which we assume is less than $1/2$. Let $\oplus$ be the XOR operation.

Input: $\{\mathbf{X}^{(1)},\ldots,\mathbf{X}^{(T)}\}$, where $\mathbf{X}^{(t)} \in \{0,1\}^n$ for each $t \in [T]$.
Let $\texttt{state}_0$ be the initial state of $\mathcal{A}$.
for each $t \in [T]$ **do**
  Player t does the following:
    Sample $\texttt{mask}_t$ from an independent fair coin.
    $\texttt{maskedX}_t \leftarrow \mathbf{X}^{(t)} \oplus (\texttt{mask}_t)^n$.
    Compute prediction and next state: $\texttt{prediction}_t, \texttt{state}_t \leftarrow \mathcal{A}(\texttt{state}_{t-1}, \texttt{maskedX}_t)$
    $\mathcal{A}$ is correct on day t iff $\texttt{prediction}_t \oplus \texttt{mask}_t = 1$
    if $t < T$ **then** write $\texttt{state}_t$ to the blackboard
end for
Let S be the fraction sample days that are correct (each player communicates a bit indicating whether the algorithm was correct on each day).
if $S < \frac{1+\delta c}{2}$ **then return** 0 **else return** 1

Theorem 1. *Let* δ, $p < \frac{1}{2}$ *be fixed constants, i.e., independent of other input parameters. Any algorithm that achieves* δ *regret for the experts problem with probability at least* $1-p$ *must use at least* $\Omega\left(\frac{n}{\delta^2 T}\right)$ *space. This lower bound holds even when the costs are binary and expert predictions, as well as the correct answer, are constrained to be i.i.d. across the days, albeit with different distributions across the experts.*

3 PREDICTION WITH EXPERTS IN THE STANDARD STREAMING MODEL

As a warm-up to our near-optimal algorithm for online learning with experts in the random-order model, we show an algorithm that can handle arbitrary-order streams. To build intuition, we first consider the simpler discrete prediction with experts problem, and then generalize our algorithm for general costs in $[0, 1]$. Recall that in the discrete prediction problem, the cost of each decision is either 0 or 1. We first propose a space constrained version of the simplest version of the majority elimination algorithm where experts that are incorrect for a "significant" fraction of days are eliminated. The algorithm uses the desired $\tilde{O}\left(\frac{n}{T}\right)$ space complexity for correctness on a constant fraction of days, even when the best expert makes as many as $O(T/\log^2 n)$ mistakes. When errors on only a δ fraction of days are permitted for some subconstant δ, the algorithm uses $\tilde{O}\left(\frac{n}{T\delta}\right)$ space but demands that the number of mistakes made by the best expert be at most $O\left(\frac{\delta T}{\log^2 n}\right)$.

3.1 Discrete Prediction in the Standard Streaming Model

Our algorithm for this upper bound is to run the majority elimination algorithm in small chunks at a time. The typical majority elimination algorithm (not in the space constrained setting) maintains a voting pool of experts, that starts by including all of the experts. On each day, the algorithm predicts the majority vote of the experts in the pool. When the outcome is revealed, the algorithm removes any expert who made an incorrect prediction from the pool. If the pool is empty, the algorithm resets by adding all of the experts back into the pool. In this formulation, the algorithm makes at most $\log n$ times as many mistakes as the best expert. We modify this algorithm to use less space, and impose a laxer requirement on the experts in our pool to achieve a better bound.

The algorithm proceeds in rounds. At the beginning of each round, the algorithm initializes a pool, P, of $k = \frac{16n\log^2 n}{T\delta}$ different experts and the variable u to mark the time that the round begins. On each day, the algorithm temporarily stores (for just the current day) the predictions of the experts in the pool, and outputs their majority vote. When, the correct answer for the day is revealed, any expert that is inaccurate for at least a $\frac{\delta}{8\log n}$ fraction of the days since time u is deleted. Once the pool P is completely depleted of experts, the current round ends and the next round begins. A complete description is given in Algorithm 4.

We first bound the number of mistakes made by the algorithm in a particular round.

Lemma 4. *Fix a $\delta > \frac{16\log^2 n}{T}$, and suppose a pool, P, of size $k = \frac{16n\log^2 n}{T\delta}$ is initiated by Algorithm 4 at time t_0 and $P \neq \emptyset$ before some later time t. Then the number of mistakes by the algorithm between times t_0 and t is at most $\frac{(t-t_0)\delta}{2} + 4\log n$.*

