Adversarial Online Learning with Temporal Feedback Graphs

Khashayar Gatmiry GATMIRY@MIT.EDU
MIT
Jon Schneider JSCHNEI@GOOGLE.COM
Google Research

Editors: Shipra Agrawal and Aaron Roth

Abstract

We study a variant of prediction with expert advice where the learner's action at round t is only allowed to depend on losses on a specific subset of the rounds (where the structure of which rounds' losses are visible at time t is provided by a directed "feedback graph" known to the learner). We present a novel learning algorithm for this setting based on a strategy of partitioning the losses across sub-cliques of this graph. We complement this with a lower bound that is tight in many practical settings, and which we conjecture to be within a constant factor of optimal. For the important class of transitive feedback graphs, we prove that this algorithm is efficiently implementable and obtains the optimal regret bound (up to a universal constant).

Keywords: Online learning, feedback graphs

1. Introduction

Prediction with expert advice is one of the most fundamental problems in online learning. In its simplest form, a learner must choose from one of K actions (possibly choosing a randomized mixture of actions) every round for T rounds. An adversary then reveals a loss vector containing the loss for each action to the learner, the learner incurs their appropriate loss, and play proceeds to the next round. The goal of the learner in such settings is usually to minimize their regret: the gap between their total utility at the end of the game and the maximum utility they could have received if they played the best fixed action in hindsight. Notably, it is possible to construct algorithms for the learner which achieve regret sublinear in T against any adversarially chosen sequence of losses.

Traditionally, a learner may use the entire history of losses up until round t to decide their action at round t. In this paper, we investigate the question of what happens if we restrict the learner's action at time t to depend on the losses in some subset of these rounds. Formally, we require the learner's (randomized) action at time t to be a function of the losses in some subset of rounds S_t, for some fixed collection of subsets $\mathcal{S} = \{S_t\}_{t=1}^{T}$ known to the algorithm a priori. We call this problem the problem of *online learning with temporal feedback graphs*, in analogy with the use of feedback graphs to understand the value of partial information in online learning settings (e.g. Mannor and Shamir (2011)).

In addition to being a mathematically natural extension of the classic problem of prediction with expert advice, this framework is general enough to model many problems of practical interest, including:

- **Batched learning**: In batched learning, the time horizon T is divided into "batches" of size $T_1, T_2, \ldots, T_B$. Losses within a batch are only reported at the beginning of the next batch, so e.g. the action at a round t in the bth batch must only depend on losses from the first $b-1$ batches.

- **Learning with delayed feedback**: In learning with delayed feedback, the loss at round t is only reported to the learner at the end of round $t + \Delta$, after Δ rounds of delay (it is also possible to consider a round dependent delay Δ_t, in which case this subsumes batched learning).

- **Learning with bounded recall**: In learning with bounded recall, the learner is only allowed to use losses from the past M rounds to decide their action. This captures the notion of bounded recall strategies for playing in repeated games introduced by Aumann and Sorin (1989).

- **"Learning from the future"**: Finally, nothing requires us to impose the constraint that S_t only contains rounds before t: if the adversary fixes their sequence of loss vectors at the beginning of the game, then we can let the learner play a function of the losses in S_t for any subset S_t of $\{1, 2, \ldots, T\}$. More practically, this can be used to model adversarial variants of prediction tasks used in the training of large language models: for example, in the task of "masked language modelling" (used in the training of BERT), the goal is to predict a masked token from a surrounding window of tokens (Devlin et al., 2018).

1.1. Our results

We investigate the problem of designing low-regret algorithms and proving regret lower bounds for the problem of online learning with temporal feedback graphs.

Algorithms (Section 3). On the topic of algorithms, we first remark that the seemingly natural algorithm of simply approximately best-responding to the set of losses you can see can actually be very far from optimal. This follows immediately from a result of Schneider and Vodrahalli (2022), who show that such algorithms can incur linear regret in T in bounded recall settings (even when the recall window M is large enough to obtain $O(\sqrt{T})$ regret by restarting every M rounds).

Instead, we propose a more sophisticated algorithm (Algorithm 1) based on constructing a fractional decomposition of the directed feedback graph $\mathcal{S}$ into ordered cliques, which we call "orders" for short. The key observation here is that the temporal feedback graph corresponding to the classic online learning setting (where you can observe all rounds in the past) consists of a single order of size T. Therefore, by partitioning the loss vector among the orders that are subgraphs of $\mathcal{S}$, we can reuse the regret guarantees we have from the standard prediction with experts problem and get strong regret bounds for our algorithm.

We analyze the regret of this algorithm and show it is at most $O(\mathsf{UB}(\mathcal{S})\sqrt{\log K})$, where $\mathsf{UB}(\mathcal{S})$ is the optimal value of a specific convex program that we call the "upper bound program" (Theorem 3). Unfortunately, the size of this convex program (and in particular, the number of variables) is proportional to the number of maximal orders in $\mathcal{S}$, which can be small for some graphs but in general is exponentially large in T. Moreover, actually running Algorithm 1 requires a feasible solution $\boldsymbol{\lambda}$ to the upper bound program as input, and takes time proportional to the sparsity of $\boldsymbol{\lambda}$.

Luckily, by examining the dual convex program, we show that there exists an optimal solution $\boldsymbol{\lambda}^*$ to the upper bound program supported on at most T distinct orders (Lemma 6). This implies that, for a fixed $\mathcal{S}$, there always exists an efficiently implementable learning algorithm achieving the optimum regret bound stated above (Corollary 8), even if it may be hard to actually find this solution and construct this algorithm.

Lower bounds (Section 4). We present two different methods for obtaining worst-case regret lower bounds for a fixed feedback graph $\mathcal{S}$ (in the binary action case $K = 2$). Similarly as with $\mathsf{UB}(\mathcal{S})$, in both methods the lower bound is obtained by solving a convex program defined by the structure of $\mathcal{S}$.

We obtain the first lower bound by extending the classic stochastic lower bound for prediction with expert advice, with the key difference that instead of sampling losses in an iid fashion, we let the loss in round t have bias of magnitude ε_t for some set of $\varepsilon_t \geq 0$. This results in a lower bound $\mathsf{LB}(\mathcal{S})$ on the regret of any algorithm, where $\mathsf{LB}(\mathcal{S})$ is given by the value of a polynomial-sized (and hence, efficiently solvable) convex program in the variables ε_t (Theorem 9). Interestingly, the lower bound program is very similar to the dual of the upper bound program, which lets us immediately conclude that the upper bound $\mathsf{UB}(\mathcal{S})$ is near optimal for a wide variety of feedback graphs. In particular, we show that if the in-neighborhood S_t of each round can be covered by at most R orders of $\mathcal{S}$, then $\mathsf{UB}(\mathcal{S})/\mathsf{LB}(\mathcal{S})$ is at most $O(\sqrt{R})$ (Theorem 10).

Unfortunately, there are many cases where the gap between $\mathsf{LB}(\mathcal{S})$ and $\mathsf{UB}(\mathcal{S})$ can be quite large (even polynomial in T). Inspired by this, we introduce a second lower bound which further generalizes the existing stochastic lower bound by allowing correlations between the losses in different rounds. In particular, we construct an adversary who, instead of sampling losses independently for each round, samples a random variable for each independent set of $\mathcal{S}$, and builds the loss for round t out of the random variables of all the independent sets containing t. By doing so, we introduce another convex program we call the *independent set program*. Although this program is large and hard to solve in general (with number of variables equal to the number of independent sets in $\mathcal{S}$), we prove that its value $\mathsf{ILB}(\mathcal{S})$ is a lower bound on the regret of any algorithm (Theorem 11) and conjecture that $\mathsf{ILB}(\mathcal{S})$ is within a constant factor of $\mathsf{UB}(\mathcal{S})$ (thus implying both this bound and our algorithm are within a constant factor of optimal).

Transitive feedback graphs (Appendix A). Finally, we consider the interesting subclass of *transitive feedback graphs*. These are feedback graphs $\mathcal{S}$ where if $s \in S_t$ and $r \in S_s$, then $r \in S_t$; that is, if the learner can see loss ℓ_s at round t, they can also see all the losses ℓ_r that were visible at round s. Transitive feedback graphs that are a natural class of graphs that include many of our aforementioned applications, such as batched learning and learning with delayed feedback (both the round-dependent delay and round-independent delay case).

For transitive feedback graphs $\mathcal{S}$, we show that we can indeed efficiently construct a sparse solution of the upper bound program, and hence efficiently construct and implement Algorithm 1 given $\mathcal{S}$ (Theorem 14 in Appendix A). Doing so requires two technical insights: i. first, we show that we can construct an efficient separation oracle for the dual of the upper bound program via dynamic programming, ii. second, we show how we can use a solution $\boldsymbol{\mu}^*$ to the dual problem to reduce the problem of constructing a sparse solution of the upper bound problem to a flow-decomposition problem, which we can solve with standard techniques.

Finally, we show that for any transitive graph $\mathcal{S}$ our upper bound $\mathsf{UB}(\mathcal{S})$ is within a constant factor of optimal (Theorem 15 in Appendix A). To do so, we show how the adversary can take any feasible solution to the upper bound dual program and construct a correlated set of losses by observing a one-dimensional Brownian motion at various points in time. Specifically, the adversary associates each round t with an interval $[p_t, q_t]$ of time, and sets the loss at round t based on the sign of $L(q_t) - L(p_t)$ for a biased Brownian motion $L(\tau)$. By doing so, we effectively construct a

feasible solution to our independent set program (which we can show has value within a constant factor of that of the value of our initial feasible solution).

1.2. Related work

Since the seminal work of Mannor and Shamir (2011) introducing the problem of learning under partial information with feedback graphs, there has been an extensive body of literature extending and refining these results, including (Cortes et al., 2018, 2019, 2020; Balseiro et al., 2019; Alon et al., 2015; Cohen et al., 2016; Erez and Koren, 2021). Although we take the name "feedback graph" from this line of work, the similarities between this line of work and the problem we study seem relatively superficial, limited mostly to the fact that both problems are parameterized by a directed graph and have regret bounds that depend on graph-theoretic properties of said graph. It would be interesting to show a stronger connection between these two questions.

