Efficient Learning & Decision Making in Environments with Structured Uncertainty

Dhruv Malik

November 2024
CMU-ML-24-112

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee
Aarti Singh (Chair)
Tom Mitchell
Akshay Krishnamurthy
Aldo Pacchiano

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by NSF-USDA AI institute for Resilient Agriculture (AIIRA) award number 2021-67021-35329.

Keywords: Reinforcement Learning, Bandits, Online Learning,
Robust Empirical Risk Minimization, CVXPY

Abstract

A prominent feature of modern machine learning is acting in environments with high degrees of uncertainty. Without enforcing structure on the environment or its type of uncertainty, efficiently making intelligent decisions is impossible. This thesis studies and formalizes structure under which machine learning systems can efficiently learn and make decisions that maximize our utility. It is split into two parts.

The first part focuses on research that has identified the presence of such structure in a variety of sequential decision making problems, particularly in reinforcement learning and online learning settings. This structure is motivated by real world problems. We present formal theoretical results which guarantee that such structure permits efficient learning. Various notions of efficiency are considered, including both statistical and computational.

The second part describes research on solving empirical risk minimization (ERM) problems, while being robust to uncertainty in the data. Under mild assumptions on the loss functions and uncertainty sets, we provide a framework via which a practitioner can specify and solve robust ERM problems. Notably, this can be done in just a few lines of code, in a manner that naturally follows the math.

Acknowledgements

I thank my advisors, thesis committee, collaborators,
teachers, friends and family.

Contents

1 Introduction

When we deploy a machine learning (ML) system in the real world, we must ensure that it acts reliably in the face of uncertainty or in a stochastic environment. This problem is paramount in both the *supervised learning* regime and also in the *sequential decision making* regime. For instance, in the sequential regime, ML algorithms are deployed in online environments (such as recommendation systems) where they make decisions based on interactive and highly stochastic user feedback. And in the supervised regime, it is common today to train massive ML models on immense amounts of stochastic and possibly adversarial data. In both settings, the uncertainty in the data and feedback must be accounted for.

It is thus natural to consider when and how we can train an ML algorithm to successfully and efficiently handle such uncertainty. A long line of work has studied this question in various forms, under various criteria for success, and under various notions of efficiency. Typically, one desires an algorithm which is both statistically and computationally efficient. In particular, an ideal algorithm should require an amount of data that is only a lower order polynomial function of the environment parameters, and should not only run in polynomial time but also be easily implementable. And in some cases, due to inherent difficulties in the problem, one trades off computation for a statistically efficient algorithm. Continuing in this line of work, this thesis focuses on the following high level questions.

In the sequential decision making regime, can we develop algorithms that exploit structure in the uncertain environment to more efficiently make decisions that maximize their cumulative reward? Towards answering this question, this thesis is organized as follows.

- Section 2 focuses on reinforcement learning with function approximation, and when statistically efficient RL is possible in this setting. A vast prior literature has shown that when we impose linearity on the environment, or restrict the complexity of the function approximation class, then statistically efficient RL is possible. However, such assumptions are unlikely to be satisfied in many practical scenarios, such as the simple video game benchmarks that are popular for deep RL. Motivated by these gaming benchmarks, we take an orthogonal approach, and consider structure that makes no linearity restrictions and allows for powerful neural network function approximation. We provide an algorithm which exploits this structure and is provably statistically efficient.

- Section 3 focuses on online learning. We are motivated by recommender systems, where there is an interactive feedback loop between the algorithm and the user, and we desire a formalism that can handle this interaction and is also efficiently solvable. On one end, the classical stochastic multi armed bandit is incapable of expressing such interaction, while on the other end, a generic reinforcement learning formulation

is expressive enough to handle this interaction but is impossible to solve efficiently. We study a suitable middle ground, which generalizes the multi armed bandit by incorporating a notion of memory, while also specializing the RL setting by ensuring the dynamics are deterministic and known. More concretely, we formalize structure where the algorithm's reward for playing an action is a function of the number of times that action was recently played. We provide an algorithm which is provably statistically efficient, and complement this with a lower bound which demonstrates the algorithm's near optimality.

- Section 4 continues the study of online learning initialized in Section 3. Motivated by tasks where a player requires some visuomotor calibration, we specialize the structure that was introduced in the previous chapter (while still strictly generalizing the stochastic multi armed bandit), by incorporating the additional assumption that there exists some action which yields optimal value after it is played repetitively. We find that this significantly larger and ostensibly more difficult class of problems can be solved with essentially the same statistical and computational efficiency as the classical stochastic multi armed bandit.

In the supervised learning regime, can we identify structure in possibly adversarial training data that allows us to efficiently train a more robust ML model? Towards answering this question, this thesis is organized as follows.

- Section 5 considers robust empirical risk minimization. We assume convex uncertainty sets that are guaranteed to contain the data and are independent from data point to data point. Our focus is on identifying a practical software framework, that for such uncertainty sets and convex loss functions, allows a practitioner to both specify and solve the robust ERM problem. Notably, we desire a methodology that is convenient and does not require the practitioner to be an expert. Our proposed framework leverages CVXPY, a popular Python-embedded modeling language for convex optimization, and enables a practitioner to specify and solve the robust ERM problem with complex uncertainty sets in just a few lines of code.

2 Sample Efficient Reinforcement Learning in Continuous State Spaces: A Perspective Beyond Linearity

The content of this section is based on [MPSL21].

Abstract

Reinforcement learning (RL) is empirically successful in complex nonlinear Markov decision processes (MDPs) with continuous state spaces. By contrast, the majority of theoretical RL literature requires the MDP to satisfy some form of linear structure, in order to guarantee sample efficient RL. Such efforts typically assume the transition dynamics or value function of the MDP are described by linear functions of the state features. To resolve this discrepancy between theory and practice, we introduce the Effective Planning Window (EPW) condition, a structural condition on MDPs that makes *no* linearity assumptions. We demonstrate that the EPW condition permits sample efficient RL, by providing an algorithm which provably solves MDPs satisfying this condition. Our algorithm requires minimal assumptions on the policy class, which can include multi-layer neural networks with nonlinear activation functions. Notably, the EPW condition is directly motivated by popular gaming benchmarks, and we show that many classic Atari games satisfy this condition. We additionally show the necessity of conditions like EPW, by demonstrating that simple MDPs with slight nonlinearities cannot be solved sample efficiently.

2.1 Introduction

Over the past decade, reinforcement learning (RL) has emerged as the dominant paradigm for sequential decision making in modern machine learning. During this time period, video games have served as popular means to benchmark the incremental improvement in state of the art RL. The Arcade Learning Environment (ALE), comprising a suite of classic Atari games, is an archetypical example of such a benchmark [BNVB13]. Agents trained by RL efficiently learn to surpass human level performance in such games [MKS+13, M+15, BPK+20].

Motivated by these empirical accomplishments, there has been a major thrust to theoretically characterize the conditions which permit sample efficient RL. A significant line of work has greatly advanced our understanding of the tabular RL setting, where the number of states is finite and relatively small [SJ19, PW21]. Sample efficiency bounds in this setting scale with cardinality of the state space. However, in practice this cardinality is often large or infinite. For instance, many gaming applications of RL, ranging in complexity from Atari to Dota, all have continuous state spaces [B+19]. These scenarios are handled in the function approximation setting [DLWZ19, DKWY20]. Here, each state is associated with a known feature, and one desires a sample efficiency bound that scales with the dimensionality of the features (instead of the cardinality of the state space).

To understand when RL is sample efficient in continuous state spaces, theoreticians make certain assumptions on the features or the underlying Markov Decision Process (MDP). A prominent assumption, which has appeared in various forms, is that the problem satisfies

some sort of *linear* structure. For instance, in the well studied linear MDP, the transitions and rewards are described by linear functions of the features [YW19, JYWJ20, YW20]. In particular, the transition probabilities at a state-action pair are defined by linear functions of the feature corresponding to that state-action pair. A weaker, but frequently occurring, form of this assumption is that value function of any policy is nearly linear [DKWY20, LSW20], or that the optimal value function is linear [DLWZ19, WAJ+21, WAS21]. Such linear structure is amenable to theoretical analysis, since it permits analysts to leverage the vast literature on supervised and online learning.

To obtain a holistic understanding of RL, examining such linear structure is certainly important. Nevertheless, it is unclear whether the aforementioned linearity conditions actually hold in practical scenarios. We illustrate this via a very simple example. Consider an MDP where there is a set of n states with the property that taking any action at one of these states leads to the same state. To cast this MDP in the aforementioned linear MDP setting, the dimensionality of the state features would have to scale linearly with n. This precludes the existence of algorithms that can solve this MDP with sample complexity independent of the cardinality of the state space.

Moreover, it has recently been shown both theoretically and empirically that the optimal value function and optimal policy can be very complex, even in ostensibly elementary continuous state space MDPs [DLY+20]. Since even powerful linear functions such as the Neural Tangent Kernel [JGH18, LL18a, ALS19] are significantly worse in terms of representation power and robustness than nonlinear neural networks [AZL19, LMZ20, AZL20, AL20], it is unclear whether such weaker linear functions can be used to approximate the value function or the underlying policy well.

Even in simple RL gaming benchmarks, there is no evidence that the aforementioned linearity assumptions hold. Indeed, nonlinear neural networks are the predominant means to approximate policies and value functions, when solving these games in practice. For instance, consider the Pong game from the ALE benchmark, which is depicted in Figure 1. In this game, the agent must use its paddle to prevent the ball from crossing a boundary, while playing against a pseudorandom opposing paddle. Despite the simplicity of Pong, state of the art methods solve this game using neural networks [MKS+13, M+15, BPK+20], and it is not apparent whether this game is linear in any sense.

This reveals a significant gap between the theory and practice of RL. In theory, one usually employs some sort of linearity assumption to ensure efficient RL. Yet, in practice, RL appears to succeed in domains which do not satisfy such linear structure. In an effort to resolve this discrepancy, we ask the following question:

Which (*non-linear*) structure is typical of popular RL domains, and how does this structure permit sample efficient RL?

This question underlies the analysis of our paper. Towards answering this question, we make the following contributions:

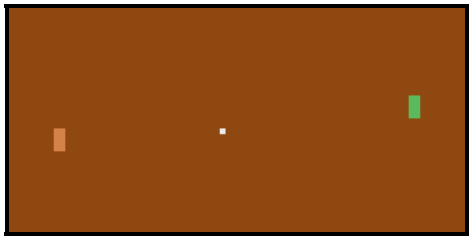


Figure 1: An image of the Atari Pong game. The green paddle must move up and down to hit the ball (the white dot) while playing against the opposing orange paddle.

- We propose the Effective Planning Window (EPW) condition, a structural condition for MDPs which goes beyond linearity. Indeed, this condition is compatible with neural network policies, and MDPs satisfying EPW can have highly nonlinear (stochastic) transitions. Informally, this condition requires the agent to *consistently plan C timesteps ahead*, for a value of C significantly smaller than the horizon length. We show that popular Atari benchmark games satisfy this condition.
- We provide a simple algorithm, which exploits the EPW condition to provably solve MDPs satisfying EPW. We prove the sample efficiency of our algorithm, and show that it requires a number of trajectories that is a lower order polynomial of the horizon length and other relevant problem dependent quantities.
- We argue that one must look beyond linear structure, and further motivate the study and necessity of conditions like EPW, by demonstrating that even slightly nonlinear MDPs cannot be solved sample efficiently.

2.2 Related Work

Linear Function Approximation. The majority of RL literature in the function approximation setting focuses on MDPs that satisfy some form of linear structure. A notable example is the linear MDP, where the transitions and rewards are described by linear functions of the state features [YW19, JYWJ20, YW20]. A weaker form of this assumption is that the value function for each policy is nearly linear [DKWY20, LSW20], or that the optimal value function is linear [DLWZ19, WAJ+21, WAS21]. We note that the algorithm of Weisz et al. [WAJ+21] requires a generative model, while we work in the standard episodic RL setting. As argued earlier, such linear assumptions are unlikely to hold true in practice. In our work, we eschew any sort of linearity assumption.

Nonlinear Function Approximation. Empirically, it is typical to use nonlinear function approximators such as neural networks [SLA+15, LFDA16]. But from a theoretical perspective, the understanding of nonlinear function approximation is limited. There is

prior work which studies the sample complexity of RL when using function approximation with nonlinear function classes [WVR13, JKA+17, VRD19, DPWZ20, DLMW20, WSY20, JLM21, WWDK21]. However, these works often are restricted to MDPs with deterministic transitions [WVR13, VRD19, DLMW20]. In this deterministic setting, an algorithm can repeatedly visit the same state and simply memorize an optimal path. By contrast, we focus on MDPs with *stochastic* transitions, as is typical in many Atari games. Here, an algorithm generally cannot visit the same state more than once, and must generalize beyond the trajectories it samples to learn something global. Moreoever, the aforementioned analyses of nonlinear function approximation typically make some stringent assumption on the complexity of the function class [JKA+17, DPWZ20, WSY20, JLM21, WWDK21]. Such complexity measures either cannot or are not known to handle neural networks. By contrast, our results place minimal restrictions on the function class, and we can handle nonlinear multilayer neural networks. In a different line of work, Dai et al. [DSL+18] study RL with nonlinear function approximators and provide a convergent algorithm for this setting. However, they do not precisely relate the quality of the solution found by their algorithm to the approximation error of the function class.

Linear Quadratic Regulator. To characterize the sample complexity of RL in continuous state spaces, a different line of work investigates the linear quadratic regulator (LQR) [FGKM18, DMM+19, MPB+20]. Here, the transition dynamics of the MDP are assumed to be noisy linear functions of the state and action, and the rewards are quadratic functions of the state and action. We remark that in this setting, the action space is continuous. By contrast, we exclusively study MDPs with finite action spaces, since these are most typical in the RL video game domains that motivate our paper.

2.3 Problem Formulation

2.3.1 Problem Statement

Notation & Preliminaries. We use the notation $[n]$ to denote $\{0, 1 \dots n-1\}$ for any positive integer n. Recall that an undiscounted, finite horizon MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, R, H)$ is defined by a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition function $\mathcal{T}$ which maps from state-action pairs to a probability density defined over states, a reward function R which maps from state-action pairs to non-negative real numbers, and a finite planning horizon H. Throughout our paper, we assume that $\mathcal{S} \subseteq \mathbb{R}^d$ and $\mathcal{A}$ is a finite set. Without loss of generality, we assume a single initial state s_0. For simplicity, we assume that $\mathcal{S}$ can be partitioned into H different levels. This means that for each $s \in \mathcal{S}$ there exists a unique $h \in [H]$ such that it takes h timesteps to arrive at s from s_0. We say that such a state s lies on level h, and denote $\mathcal{S}_h$ to be the set of states on level h. Note this assumption is without loss of generality, and our results apply to generic MDPs which cannot be partitioned into levels. This is because we can always make the final coordinate of each state encode the

number of timesteps that elapsed to reach the state. Taking any action from level $H-1$ exits the game. The notation $\|x\|_2$ denotes the Euclidean norm of x.

A policy maps each state to a corresponding distribution over actions. In practice, one typically uses a policy that is parameterized by parameters belonging to some set $\Theta \subseteq \mathbb{R}^k$. We study such policies, and use $\pi(\theta)$ to denote the policy induced by using parameter $\theta \in \Theta$. When discussing a policy which is not parameterized, we simply use π to denote the policy. We use $\pi_s^a(\theta)$ to denote the probability of taking action a at state s when using the policy $\pi(\theta)$. We use $\pi(\Theta)$ to denote $\{\pi(\theta) \text{ s.t. } \theta \in \Theta\}$, which is the set of feasible policies and defines our policy class. Given a vector $\overline{\theta} \in \Theta^H$, we let $\pi(\overline{\theta})$ denote the policy which executes $\pi(\overline{\theta}_h)$ at for any state lying on level $h \in [H]$, where $\overline{\theta}_h$ denotes the h^{th} entry of $\overline{\theta}$. The value of a policy $\pi(\theta)$ in a (stochastic) MDP $\mathcal{M}$ when initialized at state s is denoted $V_{\mathcal{M}}^s(\pi(\theta))$. It is given by $V_{\mathcal{M}}^s(\pi(\theta)) = \mathbb{E}\left[\sum_{h=\text{level}(s)}^{H-1} R(s_h, a_h) \mid \pi(\theta)\right]$, where the expectation is over the trajectory $\{(s_h, a_h)\}_{h=\text{level}(s)}^{H-1}$ conditioned on the fact that the first state in the trajectory is s. Given an accuracy ϵ and failure probability tolerance δ, the goal of RL is to find a policy π which satisfies $V_{\mathcal{M}}^{s_0}(\pi) \geq \max_{\pi'} V_{\mathcal{M}}^{s_0}(\pi') - \epsilon$ with probability at least $1-\delta$.

Query Model. We adopt the standard episodic RL setup. During each episode, an agent is allowed to interact with the MDP by starting from s_0, taking an action and to observe the next state and reward, and repeating. The episode terminates after the agent takes H actions, and the next episode starts at s_0. The agent thus takes a single trajectory in each episode, and the total query complexity of the agent is measured by the total number of trajectories. Given a desired solution accuracy ϵ and failure probability tolerance δ, we are interested in algorithms which can successfully solve an MDP using a number of trajectories that is at most polynomial in H, $|\mathcal{A}|$, d, k, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$. If an algorithm provably accomplishes this, we call such an algorithm *sample efficient* or *tractable*. Notably, such algorithms cannot depend on the (possibly uncountable) number of states.

Without any assumptions on the MDP, approximating an optimal policy is intractable. To permit sample efficient RL, prior theoretical work has often assumed that MDP satisfies some form of linear structure. For instance, the transition or value function might be described by a linear function of the states. However, it is well documented that RL is empirically successful in highly nonlinear domains [SLA+15, LFDA16]. We aim to bridge this gap between theory and practice. We now formally state the problem that we consider throughout our paper.

Our goal is to present nonlinear characteristic conditions which permit sample efficient RL, and argue that these conditions are satisfied in practice by popular RL domains.

2.3.2 Effective Planning Window Condition

We first state basic conditions that are satisfied by most RL problems encountered in practice. We will later refine these to obtain our Effective Planning Window (EPW) condition, and then show that EPW enables sample efficient RL.

Let us begin by observing that in practice, the policy class $\pi(\Theta)$ typically satisfies some mild regularity assumptions. We formalize this in the following condition.

Condition 1 (Regular Policy Class). *A policy class $\pi(\Theta)$ is said to be Regular when:*

(a) ***Bounded Domain.*** *There exists $B > 0$ such that each $\theta \in \Theta$ satisfies $\|\theta\|_2 \leq B$.*

(b) ***Lipschitz Continuous Policies.*** *There exists $\phi > 0$ such that for any $\theta, \theta' \in \Theta$ and any $(s, a) \in \mathcal{S} \times \mathcal{A}$, we have $|\pi_s^a(\theta) - \pi_s^a(\theta')| \leq \phi \|\theta - \theta'\|_2$.*

We stress that this is a very mild condition, and places minimal restrictions on $\pi(\Theta)$. Indeed, a policy parameterized by a multi-layer neural network with a nonlinear activation function satisfies this condition [FRH+19]. Using this condition on the policy class, we now introduce the following Generic Game condition. As we will discuss in the sequel, many popular gaming RL benchmarks such as Atari games satisfy this condition.

Condition 2 (Generic Game). *An MDP and Regular policy class pair $(\mathcal{M}, \pi(\Theta))$ form a Generic Game if:*

(a) ***Failure States.*** *There is a set of failure states $\mathcal{F} \subset \mathcal{S}$, and taking any action from a state in $\mathcal{F}$ exits the game.*

(b) ***Complete Policy Class.*** *There exists some $\theta^\star \in \Theta$ such that executing $\pi(\theta^\star)$ from s_0 arrives at some state in $\mathcal{S}_{H-1} \setminus \mathcal{F}$ almost surely[1]. We define $\mathcal{S}^\star$ to be the set of all states $s \in \mathcal{S} \setminus \mathcal{F}$ such that executing $\pi(\theta^\star)$ from s reaches $\mathcal{S}_{H-1} \setminus \mathcal{F}$ almost surely. If a state lies in $\mathcal{S}^\star$ we call it a safe state.*

(c) ***Binary Rewards.*** *For any state $s \in \mathcal{S}_{H-1} \setminus \mathcal{F}$ and any $a \in \mathcal{A}$, $R(s, a) = 1$. For any other state s and any $a \in \mathcal{A}$, $R(s, a) = 0$.*

A few comments are in order. Note that $\mathcal{F}$ is essentially used to describe states where the agent has lost the game. Also, observe that in Generic Games, an optimal policy is one that arrives at a non-failure state in level $H - 1$ almost surely. Hence $\pi(\theta^\star)$ is indeed an optimal policy.