Proof. Let $u_1, \ldots, u_y$ be a sequence of times defined so that u_i is the first time at which at most $\frac{k}{2^i}$ experts remain in the pool. Note that $y \leq \lceil \log k \rceil$ by definition. Let n_i be the total number of mistakes the algorithm has made by time u_i. We note that $n_{i+1} - n_i$ mistakes are made by the algorithm between times u_i and u_{i+1}, and each mistake requires at least $\frac{k}{2^{i+2}}$ mistakes across all experts, since there at least $\frac{k}{2^{i+1}}$ experts before time u_{i+1}, and at least half of them must be wrong for a mistake to be made. Consequently, the total number of mistakes made by the experts between times u_i and u_{i+1} is at least $(n_{i+1} - n_i) \cdot \frac{k}{2^{i+2}}$. We can also see that at most $\left\lceil \frac{(u_{i+1}-t_0)\delta}{8\log n} \right\rceil$ mistakes can be made by each of the $\frac{k}{2^i}$ experts that are not deleted by time u_i, so the total number of mistakes made by the experts between times u_i and u_{i+1} is at most

$$\left\lceil \frac{(u_{i+1}-t_0)\delta}{8\log n} \right\rceil \cdot \frac{k}{2^i} \leq \frac{(u_{i+1}-t_0)k\delta}{8\cdot 2^i \log n} + \frac{k}{2^i}.$$

Hence we have $(n_{i+1} - n_i) \cdot \frac{k}{2^{i+2}} \leq \frac{(u_{i+1}-t_0)k\delta}{8\cdot 2^i \log n} + \frac{k}{2^i}$ so that $(n_{i+1} - n_i) \leq \frac{(u_{i+1}-t_0)\delta}{2\cdot \log n} + 4$. Therefore,

$$\sum_{i=1}^{y-1} (n_{i+1} - n_i) \leq \sum_{i=1}^{y-1} \left(\frac{(u_{i+1}-t_0)\delta}{2\cdot \log n} + 4 \right) \leq \frac{(t-t_0)\delta}{2} + 4\log k.$$

□

Algorithm 4 An expert algorithm that maintains a pool of experts occupying $\widetilde{O}(n/T\delta)$ space, and eliminates an expert if its accuracy drops below $1 - \frac{\delta}{8\log n}$. This algorithm is a streaming analogue of the typical majority elimination algorithm for experts.

Input: Number n of experts, number T of rounds, fraction $1 - \delta$ of mistakes

```
k ← 16n log² n / (Tδ), u ← 1
Let P be a random set of k unique indices of [n].
for each time t ∈ [T] do
    Store (for only the current day t) the predictions of the experts in P.
    Output the majority vote of the experts in P.
    Discard any experts in P with lower than 1 − δ/(8 log n) accuracy since time u.
    if P = ∅ then
        Let P be a random set of k unique indices of [n].
        u ← t
    end if
end for
```

We now give the full guarantees for our algorithm.

Theorem 6. *Fix a $\delta > \frac{16\log^2 n}{T}$, and suppose the best expert makes at most $M \leq \frac{\delta^2 T}{128\log^2 n}$ mistakes. Then Algorithm 4 for the discrete prediction with experts problem uses $\widetilde{O}\left(\frac{n}{\delta T}\right)$ space and achieves regret at most δ, with probability at least $1 - \frac{1}{n}$.*

Proof. Algorithm 4 only makes more than δT total mistakes if it completes at least $\frac{\delta T}{8\log n}$ rounds. This is because, by Lemma 4, the bound on mistakes after $\frac{\delta T}{8\log n}$ rounds is $\frac{T\delta}{2} + \frac{T\delta}{8\log n} \cdot 4\log n \leq T\delta$.

Thus our goal is to show that the probability that Algorithm 4 completes at least $\frac{\delta T}{8\log n}$ rounds is low. We would like to say that conditioned on a large number of rounds, that the algorithm must

have sampled the best expert many times with high probability. Unfortunately, the conditional event that the algorithm completes a large number of rounds can significantly alter the distribution of events, resulting in a more involved analysis.

We thus define a sequential procedure to analyze the distribution for the total number of rounds. Let $d_0, d_1, d_2, \ldots$ be the random variables representing the times on which the pool of k experts is sampled, so that d_i is the random variable for the time on which the pool of experts is empty for the i-th time and thus resampled. We construct a sequence $t_0, t_1, t_2, \ldots$ of times so that the distribution of $t_0, t_1, t_2, \ldots$ will match the distribution of $d_0, d_1, d_2, \ldots$. Let $t_0 = 0$ and for each $i > 0$, let t_i be drawn uniformly from the distribution of possible times at which a new pool of k experts drawn at time t_{i-1} is completely removed from the pool. The sequence $\{t_i\}_i$ is terminated if the experts drawn at time t_{i-1} are not all removed from the pool by time T. Note that the process in which the sequence $t_0, t_1, \ldots$ is generated matches the process in which the sequence $d_0, d_1, \ldots$ is determined, so that their distributions are identical so we can instead work with the random sequence $t_0, t_1, \ldots$.

Let Bad be the set of times on which the best expert would be eliminated by the algorithm if it were added to the pool of experts on a time $t \in$ Bad. Because the best expert makes at most M mistakes and the algorithm deletes experts that have made mistakes on at least $\frac{\delta}{8 \log n}$ fraction of the times since they have been in the pool, then it follows that $|\text{Bad}| \leq \frac{8M \log n}{\delta}$.