Batched feedback has been studied fairly extensively in the online learning community, largely in the contexts of bandits and stochastic rewards Perchet et al. (2016); Gao et al. (2019); Esfandiari et al. (2021). The problem of learning with delayed feedback was first studied in the full-information setting by Weinberger and Ordentlich (2002), and has since been studied in a variety of other learning settings (Mesterharm, 2005; Agarwal and Duchi, 2011; Joulani et al., 2013; Quanrud and Khashabi, 2015). In particular, Quanrud and Khashabi (2015) study the delayed feedback problem with adversarial delays – our framework allows us to immediately recover optimal bounds for this setting given knowledge of the delays. Studying the behavior of agents with bounded recall is an active area in economics (see Neyman (1997) for a survey) which has recently been studied in the online learning setting by Schneider and Vodrahalli (2022). Finally, many sequence models in machine learning are constrained (often for efficiency / training reasons) so that the tth element of their output can only be based off of some pre-determined subset of the inputs. Often this subset is given by a context window (as in the bounded-recall setting), but it also can be structured in other interesting ways, as explored by (Beltagy et al., 2020).

2. Model and Preliminaries

2.1. Online learning preliminaries

We begin with an overview of the classic problem of prediction with expert advice. This can be viewed as a repeated game that takes place over T rounds, where in each round the learner selects an action $x_t \in \mathcal{X}$ and the adversary selects a loss $\ell_t \in \mathcal{L}$. Here $\mathcal{X}$ and $\mathcal{L}$ refer to the action and loss set respectively; we will generally consider the setting $\mathcal{X} = \Delta_K$ and $\mathcal{L} = [0,1]^K$ unless otherwise specified.

The learner selects their actions in accordance with some learning algorithm. Formally, a *learning algorithm* $\mathcal{A}$ is a collection of t functions $A_t : \mathcal{L}^{t-1} \to \mathcal{X}$, with A_t describing the action the learner takes at time t as a function of the losses $\ell_1, \ell_2, \ldots, \ell_{t-1}$. The regret of an algorithm $\mathcal{A}$ on a sequence of losses $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_T)$ is given by

$$\mathsf{Reg}(\mathcal{A}, \boldsymbol{\ell}) = \sum_{t=1}^{T} \langle x_t, \ell_t \rangle - \min_{x^* \in \mathcal{X}} \sum_{t=1}^{T} \langle x^*, \ell_t \rangle, \tag{1}$$

where in (1), $x_t = A_t(\ell_1, \ell_2, \ldots, \ell_{t-1})$. We are often concerned with the worst-case regret of a learning algorithm, which we denote via $\mathsf{Reg}(\mathcal{A}) = \max_{\boldsymbol{\ell} \in \mathcal{L}^T} \mathsf{Reg}(\mathcal{A}, \boldsymbol{\ell})$.

One algorithm with asymptotically optimal regret for the above problem is the Hedge algorithm of Freund and Schapire (1997). The Hedge algorithm with learning rate $\eta > 0$ can be defined via:

$$A_t(\ell_1, \ell_2, \ldots, \ell_{t-1})_i = \frac{\exp\left(-\eta \sum_{s=1}^{t-1} \ell_{s,i}\right)}{\sum_{j=1}^{K} \exp\left(-\eta \sum_{s=1}^{t-1} \ell_{s,j}\right)} \tag{2}$$

It is possible to show that the worst-case regret of Hedge is at most $O(\sqrt{T \log K})$. We will need the following slightly finer-grained bound on the regret of Hedge.

Lemma 1 *Let $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_T)$ be a sequence of losses such that each $\ell_t \in [0, \lambda_t]^K$. If we let $\mathcal{A}$ be the Hedge algorithm initialized with learning rate $\eta = O\left(\sqrt{(\log K)/\sum_{t=1}^{T} \lambda_t^2}\right)$, then*

$$\mathsf{Reg}(\mathcal{A}, \boldsymbol{\ell}) \leq 2\sqrt{\left(\sum_{t=1}^{T} \lambda_t^2\right) \log K}.$$

Proof This follows directly from Theorem 1.5 in Hazan et al. (2016). ■

2.2. Temporal feedback graphs

We now describe the variant of learning from experts with temporal feedback graphs. A *temporal feedback graph* $\mathcal{S}$ is a collection of subsets $S_t \subseteq [T] \setminus \{t\}$ for each $t \in [T]$, where the set S_t represents the set of losses visible to the learning algorithm at time t. Similar to our previous definition, an *$\mathcal{S}$-learning algorithm* $\mathcal{A}$ is a collection of t functions $A_t : \mathcal{L}^{S_t} \to \mathcal{X}$, with each A_t describing the algorithm for mapping the visible losses to the action taken in round t. The regret $\mathsf{Reg}(\mathcal{A}, \boldsymbol{\ell})$ of $\mathcal{A}$ is defined identically as in (1), with the only difference being that x_t is now determined via $x_t = A_t(\ell_{S_t[1]}, \ell_{S_t[2]}, \ldots, \ell_{S_t[|S_t|]})$ (where $(S_t[1], S_t[2], \ldots, S_t[|S_t|])$ is some explicit enumeration of S_t). Our goal throughout this paper is to design efficient $\mathcal{S}$-learning algorithms $\mathcal{A}$ that minimize $\mathsf{Reg}(\mathcal{A})$.

In Appendix A we consider the special subclass of transitive feedback graphs $\mathcal{S}$. We say a graph $\mathcal{S}$ is *transitive* if it is acyclic and has the property that if $s \in S_t$ and $r \in S_s$, then $r \in S_t$. That is, if the learner can see loss ℓ_s at round t, then they can also see all the losses they could see at round s. This class of graphs captures many natural applications (including the batched and delayed settings mentioned in the introduction).

3. Algorithms

3.1. A sub-optimal algorithm

We begin our discussion of $\mathcal{S}$-learning algorithms with a remark about a natural algorithm which turns out to be surprisingly sub-optimal. This algorithm simply does the following: at round t, play the strategy suggested by Hedge (with some learning rate η) when run on all losses visible at time t. That is, in round t, play the strategy defined via

$$A_t(\ell^{S_t})_i = \frac{\exp\left(-\eta \sum_{s \in S_t} \ell_{s,i}\right)}{\sum_{j=1}^{K} \exp\left(-\eta \sum_{s \in S_t} \ell_{s,j}\right)}$$

for some choice of learning rate $\eta > 0$. One can also think of this strategy as approximately best responding to the average loss visible at time t.

Schneider and Vodrahalli (2022) show that in some bounded recall settings, this algorithm can incur regret that is linear in the time horizon T.

Lemma 2 *Let $K = 2$, $M = T/10$ and let $\mathcal{S}$ be the associated bounded recall feedback graph (where $S_t = \{t - M, t - M + 1, \ldots, t - 1\}$). Then for any choice of learning rate η, the above algorithm $\mathcal{A}$ has worst-case regret $\mathsf{Reg}(\mathcal{A}) = \Omega(T)$. In contrast, there exists an algorithm $\mathcal{A}'$ with $\mathsf{Reg}(\mathcal{A}') = O(\sqrt{T})$.*

Proof See Theorem 2 of Schneider and Vodrahalli (2022). The sublinear regret algorithm $\mathcal{A}'$ simply restarts Hedge every M rounds. ■

3.2. A better algorithm

In this section, we present an $\mathcal{S}$-learning algorithm that does not have this detrimental trait, and indeed that we conjecture obtains within a constant factor of the optimal regret bound. As in the previous section, we will also use Hedge as a building block to construct this algorithm – however, we will have to pay much more attention to the structural properties of the graph $\mathcal{S}$.

For a temporal feedback graph $\mathcal{S}$, we say a sequence of rounds $t_1, t_2, \ldots, t_w \in [T]$ forms an *order* if for all $u < v$, $t_u \in S_{t_v}$. That is, every node later in the order can see the losses of all nodes earlier in the order. An order C is *maximal* if no super-sequence of C forms an order. We will let $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$ denote the collection of all maximal orders in $\mathcal{S}$.

Note that in the classic online learning problem, the entire temporal feedback graph $\mathcal{S}$ is just a maximal order of size T. We can therefore think of Hedge (or any standard learning algorithm) as being a learning algorithm tuned specifically for feedback graphs that are orders. Inspired by this, we introduce an algorithm for the general case based on the idea of running multiplicative weights in parallel for every order in the graph and taking an optimal convex combination of these sub-algorithms.

To define this optimal convex combination (and characterize the resulting regret bound we get), we need to solve the following convex program. We will call this program the *upper bound (primal) program*.

$$\begin{aligned} \text{Minimize} \quad & \sum_{c=1}^{N} \sqrt{\sum_{t \in C_c} \lambda_{c,t}^2} \\ \text{Subject to} \quad & \sum_{c=1}^{N} \lambda_{c,t} = 1 \quad \text{for all } t \in [T] \\ & \lambda_{c,t} = 0 \quad \text{if } t \notin C_c \\ & \lambda_{c,t} \geq 0 \quad \text{for all } t \in [T], c \in [N] \end{aligned} \tag{3}$$

Given a set of variables $\lambda_{c,t}$ satisfying the upper bound program (3), we can define our $\mathcal{S}$-learning algorithm as in Algorithm 1. Intuitively, Algorithm 1 instantiates an instance of Hedge for every order in $\mathcal{C}$. In round t, the algorithm partitions the loss of that round among the orders that

pass through t, where the algorithm corresponding to the order C_c receives a $\lambda_{c,t}$ fraction of the loss.

Algorithm 1 Algorithm for solving online learning with temporal feedback graphs

Input: A temporal feedback graph $\mathcal{S}$ with time horizon T and K actions, and a feasible solution $\lambda_{c,t} \geq 0$ to the convex program (3).
For each $c \in [N]$, set $\eta_c = \sqrt{(\sum_{t \in C_c} \lambda_{c,t}^2) \log K}$.
for each round $t \in [T]$ **do**
  for each order C_c containing t **do**
    Let $t_1, t_2, \ldots, t_w = t$ be the prefix of C_c up to and including t.
    Let $x_t^{(c)} \in \Delta_K$ be the strategy defined via (for any $i \in [K]$):

$$x_{t,i}^{(c)} = \frac{\exp\left(-\eta_c \sum_{u=1}^{w-1} \lambda_{c,t_u} \ell_{t_u,i}\right)}{\sum_{j=1}^{K} \exp\left(-\eta_c \sum_{u=1}^{w-1} \lambda_{c,t_u} \ell_{t_u,j}\right)}. \tag{4}$$

  end for
  Play $x_t = \sum_{c;t \in C_c} \lambda_{c,t} x_t^{(c)}$.
  Receive loss vector ℓ_t (and loss $\langle x_t, \ell_t \rangle$).
end for

Note that Algorithm 1 is a valid $\mathcal{S}$-learning algorithm, since the action taken at time t only depends on loss vectors ℓ_{t_u} for $t_u \in S_t$. Let $\mathsf{UB}(\mathcal{S})$ denote the optimal value of the upper bound program (3). The following theorem bounds the regret of Algorithm 1.