Let us now describe how popular Atari games can be cast as Generic Games. Recall the famous Pong game depicted in Figure 1, which is a part of the ALE benchmark [BNVB13].

[1]The Generic Game condition can also be defined in the case when this property of $\theta^\star$ holds true with probability exponentially large in H, as would occur when using a softmax policy class. Our results hold true when using this notion of a Generic Game. We focus on the almost sure case to avoid complicating notation.

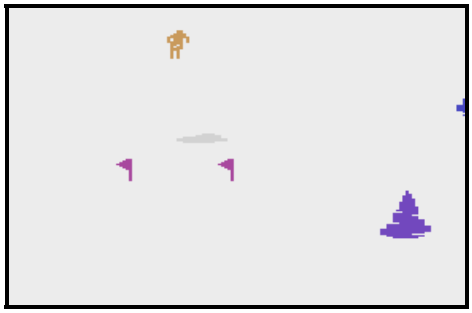


Figure 2: An image of the Atari Skiing game. The skier must move through flagged checkpoints.

In this game, an RL agent must learn to move the paddle up and down to hit the ball and prevent it from crossing its boundary. Note that in the context of RL, this is a single player game, since the opposing paddle hits the ball back according to a pre-specified stochastic decision rule (which is not trained). The agent loses the game if the ball crosses its own boundary, and wins the game if it hits the ball past the opposing paddle. Another game in the ALE benchmark is the Skiing game, depicted in Figure 2. Here, the agent must move the skier through a series of randomly appearing flagged checkpoints, which appear frequently over a long time horizon. The skier receives a penalty each time it misses a checkpoint.

We claim that an Atari game like Pong or Skiing, together with a neural network policy class, satisfy the Generic Game condition. The first two conditions of Generic Games are easy to verify. Note that the states in Pong (resp. Skiing) are images, so $\mathcal{F}$ includes any state where the ball has crossed the agent's boundary (resp. where the skier has missed a prior checkpoint), since this corresponds to the agent failing to complete the game. It is well known that Atari can be solved using a neural network policy [M+15], so a policy class parameterized by neural networks is indeed complete.

To ensure that Pong and Skiing satisfy the third condition, we need to design an appropriate binary reward function. For Pong, this is handled by redefining $\mathcal{F}$ to include any state $s \in \mathcal{S}_{H-1}$ where the ball has not crossed the opposing paddle. Similarly for Skiing, this is done by ensuring $\mathcal{F}$ includes any state where the skier has already missed a checkpoint. Then one can simply assign a reward of 1 to any state in $\mathcal{S}_{H-1} \setminus \mathcal{F}$, and 0 to all other states, as required by the Generic Game condition. Hence, playing optimally in this Generic Game framework ensures that the ball has moved past the opposing paddle, or that the skier has made all checkpoints, corresponding to winning the game.

The aforementioned reward design is an example of reward shaping, which is unavoidable in RL and ubiquitous in practice [HMMA+17]. Nevertheless, we stress that the reward function we described above is very similar to the reward function that practitioners already use. Concretely, in Pong one typically assigns a reward of 1 if the ball has moved past the opposing paddle, a reward of -1 if the ball has moved past the agent's paddle, and

a reward of 0 otherwise [BNVB13]. Similarly, in Skiing, the skier receives reward at the end of the game, in proportion to the number of checkpoints it has cleared [BPK+20]. Our reward function thus requires no more effort to design than the reward functions already in use, since they require the same information, and these are identical in spirit.

Beyond Pong and Skiing, other Atari games (and other similarly themed video games) can be cast in the Generic Game framework. In Appendix A.1, we describe this reduction for the Atari games Tennis and Journey Escape, and also for the more complex RL gaming benchmark CoinRun [CKH+19].

Let us make one more remark about the binary reward structure of Generic Games. There are many games which naturally have a set of goal states, and these immediately can be described as Generic Games. Examples include Pong, Tennis and CoinRun (latter two are described in Appendix A.1). More generally, however, EPW applies to many games which do not naturally have a binary reward structure. As we discussed, Skiing can be cast in the EPW framework with binary rewards, by ensuring that $\mathcal{F}$ includes any state where the skier has missed a checkpoint. However, in certain scenarios, one may be satisfied with only collecting a large fraction of the checkpoints in Skiiing, or more generally, obtaining a large (but not perfect) score in games where one continually collects small rewards. We emphasize that such scenarios can be cast in our Generic Game framework. For instance, in Skiing if checkpoints arrive roughly every x timesteps, and we desire to give a reward of 1 for each checkpoint collected, then we can design $\mathcal{F}$ to include any state at timestep t where the agent has not collected $\Omega(t/x)$ reward thus far. Similar reductions apply to other games where one continually needs to collect a small amount of reward at regular intervals.

Does the Generic Game condition permit sample efficient RL? Unfortunately, there exist Generic Games where the MDP is only slightly nonlinear, but even approximating an optimal policy sample efficiently is impossible. We later show this formally in Proposition 1. So we must further restrict this class of games. In order to refine our notion of a Generic Game, we first state a useful definition.

Definition 1 (x-Ancestor). *Given a Generic Game $(\mathcal{M}, \pi(\Theta))$, consider any $h \in [H]$ and any state $s' \in \mathcal{S}_h$. A state $s \in \mathcal{S}$ is an x-ancestor of s', if $s \in \mathcal{S}_{\max\{0,h-x\}}$ and there exists some $\theta \in \Theta$ such that following $\pi(\theta)$ from s will reach s' with nonzero probability.*

We are now in a position to formally state our Effective Planning Window (EPW) condition, which refines our notion of Generic Games. For the statement of the condition, recall our notion of $\mathcal{S}^\star$, which was defined in the Generic Game condition.

Condition 3 (Effective Planning Window). *A Generic Game $(\mathcal{M}, \pi(\Theta))$ satisfies the Effective Planning Window condition with parameter C if there exists $C \in [H]$ such that the following holds. Consider any $s' \in \mathcal{S} \setminus \mathcal{F}$. If s is a C-ancestor of s', then $s \in \mathcal{S}^\star$.*

Before examining RL benchmark games in the context of this condition, a few comments about the condition itself are in order. The quantity C ensures that any C-ancestor of a

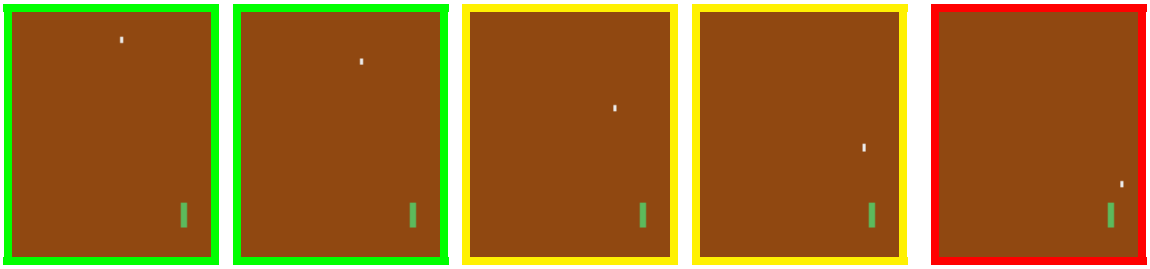


Figure 3: Five states from the Pong game. Here we let $t'' < t - C < t' < t$, and the ball is progressively moving towards the lower right corner. At timesteps t', t, the paddle has not lost the game. However, it does not have enough time to react and reach the ball in time. At timestep $t+1$ the game is over. At timesteps $t'', t - C$, the paddle has enough time to react and reach the ball.

non-failure state is a safe state. So if an agent is at timestep t and the game is not over, then at timestep $t - C$ it was in a state from where it could have achieved the highest reward possible in the MDP (if it took the correct sequence of actions). For the purposes of RL, this effectively means that at each timestep, the agent must consistently plan over the next C timesteps instead of the entire horizon length H. Thus, when C is small or a constant, then it is reasonable to believe that sample efficient RL is possible.

Of course, any Generic Game satisfies the EPW condition for a choice of $C = H - 1$. However, many popular RL benchmark games satisfy the EPW property with a value of C that is much smaller than H. Informally, the C quantity is the amount of time required by the agent to successfully *react* to scenarios in the game (without losing). Let us understand this more deeply in the Pong and Skiing games.

In Pong, after the opposing paddle hits the ball, the agent must react to the trajectory of the ball and adjust its position accordingly to hit it. If it takes too long to react before it starts adjusting its position, then it will be unable to reach the ball in time. We depict this in Figure 3. More formally, assume that at timestep t the paddle has not lost the game and the ball is moving towards its boundary. At timestep t, the ball may be too close to the boundary, and so the agent will not not have enough time to move its paddle fast enough in order to reach the ball in time. However, at timestep $t - C$ the ball is further away from the boundary, so the agent has enough time to move its paddle appropriately in order to react, reach the ball and hit it back. So at timestep $t - C$ the agent lies in a safe state in $\mathcal{S}^\star$, since it has enough time to adjust its paddle and hit the ball back, and hence play optimally. Notably, if we let C' be the number of timesteps it takes for the ball to traverse from one end of the board to the other, then $C \leq C'$. Hence, when H is large and the agent needs to control the paddle for many rounds, then C is a constant independent of H.

Similarly, in the Skiing game, the skier must react to the location of the oncoming checkpoint, and adjust its position accordingly. Formally, assume at timestep t a checkpoint

is oncoming. In such a scenario, as depicted in Figure 4, the skier might be too far from the checkpoint in order to actually clear it (even if it moves directly towards the checkpoint). However, at timestep $t - C$ the skier has enough time to adjust its position in order to clear the checkpoint. Hence at timestep $t - C$, the skier is in a safe state in $\mathcal{S}^\star$ since it can play optimally from this state. Again, if we let C' be the number of timesteps it takes for a skier to move from the left edge of the screen to the right edge, then $C \leq C'$. Hence, when H is large and there are many checkpoints to be cleared, then C is a constant independent of H, as previously observed in Pong.

Beyond Pong and Skiing, other Atari games satisfy the EPW condition, with a constant value of C. We demonstrate this for the Atari games Tennis & Journey Escape in Appendix A.1. In Appendix A.1 we additionally show that more complex games, such as CoinRun [CKH+19], also satisfy EPW with a small value of C. We stress that EPW is orthogonal to linearity. Indeed, there are MDPs satisfying linearity but not EPW, and vice versa. We conclude this section by highlighting two important aspects of the EPW condition.

The Magnitude Of C. We treat C as a constant that is independent of and much smaller than H. This is certainly reasonable given our above discussion. So an algorithm incurring $\mathcal{O}(|\mathcal{A}|^C)$ sample complexity is efficient. Furthermore, as we discuss later, there exist EPW games where $\Omega(|\mathcal{A}|^C)$ sample complexity is necessary to solve the game.

The Challenge Of Solving EPW Games. We note that a deterministic EPW game is straightforward to solve, since an agent can just try each of the $|\mathcal{A}|^C$ trajectories when it is at level h, to discover which trajectories do not lead to $\mathcal{F}$. In such a case, an agent can simply memorize a good path. However, when transitions are stochastic (as in Atari), the agent cannot simply try each trajectory to memorize one that does not lead to $\mathcal{F}$. This is because in general stochastic MDPs, a finite sample algorithm might only visit any given state at most once. Instead, the algorithm must *learn* and generalize beyond the trajectories it samples, to learn something global about the MDP. Furthermore, we emphasize that stochastic EPW games *cannot* be solved as simply as just splitting the horizon H into H/C distinct planning windows, and then solving these planning problems independently of each other. Instead, the key difficulty is that the agent must *consistently* plan C timesteps ahead. By this, we mean that just because an agent has arrived at a non-failure state at time t, does not imply that at time $t+1$ it is guaranteed to avoid $\mathcal{F}$. Indeed, if we execute a policy and the resulting trajectory ends in a failure state after t timesteps, then it is unclear at which of the prior timesteps $\{t - C \dots t - 1\}$ that we took an incorrect action. And we cannot rollback to timestep $t - C$ and rerun the same trajectory to discover when we made a mistake. This complicates the design of efficient algorithms for this setting.

$s_{t''} \in \mathcal{S}^\star$ $s_{t-C} \in \mathcal{S}^\star$ $s_{t'} \in \mathcal{S} \setminus (\mathcal{S}^\star \cup \mathcal{F})$ $s_t \in \mathcal{S} \setminus (\mathcal{S}^\star \cup \mathcal{F})$ $s_{t+1} \in \mathcal{F}$

Figure 4: Five states from the Skiing game. Here we let $t'' < t - C < t' < t$, and the skier is progressively moving towards the checkpoint. At timesteps t', t, the skier has not lost. However, it does not have enough time to react and adjust its position to reach the checkpoint in time. At timestep $t+1$ the game is over. At timesteps $t'', t - C$, the skier has enough time to react and reach the checkpoint.

2.4 Main Results

We now turn to our main results. Before diving into the details, let us provide a brief overview for the results. Our main contribution is an algorithm which sample efficiently solves games satisfying the EPW condition. We prove the efficiency of this algorithm and discuss the role of the EPW condition in permitting sample efficient RL. We then further motivate the study of conditions like EPW, as opposed to the study of linear structure, by proving a lower bound which shows that Generic Games with even slight nonlinearities cannot be solved sample efficiently. Hence, we must look beyond linear structure to characterize when sample efficient RL is possible in realistic domains. With this outline in mind, let us now formally present our main results.

Our primary contribution is an algorithm which exploits the EPW condition to sample efficiently find an optimal policy. Our algorithm is defined in Algorithm 1. For this method, recall our notation that for a vector $\overline{\theta} \in \Theta^H$, we let $\pi(\overline{\theta})$ denote the policy which executes $\pi(\overline{\theta}_h)$ at each $h \in [H]$, where $\overline{\theta}_h$ denotes the h^{th} entry of $\overline{\theta}$. We make the basic assumption that there exists $\theta_{\text{rand}} \in \Theta$ such that $\pi(\theta_{\text{rand}})$ maps each state to the uniform distribution over $\mathcal{A}$.

In our main result Theorem 1, we bound the sample complexity of Algorithm 1 and demonstrate that it efficiently finds a near optimal policy for any EPW game. Before stating the theorem, let us discuss Algorithm 1 and provide some intuition for this method. Recall that in Generic Games, a near optimal policy avoids failure states with high probability. The method initializes $\overline{\theta}(0)$ to be the parameter vector which induces a uniformly random policy regardless of the state (Line 2). It then incrementally updates $\overline{\theta}(t)$, in a fashion that ensures $\pi(\overline{\theta}(t))$ avoids failure states with high probability for each $t' \leq t$ (Line 7). Ultimately it returns the parameter vector $\overline{\theta}(H-1)$ (Line 9). More concretely, at each timestep t in the inner loop, the algorithm samples n trajectories from the policy $\pi(\overline{\theta}(t))$ that it has constructed thus far (Line 4). Via these sampled trajectories it defines the empirical loss $\widehat{L}_t$, as shown in Eq. (1). Intuitively, $\widehat{L}_t$ penalizes those parameters θ such

Algorithm 1

Inputs: MDP $\mathcal{M}$, policy class $\pi(\Theta)$, sample size n
Initialize $\overline{\theta}(0) = [\theta_{\text{rand}}, \theta_{\text{rand}} \dots \theta_{\text{rand}}] \in \Theta^{H-1}$
for $t \in \{0, 1 \dots H-2\}$ **do**
    Sample n trajectories $\{\tau_i\}_{i=1}^{n} \sim \pi(\overline{\theta}(t))$, where each trajectory $\tau_i = \{(s_{i,h}, a_{i,h})\}_{h=0}^{H-1}$
    Define the loss function $\widehat{L}_t : \Theta \to \mathbb{R}$ as

$$\widehat{L}_t(\theta) = |\mathcal{A}|^{C+1} \cdot \left[\frac{1}{n} \sum_{i=1}^{n} \left(\mathbb{I}_{s_{i,t+C+1} \in \mathcal{F}} \prod_{j=0}^{C} \pi_{s_{i,t+j}}^{a_{i,t+j}}(\theta) \right) \right] \tag{1}$$

    Minimize $\widehat{L}_t$ over Θ to obtain

$$\widehat{\theta}_t \in \operatorname*{argmin}_{\theta \in \Theta} \widehat{L}_t(\theta) \tag{2}$$

    Define $\overline{\theta}(t+1) = [\widehat{\theta}_0, \widehat{\theta}_1 \dots, \widehat{\theta}_{t-1}, \widehat{\theta}_t, \theta_{\text{rand}}, \dots, \theta_{\text{rand}}]$
end for
return $\overline{\theta}(H-1)$

that executing $\pi(\theta)$ over timesteps $\{t, t+1 \dots t+C\}$ arrives at a failure state with high probability. This intuition suggests that a minimizer $\widehat{\theta}_t$ of $\widehat{L}_t$, as defined in Eq. (2), will assign low probability to trajectories ending in a failure state when playing $\widehat{\theta}_t$. The quantity θ_t then defines the $t+1^{th}$ entry of $\overline{\theta}(t+1)$. Note that $\overline{\theta}(t+1)$ agrees with $\overline{\theta}(t)$ in its first t entries.

The form of the loss $\widehat{L}_t$ suggests that a pracitioner needs to know the exact value of C to use Algorithm 1. This may be a stringent requirement in practice. We emphasize that any upper bound $C' \geq C$ can be used in place of C in the definition of $\widehat{L}_t$. As discussed in Section 2.3.2, such an upper bound is easy to find in gaming domains where we expect the EPW condition to hold. For example, in Pong we can play the game manually (in OpenAI Gym), and observe the number of timesteps it takes the paddle to traverse its side. This yields $C \leq 15$, while $200 \leq H$. As to how this affects the sample complexity of the method, one can simply substitute C' for C in the bound provided in our main result Theorem 1.

Before formally presenting Theorem 1, a remark on the computational requirements of Algorithm 1 is imperative. The method requires oracle access to a minimizer $\widehat{\theta}_t$ of the loss $\widehat{L}_t$, which in turn is defined by the policy class $\pi(\Theta)$. In our paper, we impose minimal assumptions on $\pi(\Theta)$. Our motivation for this choice is that in practice, it is most common to parameterize a policy via a multi-layer neural network with nonlinear activation function. Beyond our extremely mild Regularity condition, is unclear which (if any) desirable properties such a policy class satisfies. Hence, for a worst case Regular policy class $\pi(\Theta)$, obtaining even an approximation of $\widehat{\theta}_t$ could be extremely computationally

intractable. Nevertheless, we stress that in both theory and practice, stochastic gradient descent and its variants have been shown to efficiently find global minima of loss functions parameterized by neural networks [LL18b, AZLS19, DLL+19]. Furthermore, as we show in our proofs, the function $\widehat{L}_t$ is Lipschitz continuous with a tolerable Lipschitz constant. Hence by Rademacher's Theorem, $\widehat{L}_t$ is differentiable almost everywhere, and it is reasonable to minimize $\widehat{L}_t$ via the stochastic gradient type methods that are popular for minimizing complex neural network losses. We also remark that it is fairly common in the RL literature to assume access to a computational oracle, when studying sample complexity [AHK+14, DKJ+19, AKKS20, MHKL20].

We now formally state Theorem 1, our main result. This theorem bounds the sample complexity required by Algorithm 1 to find a near optimal policy for any EPW game.