Let R be the random variable that corresponds to the total number of rounds in the algorithm. Let B be the random variable that corresponds to the total number of rounds in the algorithm that started on days t such that $t \in$ Bad. Let G be the random variable that corresponds to the total number of rounds in the algorithm that started on days t such that $t \notin$ Bad. Observe that $B \leq |\text{Bad}| \leq \frac{8M \log n}{\delta}$ and $R \leq B + G \leq \frac{8M \log n}{\delta} + G$. Since $M \leq \frac{\delta^2 T}{128 \log^2 n}$, then $R \leq \frac{\delta T}{16 \log n} + G$. Thus it remains to analyze the distribution of G.

Observe that if the best expert is sampled on a time t with $t \notin$ Bad, then the best expert will not be deleted and thus there will be no subsequent round. Hence, if $G = j$ for some $j \geq 1$, then the best expert must not have been sampled into the first $j - 1$ pools that were sampled on times $t_1, \ldots, t_{j-1} \notin$ Bad. Thus, $\mathbf{Pr}[G \geq j] \leq (1 - k/n)^{j-1}$ for each integer $j \geq 1$. Since $k = \frac{16n \log^2 n}{T\delta}$, we have

$$\begin{aligned}
\mathbf{Pr}\left[G \geq \frac{\delta T}{16 \log n}\right] &\leq \left(1 - \frac{16n \log^2 n}{T\delta} \cdot \frac{1}{n}\right)^{\frac{\delta T}{16 \log n}} \\
&\leq \left(1 - \frac{16 \log^2 n}{T\delta}\right)^{\frac{T\delta}{16 \log^2 n} \cdot \log n} \\
&\leq e^{-\log n} \leq \frac{1}{n}
\end{aligned}$$

Therefore, $\mathbf{Pr}\left[R \geq \frac{\delta T}{16 \log n} + \frac{\delta T}{16 \log n}\right] \leq \frac{1}{n}$ and $\mathbf{Pr}\left[R < \frac{\delta T}{8 \log n}\right] \geq 1 - \frac{1}{n}$. Conditioned on the event that $R < \frac{\delta T}{8 \log n}$, then by Lemma 4, the total number of mistakes by the algorithm is at most

$$\frac{T\delta}{2} + 4R \log n \leq \frac{T\delta}{2} + \frac{T\delta}{8 \log n} \cdot 4 \log n \leq T\delta.$$

The regret of the algorithm is defined as the difference between the error rate of the algorithm and the optimal error rate (the error rate of the best expert). This is upper bounded by the error rate of the algorithm, which is at most $T\delta/T = \delta$. □

3.2 Online Prediction for General $[0, 1]$ Costs in the Standard Streaming Model

Now, we show that with a simple change, we can modify this algorithm to work for general $[0, 1]$ costs. The trouble with this algorithm for $[0, 1]$ costs is that it is no longer clear what it means to take the "majority prediction" on each day. However, there are other sequential prediction algorithms that we could use, that work well in the case of general costs. Thus, for $[0, 1]$ costs, we can implement Algorithm 5.

Algorithm 5 An expert algorithm that works for $[0, 1]$ costs in the standard streaming model.

Input: Number n of experts, number T of rounds, fraction $1 - \delta$ of mistakes, (black-box sequential prediction algorithm of choice)

$k \leftarrow \frac{16\beta n \ln n \ln T}{T\delta}$, $u \leftarrow 1$
Let P be a random set of k unique indices of $[n]$.
Initialize a sequential prediction algorithm (Definition 1) for the experts in P with $\varepsilon = \frac{1}{2}$
for each time $t \in [T]$ **do**
  Output prediction according to sequential prediction algorithm running on P.
  if every expert in P has an error rate higher than $\frac{\delta}{8}$ since time u **then**
    Let P be a random set of k unique indices of $[n]$.
    $u \leftarrow t$
    Re-initialize sequential prediction algorithm for experts in P with $\varepsilon = \frac{1}{2}$
  end if
end for

Our analysis follows the same structure as the one for discrete costs. We can use the guarantee of the sequential prediction algorithm (Definition 1) in place of Lemma 4. The bound on the number of "bad days" that is used in Theorem 6 is now somewhat more involved and is presented as its own lemma.

Lemma 5. *Let Bad be the set of times on which the best expert would be eliminated by the algorithm if it were added to a pool of experts on a time $t \in$ Bad, and suppose the best expert incurs total cost at most M. Then $|\text{Bad}| \leq \frac{8M}{\delta}$.*

Proof. We show this via amortized analysis. For convenience, denote the threshold $L = \frac{\delta}{8}$ and let m_i be the cost of the best expert on day i for each $i \in [T]$. Let $\{t_1, \ldots, t_b\}$ be the days in Bad, so that we would like to show that $b = |\text{Bad}| \leq \frac{8M}{\delta}$. For each $t_j \in$ Bad with $j \in [b]$, i.e., the j-th bad day, let e_j be the day on which the best expert would be deleted if it were sampled on day t_j, so that $\sum_{i=t_j}^{e_j} m_i \geq (e_j - t_j + 1)L$.