Theorem 3 *If Algorithm 1 is run with an optimal solution $\boldsymbol{\lambda}^*$ to the upper bound program (3), it incurs at most $O(\mathsf{UB}(\mathcal{S})\sqrt{\log K})$ regret.*

Proof Fix an action $x^* \in [K]$, and consider the regret of this algorithm (which we will call $\mathcal{A}$) against action x^* for some fixed loss sequence $\boldsymbol{\ell}$. We can write this regret in the form:

$$\begin{aligned}
\mathsf{Reg}_{x^*}(\mathcal{A}, \boldsymbol{\ell}) &:= \sum_{t=1}^{T} \langle x_t - x^*, \ell_t \rangle \\
&= \sum_{t=1}^{T} \sum_{c=1}^{N} \left\langle \lambda_{c,t}^* \left(x_t^{(c)} - x^* \right), \ell_t \right\rangle \qquad (5) \\
&= \sum_{c=1}^{N} \left(\sum_{t=1}^{T} \left\langle x_t^{(c)} - x^*, \lambda_{c,t}^* \ell_t \right\rangle \right). \qquad (6)
\end{aligned}$$

Here in (5) we have used the fact that $\sum_{c=1}^{N} \lambda_{c,t}^* = 1$ for any fixed $t \in [T]$ (since the $\lambda_{c,t}^*$ satisfy the convex program (3)). Note that, by definition, $x_t^{(c)}$ is the output of an instance $\mathcal{A}_p$ of the Hedge algorithm initialized with learning rate η_c on losses $\boldsymbol{\ell}^{(c)}$ given by $\ell_t^{(c)} = \lambda_{c,t}^* \ell_t$; moreover, the summand of the RHS of (6) corresponding to a given c is at most $\mathsf{Reg}(\mathcal{A}_c, \boldsymbol{\ell}^{(c)})$. Therefore, by Lemma 1, we have that

$$\mathsf{Reg}_{x^*}(\mathcal{A}, \boldsymbol{\ell}) \leq \sum_{c=1}^{N} 2\sqrt{\left(\sum_{t=1}^{T} (\lambda_{c,t}^*)^2\right) \log K} = 2\sqrt{\log K} \cdot \mathsf{UB}(\mathcal{S}).$$

Since $\mathsf{Reg}(\mathcal{A}, \boldsymbol{\ell}) = \max_{x^* \in \Delta_K} \mathsf{Reg}_{x^*}(\mathcal{A}, \boldsymbol{\ell})$, the conclusion follows. ■

Theorem 3 leads to two natural questions:

1. First, although we can solve the convex program (3) in time polynomial in N, T, and K, one might notice that the parameter N is equal to the number of (maximal) orders in the temporal feedback graph $\mathcal{S}$ and can be very large (possibly exponential in T). In fact, even specifying a solution to the convex program and running Algorithm 1 requires time polynomial in N as written. Are there efficient learning algorithms (running in polynomial time in T and K) for this problem?

2. Secondly, does Algorithm 1 obtain the optimal regret bound for the $\mathcal{S}$-learning problem? That is, must any $\mathcal{S}$-learning algorithm $\mathcal{A}$ incur at least $\Omega(\mathsf{UB}(\mathcal{S})\sqrt{\log K})$ regret on some sequence of losses?

The remainder of this paper will largely be concerned with providing answers to both of these questions (especially for the specific case of transitive feedback graphs $\mathcal{S}$). For both of these questions, we will find it useful to examine the dual of the upper bound convex program, which we explore in the next section.

3.3. The dual convex program and an efficient learning algorithm

For multiple reasons, we will find it easier to work with the dual of the upper bound convex program. In contrast with the original upper bound program, which is a minimization problem involving (up to) $T \cdot N$ variables with T constraints, the dual program is a maximization problem on T variables with N constraints. We state this program (which we call the *upper bound dual program*) below.

$$\begin{aligned} \text{Maximize} \quad & \sum_{t=1}^{T} \mu_t \\ \text{Subject to} \quad & \sum_{t \in C_c} \mu_t^2 \leq 1 \quad \text{for all } c \in [N] \\ & \mu_t \geq 0 \quad \text{for all } t \in [T] \end{aligned} \tag{7}$$

The following theorem proves that the program (7) is in fact the dual of the upper bound program.

Theorem 4 *The optimal value of the upper bound dual program for temporal feedback graph $\mathcal{S}$ is equal to* $\mathsf{UB}(\mathcal{S})$.[1]

Having established duality, complementary slackness allows us to infer strong structural statements about the optimal primal solution given an optimal dual solution.

1. For the sake of brevity, we defer many of the longer or more standard proofs to Appendix D.

Lemma 5 *Let $\boldsymbol{\mu}^*$ be an optimal solution to the upper bound dual program (7). Then there exists a primal solution $\boldsymbol{\lambda}^*$ to (3) such that, for any $c \in [N]$ where*

$$\sum_{t \in C_c} (\mu_t^*)^2 < 1, \tag{8}$$

we have that $\lambda_{c,t}^* = 0$ for all $t \in [T]$, and for any $c \in [N]$ where

$$\sum_{t \in C_c} (\mu_t^*)^2 = 1, \tag{9}$$

we have that $\lambda_{c,t}^* = \rho_c \mu_t^*$, for some constant ρ_c.

Given a solution $\boldsymbol{\mu}^*$ to the upper bound dual, let $\mathcal{C}(\boldsymbol{\mu}^*) \subseteq [N]$ denote the set of orders c where the equality (9) is tight. Lemma 5 implies that we only need to solve the upper bound program (and run Algorithm 1) for orders $c \in \mathcal{C}(\boldsymbol{\mu}^*)$. If $|\mathcal{C}(\boldsymbol{\mu}^*)|$ is small, this can be far more efficient than naively running Algorithm 1. In particular, since there are only T variables in the dual program, we would naively expect only T of the constraints to be binding, and therefore $|\mathcal{C}(\boldsymbol{\mu}^*)|$ should generically equal T (in which case we have an efficient solution).

Of course, for many temporal feedback graphs $\mathcal{S}$, the dual program is not generic, and it can be the case that many (or all of) the constraints bind at optimality. The following lemma shows that, even in such cases, it is possible to find an optimal solution of the primal program that is supported on at most T different orders.

Lemma 6 *Given any optimal solution $\boldsymbol{\mu}^*$ to the upper bound dual program, there exists a subset $\mathcal{B} \subseteq \mathcal{C}(\mu^*)$ with $|\mathcal{B}| \leq T$ such that there exists a optimal solution $\boldsymbol{\lambda}^*$ to the upper bound program with the property that $\lambda_{c,t}^* = 0$ if $C_c \notin \mathcal{B}$.*

Proof By Lemma 5, we can restrict our attention to optimal solutions $\boldsymbol{\lambda}^*$ where $\lambda_{c,t}^* = \rho_c \mu_t^*$ for $c \in \mathcal{C}(\mu^*)$ (for some collection of values $\rho_c \geq 0$) and $\lambda_{c,t}^* = 0$ for $c \notin \mathcal{C}(\mu^*)$. Since $\sum_{t \in C_c} (\mu_t^*)^2 = 1$ for all $c \in \mathcal{C}(\mu^*)$, we can rewrite the original upper bound primal program as the following linear program in the ρ_c:

$$\begin{aligned} \text{Minimize} \quad & \sum_{c \in \mathcal{C}(\mu^*)} \rho_c \\ \text{Subject to} \quad & \sum_{\substack{c \in \mathcal{C}(\boldsymbol{\mu}^*) \\ \text{s.t. } t \in C_c}} \rho_c = 1/\mu_t \quad \text{for all } t \in [T] \\ & \rho_c \geq 0 \quad \text{for all } c \in \mathcal{C}(\boldsymbol{\mu}^*). \end{aligned} \tag{10}$$

But any extreme point of the linear program (10) will be the intersection of at least $|\mathcal{C}(\boldsymbol{\mu}^*)|$ constraints, of which at most T are not of the form $\rho_c = 0$. It follows that there is an optimal extreme point to this LP where at most T of the ρ_c are non-zero, and hence a $\boldsymbol{\lambda}^*$ with the property we described. ■

Given a feedback graph $\mathcal{S}$, we call a subset $\mathcal{B}$ of $\mathcal{C}$ a *basis* for $\mathcal{S}$ if there exists an optimal solution to the upper bound program supported entirely on orders $c \in \mathcal{B}$. Lemma 6 shows that there always

exists a basis of size at most T. Having an optimal solution with a small basis is valuable since it allows us to run Algorithm 1 more efficiently.

Lemma 7 *Let $\boldsymbol{\lambda}$ be a feasible point for the upper bound program (3) which is supported on a basis $\mathcal{B}$ of size $|\mathcal{B}| = B$. Then we can run Algorithm 1 on $\boldsymbol{\lambda}$ in time $O(BK)$ per iteration ($O(BKT)$ overall).*

Proof We modify Algorithm 1 by maintaining one instance of Hedge for each of the B non-zero orders (each of which takes $O(K)$ time to update per iteration). ■

Together with Lemma 6, this provides us with a partial answer to our first question.

Corollary 8 *For any feedback graph $\mathcal{S}$, there exists an $\mathcal{S}$-learning algorithm which runs in time $O(KT)$ per iteration and incurs regret at most $O(\mathsf{UB}(\mathcal{S})\sqrt{\log K})$.*

The catch, of course, is that actually coming up with the learning algorithm of Corollary 8 involves computing an optimal basis $\mathcal{B}$ and associated primal solution $\boldsymbol{\lambda}^*$, which may not be computationally efficient. In Section A, we will see that for the important class of transitive feedback graphs, it is possible to compute this sparse primal solution efficiently.