Theorem 1. *Fix error tolerance $\epsilon > 0$ and failure probability tolerance $\delta > 0$. Given any $(\mathcal{M}, \pi(\Theta))$ satisfying the EPW condition, and sample size*

$$n = \frac{4H^2|\mathcal{A}|^{2C+2}}{\epsilon^2}\left(\log\left(\frac{2H}{\delta}\right) + k\log\left(1 + \frac{32H|\mathcal{A}|^{C+1}C\phi B}{\epsilon}\right)\right),$$

Algorithm 1 outputs $\overline{\theta}$ satisfying

$$V_{\mathcal{M}}^{s_0}(\pi(\overline{\theta})) \geq V_{\mathcal{M}}^{s_0}(\pi(\theta^\star)) - \epsilon$$

with probability at least $1 - \delta$.

A formal proof for the theorem is presented in Appendix A.2. A few comments are in order. Note that Algorithm 1 samples n trajectories at each timestep in its inner loop. Recall from the definition of our query model in Section 2.3.1, that we measure the total sample complexity by the total number of trajectories sampled. Hence, the total sample complexity of Algorithm 1 scales as $\mathcal{O}(\frac{H^3|\mathcal{A}|^{2C+2}k}{\epsilon^2})$, where we have discarded logarithmic factors. Observe that the total sample complexity depends only logarithmically on the failure probability tolerance δ, the bound B on the Euclidean norm of Θ and the Lipschitz constant ϕ of the policy. It is also worth noting that the sample complexity has *no* explicit dependence on the dimension d of states. However, it does depend linearly on the dimension k of the policy parameter space Θ, and of course in general k will scale with d. We note that in practical RL, one typically employs a shallow neural net, with only two or three layers. k is relatively small in this regime, in contrast to NLP or vision tasks where models are much larger.

Observe that the sample complexity bound in Theorem 1 has an exponential dependence on C. Recall that in our framework, as motivated at the end of Section 2.3.2, C is a constant, so our algorithm is indeed efficient. Nevertheless, we remark that as a direct corollary of the work of Du et al. [DKWY20], this exponential dependence on C cannot be improved by a better algorithm or sharper analysis.

More generally, in the context of existing literature, we provide some intuition for why the EPW condition permits efficient learning. A major issue that hinders sample efficient RL, in both theory and practice, is that an agent must plan over the entire horizon H. Roughly speaking, at each timestep the agent can choose any of $|\mathcal{A}|$ actions, so the total sample complexity required to plan over H timesteps scales as $\Omega(|\mathcal{A}|^H)$. This is a recurrent theme in the RL literature, and various prior works have shown that even when the MDP has some non-trivial structure, in the worst case such a scaling is unavoidable [DKWY20, MPSL21]. Assuming that the MDP satisfies significant *linear* structure, is one way to avoid this difficulty. By contrast, we are able to avoid this difficulty while making *no* linearity assumptions. Instead, the EPW condition guarantees that an agent needs only to plan over C timesteps. Hence we exchange the worst case $\Omega(|\mathcal{A}|^H)$ scaling for the benign $\mathcal{O}(|\mathcal{A}|^{2C+2})$ scaling.

We believe that the EPW condition (or other conditions that are similar in spirit) is the *correct* condition for characterizing when sample efficient RL is possible, at least in RL domains like video games. By contrast, the linearity assumptions which prominently appear in prior literature, in addition to lacking clear empirical justification, are quite brittle. To demonstrate this, we leverage prior work to show the existence of Generic Games which have only slight nonlinearities, yet cannot be solved sample efficiently. Before we state this lower bound, we recall two standard definitions.

Definition 2 (Optimal Value Function). *The optimal value function $V^\star_{\mathcal{M}} : \mathcal{S} \to \mathbb{R}$ of an MDP $\mathcal{M}$ is defined as $V^\star_{\mathcal{M}}(s) = V^s_{\mathcal{M}}(\pi^\star)$, where $\pi^\star$ is an optimal policy of $\mathcal{M}$.*

Definition 3 (Softmax Linear Policy). *For an MDP $\mathcal{M}$, a softmax linear policy $\pi(\boldsymbol{\theta})$ is parameterized by $\boldsymbol{\theta} \in \mathbb{R}^{|\mathcal{A}| \times d}$. Letting $\boldsymbol{\theta}_i$ denote the i^{th} row of $\boldsymbol{\theta}$, the policy $\pi(\boldsymbol{\theta})$ satisfies*

$$\pi_s^{a_i}(\boldsymbol{\theta}) = \frac{\exp(s^T \boldsymbol{\theta}_i)}{\sum_{a_j \in \mathcal{A}} \exp(s^T \boldsymbol{\theta}_j)}.$$

Briefly, the optimal value function takes as input a state and outputs the optimal value that one can achieve from that state. A softmax linear policy is parameterized by a matrix, whose rows are in correspondence with actions. Given a state, the softmax linear policy outputs a probability distribution over actions, where the probabilities are exponentially weighted linear functions of the state.

We now demonstrate the existence of Generic Games, which have only slight nonlinearities, where even approximating an optimal policy in a sample efficient manner is impossible. This result is heavily inspired by the recent work of Du et al. [DKWY20], and we claim no technical novelty. Rather, the purpose of this result in our setting, is to further motivate the importance of studying conditions such as EPW, instead of assuming that the MDP has linear structure.

Proposition 1. *There exists a Generic Game $(\mathcal{M}, \pi(\Theta))$, where d, k, ϕ, B are all at most polynomial in H and $|\mathcal{A}|$, and $\pi(\Theta)$ is the class of softmax linear policies, such that*

the following holds. There exists an unknown neural network $f : \mathbb{R}^d \to \mathbb{R}$, where f is a linear combination of two ReLU *neurons, such that $V^\star_{\mathcal{M}}(s) = f(s)$ for all $s \in \mathcal{S}$. Yet, any algorithm requires $\Omega(\min\{|\mathcal{A}|^H, 2^d\})$ trajectories to find, with probability at least $3/4$, a policy π satisfying*

$$V^{s_0}_{\mathcal{M}}(\pi) \geq V^{s_0}_{\mathcal{M}}(\pi(\theta^\star)) - 1/4.$$

We stress that the essence of this result follows from Du et al. [DKWY20], and we only make small modifications to their proof to fit it in our Generic Game setting. We nevertheless provide a proof sketch in Appendix A.4. The result shows the existence of MDPs for which a softmax linear policy is optimal, and where the optimal value function can be expressed as a neural network with only two ReLU neurons. Despite only this slight nonlinearity, sample efficient RL is impossible. Notice that in the statement of this result, there is no dependency of π on θ. This is because we do not restrict the algorithm to only search over policies lying in $\pi(\Theta)$. In particular, the policy π mentioned in the result can be *arbitrary*, and does not have to lie in $\pi(\Theta)$.

Proposition 1 demonstrates that if the Generic Game is even slightly nonlinear, as one would expect in practice, sample efficient RL is impossible. So we must look beyond linearity to obtain a realistic characterization of when sample efficient RL is possible. Our EPW condition, which makes no linearity assumptions, is one example of this.

2.5 Discussion

In this paper, we studied structural conditions which permit sample efficient RL in continuous state spaces, with a focus on conditions that are typical in popular RL domains such as Atari games. We introduced the EPW condition, which in contrast to prior work, makes no linearity assumptions about the MDP structure. We provided an algorithm which provably solves MDPs satisfying EPW. We analyzed the sample complexity of this algorithm, and showed it requires a number of trajectories that is a lower order polynomial of the horizon length and other relevant problem dependent quantities. We also showed that MDPs which have very slight nonlinearities (but do not satisfy EPW) cannot be solved sample efficiently. Our analysis thus highlights the important need to look beyond linear structure, in order to establish the sample efficiency of RL in popular domains.

A number of open questions remain. First, while our EPW condition is directly motivated by RL gaming domains such as Atari, it is unclear whether EPW is satisfied by other RL application domains such as robotics. A natural direction for future work is to study these domains more closely, and identify structure that permits sample efficient RL in such domains. Second, recall that our algorithm requires access to a particular computational oracle. As discussed, we made this computational abstraction since we placed minimal restrictions on the policy class, so in the worst case obtaining such an oracle could be intractable. Nevertheless, we suspect that when using a neural network policy class with an appropriate architecture, one could approximate this oracle efficiently. It would be interesting to precisely characterize when this is possible. Third, it would be

interesting to see whether a variant of our theoretically justified algorithm can be deployed in practice. Using our theoretical insight to design a pragmatic method, with strong empirical performance, is an important direction for future work.

3 Complete Policy Regret Bounds for Tallying Bandits

The content of this section is based on [MLS22].

Abstract

Policy regret is a well established notion of measuring the performance of an online learning algorithm against an adaptive adversary. We study restrictions on the adversary that enable efficient minimization of the *complete policy regret*, which is the strongest possible version of policy regret. We identify a gap in the current theoretical understanding of what sorts of restrictions permit tractability in this challenging setting. To resolve this gap, we consider a generalization of the stochastic multi armed bandit, which we call the *tallying bandit*. This is an online learning setting with an m-memory bounded adversary, where the average loss for playing an action is an unknown function of the number (or tally) of times that the action was played in the last m timesteps. For tallying bandit problems with K actions and time horizon T, we provide an algorithm that w.h.p achieves a complete policy regret guarantee of $\widetilde{\mathcal{O}}(mK\sqrt{T})$, where the $\widetilde{\mathcal{O}}$ notation hides only logarithmic factors. We additionally prove an $\widetilde{\Omega}(\sqrt{mKT})$ lower bound on the expected complete policy regret of any tallying bandit algorithm, demonstrating the near optimality of our method.

3.1 Introduction

When decision making algorithms are deployed in the real world, the reward associated with choosing a decision is rarely static. Instead, an algorithm's decision impacts the state of its environment, which in turn influences the quality of that same decision in the future. For instance, in recommender systems such as YouTube and Netflix, the choice to recommend a type of content is often instrumental in shaping the preferences of the user for that content genre. This creates a feedback loop between an algorithm and its environment, and results in a complex back and forth interaction.

Such dynamic and interactive settings are well modeled as online learning problems, where a player competes against an adaptive adversary. To measure the performance of the player, most of the literature on online learning has focused on a performance metric called the *traditional regret* [ACBFS02, FKM05, AHR08, HK12]. However, a significant line of work has established that when the adversary is adaptive, the traditional regret is a poor indicator of the performance of an algorithm [MOSW02, ADT12, CBDS13, HKR16, LHK21]. Instead, one typically opts for a stronger performance metric, known as *policy regret*. The policy regret accumulated over a time horizon T is defined with respect to a competitor class $\mathcal{C}_T$ of deterministic policies (or length T action sequences). The policy regret with respect to $\mathcal{C}_T$, which we denote $\mathcal{R}^{\text{pol}}_{\mathcal{C}_T}$, compares the algorithm's cumulative loss to that of the best policy in $\mathcal{C}_T$.

Much of the prior work on policy regret has focused on the restrictive assumption that $\mathcal{C}_T$ contains only those action sequences that repeatedly play the same action [ADT12, CBDS13, DDKP14, ADMM18]. We allude to the policy regret with this choice of $\mathcal{C}_T$ as

the *constant action policy regret*. The strongest possible version of policy regret is when $\mathcal{C}_T$ is the complete policy class (i.e., the set of all deterministic policies), and we abbreviate this as the *complete policy regret*. This challenging setting has recently received attention from the online learning community [HKR16, SMLV20, LHK21]. On the other hand, this performance metric is equivalent to the one that is standard in the closely related field of reinforcement learning, where a vast literature explores how to efficiently maximize cumulative reward [KMN99, SMSM00, KKL03, JAZBJ18, WDYK20, MLR21].

Unfortunately, prior work has shown that without restrictions on the adversary, obtaining non-trivial guarantees on even the constant action policy regret is impossible [ADT12]. Hence, to attain meaningful guarantees on the complete policy regret (CPR), it is necessary to restrict the adversary. Prior literature on policy regret studies different competitor classes $\mathcal{C}_T$, along with varying types of restrictions on the adversary, to demonstrate non-trivial guarantees on the corresponding policy regret $\mathcal{R}^{\text{pol}}_{\mathcal{C}_T}$. We comprehensively survey these restrictions, and identify a gap in the current theoretical understanding of when it is possible to attain meaningful guarantees on CPR. To resolve this gap, we make the following contributions:

- We introduce an online learning setting known as the *tallying bandit*. Here the average loss for playing an action is a function of the number (or tally) of times that action was played in the last m timesteps. The stochastic multi armed bandit (sMAB) is a special case of the tallying bandit, via a choice of $m = 1$. From a more practical angle, we view the tallying bandit as a step towards handling feedback loops that arise in applications such as recommender systems.

- For tallying bandit problems with K actions and time horizon T, we provide an algorithm, that given any $\delta \in (0, 1)$, achieves with probability at least $1 - \delta$ a complete policy regret guarantee of $\widetilde{\mathcal{O}}\left(mK\sqrt{T}\log\left(T, m, K, 1/\delta\right)\right)$.

- We complement our algorithmic development with an $\widetilde{\Omega}(\sqrt{mKT})$ minimax lower bound on the expected complete policy regret of any method designed for tallying bandits. This demonstrates the near optimality of our algorithm.

3.2 Problem Formulation

3.2.1 Online Learning & Complete Policy Regret

We begin by providing a generic formulation of online learning against adaptive adversaries, following rather closely the description of Arora et al. [ADT12]. An online learning problem with time horizon T and action set $\mathcal{X}$ is an iterative game between a player and an adaptive adversary. Throughout, we let K denote the cardinality of $\mathcal{X}$. Before the game begins, the adversary fixes a sequence of history dependent loss functions $\{f_t\}_{t=1}^{T}$, where f_t maps $\mathcal{X}^t$ to the interval $[0, 1]$. At each timestep t of the game, the player chooses an action $a_t \in \mathcal{X}$.

In the bandit feedback model, the player then observes the loss value $f_t(a_{1:t})$, where we have used $a_{1:t}$ as shorthand for $a_1, a_2 \dots a_t$. By contrast, in the full information model, the player observes $f_t(a_{1:t-1}, x)$ for all $x \in \mathcal{X}$.

The cumulative loss experienced by the player during this game is $\sum_{t=1}^{T} f_t(a_{1:t})$. Note that this is a random variable, since the player's strategy can be random. In order to evaluate the performance of the player, we compare this cumulative loss to a baseline. In particular, we let $\mathcal{C}_T \subseteq \mathcal{X}^T$ be a competitor class of policies (or length T action sequences). Given some $\mathcal{C}_T$, one typically measures the player's performance via either the *policy regret*, which we denote $\mathcal{R}^{\text{pol}}_{\mathcal{C}_T}$, or the *traditional regret*, which we denote $\mathcal{R}^{\text{trad}}_{\mathcal{C}_T}$. The policy regret [ADT12, CBDS13, ADMM18] is defined as

$$\mathcal{R}^{\text{pol}}_{\mathcal{C}_T} = \sum_{t=1}^{T} f_t(a_{1:t}) - \min_{(y_1, y_2 \dots y_T) \in \mathcal{C}_T} \sum_{t=1}^{T} f_t(y_1, y_2 \dots y_t). \tag{3}$$

Notably, this definition differs substantially from the traditional regret [ACBFS02, FKM05, AHR08, HK12], given by

$$\mathcal{R}^{\text{trad}}_{\mathcal{C}_T} = \sum_{t=1}^{T} f_t(a_{1:t}) - \min_{(y_1, y_2 \dots y_T) \in \mathcal{C}_T} \sum_{t=1}^{T} f_t(a_{1:t-1}, y_t). \tag{4}$$

In the aforementioned adversarial online learning setup, the traditional regret lacks meaningful interpretation. Instead, one opts for the policy regret to measure the player's performance. We refer the interested reader to Arora et al. [ADT12] for further details on the motivation for this choice.

Let $\mathcal{X}_{\text{const}}$ denote the set of constant action sequences, so that $\mathcal{X}_{\text{const}} = \{(x, x \dots x) \text{ s.t. } x \in \mathcal{X}\}$. We refer to $\mathcal{R}^{\text{pol}}_{\mathcal{X}_{\text{const}}}$ and $\mathcal{R}^{\text{trad}}_{\mathcal{X}_{\text{const}}}$ respectively as the *constant action policy regret* and *constant action traditional regret*. The constant action policy regret (and hence the constant action traditional regret) yields a weak measure of performance, since we are comparing the player's performance to a very restricted baseline policy class. Expanding the competitor class $\mathcal{C}_T$ yields stronger notions of performance. In our paper, we are interested in the challenging setting where $\mathcal{C}_T$ is the complete policy class (the set of all length T action sequences), or equivalently where $\mathcal{C}_T = \mathcal{X}^T$. This choice of $\mathcal{C}_T$ in Eq. (3) yields the strongest version of policy regret, and we refer to it as *complete policy regret (CPR)*, and denote it by $\mathcal{R}^{\text{cp}}$. An *optimal policy* is a policy in $\mathcal{X}^T$ that minimizes the cumulative loss. We will often use the terminology "efficiently minimize the CPR", which means to obtain a CPR bound that is sublinear in T and at most polynomial in all other problem dependent parameters. Our exclusive focus is on statistical (rather than computational) efficiency.

3.2.2 Restricting The Adversary

Prior work due to Arora et al. [ADT12] has shown that without any restrictions on the adversary, and even when $\mathcal{C}_T = \mathcal{X}_{\text{const}}$, for any player there exists an adversary such

that the player's constant action policy regret satisfies $\mathcal{R}^{\text{pol}}_{\mathcal{X}_{\text{const}}} = \widetilde{\Omega}(T)$. To prove this lower bound, Arora et al. [ADT12] construct an adversary that is wholly unrestricted and hence extremely powerful. Thus, to obtain non-trivial upper bounds on even the constant action policy regret, it is necessary to weaken the adversary. One natural type of restriction that has been well studied in prior work, is to restrict the memory of the adversary [ADT12, CBDS13, ADMM18].

Definition 2. *We say that an adversary is m-memory bounded if for all $t \geq m$, all $a_{1:t} \in \mathcal{X}^t$, all $a'_{1:t-m} \in \mathcal{X}^{t-m}$ and all f_t we have that*

$$f_t(a_{1:t}) = f_t(a'_1, a'_2 \dots a'_{t-m}, a_{t-m+1} \dots a_t).$$

Hence, an m-memory bounded adversary is only permitted to define its loss function based on the player's most recent m actions. Prior work has shown that when m is sublinear in T and $\mathcal{C}_T$ is sufficiently restricted (informally, when $\mathcal{C}_T$ equals or is only slightly larger than $\mathcal{X}_{\text{const}}$), then the player can achieve policy regret $\mathcal{R}^{\text{pol}}_{\mathcal{C}_T}$ that is sublinear in T [ADT12, CBDS13, DDKP14, ADMM18].

When $m = 1$, then the adversary is oblivious, and we overload notation and write $f_t(a_{1:t}) = f_t(a_t)$. Notably, for a fixed $\mathcal{C}_T$, the policy regret $\mathcal{R}^{\text{pol}}_{\mathcal{C}_T}$ equals the traditional regret $\mathcal{R}^{\text{trad}}_{\mathcal{C}_T}$ in this scenario. It is well known that against an oblivious adversary, a player can achieve sublinear constant action policy regret [ADT12]. Hence, it is natural to question whether a player can achieve sublinear CPR, when the adversary is oblivious. We show via the following counterexample that this is impossible. Note that the result of the counterexample holds even in the full information feedback model (as opposed to just bandit feedback). A similar result is given by Mohri and Yang [MY18].

Counterexample 1. *Let $\mathcal{X} = \{x_1, x_2\}$. To define its sequence of loss functions, the adversary first samples a bit string b uniformly at random from $\{0,1\}^T$. For each t, it then defines*

$$f_t(x_1) = b_t \text{ and } f_t(x_2) = 1 - b_t.$$

Attaining sublinear complete policy regret is then equivalent to making a sublinear number of mistakes when guessing the value of b_t. Since this is impossible, we have that $\mathbb{E}[\mathcal{R}^{cp}] = \widetilde{\Omega}(T)$, where the expectation is over the sampling of b and the player's (possibly randomized) strategy.

Crucially, the above counterexample relies on the fact that even though the adversary is oblivious, its loss functions $f_t, f_{t'}$ for $t \neq t'$ are time varying and are constructed independent of each other. In this scenario, the player cannot predict anything about f_t via knowledge of $f_{t'}$ for $t' < t$. To evade such counterexamples, a different type of restriction on the adversary's power is to ensure that some knowledge of $f_{t'}$ leaks some information about f_t. This motivates the following definition.