We define a subsequence $t_{a_1}, \ldots, t_{a_{b'}}$ of $t_1, \ldots, t_b$ as follows. Let $a_1 = 1$ and for each $k > 1$, let $a_k = \min_{i \in [b]}\{i : t_i > e_{a_{k-1}}\}$. In

other words, t_{a_k} is the first bad day after the deletion of the $(k-1)$-th term in the subsequence. Thus the sequence $t_{a_1}, \ldots, t_{a_{b'}}$ is a subsequence of $t_1, \ldots, t_b$ and the intervals $[t_{a_1}, e_{a_1}], \ldots, [t_{a_{b'}}, e_{a_{b'}}]$ are disjoint, so we have that $\sum_{i=1}^{T} m_i \geq \sum_{j=1}^{b'} \sum_{i=t_{a_j}}^{e_{a_j}} m_i$. We also know that each of the b bad days is in one of these intervals, so $\sum_{j=1}^{b'} e_{a_j} - t_{a_j} + 1 \geq b$. Therefore, we have

$$M = \sum_{i=1}^{T} m_i \geq \sum_{j=1}^{b'} \sum_{i=t_{a_j}}^{e_{a_j}} m_i \geq \sum_{j=1}^{b'} (e_{a_j} - t_{a_j} + 1)L \geq bL.$$

It follows that $b \leq \frac{M}{L}$, where $b = |\textsc{Bad}|$ is the number of bad days and $L = \frac{\delta}{8}$. Thus, $|\textsc{Bad}| \leq \frac{8M}{\delta}$. □

This allows us to give an algorithm for general costs.

Theorem 7. *Fix any $\delta > \frac{16\beta \ln^2 n}{T}$, and suppose the best expert makes at most $M \leq \frac{\delta^2 T}{128\beta \ln n}$ mistakes, where β is a fixed constant that depends on the black-box sequential prediction algorithm that is used. Then Algorithm 5 for the online learning with experts problem uses $\widetilde{O}\left(\frac{n}{\delta T}\right)$ space and achieves regret at most δ in expectation,*

4 GENERAL COSTS IN THE RANDOM-ORDER STREAMING MODEL

In this section, we consider the online learning with experts problem, in which the cost of the decision of an expert on each day can range from $[0, \rho]$, where $\rho > 0$ is the width of the problem. Without loss of generality, we assume $\rho = 1$ throughout this section and instead incur a multiplicative factor in the regret in the guarantees of our algorithms, i.e., our algorithms will have regret $\rho\delta$ rather than δ. Whereas the previous algorithm provided guarantees on arbitrary streams, our main algorithm in this section will focus on the random-order streaming model.

The main result in the previous section, for arbitrary-order streams, relied on an assumption that the best expert incurred sub-constant regret. This allowed us to conclude that there were not too many "bad" days in the stream, where a "bad" day is one where if we start a round with the best expert on that day, the best expert will appear to do badly, causing the round to end.

For random-order streams, this is no longer a problem, because the best expert will effectively do uniformly well across the entire stream. This means that any day on which we sample the best expert will likely be a "good" day. This allows us to remove the condition on the best expert.

We will first show an algorithm that achieves δ regret, if it knows M, the number of mistakes made by the best expert. Then we show how to modify the algorithm to include a searching phase which allows us to estimate M and so the algorithm does not need to know it in advance. The algorithm for arbitrary-order streams did not need to know M in advance because we assumed an upper bound on M. However, for this algorithm, we remove this upper bound assumption on M, though we will need to look at a prefix of days to estimate M for use in our algorithm.

We first note that the best expert in the random-order model cannot incur high cost. The following is Hoeffding's bound.

Algorithm 6 An expert algorithm that maintains a pool of experts occupying $\widetilde{O}(n/(\delta^2 T))$ space, and resamples the pool if its expected cost is too high.

Input: Number n of experts, number T of rounds, regret δ, number M of mistakes of the best expert. We later show how to instead estimate M.

$u \leftarrow 1, k \leftarrow O\left(\frac{n \log^2 n}{\delta^2 T}\right)$
Let P be a random set of k unique indices of $[n]$.
for each time $t \in [T]$ **do**
  Run a sequential prediction algorithm (Definition 1) with $\varepsilon = \delta/2$ for the experts in P.
  if the cost of every expert in the pool exceeds $\frac{M}{T}(t-u) + 4\sqrt{(t-u)\log T}$ since time u **then**
    Let P be a random set of k unique indices of $[n]$.
    $u \leftarrow t$
  end if
end for

Lemma 6. *Let $X_1, \ldots, X_t$ be independent random variables such that $X_i \in [0,1]$ with $\mathbb{E}[X_i] = \alpha$ for all $i \in [t]$ and let $X = \sum_{i=1}^{t} X_i$. Then for any $T > 1$, $\mathbf{Pr}\left[|X - \alpha t| \geq 4\sqrt{t \log T}\right] \leq \frac{1}{T^2}$.*

We can apply Lemma 6 in conjunction with the distributional properties of random-order streams and a union bound to show that with high probability, a pool with the best expert will be retained.