4. Lower bounds for online learning with temporal feedback graphs

We now turn our attention to the second of the two questions: is our regret bound of $O(\mathsf{UB}(\mathcal{S})\sqrt{\log K})$ asymptotically tight? Throughout this section we will focus on the two-action setting ($K = 2$) for clarity of exposition. We expect that all lower bounds we present should extend to the K-action setting (with an additional factor of $\sqrt{\log K}$ in the lower bound).

4.1. A (naive yet efficient) lower bound program

We begin by examining what happens when we try to extend the original lower bound proof for the standard problem of learning with experts to the temporal feedback graph setting.

At a high-level, the original lower bound proof proceeds as follows. First, the adversary uniformly samples a bit $B \in \{-1, 1\}$ unknown to the learner. The adversary then fixes an $\varepsilon \in (0, 1/2)$ and generates a sequence of i.i.d. random variables $X_1, X_2, \ldots, X_T$ where each $X_t \in \{0, 1\}$ is drawn independently from a Bernoulli $(\frac{1}{2} + B\varepsilon)$ distribution. The adversary then chooses losses defined via $\ell_t = (X_t, 1/2)$.

Now, the learner cannot learn much about the random bit B until they have seen at least $\Omega(1/\varepsilon^2)$ samples from $\mathsf{Bern}(\frac{1}{2}+B\varepsilon)$, and hence until they have seen at least $\Omega(1/\varepsilon^2)$ loss vectors. But during that time, they incur $\Omega(\varepsilon)$ regret per round. Altogether, this implies that the adversary can force the learner to incur a total regret of at least $\Omega(\varepsilon \cdot \max(1/\varepsilon^2, T))$. Picking $\varepsilon = 1/\sqrt{T}$, we obtain the well-known $\Omega(\sqrt{T})$ lower bound.

We will start our discussion of lower bounds for the problem of $\mathcal{S}$-learning with a similar approach. Again, the adversary will begin by uniformly sampling a bit $B \in \{-1, 1\}$. But now, to account for the additional potential asymmetry in the feedback structure across rounds, the adversary will select a sequence of "scales" $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_T \geq 0$, and sample the r.v. X_t from the distribution $\mathsf{Bern}(\frac{1}{2} + B\gamma\varepsilon_t)$ (for some "global scale" $\gamma > 0$). Finally, the adversary again sets $\ell_t = (X_t, 1/2)$.

Intuitively, we can understand the performance of this strategy as follows. As long as the learner cannot figure out the value of B based on the losses observable at round t, the learner will incur an expected regret of $\Omega(\varepsilon_t)$ in that round. In order to prevent the learner from figuring out the value of B in round t, the amount of information leaked by the losses in S_t about B should be small. This quantity is roughly proportional to $\sum_{s \in S_t} \varepsilon_s^2$ (in particular, observing ℓ_s leaks $\Theta(\varepsilon_s^2)$ bits of information about B). This motivates writing down the following convex program (11), which we call the *lower bound program*.

$$\begin{aligned} \text{Maximize} \quad & \sum_{t=1}^{T} \varepsilon_t \\ \text{Subject to} \quad & \sum_{t \in S_{t'}} \varepsilon_t^2 \leq 1 \quad \text{for all } t' \in [T] \\ & \varepsilon_t \geq 0 \quad \text{for all } t \in [T] \end{aligned} \tag{11}$$

We will write $\mathsf{LB}(\mathcal{S})$ to denote the optimal value of the lower bound program. Note that this program shares many structural similarities with the upper bound dual program – the main difference is that whereas in the upper bound dual program (7), the constraints bound the sum of the squares of the variables over all *orders* in $\mathcal{S}$, here the sums are over all *neighborhoods* $S_{t'}$ in $\mathcal{S}$. Since every order C_c is a subset of the neighborhood $S_{t'}$ of its last element, the constraints of the lower bound program are stronger than that of the upper bound dual (and so $\mathsf{LB}(\mathcal{S}) \leq \mathsf{UB}(\mathcal{S})$, as we would expect).

The following theorem formalizes the above intuition and shows that $\mathsf{LB}(\mathcal{S})$ is indeed a valid lower bound for the $\mathcal{S}$-learning problem (up to a universal constant factor).

Theorem 9 *Every $\mathcal{S}$-learning algorithm $\mathcal{A}$ must incur worst-case regret* $\mathsf{Reg}(\mathcal{A}) \geq \mathsf{LB}(\mathcal{S})/100$.

One nice property of the lower bound program (11) is that it is polynomial-sized, and can therefore be optimized efficiently. Another feature of this program is that, due to its similarity with the upper bound dual program, it immediately gives us the following multiplicative bound on its optimality.

Theorem 10 *Let $\mathcal{S}$ be a temporal feedback graph where every neighborhood S_t is contained in the union of at most R orders C_c. Then $\mathsf{UB}(\mathcal{S})/\mathsf{LB}(\mathcal{S}) \leq \sqrt{R}$ (and hence the regret bound of Algorithm 1 is within $\Omega(\sqrt{R})$ of optimal).*

Proof Let $\boldsymbol{\mu}^*$ be an optimal solution to the upper bound dual program. Note that in such a graph, μ_t would satisfy $\sum_{t \in S_{t'}} \mu_t^2 \leq R$, and therefore setting $\varepsilon_t = \mu_t/\sqrt{R}$ forms a feasible solution to (11) with value at least $\mathsf{UB}(\mathcal{S})/\sqrt{R}$ and at most $\mathsf{LB}(\mathcal{S})$. The conclusion follows. ■

There are some classes of feedback graphs where the parameter R in Theorem 10 is $O(1)$ and hence where we can immediately conclude that Algorithm 1 is nearly optimal (one interesting such class of graphs is those arising from bounded recall, where each S_t is itself an order). The downside, however, is that in general the gap between $\mathsf{LB}(\mathcal{S})$ and $\mathsf{UB}(\mathcal{S})$ can be quite large (at least $\Omega(T^{1/4})$ – see Appendix B). In the next section we present a stronger lower bound that comes from a much larger convex program.

4.2. The independent set lower bound

One of the main reasons the lower bound in the previous section is lax is every loss which is visible to the learner at round t reveals independent information about B. But since not all of these rounds can observe each other, this is inherently a little wasteful – it would be better if we could "reuse" the same signal about B across multiple different rounds.

To implement this idea, let $\mathcal{I}(\mathcal{S})$ denote the collection of independent sets of the (undirected version) of $\mathcal{S}$. The adversary then begins by generating a random variable with some bias Bw_I for every independent set $I \in \mathcal{I}(\mathcal{S})$. Finally, to generate the loss at round t, the adversary combines the random variables for sets I containing t (by e.g. taking their majority). This motivates us to write down the following convex program:

$$\begin{aligned} \text{Maximize} \quad & \sum_{t=1}^{T} \sqrt{\sum_{\substack{I \in \mathcal{I}(\mathcal{S}) \\ \text{s.t. } t \in I}} w_I^2} \\ \text{Subject to} \quad & \sum_{\substack{I \in \mathcal{I}(\mathcal{S}) \text{ s.t.} \\ I \cap \mathcal{S}_t \neq \emptyset}} w_I^2 \leq 1 \quad \text{for all } t \in [T] \end{aligned} \tag{12}$$

We call the above program the *independent set (lower bound) program*, and denote its optimal value by $\mathsf{ILB}(\mathcal{S})$. Note that if we restrict the set of feasible set to w that take nonzero values on single nodes, then the independent set program reduces exactly to our original lower bound program, and thus implies that $\mathsf{ILB}(\mathcal{S}) \geq \mathsf{LB}(\mathcal{S})$. But more importantly, this more general program still gives a lower bound on the achievable regret of any algorithm.

Theorem 11 *Every $\mathcal{S}$-learning algorithm $\mathcal{A}$ must incur worst-case regret* $\mathsf{Reg}(\mathcal{A}) \geq \mathsf{ILB}(\mathcal{S})/50$.

We leave it as an interesting open question to characterize the gap between $\mathsf{ILB}(\mathcal{S})$ and $\mathsf{UB}(\mathcal{S})$. We conjecture that this gap is at most a universal constant (thus showing that both Algorithm 1 and this lower bound are optimal up to constant factors).

Conjecture 12 *There exists a constant γ such that for any temporal feedback graph $\mathcal{S}$,* $\mathsf{UB}(\mathcal{S})/\mathsf{ILB}(\mathcal{S}) \leq \gamma$.

In particular, in Appendix A we employ a variant of this technique to prove a tight lower bound for all transitive graphs, providing partial evidence for this conjecture.

References

Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. *Advances in neural information processing systems*, 24, 2011.

Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows. 1988.

Noga Alon, Nicolo Cesa-Bianchi, Ofer Dekel, and Tomer Koren. Online learning with feedback graphs: Beyond bandits. In *Conference on Learning Theory*, pages 23–35. PMLR, 2015.

Robert J Aumann and Sylvain Sorin. Cooperation and bounded recall. *Games and Economic Behavior*, 1(1):5–39, 1989.

Santiago Balseiro, Negin Golrezaei, Mohammad Mahdian, Vahab Mirrokni, and Jon Schneider. Contextual bandits with cross-learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Alon Cohen, Tamir Hazan, and Tomer Koren. Online learning with feedback graphs without the graphs. In *International Conference on Machine Learning*, pages 811–819. PMLR, 2016.

Corinna Cortes, Giulia DeSalvo, Claudio Gentile, Mehryar Mohri, and Scott Yang. Online learning with abstention. In *international conference on machine learning*, pages 1059–1067. PMLR, 2018.

Corinna Cortes, Giulia DeSalvo, Claudio Gentile, Mehryar Mohri, and Scott Yang. Online learning with sleeping experts and feedback graphs. In *International Conference on Machine Learning*, pages 1370–1378. PMLR, 2019.

Corinna Cortes, Giulia DeSalvo, Claudio Gentile, Mehryar Mohri, and Ningshan Zhang. Online learning with dependent stochastic feedback graphs. In *International Conference on Machine Learning*, pages 2154–2163. PMLR, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Liad Erez and Tomer Koren. Towards best-of-all-worlds online learning with feedback graphs. *Advances in Neural Information Processing Systems*, 34:28511–28521, 2021.

Hossein Esfandiari, Amin Karbasi, Abbas Mehrabian, and Vahab Mirrokni. Regret bounds for batched bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7340–7348, 2021.