Definition 3. *An m-memory bounded adversary is said to be g-restricted if the following is true. For each action $x \in \mathcal{X}$, there exists a base function $g_x : \cup_{m'=1}^{m} \mathcal{X}^{m'} \to [0,1]$, such that*

$$f_t(a_{1:t}) \equiv f_t(a_{\max\{1,t-m+1\}:t}) = g_{a_t}(a_{\max\{1,t-m+1\}:t}).$$

Although we are not aware of prior work on online learning that uses g-restricted adversaries without additional restriction, in the sequel we will discuss prior work that consider g-restricted adversaries with significant additional restrictions [HKR16, LCM17, SLC+19, SMLV20, LHK21, ABGK22]. For now, we note that a g-restricted adversary must define loss functions whose value for a fixed input cannot vary with time. With such a restriction, the player can learn information about each g_x (and hence about f_t) as the game progresses, and this enables the player to choose better actions over time. This restriction thus precludes the setting of Counterexample 1.

When the adversary is g-restricted, it is straightforward to achieve CPR bounds that are sublinear in T and depend exponentially on m. However, throughout our paper we interested in efficiently minimizing CPR, which means we desire bounds that scale polynomially with m. Unfortunately, there exist online learning games where the adversary is m-memory bounded and g-restricted, but it is impossible to efficiently minimize CPR. We demonstrate this in the following counterexample, which holds even in the full information feedback model (as opposed to just bandit feedback).

Counterexample 2. *Let $\mathcal{X} = \{x_1, x_2\}$. Sample a tuple b of length $m-1$ uniformly at random from $\mathcal{X}^{m-1}$. Define $g_{x_1} : \cup_{m'=1}^{m} \mathcal{X}^{m'} \to [0,1]$ as*

$$g_{x_1}(a_{1:m'}) = 1 \text{ if } m' < m \text{ and } g_{x_1}(a_{1:m}) = 1 - \prod_{i=1}^{m-1} \mathbb{I}\left(a_i = b_i\right).$$

Also define $g_{x_2} = 1$. Via the base functions g_{x_1}, g_{x_2} we define the adversary's loss functions as

$$f_t(a_{1:t}) \equiv f_t(a_{t-m+1:t}) = g_{a_t}(a_{t-m+1:t}).$$

The policy that cyclically plays actions $b_1, b_2 \dots b_{m-1}, x_1$ suffers a loss of zero at least once every m timesteps. Meanwhile, suffering zero loss for the player is at least as hard as identifying b, and a standard "needle in the haystack" argument [DKWY20] shows that this requires $\widetilde{\Omega}(2^m)$ timesteps. Hence we have that $\mathbb{E}\left[\mathcal{R}^{cp}\right] = \widetilde{\Omega}(\min\{2^m, T\}/m)$, where the expectation is over the sampling of b and the player's (possibly randomized) strategy.

This counterexample demonstrates that even if the adversary is m-memory bounded and g-restricted, any player suffers CPR that scales exponentially with m. So, further restrictions on the adversary are necessary. A natural restriction is to enforce that each g_x has special structure. This is precisely the approach taken by works on *rotting* bandits [HKR16, LCM17, SLC+19, SMLV20], *improving* bandits [HKR16], *single peaked* bandits [LHK21] and *congested* bandits [ABGK22]. Concretely, these works use base functions $\{g_x\}_{x \in \mathcal{X}}$ that have the following special "tallying" structure.

Definition 4. *An m-memory bounded and g-restricted adversary is said to be h-tallying, if for each $x \in \mathcal{X}$ there exists $h_x : \{1, 2 \dots m\} \to [0, 1]$ such that*

$$f_t(a_{1:t}) \equiv f_t(a_{\max\{1,t-m+1\}:t}) = g_{a_t}(a_{\max\{1,t-m+1\}:t}) = h_{a_t}\left(\sum_{t'=\max\{1,t-m+1\}}^{t} \mathbb{I}(a_{t'} = a_t)\right).$$

As discussed by the aforementioned works, this tallying structure is often a natural model in practice. For instance, Heidari et al. [HKR16] discuss a crowdsourcing setting where an agency utilizes workers to repeatedly perform the same task (such as classifying images) at each timestep. The agency picks a worker at each timestep, with the goal of picking a sequence of workers that makes the fewest number of mistakes when performing the task. Here, it is reasonable that an individual worker's performance changes as an (unknown) function of the *number* of times that the worker has already performed the task (for example, due to fatigue), thus motivating the tallying structure.

We emphasize that in addition to assuming the adversary is h-tallying, the aforementioned works of Heidari et al. [HKR16], Levine et al. [LCM17], Seznec et al. [SLC+19], Seznec et al. [SMLV20], Lindner et al. [LHK21] and Awasthi et al. [ABGK22] make *supplemental* benign assumptions on the structure of the functions $\{h_x\}_{x \in \mathcal{X}}$, as we detail in Section 3.5. For instance, the rotting bandit setting of Heidari et al. [HKR16] assumes that h_x is an increasing function for each $x \in \mathcal{X}$. Under this benign assumption, they provide algorithms that efficiently minimize CPR. Notably, such strong assumptions on h_x enable this line of work to (often) tackle the case where $m = T$ (although algorithms designed for this case generally do not handle $m < T$), and additionally enables these works to (often) handle the more difficult scenario of when the losses are *stochastically* observed.

This exposes a gap in our understanding of when one can efficiently minimize CPR. In particular, it remains unclear whether we can attain this goal for h-tallying adversaries where we make *no* assumptions on the structure of each h_x. This motivates the following question.

Assume the adversary is m-memory bounded, g-restricted and h-tallying. Without any assumptions on the functions $\{h_x\}_{x \in \mathcal{X}}$, and in the bandit feedback model with (possibly) stochastically observed losses, when is it possible to efficiently minimize the complete policy regret?

The remainder of this paper is devoted to resolving this question. To this end, in the sequel we define the *tallying bandit*, and provide upper and lower bounds on the achievable CPR in this setting.

3.3 Tallying Bandits

Let us formally introduce the tallying bandit setting.

Definition 4. *An online learning game is an (m, g, h)-tallying bandit if the adversary is m-memory bounded, g-restricted and h-tallying, and if after playing action a_t the player observes a random variable $\widetilde{h}_{a_t}(y_t) \in [0, 1]$ satisfying*

$$\mathbb{E}\left[\widetilde{h}_{a_t}(y_t)\right] = h_{a_t}(y_t) = g_{a_t}(a_{\max\{1,t-m+1\}:t}) = f_t(a_{1:t}),$$

where $y_t = \sum_{t'=\max\{1,t-m+1\}}^{t} \mathbb{I}(a_{t'} = a_t)$.

We assume the cardinality K of the action set $\mathcal{X}$ is finite, and also that m is known (although we discuss how to relax this in Section 3.6). With this definition of the setting in hand, we can restate our goal of efficiently minimizing the CPR. Concretely, we desire an algorithm, which when given an (m, g, h)-tallying bandit problem, has a CPR bound that is polynomial in K, m and is sublinear in T. The tallying bandit strictly generalizes the well studied stochastic multi armed bandit (sMAB) [LR85, ACBF02], simply via a choice of $m = 1$. Hence, we generalize the study of sMAB to $m > 1$. We remark that tallying bandit is a special case of the *rested* bandit, a general framework for nonstationary MAB where the reward of an arm evolves when it is pulled, and we defer detailed discussion of this to Section 3.5.

Recall that in sMAB, the optimal policy plays the same optimal arm at each timestep. Hence, in sMAB obtaining zero constant action traditional regret is equivalent to obtaining zero constant action policy regret and also zero CPR. Given that the tallying bandit is highly structured and generalizes sMAB via $m = 1$, it is natural to question whether minimizing constant action policy (or traditional) regret implies minimizing CPR. The following counterexample answers this question in the negative when $m > 1$. More generally, this counterexample shows that even when the adversary is restricted (as in tallying bandits), minimizing the constant action policy (or traditional) regret can lead to solutions whose cumulative loss is $\widetilde{\Omega}(T)$ larger than the minimum achievable total loss.

Counterexample 3. *Let $\mathcal{X} = \{x_1, x_2\}$ and $m = 2$. Define $h_{x_1}(1) = h_{x_2}(1) = 0$ and $h_{x_1}(2) = h_{x_2}(2) = 1$. A policy that fixes either of the two actions, and then plays this action at every timestep, incurs cumulative loss $T - 1$ but has zero constant action policy regret and zero constant action traditional regret. Meanwhile, the complete policy regret of this policy is $T - 1$, since the optimal policy that alternates playing actions x_1 and x_2 incurs zero cumulative loss.*

Thus far, our motivation for the tallying bandit setting has been primarily theoretical, to resolve the gap in our understanding of when we can efficiently minimize CPR. Nevertheless, in similar vein to Heidari et al. [HKR16], Lindner et al. [LHK21] and Awasthi et al. [ABGK22], we believe that the tallying bandit is a simple approximation for various practical settings. For instance, in recommender systems the reward associated with an action is rarely static, because the stimulus of recommended content influences user preferences [CLA+03, SGR16]. Moreover, literature on psychology and cognition suggests that humans often forget prior

stimuli and do not always encode them in permanent memory [Kla80, CM07]. Thus, one way to model a user's preferences is via an (unknown) function of the *number* of times a content genre has been recommended in a *recent* time interval, motivating both the h-tallying structure as well as bounded m. Nevertheless, the tallying bandit is just one plausible model, and we suggest possible extensions in Section 3.6.

Let us now discuss potential avenues for efficiently minimizing CPR in tallying bandit problems. One approach is to observe that any tallying bandit problem can be cast as a reinforcement learning (RL) problem, where each state corresponds to a sequence of actions taken in the last m timesteps. However, methods for solving such RL problems typically scale with the cardinality of the state space [AMK13, AOM17, JAZBJ18], and such approaches would suffer $\widetilde{\Omega}(K^m)$ CPR.

Hence it is necessary to leverage the additional properties of tallying bandit problems. To gain intuition, let us consider the simplified setting of deterministic bandit feedback, where for each $t, a_{1:t} \in \mathcal{X}^t$ the player observes $f_t(a_{1:t})$ with no noise. Since the loss functions f_t are fully defined by the functions $\{h_x\}_{x \in \mathcal{X}}$, it is natural to consider the following algorithm, which we denote $\mathbb{ALG}_{\text{det}}$. First, the algorithm queries $h_x(y)$ at each $(x, y) \in \mathcal{X} \times \{1, 2 \dots m\}$. This yields full information about the loss functions f_t. Then, the algorithm plans offline an optimal sequence of actions for the remaining timesteps. $\mathbb{ALG}_{\text{det}}$ is formally specified as Algorithm 4 in Appendix B.12, and the following result shows that its CPR is minimax optimal (upto constant factors).

Proposition 1. *Consider any (m, g, h)-tallying bandit problem with deterministic bandit feedback. Then the complete policy regret of Algorithm 4 ($\mathbb{ALG}_{det}$) is almost surely upper bounded as $\mathcal{R}^{cp} \leq (m+1)K$. Moreover, there exists an (m, g, h)-tallying bandit problem, such that the (expected) complete policy regret of any (possibly randomized) algorithm on this problem with deterministic feedback is lower bounded as $\mathbb{E}[\mathcal{R}^{cp}] \geq mK/128$.*

The proof of Proposition 1 is included in Appendix B.12. Due to the optimality of $\mathbb{ALG}_{\text{det}}$ given deterministic feedback, it is reasonable to extend it to handle stochastic feedback. One natural way to do so is via an "explore then exploit" modification, which has been studied even for sMAB [Sli19]. Consider the following "explore then exploit" algorithm, which we denote $\mathbb{ALG}_{\text{stoch}}$. It queries repeatedly to receive stochastic realizations of $h_x(y)$ at each $(x, y) \in \mathcal{X} \times \{1, 2 \dots m\}$, and constructs tight confidence intervals for these $h_x(y)$. It then uses these estimated values of $h_x(y)$ to plan offline an optimal sequence of actions for the remaining timesteps. $\mathbb{ALG}_{\text{stoch}}$ is provably efficient, in the sense that its CPR is sublinear in T and polynomial in m, K. However, a standard argument [Sli19] shows that the dependency on T for $\mathbb{ALG}_{\text{stoch}}$ scales as $\widetilde{\Theta}\left(T^{2/3}\right)$, and it is unclear whether this dependency is optimal for tallying bandits. In the forthcoming section, we show that this dependency on T can be significantly improved.

3.4 Main Results

We now turn to our main results. In Section 3.4.1, we formulate Algorithm 2, a method designed for solving tallying bandit problems, and prove an $\widetilde{\mathcal{O}}(mK\sqrt{T})$ upper bound on its CPR, where $\widetilde{\mathcal{O}}$ hides only logarithmic factors. In Section 3.4.2, we prove an $\widetilde{\Omega}(\sqrt{mKT})$ lower bound on the CPR of any method designed to solve tallying bandit problems. This shows that Algorithm 2 is nearly optimal.

3.4.1 Upper Bound

To formulate our algorithm for tallying bandits, it is natural to exploit the h-tallying structure of the problem, by building estimates of $h_x(y)$ for each $(x, y) \in \mathcal{X} \times \{1, 2 \dots m\}$. As discussed in Section 3.3, "explore then exploit" algorithms such as $\mathbb{ALG}_{\text{stoch}}$ incur a poor dependency on T. Instead, it is critical to balance exploration and exploitation, by estimating $h_x(y)$ only by playing those actions that will not increase the regret too fast. To this end, we introduce a key definition. Recall that a deterministic policy π is length T sequence of actions. We say that a deterministic policy π is $\sqrt{T}$-cyclic if $\pi_{k\sqrt{T}+t} = \pi_t$ for each $1 \leq t \leq \sqrt{T}$ and each $0 \leq k \leq \sqrt{T} - 1$. The basis for our algorithm relies on the key claim that for any tallying bandit problem, there exists a $\sqrt{T}$-cyclic policy that is nearly optimal. This claim is formalized as Lemma 3 in Appendix B.1.

Assuming this claim to be true, to solve tallying bandits it is tempting to leverage algorithms designed for multi armed bandits with expert advice [ACBFS02], where we treat each $\sqrt{T}$-cyclic policy as an expert. However, such an approach would only guarantee $\tilde{\mathcal{O}}\left(T^{3/4}\right)$ CPR, since there are $K^{\sqrt{T}}$ experts. Instead, our algorithm draws inspiration from the successive elimination (SE) algorithm, which has been applied to sMAB [EDMM02, Sli19]. Before we apply SE in our setting, let us recall SE in the context of sMAB. The method proceeds in epochs. Within each epoch s, it maintains a set A_s of feasible arms, where an arm is feasible only if its estimated optimality gap lies within a confidence interval of size C_{s-1}. The algorithm pulls each arm in A_s repeatedly to obtain a sharper estimate of its optimality gap. Then the method uses these sharper estimates to prune A_s and create a smaller set A_{s+1}, where A_{s+1} contains only those arms in A_s whose estimated optimality gap is smaller than some $C_s < C_{s-1}$.

To apply SE in our tallying bandit setting, we define the initial set A_1 as the set of all $\sqrt{T}$-cyclic policies, and treat each such policy to be analogous to an "arm" in sMAB. Our aforementioned key claim ensures that there is some policy in A_1 that is guaranteed to be nearly optimal. However, there are two salient technical issues that prevent a naive application of SE to solve tallying bandits. First, note that the cardinality of A_1 is $K^{\sqrt{T}}$. This is too large to apply a traditional SE approach, since naively estimating the optimality gap of each policy in A_s by repeatedly playing the policy and applying a concentration inequality would incur large regret. To resolve this, we exploit the h-tallying structure of the problem to modify SE, and estimate the optimality gap of each policy in A_s by

iteratively estimating $h_x(y)$ for each $(x, y) \in \mathcal{X} \times \{1, 2 \dots m\}$. Second, note that unlike in the sMAB, in the tallying bandit the prior history of actions affects the loss of the current action, which biases the estimation of the optimality gap of the policies in A_s. To handle this, we modify SE to incorporate an additional overheard step before the estimation, and our proof shows that this overhead step removes the bias from the estimation without incurring much additional regret.

Algorithm 2 Successive Elimination for Tallying Bandits (SE-TB)

Require: memory capacity m, time horizon T, failure probability tolerance $\delta \in (0, 1)$, number of actions K

Define $S = \log_2 \left(\frac{\sqrt{T}}{4Km} + 1 \right)$.
Define $n_s = 2^s$, $T_s = 2n_s Km\sqrt{T}$ and $C_s = \sqrt{\frac{32Km}{n_s\sqrt{T}} \log\left(\frac{2KmS}{\delta}\right)}$.
Construct A_1 to be the set of all $\sqrt{T}$-cyclic policies.
for $s \in \{1, 2 \dots S\}$ **do**
    for $x \in \mathcal{X}$ **do**
        for $y \in \{1, 2 \dots m\}$ **do**
            Select $\pi_{sxy} \in \operatorname{argmax}_{\pi' \in A_s} \{N_{xy}(\pi')\}$, where N_{xy} is defined in Eq. (5).
            if $N_{xy}(\pi_{sxy}) = 0$ **then**
                Execute π_{sxy} for $2n_s$ periods and store nothing.
            else
                Execute π_{sxy} for n_s periods and store nothing.
                Execute π_{sxy} for n_s periods and store $\{\widetilde{h}_x(y)_{s,k}\}_{k=1}^{n_s N_{xy}(\pi_{sxy})\sqrt{T}}$.
            end if
        end for
    end for
    for $\pi \in A_s$ **do**
        Define $\widehat{\mu}_s(\pi) = \sum_{(x,y) \in \mathcal{X} \times \{1,2 \dots m\}} N_{xy}(\pi) \frac{1}{n_s N_{xy}(\pi_{sxy})\sqrt{T}} \sum_{k=1}^{n_s N_{xy}(\pi_{sxy})\sqrt{T}} \widetilde{h}_x(y)_{s,k}$.
    end for
    Select $\widehat{\pi}_s \in \operatorname{argmin}_{\pi \in A_s} \widehat{\mu}_s(\pi)$.
    Construct $A_{s+1} = \{\pi \in A_s \text{ s.t. } \widehat{\mu}_s(\pi) \leq \widehat{\mu}_s(\widehat{\pi}_s) + 2C_s\}$.
end for

With this outline in mind, let us present our method, which is formalized in Algorithm 2. To define Algorithm 2, we say that to *execute* a $\sqrt{T}$-cyclic policy π for $k \leq \sqrt{T}$ periods means to choose the action sequence $\pi_1, \pi_2 \dots \pi_{k\sqrt{T}}$. We also define for each $\sqrt{T}$-cyclic policy π and $(x, y) \in \mathcal{X} \times \{1, 2 \dots m\}$, the quantity $N_{xy}(\pi)$ via the following procedure. Execute π for $n + 1 \leq \sqrt{T}$ periods so that we have played the action sequence $\pi_1 \dots \pi_{n\sqrt{T}}, \pi_{n\sqrt{T}+1} \dots \pi_{(n+1)\sqrt{T}}$.

Then use this action sequence to define

$$N_{xy}(\pi) = \frac{1}{\sqrt{T}} \sum_{t=n\sqrt{T}+1}^{(n+1)\sqrt{T}} \mathbb{I}(\pi_t = x) \cdot \mathbb{I}\left(y = \sum_{t'=\max\{1,t-m+1\}}^{t} \mathbb{I}(\pi_{t'} = x) \right). \tag{5}$$

Intuitively, $N_{xy}(\pi)$ is the fraction of times that the player (stochastically) observes the loss value $h_x(y)$ when they repeatedly play the $\sqrt{T}$-cyclic policy π. In Lemma 2 in Appendix B.1, we show that if $m \leq \sqrt{T}$, then as long as $n \geq 1$, the number $N_{xy}(\pi)$ is well defined and independent of n, and also independent of any action sequence that was played before we executed π for $n+1$ periods. It suffices to consider tallying bandit problems where $m \leq \sqrt{T}$ (as we show in our proofs). Hence, lines 11 and 12 in Algorithm 2 are well defined, because when we execute π for $2n_s \geq 2$ periods, then in the latter $n_s \geq 1$ periods we observe (stochastic instantiations of) the loss value $h_x(y)$ for a total of $n_s N_{xy}(\pi)\sqrt{T}$ times, and we have denoted these observations as $\{\widetilde{h}_x(y)_{s,k}\}_{k=1}^{n_s N_{xy}(\pi_{sxy})\sqrt{T}}$. Note that various steps in Algorithm 2, such as line 7, require knowledge of $N_{xy}(\pi)$, but this can be computed offline when m is known. Indeed, the only steps of Algorithm 2 that are online (or incur regret) are lines 9, 11 and 12. Let us now analyze the performance of Algorithm 2.