Corollary 1. *In the random-order model, with prob. at least $1 - \frac{1}{T}$, Algorithm 6 will not resample a pool including the best expert.*

We now analyze Algorithm 6, which assumes that the cost M of the best expert is given as input to the algorithm. We remove this assumption afterwards.

Theorem 8. *For any $\delta > \sqrt{\frac{16 \log^2 n}{T}}$, there exists an algorithm that takes as input a number M, which is the cost of the best expert, and achieves regret at most δ in expectation on random-order streams. The algorithm uses $O\left(\frac{n \log^2 n}{\delta^2 T}\right)$ space.*

Proof. Consider Algorithm 6 and suppose by way of contradiction, that its expected cost is at least $M + \delta T$. Let $\alpha = \frac{M}{T}$. Suppose the j-th pool of experts was run for time t_j. Then the best expert in the pool has cost at most $\alpha t_j + 4\sqrt{t_j \log n} + 1$. Then by Definition 1 for $\varepsilon = \frac{\delta}{2}$, the expected cost of running the sequential prediction algorithm on the j-th pool of experts is at most

$$\left(1 + \frac{\delta}{2}\right)\left(\alpha t_j + 4\sqrt{t_j \log n} + 1\right) + \frac{2\beta \ln n}{\delta},$$

for some fixed constant β. Thus, if there are r total rounds over time t, the expected cost of the algorithm by linearity of expectation is at most

$$\left(1 + \frac{\delta}{2}\right)\left(\alpha t + 4\sqrt{rt \log n} + r\right) + \frac{2\beta r \ln n}{\delta}$$
$$= \alpha t + O\left(\delta \alpha t + \sqrt{rt \log n} + \frac{r \log n}{\delta}\right).$$

Hence, if the expected cost is at least $M + \frac{3\delta T}{4}$ and $\delta > \sqrt{\frac{16\log^2 n}{T}}$, then the algorithm must have used $r = \Omega\left(\frac{\delta^2 T}{\log n}\right)$ rounds.

We now analyze the probability distribution for the number of rounds that the algorithm uses. By Corollary 1, any pool that includes the best expert will not be resampled in the random-order model, with probability at least $1 - \frac{1}{T}$. Thus, conditioning on the event that the first pool sampled that includes the best expert is not resampled, then if the algorithm uses j total rounds, the first $j-1$ rounds must have not sampled the best expert. Therefore, if Z is a random variable that represents the number of rounds, we have $\mathbf{Pr}[Z \geq j] \leq \left(1 - \frac{k}{n}\right)^{j-1}$. Since $k = O\left(\frac{n\log^2 n}{\delta^2 T}\right)$ with a sufficiently large constant in the big-Oh, then $\mathbf{Pr}[Z \geq r] \leq \frac{1}{\mathrm{poly}(T)}$ for $r = \Omega\left(\frac{\delta^2 T}{\log n}\right)$. Hence, we have that with probability at least $1 - \frac{2}{T}$, the expected cost of the algorithm is at most $M + \frac{3\delta T}{4}$. Otherwise, the cost of the algorithm is at most T. Thus, the overall expected cost of the algorithm is at most $M + \delta T$. □

Unknown cost of the best expert in the random-order model. We remark that Algorithm 6 assumes the cost M incurred by the best expert is known. We now describe how this assumption can be easily removed in the random-order model. Note that since the overall expected cost of Algorithm 6 is at most $M + \delta T$, then even if we use a $(1 + O(\delta))$-approximation of M as input to the algorithm, then the overall expected cost is $(1 + O(\delta))M + \delta T = M + O(\delta T)$, which can be then adjusted to $M + \delta T$ by a rescaling of δ. Thus, it suffices to find a $(1 + O(\delta))$-approximation to M.

Let γ be an estimate for the average cost $\frac{M}{T}$, and we initialize γ to $\frac{1}{2}$. Note that a $(1 + O(\delta))$-approximation to γ corresponds to a $(1 + O(\delta))$-approximation to M. We obtain a $(1 + O(\delta))$-approximation to γ through a binary search. We proceed through $\ell := 2\log\frac{1}{\delta}$ epochs so that in each epoch $j \in [\ell]$, γ is a $\left(1 + \frac{1}{2^j}\right)$-approximation to $\frac{M}{T}$. Each epoch $j \in [\ell]$ has length $\frac{\delta T}{2\log\frac{1}{\delta}}$. We run Algorithm 6 on this epoch with input $\gamma \cdot \frac{\delta T}{2\log\frac{1}{\delta}}$ as the estimate for the cost and $\frac{1}{100}$ as the target regret. We can also track the average cost β_j of the best expert in each epoch j. If $\gamma > (1+\delta)\beta_j$, then we update $\gamma \leftarrow \gamma - \frac{1}{2^{j+1}}$. Similarly if $\gamma < (1-\delta)\beta_j$, then we update $\gamma \leftarrow \gamma + \frac{1}{2^{j+1}}$. After the ℓ epochs, we will fix γ as the estimated average cost for the remainder of the stream and run Algorithm 6.