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

Zijun Gao, Yanjun Han, Zhimei Ren, and Zhengqing Zhou. Batched multi-armed bandits problem. *Advances in Neural Information Processing Systems*, 32, 2019.

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Pooria Joulani, Andras Gyorgy, and Csaba Szepesvári. Online learning under delayed feedback. In *International Conference on Machine Learning*, pages 1453–1461. PMLR, 2013.

Shie Mannor and Ohad Shamir. From bandits to experts: On the value of side-observations. *Advances in Neural Information Processing Systems*, 24, 2011.

Chris Mesterharm. On-line learning with delayed label feedback. In *International Conference on Algorithmic Learning Theory*, pages 399–413. Springer, 2005.

Abraham Neyman. *Cooperation, repetition, and automata*. Springer, 1997.

Vianney Perchet, Philippe Rigollet, Sylvain Chassang, and Erik Snowberg. Batched bandit problems. 2016.

Kent Quanrud and Daniel Khashabi. Online learning with adversarial delays. *Advances in neural information processing systems*, 28, 2015.

RT Rockafellar. Convex analysis. *Princeton Math. Series*, 28, 1970.

Jon Schneider and Kiran Vodrahalli. Online learning with bounded recall. *arXiv preprint arXiv:2205.14519*, 2022.

Marcelo J Weinberger and Erik Ordentlich. On delayed prediction of individual sequences. *IEEE Transactions on Information Theory*, 48(7):1959–1976, 2002.

Appendix A. Transitive feedback graphs

In this appendix, we turn our attention to the case of transitive feedback graphs $\mathcal{S}$. Recall that these are acyclic feedback graphs $\mathcal{S}$ where if $u \in S_t$ and $v \in S_u$, then $v \in S_t$ (that is, if you can see loss ℓ_u at round t, you can also see all losses you could see at round u).

A.1. An efficient algorithm

We will begin by showing that we can efficiently implement Algorithm 1 for any transitive feedback graph $\mathcal{S}$. In particular, it suffices to show that we can efficiently find an optimal basis $\mathcal{B}$ and associated primal solution $\boldsymbol{\lambda}^*$ in time polynomial in T.

Note that for a transitive feedback graph $\mathcal{S}$, any directed path $t_1 \to t_2 \to \cdots \to t_w$ in $\mathcal{S}$ forms an order. This allows us to construct an efficient separation oracle (and hence efficiently solve) the upper bound dual program (7).

Lemma 13 *If $\mathcal{S}$ is a transitive feedback graph, then we can find an optimal solution $\boldsymbol{\mu}^*$ to the upper bound dual program (to within ε additive error) in time $\mathrm{poly}(T, 1/\varepsilon)$. Moreover, there exists an efficiently computable subgraph $\mathcal{S}'$ of $\mathcal{S}$ and two sets of rounds $\mathrm{src}(\mathcal{S}')$ and $\mathrm{dest}(\mathcal{S}')$ such that $\mathcal{C}(\boldsymbol{\mu}^*)$ is equal to the set of all directed paths contained within $\mathcal{S}'$ that start at a node in $\mathrm{src}(\mathcal{S}')$ and end at a node in $\mathrm{dest}(\mathcal{S}')$.*

Proof We will first provide an efficient separation oracle which, given a $\boldsymbol{\mu} \geq 0$, will either describe which of the order constraints $c \in [N]$ $\boldsymbol{\mu}$ violates, or report that $\boldsymbol{\mu}$ is a valid dual solution. With such an oracle, we can use the ellipsoid method to solve the convex program to within ε additive error in time $\mathrm{poly}(T, 1/\varepsilon)$.

Since every directed path in $\mathcal{S}$ corresponds to some order C_c, it suffices to be able to find the directed path P in $\mathcal{S}$ which maximizes $\sum_{t \in P} \mu_t^2$ (if this maximum is larger than 1, we have a violating constraint given by the order P corresponds to). But this is equivalent to computing the longest weighted path in a directed acyclic graph (where vertex t has weight μ_t^2), which can be solved efficiently in $O(T^2)$ time via dynamic programming.

Now, note that $C(\boldsymbol{\mu}^*)$ consists of all directed paths P where $\sum_{t \in P} \mu_t^2 = 1$. We can use the same dynamic program to show that $C(\boldsymbol{\mu}^*)$ simply contains all paths in a subgraph $\mathcal{S}'$ of $\mathcal{S}$ that start in a source set $\mathrm{src}(\mathcal{S}')$ and end in a target set $\mathrm{dest}(\mathcal{S}')$. For each node $t \in [T]$, let V_t be the maximum value of $\sum_{s \in P} \mu_s^2$ over any directed path P ending at t. We can use the following procedure to construct $\mathcal{S}'$:

- Start by adding the set of vertices t where $V_t = 1$ to $\mathcal{S}'$.
- For every round t in $\mathcal{S}'$ that hasn't been processed, find all ancestors $s \in S_t$ with the property that $V_t - V_s = \mu_t^2$. For each such ancestor s, add the edge $s \to t$ to $\mathcal{S}'$ (i.e., add s to S'_t), and process s if it has not already been processed.
- Finally, let $\mathrm{src}(\mathcal{S}')$ equal the set of rounds $t \in \mathcal{S}'$ where $V_t = \mu_t^2$, and let $\mathrm{dest}(\mathcal{S}')$ equal the set of rounds t in $\mathcal{S}'$ where $V_t = 1$ (equivalently, these are the sources and sinks of the DAG $\mathcal{S}'$).

To see why this procedure works, note that any path $P = (t_1, t_2, \ldots, t_k)$ from $\mathrm{src}(\mathcal{S}')$ to $\mathrm{dest}(\mathcal{S}')$ along edges of $\mathcal{S}'$ will satisfy $\sum_{t \in P} \mu_t^2 = \sum_{i=1}^{k} (V_i - V_{i-1}) = V_k = 1$. Moreover,

any edge this algorithm does not select cannot possibly be included in a path in $C(\boldsymbol{\mu}^*)$ (the sum of μ_t^2 over a path from t_i to t_j containing this edge must be strictly less than $V_j - V_i \leq 1$). It follows that $C(\boldsymbol{\mu}^*)$ simply contains all source-destination paths in $\mathcal{S}'$. ■

We can now use the dual solution (and the characterization of $\mathcal{C}(\boldsymbol{\mu}^*)$) provided by Lemma 13 to find an efficient, sparse solution to the primal.

Theorem 14 *If $\mathcal{S}$ is a transitive feedback graph, then we can find an optimal solution $\boldsymbol{\lambda}^*$ to the upper bound program (to within additive ε error) supported on a basis $\mathcal{B}$ of size $|\mathcal{B}| \leq T$ in time* $\mathrm{poly}(T, 1/\varepsilon)$.

Proof By Lemma 13, we can efficiently construct a dual solution $\boldsymbol{\mu}^*$ and corresponding set $\mathcal{C}(\boldsymbol{\mu}^*)$. Our approach will be to use this to find a sparse solution to the linear program (10) in the proof of Lemma 6 in the ρ_c random variables (recall that such a solution characterizes an optimal primal solution via $\lambda^*_{c,t} = \rho_c \mu^*_t$ for each $c \in \mathcal{C}(\boldsymbol{\mu}^*)$, $t \in C_c$).

The linear program in (10) has $|\mathcal{C}(\boldsymbol{\mu}^*)|$ random variables, and therefore it would be inefficient to solve directly. Instead, we will show that we can use the structure of $\mathcal{C}(\boldsymbol{\mu}^*)$ (as the set of all source-destination paths in the subgraph $\mathcal{S}'$) to rewrite it as a flow problem. Indeed, consider the following linear program in the variables f_e (for each edge $e = (s,t)$ in the edge set $E(\mathcal{S}')$ of $\mathcal{S}'$):

$$\begin{aligned}
\text{Minimize} \quad & \sum_{\substack{t \in \mathrm{dest}(\mathcal{S}') \\ (s,t) \in E(\mathcal{S}')}} f_{s,t} && (13)\\
\text{Subject to} \quad & \sum_{s|(s,t)\in\mathcal{S}'} f_{s,t} = 1/\mu_t \quad \text{for all } t \in [T] \\
& \sum_{s|(s,t)\in\mathcal{S}'} f_{s,t} = \sum_{s|(t,s')\in\mathcal{S}'} f_{t,s'} \quad \text{for all } t \in \mathcal{S}' \setminus (\mathrm{src}(\mathcal{S}') \cup \mathrm{dest}(\mathcal{S}')) \\
& f_{s,t} \geq 0 \quad \text{for all } (s,t) \in E(\mathcal{S}').
\end{aligned}$$

The linear program (13) is a polynomial-sized LP, so we can solve it efficiently. At the same time, it is equivalent to the linear program (10) in the following sense: first, given any solution ρ_c to (10), if we let $f_e = \sum_{c|e\in c} \rho_c$ equal the sum of ρ_c over all paths C_c that contain e, then f_e satisfies (13) (with the same objective value). Conversely, given any flow f_e solving (13), we can decompose it into a positive combination of source-destination paths. If we let ρ_c be the weight of the path C_c in this decomposition, we can likewise check that ρ_c satisfies (10) (also with the same objective value).

There are well-known efficient procedures for flow-decomposition (see e.g. Ahuja et al. (1988)), which take a flow f_e and return a positive combination of at most $O(|E(\mathcal{S}')|)$ paths. By doing this, we can obtain an optimal solution ρ_c to (10) supported on a basis of size at most $O(T)$. We can then shrink this to an optimal basis $\mathcal{B}$ of size at most T by removing all other variables from the LP (10) and using any method for finding a basic feasible solution to the resulting program (e.g. optimizing a random linear functional over the face containing the optimum). ■

Theorem 14 implies that given any transitive feedback graph $\mathcal{S}$, we can efficiently construct the algorithm of Corollary 8 that runs in time $O(KT)$ per iteration and incurs regret at most $O(\mathsf{UB}(\mathcal{S})\sqrt{\log K})$.

A.2. An asymptotically tight lower bound

Finally, we prove a lower bound of $\Omega(\mathsf{UB}(\mathcal{S}))$ on the regret of any algorithm when the feedback graph $\mathcal{S}$ is transitive. This combined with our upper bound in Corollary 8 settles the optimal learning rate for the case of transitive graphs.