Theorem 1. *For any (m, g, h)-tallying bandit problem and any input $\delta \in (0, 1)$, with probability at least $1 - \delta$ the complete policy regret of Algorithm 2 (SE-TB) is upper bounded as*

$$\mathcal{R}^{cp} \leq 1200Km\sqrt{T}\left(\sqrt{\log\left(2Km\log(T)/\delta\right)} + \log_2\left(\sqrt{T}/(2Km)\right)\right).$$

The proof of Theorem 1 is deferred to Appendix B.1. This result guarantees that given any tallying bandit problem, Algorithm 2 efficiently minimizes CPR, with a favorable dependency on m, K, T. Nevertheless, we acknowledge that our result has the following two limitations.

Knowledge of m. Algorithm 2 requires m as an input, which is unrealistic in practice. It is possible to modify Algorithm 2 to be adaptive to an unknown m, albeit at the expense of polynomially worse (and not sharp) dependency on m, K. Our focus is on obtaining a sharp characterization of the achievable CPR when m is known, and so we relegate discussion of this modification to Section 3.6. Sharply characterizing the minimax CPR when m is unknown remains an important open question.

Computational Efficiency. Algorithm 2 is computationally inefficient. We emphasize that our exclusive focus is on statistical (rather than computational) efficiency, since our work is only a first step. We believe this is a worthwhile endeavor, since attaining sublinear CPR is a non-trivial task riddled with subtleties, even in settings that make much stronger assumptions than we do. For instance, the improving [HKR16] and single peaked [LHK21] bandit settings enforce $m = T$, require monotonicity and convexity conditions on $\{h_x\}_{x \in \mathcal{X}}$, and also require

the losses are observed *deterministically*. Even with these strong requirements, the best known CPR guarantees are *asymptotic* bounds that may decay *arbitrarily* slowly, and their algorithms cannot handle $m < T$. Hence we believe that our effort to provide nearly optimal non-asymptotic bounds on the CPR, in our realistic and practically motivated setting where $m \leq T$ and losses are observed stochastically, is worthwhile. Nevertheless, devising computationally efficient algorithms for the tallying bandit remains an important future direction, and we believe this is a non-trivial task. Indeed, even for the congested bandit [ABGK22], which is the tallying bandit with the additional strong assumption that $\{h_x\}_{x\in\mathcal{X}}$ are increasing, existing algorithms are computationally inefficient.

3.4.2 Lower Bound

It is reasonable to question whether the dependency of Algorithm 2 on m, K, T is optimal. Since the tallying bandit is equivalent to sMAB when $m = 1$, a classical result [Sli19] shows that any tallying bandit algorithm suffers $\widetilde{\Omega}\left(\sqrt{KT}\right)$ expected CPR. However, the correct dependency on m is unclear when $m > 1$, due to the highly structured nature of the tallying bandit. For instance, the proof of Theorem 1 shows that any tallying bandit problem can be equivalently cast as a Markov decision process (MDP), where it takes at most m timesteps to transition from any state to any other state in this MDP. One may wonder whether we can utilize such structure to design a smarter algorithm that exchanges the *multiplicative* dependence on m in the result of Theorem 1 for an *additive* dependence on m. Concretely, one may desire a bound that scales as $\widetilde{\mathcal{O}}\left(\text{poly}(m, K) + K\sqrt{T}\right)$. The following result shows that this is impossible.

Theorem 2. *There exists an (m, g, h)-tallying bandit problem and a numerical constant $c > 0$, such that the (expected) complete policy regret of any (possibly randomized) algorithm on this problem is lower bounded as*

$$\mathbb{E}\left[\mathcal{R}^{cp}\right] \geq c \cdot \max\left\{mK, \sqrt{mKT}\right\}.$$

The proof of Theorem 2 is deferred to Appendix B.11. This result demonstrates that Algorithm 2 is nearly minimax optimal, and its suboptimality is bounded by $\widetilde{\mathcal{O}}\left(\sqrt{mK}\log\left(T, m, K\right)\right)$. A comment on the proof technique of Theorem 2 is in order. Our proof reduces the tallying bandit setting to that of best arm identification in sMAB problems [AB10, Sli19]. We construct a tallying bandit problem where minimizing CPR is at least as hard as identifying the best arm in an sMAB problem with $\widetilde{\Theta}\left(mK\right)$ arms. Indeed, when $m = 1$ then tallying bandit is equivalent to sMAB, and Theorem 2 recovers the classical lower bound on the expected regret suffered by any algorithm designed for sMAB with K arms. Notably, a key component of our proof is to non-trivially upper bound the cumulative loss of the optimal policy in our construction, to ensure that we get a multiplicative (in lieu of additive) dependence on m in our lower bound.

3.5 Related Work

Policy Regret. The incompatibility of the traditional regret with an adaptive adversary was first identified by Merhav et al. [MOSW02], who studied the full information feedback model. The notion of policy regret was formalized by the foundational work of Arora et al. [ADT12]. They provide an algorithm which efficiently minimizes constant action policy regret $\mathcal{R}^{\text{pol}}_{\mathcal{X}_{\text{const}}}$ against generic m-memory bounded adversaries. It is unclear how to apply this algorithm to our setting, since our focus is on minimizing the CPR $\mathcal{R}^{\text{cp}}$. Arora et al. [ADT12] do also consider minimizing the policy regret $\mathcal{R}^{\text{pol}}_{\mathcal{C}_T}$ when $\mathcal{C}_T \supsetneq \mathcal{X}_{\text{const}}$. For instance, when $\mathcal{C}_T$ is the set of all piecewise constant sequences with at most s switches, they provide an algorithm whose policy regret satisfies $\mathcal{R}^{\text{pol}}_{\mathcal{C}_T} \leq \widetilde{\mathcal{O}}\left(m(Ks)^{1/3}T^{2/3}\right)$. Our Counterexample 3 shows that this algorithm cannot minimize CPR in the tallying bandit setting, since the optimal policy in Counterexample 3 has $\widetilde{\Theta}(T)$ switches. Cesa-Bianchi et al. [CBDS13], Dekel et al. [DDKP14] and Arora et al. [ADMM18] study the constant action policy regret, and do not discuss CPR. The results of Mohri and Yang [MY18] can be extended to yield policy regret guarantees relative to rather large comparator classes, but do not provide CPR guarantees.

Reinforcement Learning (RL). Minimizing CPR is equivalent to the performance metric used in RL, which is to maximize the total collected reward [SB18]. As we show (see Lemma 1), any tallying bandit problem can be cast as an RL problem. However, RL methods typically scale with the cardinality of the state space [AMK13, AOM17, JAZBJ18]. Hence, applying off the shelf RL algorithms to solve tallying bandit problems would incur $\widetilde{\Omega}\left(K^m\right)$ CPR.

Restless & Rested Bandits. In restless bandits, the reward of an arm evolves according to a stochastic process, independently of the actions chosen by the player [Whi81, GM11, BGZ14]. This is incompatible with tallying bandits. By contrast, tallying bandits is a special case of rested bandits, which is a general non-stationary MAB framework where an arm's reward changes when it is pulled. Tekin and Liu [TL12] and Cortes et al. [CDK+20] both study rested bandits where an arm's reward evolves according to a stochastic process, but both consider notions of regret that are significantly weaker than the CPR. A different variant of rested bandits is studied by Bouneffouf and Féraud [BF16], who assume that the dynamics of how the reward changes is known upto a constant factor, and hence their results are incomparable to ours.

Recharging, Recovering, Blocking, Delay-Dependent & Last Switch Dependent Bandits. This line of work studies settings where an arm's reward changes according to the number of timesteps that have passed since the arm was last pulled [KI18, BSSS19, PBG19, CCB20, LCCG21]. Such settings typically cannot be cleanly classified as either rested or restless bandits, but are related to both. The models in these works for how the

reward evolves are different than our tallying structure, where the reward of an arm instead depends on the number of times that arm was played.

Rotting Bandits. The rotting bandits setting [HKR16, LCM17, SLC+19, SMLV20] is a special case of our tallying bandit formulation, and merits close comparison to our work. This setting enforces $m = T$, and an arm's reward is assumed to be a decreasing function of the number of times that arm has been pulled. In our language, this means the $\{h_x\}_{x\in\mathcal{X}}$ functions are increasing. This strong assumption enables these works to efficiently minimize CPR even though $m = T$, although we note that algorithms for this setting cannot handle general $m < T$. By contrast, in our setting we make no assumptions on the structure of the functions $\{h_x\}_{x\in\mathcal{X}}$, and assuming $m < T$ is necessary. Indeed, in our setting, when m is $\widetilde{\Omega}(T)$ then the result of Theorem 2 shows that any player suffers $\widetilde{\Omega}(T)$ worst case CPR.

Improving & Single Peaked Bandits. The improving bandit [HKR16] and single peaked bandit [LHK21] are both special cases of our tallying bandit setting, and deserve special attention. In improving bandits, the reward of an arm is an increasing, concave function of the number of times it has been pulled. The single peaked bandit generalizes this, so that the reward function of an arm is initially increasing and concave, but may become decreasing at some point. In our language, this means that the $\{h_x\}_{x\in\mathcal{X}}$ functions are decreasing and convex, or decreasing and convex and then possibly increasing after some point. Both settings enforce $m = T$, and algorithms for these settings do not handle general $m < T$. By contrast, since we make no assumptions on the structure of the functions $\{h_x\}_{x\in\mathcal{X}}$, our Theorem 2 shows that when m is $\widetilde{\Omega}(T)$ then the worst case CPR scales as $\widetilde{\Omega}(T)$. We remark that the CPR bounds in these works are asymptotic, whereas we provide non-asymptotic guarantees. We also remark that these works require the losses to be observed deterministically, and they only provide a heuristic to handle stochastic observations of the loss.

Congested Bandits. In concurrent work, Awasthi et al. [ABGK22] introduced the congested bandit, which is a special case of the tallying bandit. Their formulation considers arbitrary $m \leq T$, and the reward of an arm is a decreasing function of the number of times it has been pulled. Hence the congested bandit is the tallying bandit with the additional assumption that the $\{h_x\}_{x\in\mathcal{X}}$ functions are increasing. This additional assumption enables them to provide an $\widetilde{\mathcal{O}}\left(\sqrt{mKT}\right)$ CPR bound, although we note that their algorithm is computationally inefficient.

3.6 Discussion

In this paper, we studied conditions under which it is possible to efficiently minimize CPR in online learning. To this end, it is necessary the restrict the adversary, and we considered several natural restrictions on the adversary that have appeared in prior work.

We then exposed a gap in our understanding of when it is possible to efficiently minimize CPR. To resolve this gap, we introduced the tallying bandit setting, and formulated an algorithm whose CPR (after discarding logarithmic factors) is w.h.p at most $\widetilde{\mathcal{O}}\left(mK\sqrt{T}\right)$. We also provided a lower bound of $\widetilde{\Omega}\left(\sqrt{mKT}\right)$ on the expected CPR of any tallying bandit algorithm, demonstrating the near optimality of our method.

Our Algorithm 2 required as input the true value of m. In practice, this knowledge is unrealistic, and one instead might only have an upper bound $\overline{m}$ on the true value of m. Let us describe a modified version of Algorithm 2 that can be used in this setting. Recall from the proof of Theorem 1 that Algorithm 2 proceeds in epochs, and in each epoch s it stores a set A_s of policies whose (average) loss is $\widetilde{\mathcal{O}}\left(2^{-s}T^{-1/4}\right)$ greater than that of the optimal policy. Hence, for the setting with unknown m, we can run $\overline{m}$ instantiations of Algorithm 2. After each epoch s, we identify the instantiation that has the policy with the minimum estimated (average) loss, and denote this instantiation as m_s. We then discard those instantiations whose policies have (average) loss that is $\widetilde{\Omega}\left(2^{-s}T^{-1/4}\right)$ greater than the (average) loss of the best policy stored by m_s. With such an approach, we are guaranteed to never discard the instantiation corresponding to the true m. And via the techniques used in the proof of Theorem 1, we can show that if we do not discard an instantiation corresponding to $m' \neq m$, then playing policies stored by this instantiation does not incur large regret. This approach yields a $\widetilde{\mathcal{O}}\left(\sqrt{T}\right)$ upper bound on the CPR, at the expense of polynomial factors of $\overline{m}, K$.

A number of open directions remain. A natural open question is resolving the gap between our upper and lower bounds on the achievable CPR in the tallying bandit. Separately, although Algorithm 2 is nearly statistically optimal, it is computationally inefficient. However, since the tallying bandit is highly structured, it is possible that the computational efficiency of even our own Algorithm 2 can be improved. For instance, can we design a data structure, which stores policies in a manner that allows efficient elimination of suboptimal policies after each epoch? Devising computationally efficient algorithms for tallying bandits is an important direction for future work. Finally, we view the tallying bandit as only a first step towards modeling interactive settings like recommender systems. For example, consider the following generalization of tallying bandits, where the loss of an action is a function of a *weighted* sum of the number of times the action has been played in the past, where more recent plays are given more weight. This naturally corresponds to a model of human memory, where more importance is placed on more recent events. Can we design efficient algorithms for such settings?

4 Weighted Tallying Bandits: Overcoming Intractability via Repeated Exposure Optimality

The content of this section is based on [MILS23].

Abstract

In recommender system or crowdsourcing applications of online learning, a human's preferences or abilities are often a function of the algorithm's recent actions. Motivated by this, a significant line of work has formalized settings where an action's loss is a function of the number of times that action was recently played in the prior m timesteps, where m corresponds to a bound on human memory capacity. To more faithfully capture decay of human memory with time, we introduce the Weighted Tallying Bandit (WTB), which generalizes this setting by requiring that an action's loss is a function of a *weighted* summation of the number of times that arm was played in the last m timesteps. This WTB setting is intractable without further assumption. So we study it under Repeated Exposure Optimality (REO), a condition motivated by the literature on human physiology, which requires the existence of an action that when repetitively played will eventually yield smaller loss than any other sequence of actions. We study the minimization of the complete policy regret (CPR), which is the strongest notion of regret, in WTB under REO. Since the precise value of m is typically unknown, we assume that we only have access to an upper bound M on m. We show that for problems with K actions and horizon T, a simple modification of the successive elimination algorithm has (upto a logarithmic factor) a CPR bound of $\widetilde{\mathcal{O}}\left(\sqrt{KT} + (m+M)K\right)$. Interestingly, upto an additive (in lieu of mutliplicative) factor in $(m+M)K$, this recovers the classical guarantee for the far simpler stochastic multi-armed bandit with traditional regret. We additionally show that in our setting, any algorithm will suffer an additive CPR factor of $\widetilde{\Omega}\left(mK + M\right)$, demonstrating that our result is nearly optimal. Our algorithm is computationally efficient, and we experimentally demonstrate its practicality and superiority over natural baselines in various simulation domains.

4.1 Introduction

When online learning algorithms are deployed in interactive applications, the algorithm's decisions impact the state of the environment. In turn, this impacts the quality of subsequent decisions made by the algorithm. This is especially true in human-centered applications such as recommender systems or crowdsourcing. For instance, consider a crowdsourcing setting where at each timestep we want to select a worker to perform a task, without prior knowledge of any worker's ability. The task may be complex or require some fine-tuning, and each worker might need a calibration period where they repeatedly perform the task, before they start exhibiting their true performance. The existence of such a calibration period has been extensively demonstrated in tasks that require visuomotor calibration [Ada61], such as throwing darts [WHFM20] or shooting a basketball [PMR+20]. Hence, an algorithm that asks workers to alternately perform the task, without intelligently allowing each worker time

to calibrate themselves to the task, may bias its estimation of each worker's true ability. This interaction between an algorithm and its environment separates this scenario from classical non-interactive frameworks such as the multi-armed bandit.

To capture one aspect of this interactivity, a significant research thrust in online learning has studied settings where an action's loss is described by the number of times that action was recently played in the prior m timesteps [HKR16, LCM17, SLC+19, SMLV20, LHK21, ABGK22, MLS22]. The quantity m typically corresponds to a bound on human memory capacity or capability. For example, in the aforementioned scenario m would be the number of timesteps required by a worker to fine-tune and calibrate themselves to the task, before revealing their true ability.

Of course, such settings are an approximation to reality. For instance, psychological research demonstrates that humans typically display a better memory for more recently occurring events [Kla80, RVC16]. So if we play an action once in the previous m timesteps, its impact on the present may greatly differ depending on whether it was played on the previous timestep or m timesteps ago. In the context of the aforementioned crowdsourcing setting, a worker may need a shorter calibration period if they performed the task on the previous timestep, as opposed to many timesteps ago. However, prior formalizations are oblivious to this difference. Motivated by these considerations, we make the following contributions:

- We introduce the Weighted Tallying Bandit (WTB), which generalizes prior formalizations by requiring that an action's loss is described by a *weighted summation* of the number of times that action was played in the prior m timesteps. Since this setting is dynamic and interactive, we eschew the traditional regret, and instead study the minimization of the strongest notion of regret known as the complete policy regret (CPR).

- We show that minimizing CPR in WTB is generally intractable. So we study it under the additional condition of Repeated Exposure Optimality (REO), which enforces the existence of an action that when repetitively played m times will yield smaller loss than other action sequences. In the context of the aforementioned example, REO is interpreted as the existence of a worker that once calibrated to the task, will perform better than other (calibrated or uncalibrated) workers. We motivate this condition via literature on human physiology.

- For WTB problems with K actions and horizon T that satisfy REO, and in the regime where only an upper bound M on the true value of m is known, we show that a slight modification of the classical successive elimination algorithm achieves a CPR guarantee (upto a logarithmic factor) of $\widetilde{\mathcal{O}}\left(\sqrt{KT} + (m+M)K\right)$. Besides an additive factor in $(m+M)K$, this matches the lower bound on the weaker traditional regret of the stochastic multi-armed bandit (which is the special $m = 1$ case of WTB with REO).

- While one may desire an algorithm that is fully adaptive to m and requires no such upper bound M, we show this is impossible. Concretely, we show that any algorithm with sublinear CPR must require such an upper bound M, and then show that a linear dependency on this input M is necessary. This implies a lower bound of $\widetilde{\mathcal{O}}\left(\sqrt{KT} + mK + M\right)$ on the achievable CPR of any algorithm in our setting, highlighting our algorithm's near optimality.
- Via diverse numerical simulations, we demonstrate our (computationally efficient) method's practicality and superiority over various baselines.

4.2 Problem Formulation

4.2.1 Weighted Tallying Bandit

We begin by formally introducing the Weighted Tallying Bandit as an online learning game with bandit feedback over time horizon T, where the player has access to an action set $\mathcal{X}$ with finite cardinality K. A long line of prior work has studied the scenario where an action's loss at any timestep is a function of the number of timesteps it was played in the prior m timesteps [HKR16, LCM17, SLC+19, SMLV20, LHK21, ABGK22, MLS22]. We refer to these settings as "tallying" settings. Our goal is to generalize this work, to the case where an action's loss is a function of a weighted tally of the number of times it was played in the past m timesteps.

To this end, we first introduce some notation. Assume the player has played the game for t timesteps, and for each timestep $1 \leq t' \leq t$ the player plays action $a_{t'}$. For a fixed action $x \in \mathcal{X}$, we define the vector $y^{t,x,m} \in \{0,1\}^m$ in a componentwise fashion as

$$y_i^{t,x,m} = \begin{cases} \mathbb{I}(a_{t-i+1} = x) & \text{if } t-i+1 \geq 1 \\ 0 & \text{if } t-i+1 < 1 \end{cases},$$

for each component $1 \leq i \leq m$. Hence, the vector $y^{t,x,m}$ stores the timesteps where action x was played in the previous m timesteps upto (and including) the current timestep t. With this notation in hand, we are now in a position to formally define the Weighted Tallying Bandit.