Theorem 9. *For any $\delta > \sqrt{\frac{16\log^2 n}{T}}$, there exists an algorithm that achieves regret at most δ in expectation on random-order streams. The algorithm uses $O\left(\frac{n}{\delta^2 T}\log^2 n\right)$ space.*

Proof. It suffices to (1) show that γ converges to a $(1+\delta)$-approximation of the true average cost $\frac{M}{T}$ by the best expert and (2) analyze the regret induced by the procedure until γ converges. The expected regret of the algorithm afterward is upper bounded by Theorem 8.

To show that γ converges to a $(1+\delta)$-approximation of the true average cost $\frac{M}{T}$ by the best expert, we consider casework on γ. Suppose $\gamma > (1+\delta)\cdot\frac{M}{T}$. Then by Lemma 6, no experts sampled by the pool will achieve average cost γ in the random-order model. Thus γ will be decreased accordingly. On the other hand, if $\gamma < (1-\delta)\cdot\frac{M}{T}$, then again by Lemma 6, the best expert will be sampled by the pool and have average cost at least $(1-\delta)\cdot\frac{M}{T}$ in the random-order model and then γ will be increased accordingly. Hence, with probability at least $1 - \frac{1}{T}$, it holds that γ converges to a $(1+\delta)$-approximation of the true average cost $\frac{M}{T}$ of the best expert.

On the other hand, since each epoch $j \in [\ell]$ only has length $\frac{\delta T}{2\log\frac{1}{\delta}}$ and $\ell = 2\log\frac{1}{\delta}$, then the total cost that can be incurred across the ℓ epochs is only δT. Hence the regret can only be increased by an additive δ due to not knowing the average cost of the best expert. Finally, to analyze the space complexity, recall that each epoch has length $\frac{\delta T}{2\log\frac{1}{\delta}}$ and $\frac{1}{100}$ is the target regret. With high probability, the best expert makes at least $\frac{\delta M}{200\log\frac{1}{\delta}}$ mistakes in each epoch. Thus by Theorem 8, it suffices to use $O\left(\frac{n}{\delta^2 T}\log^2 n\right)$ space. □

ACKNOWLEDGEMENTS

We thank Santosh Vempala for pointing out the connection to follow the perturbed leader. D.P.W. and S.Z. were supported by a Simons Investigator Award and by the NSF Grant No. CCF-1815840. V.S. was partially supported by NSF Grant No. CCF-1652491. Z.X. was partially supported by a PricewaterhouseCoopers Research Grant.

REFERENCES

[1] Idan Amir, Idan Attias, Tomer Koren, Yishay Mansour, and Roi Livni. 2020. Prediction with Corrupted Expert Advice. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 14315–14325. https://proceedings.neurips.cc/paper/2020/file/a512294422de868f8474d22344636f16-Paper.pdf

[2] Yossi Arjevani and Ohad Shamir. 2015. Communication Complexity of Distributed Convex Learning and Optimization. In *Advances in Neural Information Processing Systems 28*. 1756–1764.

[3] Sanjeev Arora, Elad Hazan, and Satyen Kale. 2012. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing* 8, 1 (2012), 121–164.

[4] Sepehr Assadi and Chen Wang. 2020. Exploration with Limited Memory: Streaming Algorithms for Coin Tossing, Noisy Comparisons, and Multi-Armed Bandits. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2020)*. 1237–1250.

[5] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, and D Sivakumar. 2004. An information statistics approach to data stream and communication complexity. *J. Comput. System Sci.* 68, 4 (2004), 702–732.

[6] Arnab Bhattacharyya, Palash Dey, and David P Woodruff. 2018. An optimal algorithm for ℓ_1-heavy hitters in insertion streams and related problems. *ACM Transactions on Algorithms (TALG)* 15, 1 (2018), 1–27.

[7] Avrim Blum. 1998. On-Line Algorithms in Machine Learning. In *Online Algorithms*. Vol. 1442. Springer Berlin Heidelberg, 306–325.

[8] Mark Braverman, Faith Ellen, Rotem Oshman, Toniann Pitassi, and Vinod Vaikuntanathan. 2013. A tight bound for set disjointness in the message-passing model. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. 668–677.

[9] Mark Braverman, Ankit Garg, Tengyu Ma, Huy L Nguyen, and David P Woodruff. 2016. 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*. 1011–1020.

[10] Vladimir Braverman, Stephen R Chestnut, Nikita Ivkin, Jelani Nelson, Zhengyu Wang, and David P Woodruff. 2017. BPTree: an ℓ_2 heavy hitters algorithm using constant memory. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 361–376.