Theorem 15 (Tight lower bound for transitive $\mathcal{S}$) *For any transitive graph $\mathcal{S}$, every $\mathcal{S}$-learning algorithm $\mathcal{A}$ must incur worst-case regret* $\mathsf{Reg}(\mathcal{A}) \geq \mathsf{UB}(\mathcal{S})/50$.

Proof To show this lower bound, we use the fact that the value of the upper bound program $\mathsf{UB}(\mathcal{S})$ is equal to its dual (7). Then, for every instance $(\mu_t)_{t=1}^T$ of (7), we show $\Omega(\sum_{t=1}^T \mu_t)$ as a lower bound for the value of the regret. To show this lower bound, we take a similar strategy as in the proof of Theorem 9; the adversary flips a coin $B \in \{-1, +1\}$ and then consider loss vectors $\ell_t = (X_t, \frac{1}{2})$ where X_t is a $\gamma\epsilon_t$-biased Bernoulli variable, where the bias is to one if $B = 1$ and to zero if $B = -1$. Here again the adversary attempts to use shared randomness between ℓ_t's to block the chances of the player to learn B. At the same time, the adversary has to be careful not to reveal any information about the ℓ_t given the loss vectors ℓ_s's, $s \in \mathcal{S}_t$ which it can observe at time t. Here, the adversary exploits the transitive nature of $\mathcal{S}$ to cook up these random variables and considers a linearly shifted Brownian motion with rate γ:

$$L_t = \gamma B t + B_t.$$

The adversary defines each X_t as a positivity indicator variable of a chunk of the process L. Namely, for times $p_t \geq q_t$,

$$X_t = 1\{L_{p_t} - L_{q_t} \geq 0\}$$

Specifically, for each time $t \in [T]$, the adversary defines

$$q_t = \max_{s \in \mathcal{S}_t} \{p_s\},$$
$$p_t = q_t + \mu_t^2.$$

With this definition, first we show by induction that $p_t \leq 1$ for all t. We strengthen the hypothesis of induction and concurrently show the argument that for any t, there is an ordered clique p in $\mathcal{S}$ ending at t and with $\sum_{s \in p} \mu_s^2 = p_t$. The hypothesis of induction is trivial since $q_1 = 0$ and $p_1 = \mu_1^2$. Now for arbitrary $t \leq T$, let $s \in \mathcal{S}_t$ be the index with maximum p_s in $\mathcal{S}_t$. From the hypothesis of induction we know there is an ordered clique p ending at s with $\sum_{s' \in p} \mu_{s'}^2 = p_s$. Now from definition we have $p_t = p_s + \mu_t^2$. Therefore, $p_t = \sum_{s \in p'} \mu_s^2$ where p' is the ordered clique of p concatenated with t. This shows the step of induction. Finally, note that from the argument that we showed, that there is an ordered clique p ending at t with $\sum_{s \in p} \mu_s^2 = p_t$, we conclude $p_t = \sum_{s \in p} \mu_s^2 \leq 1$ because of the constraint in the dual of the upper bound program.

The second observation is that with this definition, X_t becomes independent of X_s for $s \in \mathcal{S}_t$. This follows from the independence of disjoint increments of Brownian motion. Next, we show that with small enough choice of constant γ, the adversary cannot distinguish between $B = \pm 1$ cases with constant probability. For this, similar to the proof of Theorem 9 it is enough to bound the total variation distance between $Q^{-1}(\cup_{s \in \mathcal{S}_t}(L_{p_s} - L_{q_s}))$ and $Q^{+1}(\cup_{s \in \mathcal{S}_t}(L_{p_s} - L_{q_s}))$, where here we

use the notation $Q^+(X)$ and $Q^-(X)$ to refer to the distribution of the random variable (or more generally random process) X given $B = 1$ and $B = -1$, respectively. But again from the data processing inequality, we have

$$\mathrm{TV}\Big(Q^-(\cup_{s\in\mathcal{S}_t}(L_{p_s} - L_{q_s})), Q^+(\cup_{s\in\mathcal{S}_t}(L_{p_s} - L_{q_s}))\Big) \tag{14}$$

$$\leq \sqrt{D\Big(Q^-(\cup_{s\in\mathcal{S}_t}(L_{p_s} - L_{q_s}))||Q^+(\cup_{s\in\mathcal{S}_t}(L_{p_s} - L_{q_s}))\Big)} \tag{15}$$

$$\leq \sqrt{D\Big(Q^-\big(L[0:1]\big)||Q^+\big(L[0:1]\big)\Big)}, \tag{16}$$

where in the last Equation, $Q^{\pm}(L[0:1])$ refers to the distribution corresponding to the whole sample path of the process L in the interval $[0,1]$ and we used the fact that for all times s, $0 \leq q_s \leq p_s \leq 1$.

Next, we use the following Lemma, proved in Appendix C.3, to upper bound the RHS in Equation (16). This Lemma provides a formula for the KL divergence of two shifted Brownian motions.

Lemma 16 (KL divergence between shifted Brownian motions) *For linearly shifted Brownian motions $L_t = \gamma t + B_t$, the KL divergence between the measures corresponding to L_t and B_t in the interval $[0,1]$ is equal to*

$$D\Big(Q\big(L[0:1]\big)||Q\big(B[0:1]\big)\Big) = \gamma^2/2.$$

Applying Lemma 16, we get

$$D\Big(Q^-\big(L[0:1]\big)||Q^+\big(L[0:1]\big)\Big) \leq 2\gamma^2,$$

Plugging this into Equation (16):

$$\mathrm{TV}\Big(Q^{-1}(\cup_{s\in\mathcal{S}_t}(L_{p_s} - L_{q_s})), Q^{+1}(\cup_{s\in\mathcal{S}_t}(L_{p_s} - L_{q_s}))\Big) \leq 2\gamma.$$

Therefore, similar to the proof of Theorem 11,

$$\mathbb{E}[\epsilon_t|x_t - x_t^*|] \geq \gamma\epsilon_t(1/2 - \gamma) = \frac{\gamma\epsilon_t}{4}.$$

by picking $\gamma = 1/4$. Moreover, from Lemma 19 we have $\epsilon_t \geq \frac{\mu_t}{\sqrt{2\pi}}$. Therefore,

$$\mathsf{Reg}(\mathcal{A}) \geq \gamma \sum_{t=1}^{T} \frac{\mu_t}{4\sqrt{2\pi}} = \frac{\sum_t \mu_t}{16\sqrt{2\pi}} = \frac{\mathsf{UB}(\mathcal{S})}{16\sqrt{2\pi}}.$$

■

We briefly remark on the connection between Theorem 15 and the independent set program of Section 4.2. Although we have presented Theorem 15 in a completely self-contained way, we can view the construction in the proof of this theorem as constructing a feasible point to the independent set program.

Indeed, in the proof of the above theorem, we associate to each round $t \in [T]$ an interval $[p_t, q_t]$ contained within the unit interval. The intersection of all these intervals induces a partition of the

unit interval into sub-intervals, each of which is labeled with a subset of rounds of $[T]$. In addition, by our construction of these intervals, two intervals $[p_s, q_s]$ and $[p_t, q_t]$ can intersect only if s and t are not adjacent in $\mathcal{S}$. Therefore, each sub-interval is actually labeled with some *independent set* I belonging to $\mathcal{I}(\mathcal{S})$. Taking the weight w_I to be the square root of the length of this interval, the analysis in the above proof implies that w_I form a feasible solution to the independent set program with value equal to $\sum_t \mu_t = \mathsf{UB}(\mathcal{S})$.

Appendix B. The lower bound $\mathsf{LB}(\mathcal{S})$ is not tight

In this appendix, we show that the gap between the value of the upper bound program $\mathsf{UB}(\mathcal{S})$ and the lower bound program $\mathsf{LB}(\mathcal{S})$ can grow without bound.

Theorem 17 *For any T, there exists a temporal feedback graph $\mathcal{S}$ on T rounds where $\mathsf{UB}(\mathcal{S})/\mathsf{LB}(\mathcal{S}) \geq \Omega(T^{1/4})$.*

Proof Consider the feedback graph $\mathcal{S}$ formed by batched learning setting where the time horizon T is divided into $\sqrt{T}$ batches of $\sqrt{T}$ rounds each (so, S_t only contains rounds $s < \lfloor t/\sqrt{T} \rfloor \sqrt{T}$).

Since each order in $\mathcal{S}$ contains at most $\sqrt{T}$ rounds, one feasible solution for the upper bound dual program is to set $\mu_t = T^{-1/4}$ for all $t \in [T]$, which implies $\mathsf{UB}(\mathcal{S}) \geq T^{3/4}$.

On the other hand, note that in the lower bound program, we have that $\sum_{t \in S_T} \varepsilon_t^2 \leq 1$. Since S_t contains at most T elements, this implies that $\sum_{t \in S_T} \varepsilon_t \leq \sqrt{T}$. But on the other hand, S_T can see all the rounds except the rounds in the very last batch (of which there are at most $\sqrt{T}$). So $\sum_{t \notin S_t} \varepsilon_t \leq \sqrt{T}$, and therefore $\mathsf{LB}(\mathcal{S}) \leq \sum_t \varepsilon_t \leq 2\sqrt{T}$.

It follows that $\mathsf{UB}(\mathcal{S})/\mathsf{LB}(\mathcal{S}) \geq T^{1/4}/2 = \Omega(T^{1/4})$, as desired. ■

Appendix C. Lemmas from probability and information theory

In this appendix, we establish some standard results from probability and information theory that we make use of in our proofs of our lower bounds.