Definition 5 (Weighted Tallying Bandit (WTB)). *An online learning game is said to be an (m, w, h)-weighted tallying bandit with memory capacity m, if there exists an integer $m \geq 1$, a set of vectors $\{w_x\}_{x \in \mathcal{X}} \subset (0,1]^m$, and a set of functions $\{h_x\}_{x \in \mathcal{X}}$ each mapping from $\mathbb{R}$ to $[0,1]$, such that the following is true. For each $x \in \mathcal{X}$, the expected loss incurred at timestep t by playing action $a_t = x$ is given by*

$$h_x\left(w_x^\top y^{t,x,m}\right),$$

and the player observes as feedback a random observation $\widetilde{h}_x(w_x^\top y^{t,x,m}) \in [0,1]$*, that is independent of all other random observations, and satisfies*

$$\mathbb{E}\left[\widetilde{h}_x(w_x^\top y^{t,x,m})\right] = h_x\left(w_x^\top y^{t,x,m}\right).$$

In general, the quantities $m, \{w_x\}_{x\in\mathcal{X}}, \{h_x\}_{x\in\mathcal{X}}$ are all unknown, and the player only learns about them via bandit feedback over time. When $m = 1$, then WTB recovers the stochastic multi-armed bandit (sMAB) [LR85, ACBF02]. However, WTB with $m \geq 2$ is often a better model for human-centered applications that require calibration. To understand this, let us concretize the crowdsourcing application introduced in Section 4.1. Assume that the task to be performed is throwing a dart at a dartboard, and that each worker is a different darts player. Without prior knowledge of any player's true ability to hit the dartboard, our goal is to discover which of the K players is the best, by picking (at each timestep) a player to throw a dart and seeing whether they hit or miss. At first glance, this seems to be a classical sMAB problem, where each player has some true ability, and each time we query a player we (stochastically) observe their true ability.

Unfortunately, this sMAB formulation is agnostic to the calibration period that darts players require before they can exhibit their true performance. The existence of such a calibration period has been demonstrated in the literature on visuomotor calibration. For instance, Wunderlich et al. [WHFM20] show that when professional darts players toss darts in a row, the first toss is significantly less accurate than the remainder of the darts, although the performance stabilizes after the first dart toss. They attribute this phenomenon to the *warm-up decrement* [Ada61, AW93, Ans95], which describes the decline in performance due to a break in a specific motor skill, as well as its recovery once the skill is resumed. Simply put, a player performs better once they are "in motion" and have fine-tuned their movement parameters after their first toss.

This phenomenon affects the design of algorithms for our darts setting, since we do not observe the true performance of a dart player until after their first toss. Furthermore, this *cannot* be resolved by simply having each player toss once, so that they are calibrated, and then running a standard sMAB algorithm while assuming that the players stay calibrated forever. Indeed, Wunderlich et al. [WHFM20] demonstrate that even small interruptions in the dart tosses (such as the few seconds required for the player to retrieve their darts from the board) can cause the player to "reset", and subconsciously lose their fine-tuned movement parameters. Hence, the sMAB is hence a poor model for this setting. By contrast, the WTB with $m \geq 2$ is a more faithful model, since m describes the number of times a player must toss a dart in a row before we (stochastically) observe their true performance. The "reset" phenomenon that exists in this motivating example (as well as our forthcoming examples) requires that if we model this problem with WTB, then m should be non-trivially smaller than the horizon T. We will assume this throughout our paper.

WTB more naturally models this phenomenon than the aforementioned tallying settings [HKR16, LCM17, SLC+19, SMLV20, LHK21, ABGK22, MLS22], which are all special

cases of the WTB where w_x is the all ones vector $\vec{1}$ for each $x \in \mathcal{X}$. As a stylized example, assume the task is shooting basketball free-throws, and that we need to find the best of two players x_1, x_2. Consider two different sequences of selecting players — x_1, x_2, x_1 versus x_2, x_1, x_1. Phatak et al. [PMR+20] show that players require a calibration period of length at least 3 while shooting free-throws, and that their shooting performance monotonically improves with each successive free-throw. This implies that picking x_1, x_2, x_1 (i.e., x_1 shoots, then x_2, then x_1 again) will cause x_1 to have a worse expected performance on her final shot, relative to her performance if we select x_2, x_1, x_1 (i.e., x_2 shoots, then x_1 shoots twice). If we model this with WTB where $m = 3$ and $w_x = \vec{1}$, then we cannot distinguish these two scenarios, since in both cases x_1 shot twice in the past m timesteps. By contrast, WTB with $w \neq \vec{1}$ allows us to model different losses for these two scenarios. For instance, if we use the model $w_{x_1} = [1, 1/2, 1/4]$ and $h_{x_1}(z) = 1 - z/3$, then this model says that selecting x_2, x_1, x_1 will cause x_1 to have better expected performance on her final shot than if we selected x_1, x_2, x_1.

More broadly, WTB significantly generalizes prior tallying settings, by allowing us to better approximate the decay in memory strength that occurs with passage of time, that has been documented extensively by studies on short and long term human memory [Kla80, RVC16]. This more naturally models the human-centered applications that motivate tallying settings. For instance, Malik et al. [MLS22] motivate their study via recommender systems, arguing that recommended content impacts human preferences, and assume the quantity $m \ll T$ bounds the length of time that a human remembers past recommendations. But their formulation is agnostic to how recently a piece of content was recommended *within* this window of length m. So if some content was recommended k times in the past m timesteps, then their framework requires that this incurs the same loss regardless of the ordering of those k recommendations. This is rather limiting, since human preferences today may depend only mildly on recommendations that occurred $\widetilde{\Omega}(m)$ timesteps ago. Our WTB formulation is more fine grained, and allows for the possibility of different losses incurred by each of the different orderings of those k recommendations.

4.2.2 Complete Policy Regret

A key property of WTB is that the loss incurred by an action depends on the past actions of the algorithm. In such dynamic scenarios, it has been well established that the popular traditional regret is inappropriate to measure the performance of an algorithm [ADT12]. Instead, one typically opts for the stronger notion of policy regret [CBDS13, ADMM18]. In line with prior work on tallying settings [HKR16, LCM17, SLC+19, SMLV20, LHK21, ABGK22, MLS22], we study the minimization of the complete policy regret (CPR), which is the strongest possible notion of regret. Given an (m, w, h)-weighted tallying bandit and an algorithm that plays action sequence $(a_1, a_2 \dots a_T) \in \mathcal{X}^T$, the CPR $\mathcal{R}^{\text{cp}}$ of the algorithm

is defined as

$$\mathcal{R}^{\text{cp}} = \sum_{t=1}^{T} h_{a_t} \left(w_{a_t}^\top y^{t,a_t,m} \right) - \min_{(x_1,x_2 \ldots x_T) \in \mathcal{X}^T} \sum_{t=1}^{T} h_{x_t} \left(w_{x_t}^\top y^{t,x_t,m} \right). \tag{6}$$

Following prior convention, we refer to any length T sequence of actions (i.e., an element of $\mathcal{X}^T$) as a policy. The CPR is hence the cumulative loss experienced by the algorithm, relative to the minimum loss achieved by the best policy in $\mathcal{X}^T$. Minimizing CPR is hence equivalent to minimizing the cumulative expected loss of the algorithm, and we note that this performance metric is identical to the one used in reinforcement learning [JAZBJ18, WDYK20, IGS22]. If the CPR of an algorithm is sublinear in T and polynomial in m, K then we say it has statistically efficient CPR.

Prior work has shown that in the case of WTB with $w_x = \vec{1}$ for each $x \in \mathcal{X}$, without any further assumption, there exists an algorithm with statistically efficient CPR [MLS22]. Unfortunately, the following result shows that such an algorithm does not exist in WTB with $w_x \neq \vec{1}$.

Proposition 2. *For any $m \geq 1$, there exists an (m, w, h)-weighted tallying bandit with $K = 2$ such that the following is true. Any (possibly randomized) algorithm has expected CPR satisfying $\mathbb{E}[\mathcal{R}^{cp}] = \widetilde{\Omega}(\min\{2^m, T\}/m)$.*

The proof of Proposition 2 is deferred to Appendix C.3. At a high level, the proof shows that if $w_x \neq \vec{1}$ then h_x can take on $\widetilde{\Omega}(2^m)$ different values, and so discovering the optimal sequence of actions requires $\widetilde{\Omega}(2^m)$ queries. This result demonstrates that if we desire an algorithm with statistically efficient CPR, then we must impose structure on the WTB setting that restricts the set of optimal action sequences. We motivate and formalize such structure in the sequel.

4.2.3 Repeated Exposure Optimality

To motivate additional structure in the types of problems that are modeled by WTB, we recall the darts setting illustrated in Section 4.2.1. Notably, if we ask a player to toss darts in a row, then on the *first* toss, their uncalibrated performance is poor and not necessarily indicative of their subsequent performance. But on successive tosses after the first toss, Wunderlich et al. [WHFM20] show that their calibrated performance stabilizes and is *better* than the uncalibrated performance on the first toss. A similar observation holds for shooting free-throws [PMR+20]. So if we let $x^\star \in \mathcal{X}$ denote the player with the best calibrated performance, then this implies that the calibrated performance $x^\star$ is better than not only the calibrated performances of player $x \neq x^\star$, but also the uncalibrated performances of all players. We formalize this insight in the following condition.

Definition 6 (Repeated Exposure Optimality (α-REO)). *An (m, w, h)-weighted tallying bandit satisfies the Repeated Exposure Optimality condition with parameter α, if*

there exists an action $x^\star \in \mathcal{X}$, such that for each $x \in \mathcal{X}$ and each $y \in \{1\} \times \{0,1\}^{m-1}$ we have

$$h_{x^\star}\left(\|w_{x^\star}\|_1\right) \leq h_x\left(w_x^\top y\right) + \alpha.$$

The α-REO condition thus requires that there is some action $x^\star \in \mathcal{X}$, which when played repetitively for at least m times in a row, will have smaller loss (upto the suboptimality α) than other action sequences. Two remarks are in order, to understand this condition in the context of prior work. First, observe that even when we additionally impose the α-REO condition on WTB, the sMAB remains a special case of this setting via a choice of $\alpha = 0, m = 1$. Second, significant prior work on tallying settings has focused on when the loss functions $\{h_x\}_{x\in\mathcal{X}}$ are monotonic. For instance, the improving bandit [HKR16] is a special case of WTB under significant additional restrictions, including (but not limited to) the facts that $\{w_x\}_{x\in\mathcal{X}} = \{\vec{1}\}$ and $\{h_x\}_{x\in\mathcal{X}}$ are decreasing. We note that this property of decreasing $\{h_x\}_{x\in\mathcal{X}}$ functions is a special case of the 0-REO condition.

We have motivated REO via the warm-up decrement phenomenon that has been extensively demonstrated in the psycho-physiological literature. And we believe REO may also be relevant in other interactive settings such as recommender systems, as we discuss in Section 4.6. Nevertheless, we acknowledge that our setting is ultimately a mathematical abstraction that falls short of ground truth reality, and fails to model many subtleties that make human-centered applications challenging. A complete formulation and study of all these subtleties is beyond the scope of our paper, and we relegate discussion of important avenues for future work to Section 4.6.

With the REO condition thus motivated and formalized, the following question is natural:

Consider any (m, w, h)-weighted tallying bandit satisfying α-REO. Is there a computationally efficient and practical algorithm that solves this problem with a statistically efficient CPR guarantee?

The remainder of our paper's analysis is devoted to answering this question.

4.3 Main Results

We present two categories of results. In Section 4.3.1 we present a statistically and computationally efficient algorithm that can solve WTB problems satisfying REO. This method requires only an upper bound M on the true memory capacity m, whose exact value is often unknown. In Section 4.3.2, we show the impossibility of an algorithm that is fully adaptive to an unknown m (i.e., does not require knowledge of an upper bound $M < T$ on m). We also show that if such an upper bound $M < T$ on m is known, then the dependency of our method on M is optimal.

4.3.1 A Statistically & Computationally Efficient Algorithm

Before we present our algorithm, let us consider some natural approaches. Since the WTB is a subclass of reinforcement learning (RL) problems, one may attempt to use RL algorithms to solve it. But even when $\{w_x\}_{x\in\mathcal{X}} = \{\vec{1}\}$, such algorithms will suffer $\widetilde{\Omega}(K^m)$ CPR [ABGK22, MLS22]. One may also attempt to extend the classical UCB algorithm from sMAB to WTB as follows. Solve the problem in epochs of length m, where at the beginning of each epoch, we select the action that minimizes the usual UCB estimate, and then play it m times in a row instead of just once. Then we record the loss observed in the most recent play, since this is an unbiased estimate of the action's eventual loss, and use it to update the action's UCB estimate. While this seems like a reasonable heuristic, each epoching has an m-length overhead which can substantially increase regret.

Algorithm 3 Successive Elimination for WTB

Require: upper bound M on memory capacity m, time horizon T, failure probability tolerance $\delta \in (0,1)$, number of actions K

Define $S = \log_2\left(\frac{T}{4KM} + 1\right)$.
Define $A_1 = \mathcal{X}$.
Define $n_s = KM2^s/|A_s|$, $T_s = 2|A_s|n_s$ and $C_s = \sqrt{\frac{32}{n_s}\log\left(\frac{2KS}{\delta}\right)}$.
for $s \in \{1, 2 \dots S\}$ **do**
    for $x \in A_s$ **do**
        Execute action x for $n_s \geq m$ times and store nothing.
        Execute action x for n_s times and store $\{\widetilde{h}_x(\|w_x\|_1)_{s,k}\}_{k=1}^{n_s}$.
        Define $\widehat{\mu}_s(x) = \frac{1}{n_s}\sum_{k=1}^{n_s}\widetilde{h}_x(\|w_x\|_1)_{s,k}$.
    end for
    Select $\widehat{x}_s \in \operatorname{argmin}_{x\in A_s}\widehat{\mu}_s(x)$.
    Construct $A_{s+1} = \{x \in A_s \text{ s.t. } \widehat{\mu}_s(x) \leq \widehat{\mu}_s(\widehat{x}_s) + 2C_s\}$.
end for

A different idea is to adapt algorithms from prior tallying settings for our problem. But prior tallying settings that are most comparable to ours all have CPR bounds that scale multiplicatively with m (see Section 4.5 for details). Our key theoretical contribution is to demonstrate that due to the additional presence of REO, we can solve not just these tallying settings but also WTB with a CPR guarantee that is only *additive* (in lieu of multiplicative) in m. The algorithm that achieves this bound is a slightly modified version of successive elimination (SE), and is presented in Algorithm 3. Our inspiration for this is due to Malik et al. [MLS22], who adapt SE for their tallying bandit setting, although their modification is more involved. By contrast, our modification is very simple, since REO permits us to only estimate the eventual losses of each action. We now present our main result, which bounds the CPR of this algorithm.

Theorem 3. *Fix any (m, w, h)-weighted tallying bandit problem satisfying Repeated Exposure Optimality with parameter α. When Algorithm 3 is run with inputs $M \geq m$ and $\delta \in (0, 1)$, then with probability at least $1 - \delta$ it has complete policy regret upper bounded as*

$$\mathcal{R}^{cp} \leq 4KM + Km\log(T) + 800\sqrt{KT\log\left(\frac{2K\log(T)}{\delta}\right)} + \alpha T. \tag{7}$$

The proof of Theorem 3 is deferred to Appendix C.1. Let us highlight some key aspects of this result.

Comparison to sMAB & Tallying Settings. Recall that in the classical sMAB, which is a special case of WTB with 0-REO via $m = 1$, any algorithm suffers $\widetilde{\Omega}\left(\sqrt{KT}\right)$ traditional regret. Theorem 3 thus shows that the much larger class of WTB with $\widetilde{\mathcal{O}}\left(\sqrt{K/T}\right)$-REO problems can be solved with essentially this guarantee on CPR, upto a logarithmic factor and an additive dependence on mK. Our guarantee scales more favorably than those obtained for prior comparable tallying settings (see Section 4.5 for details).

Efficiency & Practicality. Algorithm 3 is computationally efficient and scalable. This is in contrast to results on prior comparable tallying settings (see Section 4.5 for details). Moreover, implementing Algorithm 3 does not require exact knowledge of unknown quantities such as $\{h_x\}_{x\in\mathcal{X}}$, $\{w_x\}_{x\in\mathcal{X}}$, α or m; an upper bound M on m suffices.

Statistical Optimality In Various Regimes. In the regime where m is known (so $M = m$) and REO is satisfied with $\alpha = 0$, the guarantee of Theorem 3 is optimal within a single logarithmic factor. To see this, note that in the RHS of Eq. (7), the Km term cannot be improved due to Proposition 1 of Malik et al. [MLS22], and the $\sqrt{KT}$ term is of course tight due to the classical sMAB lower bound. Moreover, when m is known and REO is satisfied with $\alpha = \widetilde{\Theta}\left(\sqrt{mK/T}\right)$, then the proof of Theorem 2 of Malik et al. [MLS22] shows that there is a regime of non-trivial $0 < \alpha \ll 1$ where the dependence on αT in Theorem 3 cannot be improved, and so Theorem 3 is optimal (within a logarithmic factor). We note that it is unclear whether the αT term in Eq. (7) is optimal for *all* $\alpha > 0$, and investigating this is an interesting future direction. We defer our investigation into the optimal dependency on M, in the regime where m is unknown, to Section 4.3.2.

The proof of Theorem 3 requires some care to ensure optimal dependencies, but the technique is standard, and our contribution is not a novel analysis route. Rather, our contribution is to demonstrate that a classical algorithm for the canonical sMAB can be easily adopted to solve a much more general, and ostensibly more complex, class of problems that are very well motivated in practice. The prior tallying settings that are comparable to our WTB have inherent computational and statistical difficulties (see Section 4.5 for details). We

believe that our formalization of REO and Theorem 3 is an important identification of well motivated structure that permits statistically and computationally efficient solutions to problems arising in interactive human-centered applications.

4.3.2 Adaptivity To Memory Capacity

While Algorithm 3 does not require knowledge of the true memory capacity m, it does require an upper bound M on m. Theorem 3 suggests that the CPR of Algorithm 3 scales linearly in this input M, which is disadvantageous in scenarios where it is difficult to non-trivially upper bound m. In general, we desire an algorithm which scales more favorably (or not at all) with the input M. For instance, this could be achieved via an algorithm that maintains a confidence interval of the true value m, and adaptively queries to refine its estimate of m, in order to improve or remove its dependency on M. We now show that such an algorithm cannot exist, even in the simpler "tallying setting" that is a special case of WTB, and in the case when REO is satisfied with parameter $\alpha = 0$.

To this end, we introduce some notation. For any positive integers T, M, K with $M \leq T$, let $\mathsf{UTB}_{T,M,K}$ denote the set of unweighted tallying bandit problems (i.e., WTB problems where w_x is the all ones vector for each action x), that each have horizon length T, number of actions K, and memory capacity $m \in \{1, 2 \dots M\}$, and that satisfy 0-REO. For any possibly randomized algorithm $\mathcal{A}$ and any unweighted tallying bandit problem tb, let m_{tb} denote the memory capacity of tb, and let $\mathcal{R}^{\mathrm{cp}}(\mathcal{A}, \mathsf{tb})$ denote the expected CPR of algorithm $\mathcal{A}$ when it is used to solve tb. And for a choice of $\epsilon = (\epsilon_1, \epsilon_2, \epsilon_3)$ satisfying $\epsilon_1, \epsilon_2 \in (0, 1)$ and $\epsilon_3 \in [0, \epsilon_2)$, and a choice of function $f : \mathbb{R}^2 \to \mathbb{R}$, let $\overline{\mathcal{A}}_{\epsilon,f}$ be the set of algorithms $\mathcal{A}$ which, when given as input any positive integers T, M, K with $M \leq T$ (and no other information), satisfy for each problem instance $\mathsf{tb} \in \mathsf{UTB}_{T,M,K}$ that

$$\mathbb{E}[\mathcal{R}^{\mathrm{cp}}(\mathcal{A}, \mathsf{tb})] \leq \min\left\{T/4, f(m_{\mathsf{tb}}, K)\left(T^{1-\epsilon_1} + T^{\epsilon_3} M^{1-\epsilon_2}\right)\right\}. \tag{8}$$

An algorithm $\mathcal{A}$ in the set $\overline{\mathcal{A}}_{\epsilon,f}$ thus has a benign dependence on M in the following sense. When given any positive integers T, M, K with $M \leq T$, and any problem instance $\mathsf{tb} \in \mathsf{UTB}_{T,M,K}$, the algorithm $\mathcal{A}$ does not a priori know the memory capacity m_{tb} of tb, and only knows the upper bound M. Nevertheless, the CPR of $\mathcal{A}$ when solving tb scales *sublinearly* in the bound M. Unfortunately, the following result demonstrates that such an algorithm does not exist.