[11] Vladimir Braverman, Stephen R Chestnut, Nikita Ivkin, and David P Woodruff. 2016. Beating countsketch for heavy hitters in insertion streams. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 740–753.

[12] David Brayshaw, Paula Gonzalez, and Florian Ziel. 2020. A new approach to subseasonal multi-model forecasting: Online prediction with expert advice. In *EGU General Assembly Conference Abstracts*. 17663.

[13] George W. Brown. 1951. Iterative solution of games by fictitious play. In *Analysis of Production and Allocation*. Wiley, 374–376.

[14] Nicolò Cesa-Bianchi and Gábor Lugosi. 2006. *Prediction, learning, and games*. Cambridge university press.

[15] Nicolò Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. 2005. Improved Second-Order Bounds for Prediction with Expert Advice. In *Learning Theory*. 217–232.

[16] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. 2003. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *18th IEEE Annual Conference on Computational Complexity, 2003. Proceedings.* 107–117.

[17] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. 2004. Finding frequent items in data streams. *Theor. Comput. Sci.* 312, 1 (2004), 3–15.

[18] Chao-Kai Chiang, Tianbao Yang, Chia-Jung Lee, Mehrdad Mahdavi, Chi-Jen Lu, Rong Jin, and Shenghuo Zhu. 2012. Online Optimization with Gradual Variations. In *Proceedings of the 25th Annual Conference on Learning Theory*. 6.1–6.20.

[19] Graham Cormode and S. Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms* 55, 1 (2005), 58–75.

[20] T.M. Cover. 1996. Universal data compression and portfolio selection. In *Proceedings of 37th Conference on Foundations of Computer Science*. 534–538. https://doi.org/10.1109/SFCS.1996.548512

[21] Thomas M Cover. 2011. Universal portfolios. In *The Kelly Capital Growth Investment Criterion: Theory and Practice*. World Scientific, 181–209.

[22] Thomas M Cover and Erik Ordentlich. 1996. Universal portfolios with side information. *IEEE Transactions on Information Theory* 42, 2 (1996), 348–363.

[23] Michael Crouch, Andrew McGregor, Gregory Valiant, and David P. Woodruff. 2016. Stochastic Streams: Sample Complexity vs. Space Complexity. In *24th Annual European Symposium on Algorithms (ESA 2016)*. 32:1–32:15.

[24] Yuval Dagan and Ohad Shamir. 2018. Detecting Correlations with Little Memory and Communication. *CoRR* abs/1803.01420 (2018). arXiv:1803.01420 http://arxiv.org/abs/1803.01420

[25] Dylan J Foster, Alexander Rakhlin, and Karthik Sridharan. 2015. Adaptive Online Learning. In *Advances in Neural Information Processing Systems*, Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/19de10adbaa1b2ee13f77f679fa1483a-Paper.pdf

[26] Dean P. Foster and Rakesh Vohra. 1999. Regret in the On-Line Decision Problem. *Games and Economic Behavior* 29, 1 (1999), 7–35. https://doi.org/10.1006/game.1999.0740

[27] Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55, 1 (1997), 119–139.

[28] Yoav Freund and Robert E Schapire. 1999. Adaptive game playing using multiplicative weights. *Games and Economic Behavior* 29, 1-2 (1999), 79–103.

[29] Ankit Garg, Tengyu Ma, and Huy L. Nguyen. 2014. On Communication Cost of Distributed Statistical Estimation and Dimensionality. In *Advances in Neural Information Processing Systems 27*. 2726–2734.

[30] Naveen Garg and Jochen Könemann. 2007. Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems. *SIAM J. Comput.* 37 (01 2007), 630–652. https://doi.org/10.1109/SFCS.1998.743463

[31] Sumegha Garg, Pravesh K Kothari, and Ran Raz. 2020. Time-Space Tradeoffs for Distinguishing Distributions and Applications to Security of Goldreich's PRG. *arXiv preprint arXiv:2002.07235* (2020).

[32] Sumegha Garg, Ran Raz, and Avishay Tal. 2017. Extractor-based time-space tradeoffs for learning. *Manuscript. July* (2017).

[33] Sumegha Garg, Ran Raz, and Avishay Tal. 2019. Time-space lower bounds for two-pass learning. In *34th Computational Complexity Conference (CCC 2019)*.

[34] Nick Gravin, Yuval Peres, and Balasubramanian Sivan. 2017. Tight Lower Bounds for Multiplicative Weights Algorithmic Families. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*.

[35] Sudipto Guha and Andrew McGregor. 2009. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM J. Comput.* 38, 5 (2009), 2044–2059.

[36] David Haussler, Jyrki Kivinen, and Manfred K. Warmuth. 1995. Tight Worst-Case Loss Bounds for Predicting with Expert Advice. In *Computational Learning Theory (Lecture Notes in Computer Science)*. Springer, 69–83.

[37] Elad Hazan. 2016. Introduction to Online Convex Optimization. *Found. Trends Optim.* 2, 3-4 (2016), 157–325.