C.1. Bound of KL-divergence for Bernoulli random variables

Lemma 18 *Fix a δ such that $0 \leq \delta \leq 1/4$. If $Q^+ = \mathsf{Bern}(1/2 + \delta)$, $Q^- = \mathsf{Bern}(1/2 - \delta)$, then $D(Q^- \parallel Q^+) \leq 12\delta^2$.*

Proof Computing the KL divergence explicitly, we have that:

$$\begin{aligned} D(Q^+ \parallel Q^-) &= \left(\frac{1}{2} - \delta\right) \log \frac{\frac{1}{2} - \delta}{\frac{1}{2} + \delta} + \left(\frac{1}{2} + \delta\right) \log \frac{\frac{1}{2} + \delta}{\frac{1}{2} - \delta} \\ &= 2\delta(\log(1 + 2\delta) - \log(1 - 2\delta)) \\ &\leq 2\delta \cdot (6\delta) = 12\delta^2. \end{aligned}$$

Here we have used the fact that $\log(1 + x) - \log(1 - x) \leq 3x$ for $x \in [0, 1/2]$. ■

C.2. From Gaussian to biased Bernoulli

Lemma 19 *Given a Gaussian variable $Y \sim \mathcal{N}(c,1)$ with $c > 0$, and Bernoulli variable $X = 1(Z \geq 0)$, we have*

$$\mathbb{P}(X=1) \geq 1/2 + c/(2\pi).$$

Proof Note that

$$\begin{aligned}\mathbb{P}(Z \geq 0) &= \frac{1}{2} + \int_0^c \frac{1}{\sqrt{2\pi}} e^{-(z-c)^2/2} \\ &\geq \frac{1}{2} + \int_0^c \frac{1}{\sqrt{2\pi}} e^{-c^2/2} \\ &\geq \frac{1}{2} + \frac{c}{\sqrt{2\pi}}(1 - c^2/2) \\ &\geq \frac{1}{2} + \frac{c}{2\pi}.\end{aligned}$$

This completes the proof. ■

C.3. KL divergence in Gaussian processes

Lemma 20 (KL divergence between two Gaussians)

$$D(\mathcal{N}(\mu_1,1)||\mathcal{N}(\mu_2,1)) = (\mu_1-\mu_2)^2/2.$$

Proof We can write

$$\begin{aligned}D(\mathcal{N}(\mu_1,1)||\mathcal{N}(\mu_2,1)) &= \mathbb{E}_{N(\mu_1,1)} \ln(\frac{e^{-(y-\mu_1)^2/2}}{e^{-(y-\mu_2)^2/2}}) \\ &= \mathbb{E}_{N(\mu_1,1)}((y-\mu_2)^2/2 - (y-\mu_1)^2/2) \\ &= (\mu_1-\mu_2)^2/2.\end{aligned}$$

■

Lemma 21 (Restatement of Lemma 16) *For a linearly shifted Brownian motions $L_t = \gamma t + B_t$, the KL divergence between the measures corresponding to L_t and B_t in the interval $[0,1]$ is equal to*

$$D\Big(Q\big(L[0:1]\big)||Q\big(B[0:1]\big)\Big) = \gamma^2/2.$$

Proof From the Girsanov theorem applied to the exponential martingale of the process γB_t, we have

$$\frac{dQ\big(B[0:1]\big)}{dQ\big(L[0:1]\big)} = e^{\gamma \int_0^1 dB_t - \frac{1}{2}\int_0^1 \gamma^2 dt},$$

which implies

$$D\Big(Q\big(L[0:1]\big)||Q\big(B[0:1]\big)\Big) = \mathbb{E}_{B[0:1]}\Big(\gamma \int_0^1 dB_t - \frac{1}{2}\int_0^1 \gamma^2 dt\Big) = \frac{\gamma^2}{2}.$$

■

Appendix D. Omitted proofs

D.1. Proof of Theorem 4

Proof We will show that the upper bound dual program arises from Lagrangifying the original upper bound program (and thus this equality follows as a consequence of strong duality).

We will begin by weakening the upper bound program (3) by replacing the strict equality $\sum_{c=1}^{N} \lambda_{c,t} = 1$ with the weak inequality $\sum_{c=1}^{N} \lambda_{c,t} \geq 1$. Note that this does not change the optimal value of the LP (as if this inequality is strict for a specific t, one can always improve the objective by decreasing one of the non-zero $\lambda_{c,t}$ while not altering any of the other constraints). By Lagrangifying these constraints, we have that

$$\mathsf{UB}(\mathcal{S}) = \min_{\boldsymbol{\lambda} \geq 0} \max_{\boldsymbol{\mu} \geq 0} \left[\sum_{c=1}^{N} \sqrt{\sum_{t \in C_c} \lambda_{c,t}^2} + \sum_{t=1}^{T} \mu_t \left(1 - \sum_{c=1}^{N} \lambda_{c,t} \right) \right]. \tag{17}$$

Now, note that in this convex program the objective is convex, all the constraints are affine, and the program is always feasible (for every round t there is at least one order containing it, namely the singleton order $\{t\}$). Therefore we can apply the theorem of strong duality (see Chapter 28 of Rockafellar (1970)), and interchange the order of minimum and maximum in (17).

$$\mathsf{UB}(\mathcal{S}) = \max_{\boldsymbol{\mu} \geq 0} \min_{\boldsymbol{\lambda} \geq 0} \left[\sum_{c=1}^{N} \sqrt{\sum_{t \in C_c} \lambda_{c,t}^2} + \sum_{t=1}^{T} \mu_t \left(1 - \sum_{c=1}^{N} \lambda_{c,t} \right) \right]. \tag{18}$$

We can in turn rewrite (18) in the following form:

$$\mathsf{UB}(\mathcal{S}) = \max_{\boldsymbol{\mu} \geq 0} \left[\left(\sum_{t=1}^{T} \mu_t \right) + \min_{\boldsymbol{\lambda} \geq 0} \sum_{c=1}^{N} \left(\sqrt{\sum_{t \in C_c} \lambda_{c,t}^2} - \sum_{t \in C_c} \lambda_{c,t} \mu_t \right) \right]. \tag{19}$$

Note that the terms in the internal sum pertaining to C_c only depend on the variables $\lambda_{c,t}$. We can therefore interchange the order of this sum and min in (19) and obtain:

$$\mathsf{UB}(\mathcal{S}) = \max_{\boldsymbol{\mu} \geq 0} \left[\left(\sum_{t=1}^{T} \mu_t \right) + \sum_{c=1}^{N} \min_{\boldsymbol{\lambda}_c \geq 0} \left(\sqrt{\sum_{t \in C_c} \lambda_{c,t}^2} - \sum_{t \in C_c} \lambda_{c,t} \mu_t \right) \right] \tag{20}$$

(where $\boldsymbol{\lambda}_p$ represents the $|C_c|$ variables of the form $\lambda_{c,t}$).

Let us now consider each of these inner optimization problems of the form

$$\min_{\boldsymbol{\lambda}_c \geq 0} \left(\sqrt{\sum_{t \in C_c} \lambda_{c,t}^2} - \sum_{t \in C_c} \lambda_{c,t} \mu_t \right).$$

Note that if we restrict the domain of $\boldsymbol{\lambda}_c$ to the subdomain where $\sum_{t \in C_c} \lambda_{c,t}^2 = R^2$, this expression is minimized when each $\lambda_{c,t}$ is proportional to μ_t. Specifically, it is minimized when

$$\lambda_{c,t} = \frac{\mu_t}{\sqrt{\sum_{t' \in C_c} \mu_{t'}^2}} \cdot R,$$

at which point the expression has value equal to

$$R\left(1-\sqrt{\sum_{t\in C_c}\mu_t^2}\right).$$

We therefore have two cases depending on our choice of $\boldsymbol{\mu}$:

- If $\sum_{t\in C_c}\mu_t^2 > 1$, then the value of this inner minimization problem is $-\infty$ (we can set R arbitrarily large).
- Otherwise, the value of this inner minimization problem is 0, attained when $R = 0$.

It follows that any feasible $\boldsymbol{\mu}$ (that causes the outer-maximization problem to have finite value) must satisfy $\sum_{t\in C_c}\mu_t^2 \le 1$ for each $c \in [N]$. If $\boldsymbol{\mu}$ is feasible, then the value of the inner expression is just $\sum_{t=1}^{T}\mu_t$. But note that this is precisely a description of the upper bound dual program (7). The result follows. ■

D.2. Proof of Lemma 5

Proof As in the proof of Theorem 4, we write the Lagrangified objective:

$$\mathcal{L}(\boldsymbol{\lambda},\boldsymbol{\mu}) = \sum_{c=1}^{N}\sqrt{\sum_{t\in C_c}\lambda_{c,t}^2} + \sum_{t=1}^{T}\mu_t\left(1-\sum_{c=1}^{N}\lambda_{c,t}\right) = \sum_{t=1}^{T}\mu_t + \sum_{c=1}^{N}\left(\sqrt{\sum_{t\in C_c}\lambda_{c,t}^2} - \sum_{t\in C_c}\mu_t\lambda_{c,t}\right).$$

From the properties of strong duality, we know that any optimal primal solution $\boldsymbol{\lambda}^*$ must satisfy $\mathcal{L}(\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*) \le \mathcal{L}(\boldsymbol{\lambda},\boldsymbol{\mu}^*)$ for any other $\boldsymbol{\lambda} \ge 0$ (not necessarily feasible). Assume to the contrary that the inequality (8) holds but we have that $\lambda_{c,t}^* > 0$ for some t. Then by Cauchy-Schwartz, we have that

$$\sum_{t\in C_c}\mu_t\lambda_{c,t} \le \sqrt{\sum_{t\in C_c}\mu_t^2}\cdot\sqrt{\sum_{t\in C_c}\lambda_{c,t}^2} < \sqrt{\sum_{t\in C_c}\lambda_{c,t}^2}.$$

It follows that if we construct $\boldsymbol{\lambda}'$ by taking $\boldsymbol{\lambda}^*$ and setting $\lambda'_{c,t} = 0$ for all $t \in [T]$, then $\mathcal{L}(\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*) > \mathcal{L}(\boldsymbol{\lambda}',\boldsymbol{\mu}^*)$, contradicting our assumption. Similarly, if the equality (9) holds but $\lambda_{c,t}^*$ is not proportional to μ_t, then if we set $\lambda'_{c,t} = \mu_t\cdot\sqrt{\left(\sum_{t'\in C_c}\lambda_{t',c}^2\right)/\left(\sum_{t'\in C_c}\mu_{t'}^2\right)}$, we again find that $\mathcal{L}(\boldsymbol{\lambda}^*,\mu^*) > \mathcal{L}(\boldsymbol{\lambda}',\mu^*)$, contradicting our assumption. ■

D.3. Proof of Theorem 9

Proof Let ε_t be an optimal solution to the lower bound program (11). Set $\gamma = 1/10$, and consider the distribution over loss vectors $\boldsymbol{\ell}$ induced by the process described above (where the adversary first selects B uniformly from $\{-1,1\}$, selects each X_t independently from $\mathsf{Bern}(1/2+B\gamma\varepsilon_t)$, and sets $\ell_t = (X_t, 1/2)$). We will show that when faced with a loss vector $\boldsymbol{\ell}$ sampled from this process,

any $\mathcal{S}$-learning algorithm $\mathcal{A}$ must incur $\Omega(\mathsf{LB}(\mathcal{S}))$ expected regret (from which it follows that there exists a specific loss vector $\boldsymbol{\ell}$ on which it incurs this much regret).