Theorem 4. *For each ϵ satisfying $\epsilon_1, \epsilon_2 \in (0, 1)$ and $\epsilon_3 \in [0, \epsilon_2)$, and each function f, the corresponding set $\overline{\mathcal{A}}_{\epsilon,f}$ is the empty set.*

The proof is deferred to Appendix C.2. The result demonstrates a "price for adaptivity" (see, for example, [LC18] for similar results in a different context), showing that if we only have an upper bound M on the unknown true memory capacity, then any algorithm's CPR cannot be sublinear in both M and T. We concretize this via two salient corollaries. The

following corollary is stated for the case when we have no non-trivial bound on the memory capacity, or equivalently that $M = T$.

Corollary 1. *Fix any function $f : \mathbb{R}^2 \to \mathbb{R}$. There is no (possibly randomized) algorithm which has expected CPR bounded by $\widetilde{o}(T) f(m_{\mathsf{tb}}, K)$ for each $\mathsf{tb} \in \mathsf{UTB}_{T,T,K}$.*

The result of Corollary 1 shows that it is impossible to have an algorithm whose CPR is sublinear in T for all unweighted tallying bandit instances tb with horizon T that satisfy 0-REO, even at the expense of arbitrarily poor dependence on m_{tb}, K. Thus, to obtain a sublinear CPR guarantee of the sort afforded by Theorem 3, it is necessary to have knowledge of some bound $M < T$ on the true memory capacity. The next corollary is stated for when we have a non-trivial bound $M < T$ on the memory capacity.

Corollary 2. *Fix any function $f : \mathbb{R}^2 \to \mathbb{R}$. There is no (possibly randomized) algorithm, which given an input M, has expected CPR bounded by $f(m_{\mathsf{tb}}, K)\left(\widetilde{o}(T) + \widetilde{o}(M)\right)$ for each $\mathsf{tb} \in \mathsf{UTB}_{T,M,K}$.*

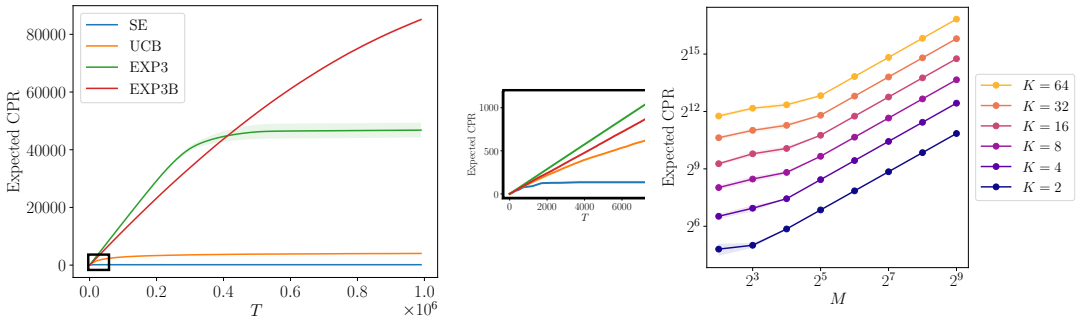
The result thus shows that we cannot hope to have an algorithm with sublinear dependency on both M and T for all unweighted tallying bandit instances tb with horizon T that satisfy 0-REO, even at the expense of arbitrarily bad dependence on K, m_{tb}. Note that ignoring logarithmic factors, Theorem 3 shows that Algorithm 3 has CPR bounded by $\widetilde{\mathcal{O}}\left(\sqrt{KT} + K(M+m)\right)$ for each $\mathsf{tb} \in \mathsf{UTB}_{T,M,K}$. Combined with our earlier discussion of Theorem 3, Corollary 2 thus shows that any algorithm must suffer $\widetilde{\Omega}\left(\sqrt{KT} + mK + M\right)$ CPR, highlighting the near optimality of Algorithm 3 for WTB problems satisfying 0-REO, even in the regime when we only have an upper bound M on the unknown true memory capacity.

4.4 Numerical Results

In this section, we evaluate the performance of Algorithm 3, which we denote SE, in different domains which are modeled as WTB problems satisfying REO. In each domain, we compare this performance to the following three baselines — (A) The EXP3 algorithm [ACBFS02], which has sublinear traditional regret in our setting (B) The batched version of EXP3 described by Arora et al. [ADT12], denoted as EXP3B, which has a statistically efficient CPR guarantee in our setting (C) The modified UCB algorithm described in Section 4.3.

4.4.1 Synthetic Loss Functions on Unweighted Tallying Bandit

We consider $\{w_x\}_{x \in \mathcal{X}} = \{\vec{1}\}$, and fix some $x^\star \in \mathcal{X}$. We define $h_x = 0.5$ for each $x \in \mathcal{X}$, with the modification that $h_{x^\star}(\|w_{x^\star}\|_1) = 0.35$. Hence, the losses are identical, except until we play $x^\star$ at least m times, implying that this instance satisfies 0-REO. To define the feedback model, we require the random variable $\widetilde{h}_x(w_x^\top y^{t,x,m})$ has distribution



(a) CPR over time horizon $T = 10^6$.

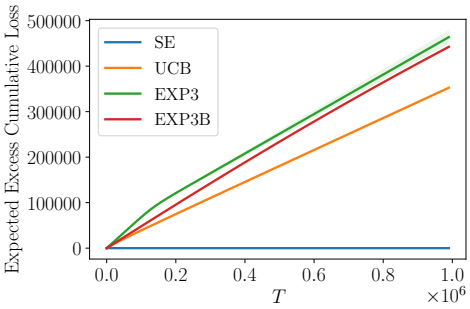
(b) CPR as a function of M.

Figure 5: We plot the expected CPR of each algorithm. In both plots, each datapoint is obtained by averaging over 20 problem instances, and the shaded region depicts ± 1 standard error around the mean. In (a) we fix $K = 5$, $m = 3$ and $M = 3$. In (b) we fix $m = 4$ and $T = 10^6$.

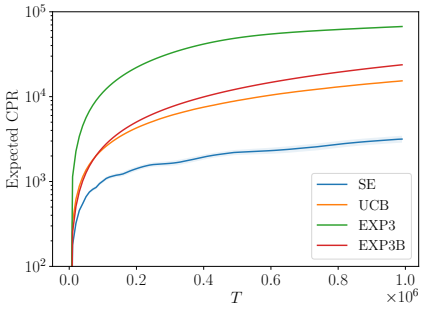
Bernoulli$(h_x(w_x^\top y^{t,x,m}))$. When $m = 1$, this is a hard instance for sMAB [Sli19], and UCB is of course optimal. For $m > 1$, we note that the UCB variant will perform best in regimes where $h_x \approx h_x(\|w_{x^\star}\|_1)$, since the loss incurred during steps of the m-length overhead are nearly equivalent to the eventual losses of repetitively playing an action. Hence, we consider our experimental design to be as favorable to the UCB variant as possible. In Figure 5a, we plot the expected CPR of each method over time. As expected, SE outperforms each baseline. In Appendix C.4.1, we present similar results for other choices of m, K, M, and also present results for a problem where α-REO is satisfied with $\alpha > 0$. Separately, we study the deterioration of the performance of SE as a function of its input M, for the same fixed m, T, K. In Figure 5b, we observe that the CPR of SE is at most a linear function of M, as one would expect from Theorem 3. However, we also see that in several cases, the scaling is sublinear and hence better than the worst case linear scaling predicted by Theorem 3.

4.4.2 Synthetic Loss Functions on Weighted Tallying Bandit

We now consider a WTB problem satisfying REO where $\{w_x\}_{x \in \mathcal{X}} \neq \{\vec{1}\}$. We relegate the discussion of the precise loss functions used to Appendix C.4.2. Since the optimal policy is difficult to compute for this problem, the CPR is also difficult to compute. So in lieu of the CPR, we plot the expected cumulative loss of each algorithm in excess of SE's loss (hence the CPR at any time is obtained by applying a constant shift to each algorithm's excess loss). The results are shown in Figure 6a, and demonstrate the superiority of our method over the baselines.



(a) Excess loss in WTB with $\{w_x\}_{x\in\mathcal{X}} \neq \{\vec{1}\}$.



(b) CPR in darts tournament.

Figure 6: In (a), we plot as a function of time the expected cumulative loss of each algorithm in excess of that of SE, in the WTB instance where $\{w_x\}_{x\in\mathcal{X}} \neq \{\vec{1}\}$ described in Section 4.4.2, with $K = 5$, $m = 4$ and $M = 4$. In (b), we plot as a function of time the expected CPR of each algorithm in the simulated darts tournament described in Section 4.4.3, and truncate the y-axis below 10^2 for illustrative purpose. In both (a) & (b), data is obtained by averaging over 20 problem instances, and the shaded region depicts ± 1 standard error around the mean.

4.4.3 Simulated Dart Throwing Tournament

Motivated by prior work showing the existence of a calibration period in motor tasks [Ada61, PMR+20, WHFM20], we simulate a simplified dart throwing tournament with $K = 20$ players. As discussed in Section 4.2.3, Wunderlich et al. [WHFM20] show that while a player's first toss is uncalibrated and not necessarily indicative of their subsequent performance, in immediately subsequent tosses the performance calibrates, stabilizes and is better than that of the first toss. We model each (random) instance of the tournament as a WTB with $m = 2$ and arbitrary w, where each player $x \in \mathcal{X}$ has expected loss function sampled from $h_x(w_{x,1}) \sim \text{Unif}[0.68, 0.72]$ and $h_x(\|w_x\|_1) \sim \text{Unif}[0.58, 0.62]$. We obtained the bounds for these distributions from Wunderlich et al. [WHFM20], who showed that most players' average performance was concentrated in these intervals. To define the feedback model, we require the random variable $\widetilde{h}_x(w_x^\top y^{t,x,m})$ has distribution $\text{Bernoulii}(h_x(w_x^\top y^{t,x,m}))$. While our experimental design eschews some real world subtleties that may occur while throwing darts (for instance, missing a throw might affect the player's confidence on the next throw), we believe that it is a reasonable preliminary model for the calibration period required to throw darts optimally. In Figure 6b, we plot the CPR of each method over time. As expected, SE performs significantly better than each baselines.

4.4.4 Simulated F1 Tournament

In a Formula One (F1) tournament, the goal is to discover the fastest driver out of a set of K drivers. Each driver in the tournament must complete a number of laps. We simulate a

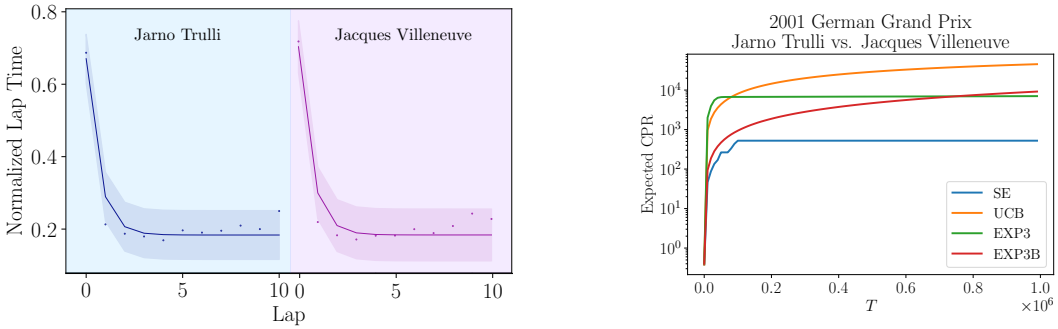
modified version of an F1 tournament, where $K = 2$, and at each timestep we pick one of the two drivers to complete a lap (i.e., only a single driver can be on the track at any given timestep). After a lap is completed, we observe (a stochastic realization of) the driver's lap time.

Notably, a driver's lap time depends on the number of laps they have previously completed. To demonstrate this, we utilize F1 lap time data from 1950 to 2022 [Rao22] to fit a probabilistic lap time model for each F1 driver (details of our probabilistic model and data processing are provided in Appendix C.4.3). In Figure 7a, we illustrate our probabilistic model of lap times for a typical driver pair, and show that their lap times tend to decrease as the lap index increases. There are several plausible reasons for this; for instance, the mass of the driver's vehicle decreases with fuel consumption, and the driver's calibration to the race track improves. We thus argue that the sMAB is a poor model for our F1 tournament. Instead it is better modeled as a WTB problem satisfying REO, and our tournament's goal is to discover the driver with the fastest calibrated lap time, which we only observe after repeated exposure.

We simulate multiple instances of our modified F1 tournament, each with $K = 2$. The two drivers for each instance are chosen such that their calibrated performance is difficult to distinguish (details in Appendix C.4.3) Here, we present results for a single instance. The results for other instances are presented in Appendix C.4.3. For this instance, we use the probabilistic model depicted in Figure 7a to a create a WTB problem with $K = 2$ and $m = 10$. We maintain a tally of the number of times each driver was chosen in the prior m timesteps. The loss associated with picking a driver is governed by the distribution parameterized by our fitted probabilistic model. In particular, if we pick driver x and we have picked them y times in the last m timesteps, then the instantaneous loss is sampled from the distribution parameterized by our fitted probabilistic model for driver x at lap index y. Note that in this setting, one has $o(T)$ CPR if and only if one plays the worse driver $o(T)$ many times. In Figure 7b, we plot each method's CPR over time for this tournament instance, showing that SE outperforms the baselines. Similar results are observed for the other tournament instances shown in Appendix C.4.3.

4.5 Related Work

Tallying Settings with $m = T$. A significant thrust of prior work studies tallying settings that are special cases of WTB with $\{w_x\}_{x\in\mathcal{X}} = \{\vec{1}\}$, and require that $m = T$ [HKR16, LCM17, SLC+19, SMLV20, LHK21, MTPR22]. Of course, to ensure tractability they enforce various additional types of assumptions, typically in the form of monotonicity on the $\{h_x\}_{x\in\mathcal{X}}$ functions. Results here do not apply to the case when $m < T$, because $m < T$ causes complications in the design of algorithms since an action's loss "resets" if it is not played. Since our paper is primarily motivated by applications where $m < T$, we do not view these works as directly comparable to ours. Nevertheless, we note that upto an additive factor in mK and a logarithmic factor, the CPR guaran-



(a) Illustration of probabilistic lap time model.

(b) CPR over time horizon $T = 10^6$.

Figure 7: In (a), we depict our fitted probabilistic lap time model for two drivers in the 2001 German Grand Prix. Our probabilistic model parameterizes a distribution over lap times for each lap index from 1 to 10. The solid line depicts the mean of this distribution, for each lap index. The shaded region contains ± 2 standard deviations of this distribution, centered around the distribution's mean. The dotted points are the actual lap times. Note that all the lap times are normalized, so that each lap time lies in the interval $[0, 1]$ (see Appendix C.4.3 for details). In (b), we plot as a function of time the expected CPR of each algorithm. Data is obtained by averaging over 20 problem instances, each with $K = 2$, $m = 10$, $M = 10$ and $T = 10^6$, and the shaded region depicts ± 1 standard error around the mean.

tees in all these works generally scale less favorably than the rates provided by our Theorem 3.

Tallying Settings with $m < T$. A different body of prior work studies tallying settings that are special cases of WTB with $\{w_x\}_{x \in \mathcal{X}} = \{\vec{1}\}$, and like us, they are motivated by applications where $m < T$ [ABGK22, MLS22]. These settings are more comparable to ours, since they do not enforce that $m = T$. The tallying bandit [MLS22] makes no assumptions beyond $\{w_x\}_{x \in \mathcal{X}} = \{\vec{1}\}$, and here any algorithm must suffer $\widetilde{\Omega}\left(\sqrt{mKT}\right)$ CPR. They adapt successive elimination, and our algorithm is heavily inspired by theirs. The congested bandit [ABGK22] specializes the tallying bandit by requiring that the $\{h_x\}_{x \in \mathcal{X}}$ functions are increasing, and even here the best known upper bounds scale as $\widetilde{\mathcal{O}}\left(\sqrt{mKT}\right)$. The best CPR bounds of these settings thus seem to scale multiplicatively with m. Moreover, the computational complexities of the best known algorithms in these settings scale exponentially in T, m respectively. By contrast, our REO condition allows for a computationally efficient algorithm achieving a statistically optimal CPR guarantee that is only additive in m.

Related Non-Tallying Settings. A massive body of work studies various settings where the loss of each action evolves over time in some structured fashion, for instance, according to some stochastic process or according to the number of timesteps since the action was last played [Whi81, GM11, TL12, BGZ14, BF16, KI18, BSSS19, PBG19, CCB20, CDK+20,

LCCG21]. The models for the evolution of loss in all these works are different than our (weighted) tallying setting.

Policy Regret. Many works study policy regret against generic m-memory bounded adversaries, and their algorithms apply to our setting [ADT12, CBDS13, DDKP14, ADMM18, MY18]. However, these results ignore the special weighted tallying structure that we consider, and a direct application would result in a suboptimal CPR bound that is worse than our Theorem 3.

4.6 Discussion

In this paper, we formulated the Weighted Tallying Bandit, which generalizes prior tallying settings so that the loss at a timestep is a function of a weighted summation of the number of times it was recently played. To ensure tractability in this challenging setting, we introduced the Repeated Exposure Optimality condition, which we motivated via human-centered applications where one's best performance requires a calibration period before it stabilizes. We showed that a simple modification of the classical successive elimination algorithm achieves an optimal complete policy regret guarantee (upto a single logarithmic factor), and our numerical results demonstrate its practicality, scalability and superiority over alternative baselines. Finally, we showed that while our algorithm required as input a non-trivial upper bound $M < T$ on m, any algorithm that has sublinear CPR requires such an input, and that our method's dependency on this input M is optimal. Collectively, this implies our algorithm's CPR is optimal for WTB problems satisfying 0-REO.

We acknowledge our work has certain limitations. From a theoretical perspective, while there is a regime of non-trivial $\alpha = \widetilde{\Theta}\left(\sqrt{mK/T}\right)$ (as discussed in Section 4.3.1) where Theorem 3 is optimal and its dependence on α cannot be improved, it is unclear whether this dependence is optimal for *all* values of α. Investigating this is an interesting direction for future work.

More practically, a limitation of our WTB and REO setting is that while it is a reasonable step to model the calibration period that arises before we see true best performance, it fails to model many other subtleties that arise in the human-centered domains that motivate our work to begin with. For instance, it is plausible that in more strenuous tasks, after repeatedly performing a task for a long $m < T$, a calibrated individual may begin to experience fatigue. In this case, the best model for losses associated with repeatedly playing an action would be an initial period where the individual calibrates and their performance improves to "sweet-spot", but then their performance eventually deteriorates as fatigue accumulates (this model is analogous to the $m = T$ setting considered by [LHK21]). Our setting can handle the initial period of calibration and performance improvement, but cannot handle the latter phase of deterioration. We believe that exploring algorithms that can handle this is a key direction for future work.

A different way to improve our model, is by studying more general settings where each

action x is associated with its own memory capacity m_x (instead of a single m for all actions), or where the memory capacity of the problem changes with time. Can we design algorithms that intelligently adapt to such complexities?