[38] Elad Hazan and Satyen Kale. 2010. Extracting certainty from uncertainty: regret bounded by variation in costs. *Mach. Learn.* 80, 2-3 (2010), 165–188.

[39] David P Helmbold and Robert E Schapire. 1997. Predicting nearly as well as the best pruning of a decision tree. *Machine Learning* 27, 1 (1997), 51–68.

[40] Adam Kalai and Santosh Vempala. 2005. Efficient algorithms for online decision problems. *J. Comput. System Sci.* 71, 3 (2005), 291–307. https://doi.org/10.1016/j.jcss.2004.10.016 Learning Theory 2003.

[41] Purushottam Kar, Bharath K. Sriperumbudur, Prateek Jain, and Harish C. Karnick. 2013. On the Generalization Ability of Online Learning Algorithms for Pairwise Loss Functions. *arXiv:1305.2505 [cs, stat]* (May 2013). arXiv:1305.2505 [cs, stat]

[42] Gillat Kol, Ran Raz, and Avishay Tal. 2017. Time-Space Hardness of Learning Sparse Parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*. 1067–1080.

[43] Wouter M Koolen, Tim van Erven, and Peter Grünwald. 2014. Learning the Learning Rate for Prediction with Expert Advice. In *Advances in Neural Information Processing Systems*, Vol. 27. https://proceedings.neurips.cc/paper/2014/file/3f67fd97162d20e6fe27748b5b372509-Paper.pdf

[44] Kasper Green Larsen, Jelani Nelson, Huy L. Nguyen, and Mikkel Thorup. 2016. Heavy Hitters via Cluster-Preserving Clustering. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*. IEEE Computer Society, 61–70.

[45] David Liau, Zhao Song, Eric Price, and Ger Yang. 2018. Stochastic Multi-armed Bandits in Constant Space. In *International Conference on Artificial Intelligence and Statistics*. 386–394.

[46] N. Littlestone and M. K. Warmuth. 1989. The Weighted Majority Algorithm. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (SFCS '89)*. 256–261. https://doi.org/10.1109/SFCS.1989.63487

[47] Wolfgang Maass and Manfred K Warmuth. 1998. Efficient learning with virtual threshold gates. *Information and Computation* 141, 1 (1998), 66–83.

[48] Erik Ordentlich and Thomas M. Cover. 1998. The Cost of Achieving the Best Portfolio in Hindsight. *Mathematics of Operations Research* 23, 4 (1998), 960–982.

[49] S.A. Plotkin, D.B. Shmoys, and E. Tardos. 1991. Fast approximation algorithms for fractional packing and covering problems. In *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*. 495–504. https://doi.org/10.1109/SFCS.1991.185411

[50] Maxim Raginsky. 2016. Strong Data Processing Inequalities and Φ-Sobolev Inequalities for Discrete Channels. *IEEE Trans. Inf. Theory* 62, 6 (2016), 3355–3389.

[51] Alexander Rakhlin and Karthik Sridharan. 2012. Online Learning With Predictable Sequences. *Journal of Machine Learning Research* 30 (08 2012).

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

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

[54] Amir Sani, Gergely Neu, and Alessandro Lazaric. 2014. Exploiting easy data in online optimization. In *Advances in Neural Information Processing Systems*, Vol. 27. https://proceedings.neurips.cc/paper/2014/file/01f78be6f7cad02658508fe4616098a9-Paper.pdf

[55] Ohad Shamir. 2016. Without-Replacement Sampling for Stochastic Gradient Methods: Convergence Results and Application to Distributed Optimization. *CoRR* abs/1603.00570 (2016).

[56] Kai Sheng Tai, Vatsal Sharan, Peter Bailis, and Gregory Valiant. 2018. Sketching Linear Classifiers over Data Streams. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD*. 757–772.

[57] Eiji Takimoto, Akira Maruoka, and Volodya Vovk. 2001. Predicting nearly as well as the best pruning of a decision tree through dynamic programming scheme. *Theoretical Computer Science* 261, 1 (2001), 179–209.

[58] Vladimir Vovk. 1990. Aggregating Strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory, COLT*. 371–386.

[59] Vladimir Vovk. 1998. A Game of Prediction with Expert Advice. *J. Comput. Syst. Sci.* 56, 2 (1998), 153–173.

[60] Vladimir Vovk. 1999. Derandomizing stochastic prediction strategies. *Machine Learning* 35, 3 (1999), 247–282.

[61] Vladimir Vovk. 2005. Defensive prediction with expert advice. In *International Conference on Algorithmic Learning Theory*. Springer, 444–458.

[62] Omri Weinstein and David P Woodruff. 2015. The simultaneous communication of disjointness with applications to data streams. In *International Colloquium on Automata, Languages, and Programming*. Springer, 1082–1093.

[63] Yuchen Zhang, John Duchi, Michael I Jordan, and Martin J Wainwright. 2013. Information-Theoretic Lower Bounds for Distributed Statistical Estimation with Communication Constraints. In *Advances in Neural Information Processing Systems*, Vol. 26.