To prove this, note that when $\mathcal{A}$ is deciding the action x_t to take at time t, it can see the variables X_s for $s \in S_t$ – we will denote this set of random variables as X_{S_t}. If $B = -1$, these r.v.s are distributed according to a distribution $Q^-_{S_t}$, and if $B = 1$, these r.v.s are distribution according to a different distribution $Q^+_{S_t}$. We will upper bound the KL divergence $D(Q^-_{S_t} \parallel Q^+_{S_t})$; this in turn will allow us to apply Pinsker's inequality to upper bound the probability $\mathcal{A}$ can successfully distinguish between the cases $B = -1$ and $B = 1$, from which we can lower bound the regret incurred by $\mathcal{A}$.

Since $Q^-_{S_t}$ is a product distribution over $|S_t|$ independent Bernoulli r.v.s (and likewise for $Q^+_{S_t}$), by the chain rule for KL divergence we have that

$$D(Q^-_{S_t} \parallel Q^+_{S_t}) = \sum_{s \in S_t} D(Q^-_s \parallel Q^+_s),$$

where Q^-_s is simply the Bernoulli distribution $\mathsf{Bern}(1/2 - \gamma\varepsilon_s)$ and Q^+_s is the oppositely biased Bernoulli distribution $\mathsf{Bern}(1/2 + \gamma\varepsilon_s)$. We can compute that for these distributions, $D(Q^-_s \parallel Q^+_s) \leq 12\gamma^2\varepsilon_s^2$ (see Lemma 18 in the Appendix), so it follows that $D(Q^-_{S_t} \parallel Q^+_{S_t}) \leq \sum_{s \in S_t} 12\gamma^2\varepsilon_s^2$, which in turn is at most 0.12 (since $\gamma = 0.1$ and ε_s satisfy (11)).

By Pinsker's inequality, the total variation distance between $Q^-_{S_t}$ and $Q^+_{S_t}$ is therefore at most

$$\sqrt{(1/2)D(Q^-_{S_t} \parallel Q^+_{S_t})} \leq \sqrt{0.06} \leq 0.3.$$

Now, consider the probability $x_t \in [0, 1]$ that $\mathcal{A}$ plays the first action at round t. Let $x^*_t = 0$ if $B = 1$ and $x^*_t = 1$ if $B = -1$. Then the expected regret $\mathcal{A}$ incurs in this round against this adversary is $(X_t - 1/2)(x_t - x^*_t)$. In expectation, this is at least $\gamma\varepsilon_t \mathbb{E}[|x_t - x^*_t|]$. But (applying Le Cam's method), $\mathbb{E}[|x_t - x^*_t|] = (1/2)\mathbb{E}[x_t \mid B = 1] + (1/2)\mathbb{E}[1 - x_t \mid B = -1] = (1/2) + (1/2)(\mathbb{E}[x_t \mid B = 1] - \mathbb{E}[x_t \mid B = -1])$. Since x_t depends solely on X_{S_t} (which are drawn from $Q^-_{S_t}$ when $B = 1$ and from $Q^+_{S_t}$ when $B = -1$), this second term is at most the total variation distance in magnitude, and thus $\mathbb{E}[|x_t - x^*_t|] \geq 0.5 - 0.5 \cdot 0.3 \geq 0.3$.

It follows that the expected regret $\mathcal{A}$ incurs this round is at least $0.3\gamma\varepsilon_t \geq \varepsilon_t/100$. Over all rounds, the expected regret $\mathcal{A}$ incurs is therefore at least $\sum_t \varepsilon_t/100 = \mathsf{LB}(\mathcal{S})/100$, as desired. ■

D.4. Proof of Theorem 11

Proof The basic idea is similar to the proof of Theorem 9; the adversary again begins by sampling uniformly a random bit $B \in \{-1, +1\}$. For each independent set $I \in \mathcal{I}(\mathcal{S})$, the adversary samples a Gaussian random variable $Y_I \sim \mathcal{N}(\gamma B w_I^2, \gamma w_I^2)$ (for some fixed $\gamma > 0$ to be decided later). Then, for each time t, she defines the Gaussian random variable Z_t as the sum of Gaussian random variables Y_I of the independent sets I that include t (letting $\mathcal{I}_t(\mathcal{S})$ denoting this sub-collection of independent sets):

$$Z_t = \sum_{I \in \mathcal{I}_t(\mathcal{S})} Y_I.$$

Note that the mean and variance of Z_t is equal to the sum of the means and variances of the Y_I r.v.s, which are $\gamma B \sum_{I \in \mathcal{I}_t(\mathcal{S})} w_I^2$ and $\sum_{I \in \mathcal{I}_t(\mathcal{S})} w_I^2$, respectively. Finally, the adversary defines the loss at time t as $\ell_t = (X_t, \frac{1}{2})$, where $X_t = 1(Z_t \geq 0)$ is a Bernoulli random variable with some bias $\gamma\epsilon_t$.

By Lemma 19 (proved in Appendix C), we can bound $\epsilon_t \geq \frac{1}{\sqrt{2\pi}} \cdot \sqrt{\sum_{I \in \mathcal{I}_t(\mathcal{S})} w_I^2}$ from our mean / variance calculations for Y_i.

Let Q_t^- and Q_t^+ be the distribution of X_t given $B = -1$ and $B = +1$, respectively. For a subset of times $S \subseteq [T]$, let Q_S^- and Q_S^+ be the product of distributions Q_t^- and Q_t^+ for all $t \in S$, respectively. The first component of the proof, similar to the proof of Theorem 9, is to bound the KL divergence $D(Q_{\mathcal{S}_t}^- \parallel Q_{\mathcal{S}_t}^+)$. But by the data processing inequality, we can relate it to the distribution of the Y_I:

$$D(Q_{\mathcal{S}_t}^- \parallel Q_{\mathcal{S}_t}^+) \leq D\left(\bigotimes_{\mathcal{S}_t \cap I \neq \emptyset} Q^{-1}(I) \parallel \bigotimes_{\mathcal{S}_t \cap I \neq \emptyset} Q^{+1}(I) \right),$$

where $Q^{-1}(I)$ and $Q^{+1}(I)$ refer to the distribution of Y_I given $B = -1$ and $B = 1$, respectively. Now from the independence of the Y_I,

$$D\left(\bigotimes_{\mathcal{S}_t \cap I \neq \emptyset} Q^{-1}(I) \parallel \bigotimes_{\mathcal{S}_t \cap I \neq \emptyset} Q^{+1}(I) \right) = \sum_{\mathcal{S}_t \cap I \neq \emptyset} D(Q^{-1}(I) \parallel Q^{+1}(I)).$$

But defining P_s^- and P_s^+ to be the distributions of Z_s given $B = 1$ and $B = -1$, the data processing inequality implies

$$D(Q^-(I) \parallel Q^+(I)) \leq D(P^-(I) \parallel P^+(I)) = 4\gamma^2 w_I^2,$$

where the last equality follows from the formula for the KL divergence between two Gaussians, as we state in Appendix C. Hence,

$$D\left(\bigotimes_{\mathcal{S}_t \cap I \neq \emptyset} Q^{-1}(I) \parallel \bigotimes_{\mathcal{S}_t \cap I \neq \emptyset} Q^{+1}(I) \right) \leq 4\gamma^2 \sum_{\mathcal{S}_t \cap I \neq \emptyset} w_I^2.$$

Therefore, due to Pinsker's inequality, the total variation distance between $\bigotimes_{\mathcal{S}_t \cap I \neq \emptyset} Q^{-1}(I)$ and $\bigotimes_{\mathcal{S}_t \cap I \neq \emptyset} Q^{+1}(I)$ is at most

$$\mathrm{TV}\Big(\bigotimes_{\mathcal{S}_t \cap I \neq \emptyset} Q^{-1}(I) \parallel \bigotimes_{\mathcal{S}_t \cap I \neq \emptyset} Q^{+1}(I) \Big) \leq 2\gamma \sqrt{\sum_{\mathcal{S}_t \cap I \neq \emptyset} w_I^2} \leq 2\gamma.$$

where the last equality follows from feasibility of w for $\mathsf{ILB}(\mathcal{S})$. Then the regret incured by $\mathcal{A}$ is $(X_t - \frac{1}{2})(x_t - x_t^*)$ with expectation $\epsilon_t |x_t - x_t^*|$. But by Le Cam's method, since x_t is only a function of Y_I,

$$\begin{aligned}
\mathbb{E}[|x_t - x_t^*|] &= (1/2)\,\mathbb{E}[x_t \mid B = 1] + (1/2)\,\mathbb{E}[1 - x_t \mid B = -1] \\
&= (1/2) + (1/2)(\mathbb{E}[x_t \mid B = 1] - \mathbb{E}[x_t \mid B = -1]) \\
&\geq (1/2) - (1/2)\mathrm{TV}\Big(\bigotimes_{\mathcal{S}_t \cap I \neq \emptyset} Q^{-1}(I) \parallel \bigotimes_{\mathcal{S}_t \cap I \neq \emptyset} Q^{+1}(I) \Big) \geq (1/2) - \gamma.
\end{aligned}$$

Therefore, picking $\gamma = 1/4$, we get $\mathbb{E}[|x_t - x_t^*|] \geq 1/4$, which implies the expected regret at round t is at least $\mathbb{E}[\epsilon_t |x_t - x_t^*|] \geq \epsilon_t/4$. Summing over t and using Lemma 19, we get

$$\mathsf{Reg}(\mathcal{A}) \geq \gamma \frac{\sum_{t=1}^{T} \sqrt{\sum_{\mathcal{S}_t \cap I \neq \emptyset} w_I^2}}{(4\sqrt{2\pi})} = \frac{\sum_{t=1}^{T} \sqrt{\sum_{\mathcal{S}_t \cap I \neq \emptyset} w_I^2}}{16\sqrt{2\pi}} \geq \mathsf{ILB}(\mathcal{S})/50.$$

■