Our concrete motivating examples for the REO condition are primarily derived from the psycho-physiological literature on the warm-up decrement phenomenon. Nevertheless, we generally expect that it may additionally apply to other interactive settings such as recommender systems. For instance, it is plausible that a user needs to see a type of content multiple times before she decides her preferences for it, but if she is not shown the content for a while, then she forgets its details and requires another exploratory calibration period to re-affirm her preference for that content relative to more recent recommendations. In this case, the goal is to explore the *eventual* preferences of the user, and then repetitively select the item that the user *eventually* prefers the most. Such a setting would be a reasonable application for WTB with REO. Quantitatively analyzing recommender system data, and showing that such settings exist and can be modeled well by WTB with REO, is a very interesting direction for future work with potentially broad practical impact.

5 Specifying and Solving Robust Empirical Risk Minimization Problems Using CVXPY

The content of this section is based on [LML+24].

Abstract

We consider robust empirical risk minimization (ERM), where model parameters are chosen to minimize the worst-case empirical loss when each data point varies over a given convex uncertainty set. In some simple cases, such problems can be expressed in an analytical form. In general the problem can be made tractable via dualization, which turns a min-max problem into a min-min problem. Dualization requires expertise and is tedious and error-prone. We demonstrate how CVXPY can be used to automate this dualization procedure in a user-friendly manner. Our framework allows practitioners to specify and solve robust ERM problems with a general class of convex losses, capturing many standard regression and classification problems. Users can easily specify any complex uncertainty set that is representable via disciplined convex programming (DCP) constraints.

5.1 Robust empirical risk minimization

Robust optimization is used in mathematical optimization, statistics, and machine learning, to handle problems where the data is uncertain. In this note we consider the robust empirical risk minimization (RERM) problem

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n \sup_{x_i \in \mathcal{X}_i} f(x_i^T\theta - y_i) \\ \text{subject to} & \theta \in \Theta, \end{array} \tag{9}$$

with variable $\theta \in \mathbf{R}^d$. Here, $\Theta \subseteq \mathbf{R}^d$ is closed and convex, $\mathcal{X}_i \subset \mathbf{R}^d$ is compact and convex for each $i = 1, \ldots, n$, $f : \mathbf{R} \to \mathbf{R}$ is convex and $\{x_i, y_i\}_{i=1}^n$ is a dataset. The objective is to find $\theta \in \Theta$ that minimizes the worst-case value of $\sum_{i=1}^n f(x_i^T\theta - y_i)$ over all possible x_i in the given uncertainty sets $\mathcal{X}_i$. Beyond convexity, we will assume that f is either non-increasing, or f is non-decreasing on $\mathbf{R}_+$ and a function of the absolute value of its argument, *i.e.*, $f(z) = f(|z|)$.

Examples. Our assumptions capture a wide range of loss functions in both regression and classification, including the following.

- *Finite p-norm loss.* $f(z) = |z|^p$ for $1 \leq p < \infty$.
- *Huber loss.* $f(z) = \frac{1}{2}z^2$ for $|z| \leq \delta$, and $f(z) = \delta|z| - \frac{\delta^2}{2}$ for $|z| > \delta$, where $\delta > 0$ is a parameter.
- *Hinge loss.* $f(z) = \max(0, 1 - z)$.

- *Logistic loss.* $f(z) = \log(1 + \exp(-z))$.
- *Exponential loss.* $f(z) = \exp(-z)$.

Our formulation includes the case of using hinge, logistic, or exponential loss for binary classification, by solving (9) with the transformed dataset $\{y_i x_i, 0\}$.

5.1.1 Solving RERM problems

The problem (9) is convex, but not immediately tractable because of the suprema appearing in the worst-case loss terms. It can often be transformed to an explicit tractable form that does not include suprema.

Analytical cases. In some simple cases we can directly work out a tractable expression for the worst-case loss. As a simple example, consider $\mathcal{X}_i = \{x_i \mid \|x_i - \tilde{x}_i\|_2 \leq \rho\}$, where $\rho > 0$. When f is non-increasing, the worst-case loss term is

$$\sup_{x_i \in \mathcal{X}_i} f(x_i^T \theta - y_i) = f(\tilde{x}_i^T \theta - y_i - \rho\|\theta\|_2).$$

When f is non-decreasing on $\mathbf{R}_+$ with $f(z) = f(|z|)$, the worst-case loss term is

$$\sup_{x_i \in \mathcal{X}_i} f(x_i^T \theta - y_i) = f(|\tilde{x}_i^T \theta - y_i| + \rho\|\theta\|_2).$$

Both righthand sides are explicit convex expressions that comply with the disciplined convex programming (DCP) rules. This means they can be directly typed into domain specific languages (DSLs) for convex optimization such as CVXPY [DB16].

Dualization. For more complex uncertainty sets the problem (9) can still be transformed to a tractable form, using *dualization* of the suprema apprearing in the worst-case loss terms. This dualization process converts the suprema in (9) to infima, so that the problem can be solved by standard methods as a single minimization problem. Unfortunately, this dualization procedure is cumbersome and error-prone. Many practitioners are not well versed in this procedure, limiting its use to experts. Moreover, a key step in this procedure involves writing down a conic representation of $\mathcal{X}_i$. Such a calculation is antithetical to the spirit of DSLs such as CVXPY, which were introduced precisely to alleviate users of this burden.

Automatic dualization via CVXPY. In this note we show how CVXPY can be used to conveniently solve (9) with just a few lines of code, even when the uncertainty sets $\mathcal{X}_i$ are complicated. We also demonstrate how DSP, a recent DSL for disciplined saddle programming [SLB23] that is based on CVXPY, can solve the RERM problem (9) with

the same ease and convenience. In both approaches no explicit dualization is needed, and the code is short and naturally follows the math. We demonstrate our approach with a synthetic regression example that, however, uses real data, where the uncertainty sets are intervals intersected with a Euclidean ball.

5.1.2 Previous and related work

Robust optimization and saddle problems. Robust optimization is an approach that takes into account uncertainty, variability or missing-ness of problem parameters [BTEGN09]. Saddle problems are robust optimization problems that include the partial supremum or infimum of convex-concave saddle functions. While (9) is not a priori a saddle problem, we can solve it via DSP [SLB23], a recently introduced DSL for saddle programming.

RERM. In machine learning and statistics, it is common to learn a robust predictor or classifier by solving (9) with appropriate choices of $f, \mathcal{X}_i$ [EGL97, XCM08, XCM09, BBC11]. When each $\mathcal{X}_i$ has benign structure, then (9) admits convenient reformulation for many choices of f [BV04]. As an example, such reformulations have been applied to learn linear regression functions when the feature matrix has missing data, and the features are known to lie with high probability in an ellipsoid, so that (9) is easily written as an SOCP [SBS06, AFG22]. However, when $\mathcal{X}_i$ is not a simple set such as an ellipsoid or box, then prior techniques reformulate (9) by writing $\mathcal{X}_i$ in conic form and then dualizing [BTEGN09].

5.2 Reformulating the RERM problem

Throughout, our only requirement on the uncertainty sets $\mathcal{X}_i$ is that each is a compact, convex set that can be expressed via DCP constraints. This includes canonical scenarios, such as when $\mathcal{X}_i$ is a polytope, or is a norm ball centered at a nominal value. But it also includes many complex uncertainty sets, such as the intersection of a norm ball and a polytope. We now reformulate (9) in a manner that permits easy specification and solution via CVXPY, under various monotonicity assumptions on f. Recall that the support function of a non-empty closed convex set C is given by $\mathcal{S}_C(\theta) = \sup\{x^T\theta : x \in C\}$, which is a fundamental object in convex analysis [BV04].

Introducing the epigraph variables $c \in \mathbf{R}^n$, the problem (9) is straightforwardly equivalent to

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n c_i \\ \text{subject to} & \theta \in \Theta, \\ & \sup_{x_i \in \mathcal{X}_i} f(x_i^T\theta - y_i) \leq c_i, \quad i = 1, \ldots, n. \end{array} \tag{10}$$

with variables $c \in \mathbf{R}^n, \theta \in \mathbf{R}^d$. We now use the assumptions on f to rewrite the constraints $\sup_{x_i \in \mathcal{X}_i} f(x_i^T\theta - y_i) \leq c_i$ in a tractable form.

Loss f is non-increasing. If f is non-increasing, then introducing an auxiliary variable z_i shows that

$$\begin{aligned}\sup_{x_i\in\mathcal{X}_i} f(x_i^T\theta - y_i) \le c_i \quad &\iff \quad f\left(\inf_{x_i\in\mathcal{X}_i} x_i^T\theta - y_i\right) \le c_i \\ &\iff \quad \inf_{x_i\in\mathcal{X}_i} x_i^T\theta - y_i \ge z_i, \quad f(z_i) \le c_i \\ &\iff \quad \sup_{x_i\in\mathcal{X}_i} -x_i^T\theta + y_i \le -z_i, \quad f(z_i) \le c_i.\end{aligned}$$

So, after eliminating the epigraph variable c from (10), we have shown (9) is equivalent to

$$\begin{array}{ll}\text{minimize} & \sum_{i=1}^n f(z_i) \\ \text{subject to} & \theta \in \Theta, \\ & \mathcal{S}_{\mathcal{X}_i}(-\theta) + y_i \le -z_i, \quad i = 1,\ldots,n,\end{array} \tag{11}$$

with variables $z \in \mathbf{R}^n, \theta \in \mathbf{R}^d$. Typical classification losses, such as the hinge, logistic and exponential losses, are non-increasing.

Loss f is non-decreasing on $\mathbf{R}_+$ and $f(a) = f(|a|)$. If f is monotone on nonnegative arguments, and depends only on its argument through the absolute value, then introducing the auxiliary variable z_i shows that

$$\begin{aligned}\sup_{x_i\in\mathcal{X}_i} f(x_i^T\theta - y_i) \le c_i \quad &\iff \quad f\left(\sup_{x_i\in\mathcal{X}_i} |x_i^T\theta - y_i|\right) \le c_i \\ &\iff \quad \sup_{x_i\in\mathcal{X}_i} |x_i^T\theta - y_i| \le z_i, \quad f(z_i) \le c_i \\ &\iff \quad \sup_{x_i\in\mathcal{X}_i} x_i^T\theta - y_i \le z_i, \quad \sup_{x_i\in\mathcal{X}_i} -x_i^T\theta + y_i \le z_i, \quad f(z_i) \le c_i.\end{aligned}$$

So, after eliminating the epigraph variable c from (10), we have shown (9) is equivalent to

$$\begin{array}{ll}\text{minimize} & \sum_{i=1}^n f(z_i) \\ \text{subject to} & \theta \in \Theta, \\ & \mathcal{S}_{\mathcal{X}_i}(\theta) - y_i \le z_i, \quad i = 1,\ldots,n, \\ & \mathcal{S}_{\mathcal{X}_i}(-\theta) + y_i \le z_i, \quad i = 1,\ldots,n,\end{array} \tag{12}$$

with variables $z \in \mathbf{R}^n, \theta \in \mathbf{R}^d$. Typical regression losses, such as p-norm and Huber losses, satisfy this requirement on f.

CVXPY code. The robust constraints in (11) and (12) include suprema over $\mathcal{X}_i$ of bilinear forms involving x_i, θ. While this ostensibly requires dualization to handle, the CVXPY transform `SuppFunc` allows one to easily specify the support function $\mathcal{S}_C(\theta)$ of a set

C created via DCP constraints. Since this function is already implemented in CVXPY, we can directly specify the robust constraints in (11) and (12) without additional reformulation or dualization. As an example, we depict below the CVXPY code that specifies and solves (12) with $f = |\cdot|^2$ and $\Theta = \mathbf{R}^d$, which is a robust least squares problem. For convenience, we assume y has already been specified as `y`.

```
1  import cvxpy as cp
2  from cvxpy.transforms.suppfunc import SuppFunc
3
4  theta, z = cp.Variable(d), cp.Variable(n)
5  constraints = []
6
7  for i in range(n):
8      # Create variables for uncertainty set
9      x = cp.Variable(d)
10
11     # Construct uncertainty set containing x (filled in by user)
12     local_constraints = []
13
14     # Implement the support function of the uncertainty set
15     G1 = SuppFunc(x, local_constraints)(theta)
16     G2 = SuppFunc(x, local_constraints)(-theta)
17
18     # Store robust constraints
19     constraints.append(G1 - y[i] <= z[i])
20     constraints.append(G2 + y[i] <= z[i])
21
22 obj = cp.Minimize(cp.sum_squares(z))
23 prob = cp.Problem(obj, constraints)
24 prob.solve()
```

To fully specify the problem, the user only needs to describe the uncertainty set $\mathcal{X}_i$ for each $i = 1, \ldots, n$ in Line 12, in terms of the `x` instantiated in Line 9. This is done exactly as one would typically do for any `Variable` in CVXPY. If, for example, $\mathcal{X}_i$ was the intersection of the Euclidean unit ball, the non-negative orthant, and the set of vectors whose first coordinate is 0.25, then replacing Line 12 with the code block below is sufficient.

```
12 local_constraints = [x >= 0, cp.sum_squares(x) <= 1, x[0] == 0.25]
```

This manner of expressing $\mathcal{X}_i$ is thus natural, user-friendly and directly follows the math.

Alternatively, one may recognize that the constraints in (11) and (12) include the

partial suprema of a convex-concave saddle function. Since DSP was designed to solve saddle problems, and a bilinear function is an atom in DSP, we can use DSP to solve (11) and (12) with the same convenience and ease. All that is required is importing DSP via `from dsp import *`, and replacing lines 8-16 above with the code block below.

```
# Creating local variables for uncertainty set
x1, x2 = LocalVariable(d), LocalVariable(d)

# Create bilinear form of theta and x
g1, g2 = saddle_inner(theta, x1), saddle_inner(-theta, x2)

# Construct uncertainty set containing x (filled in by user)
local_constraints1 = []
local_constraints2 = []

# Take suprema over x
G1 = saddle_max(g1, local_constraints1)
G2 = saddle_max(g2, local_constraints2)
```

For the user's convenience, in Appendix D we present a helper Python function that automatically converts problems of the form (9) to problems of the form (11) and (12).

5.3 Example

We consider the problem of predicting nightly Airbnb rental prices in London, from different features such as coordinates, distance from city center, and neighborhood restaurant quality index. We will consider a simulated hypothetical case where we do not have full acess to the rentals' location. We will use robust regression to handle the uncertain location features. We can then use uncertainty sets for the unknown locations, allowing us to illustrate the ease of specifying RERM problems with our framework. This example is artificial, but does use real original data. We do not advocate using robust regression in particular for this problem; replacing each unknown location with a center of the uncertainty set performs nearly as well as the best robust regression method, and is much simpler. The code to reproduce this example is available at

https://github.com/cvxgrp/rerm_code.

Data. We begin with a curated dataset from London [GN21], and remove rentals with prices exceeding 1000 Euros and those located more than 7 km from the city center, resulting in a dataset of 3400 rows and 20 columns. We then remove categorical features and randomly sub-sample to obtain a training set with 1000 data points and test data set with 500 data points. The training feature matrix is $X \in \mathbf{R}^{1000\times 9}$, with rows x_i^T. The first two columns

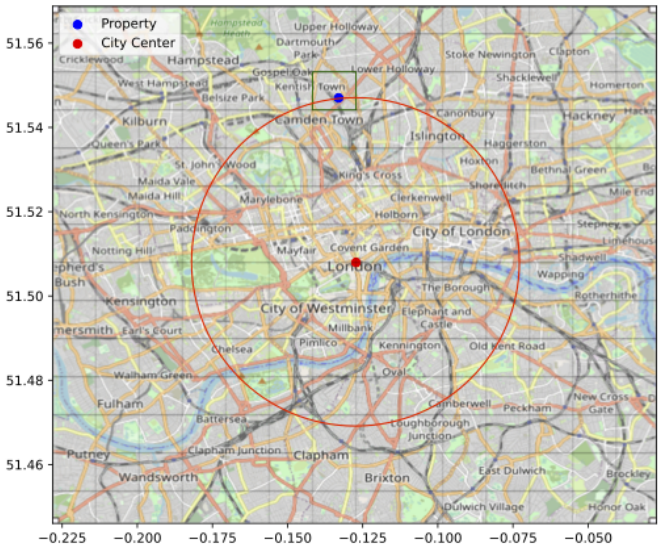


Figure 8: A visualization of D_i and S_i for a particular rental i. The large disk around the red dot corresponds to D_i. The square containing the blue dot corresponds to S_i. The overlap of the square and the disk corresponds to $D_i \cap S_i$.

of X correspond to the longitude and latitude respectively. Our baseline predictor of rental price is a simple linear ordinary least squares (OLS) regression based on all 9 features. Its test RMS error is 138 euros.

Hidden location features. To illustrate our method, we imagine a case where rental owners have elected to not release the exact longitude and latitude of their properties. (While not identical to this example, Airbnb does in fact mask rental locations.) We grid London into 1 km by 1 km squares, and for each rental only give the square S_i it is located in. This generates an uncertainty set for data point i given by

$$\mathcal{X}_i^S = \{x \in \mathbf{R}^9 \mid x_{1:2} \in S_i,\ x_{3:9} = X_{3:9}^i\}.$$

We also know the distance of each rental from the city center $c \in \mathbf{R}^2$, denoted by d_i. Using this, we can consider a more refined uncertainty set

$$\mathcal{X}_i^{S\cap D} = \mathcal{X}_i^S \cap D_i,$$

where $D_i = \{x \in \mathbf{R}^9 \mid \|x_{1:2} - c\|_2 \leq d_i\}$. See figure 8 for a visualization of these uncertainty sets.

We solve (12) with squared loss $f = |\cdot|^2$ and the two choices of the uncertainty sets described above. These choices correspond to using square or disk-intersected-with-square

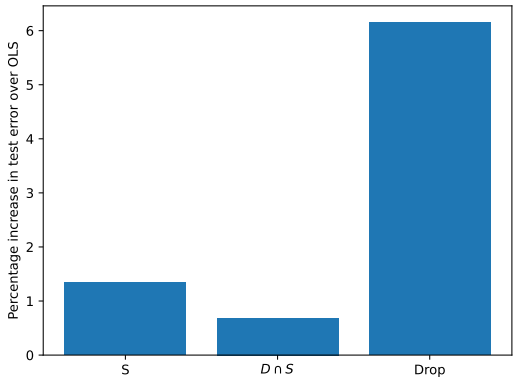


Figure 9: Excess test error in the Airbnb price prediction experiment.

uncertainty sets for the missing coordinates. Note that the square uncertainty set combined with the quadratic loss is a special case where we can derive an analytical form for the worst-case loss. However, the analytical form is lost once we intersect the square with the disk.

Comparing the methods. We depict the performance of the two RERM predictors, as well as a predictor that completely ignores coordinate information, in figure 9. Our performance metric is the mean squared error on the test set, in excess of the baseline OLS predictor trained on X without any missing entries.

We observe that the dropping scheme, denoted as Drop in figure 9, performs the worst. The robust predictors that use square uncertainty sets (denoted as S) and the intersected uncertainty sets (denoted as $S \cap D$) outperform the others. The robust predictor that uses uncertainty sets $S \cap D$ outperforms the robust predictor that uses only S. Indeed, its performance is nearly as good as the OLS predictor baseline that has access to all the columns of X. These intersected uncertainty sets are complex and do not admit the sort of convenient reformulation afforded by using square or disk uncertainty sets. Yet, our framework allows us to handle these uncertainty sets conveniently, and hence obtain less conservative predictors.

6 Conclusion

In this thesis, we considered the broad and challenging problem of ensuring that a machine learning system can learn and act reliably and efficiently in an uncertain stochastic environment. We studied both statistical and also computational notions of efficiency, in both the supervised learning and sequential decision making regimes. Since this problem is intractable in the worst case, our primary focus was on determining structure that exists in such environments, which would guarantee efficiency. We particularly emphasized structural properties that are motivated by practical problems and arise in real world scenarios.

In the sequential decision making regime, our contribution was twofold. We considered structure that appears in simple video games, that allows for a smaller effective planning window. We used this to devise a statistically efficient reinforcement learning algorithm. We then considered interactive feedback loops between an algorithm and a user, as occurs in recommender systems. We formalized this structure by generalizing the classical stochastic multi armed bandit to incorporate this interactivity, while also specializing the RL setting to ensure tractability. We provided a statistically efficient and near optimal algorithm for this problem, and under an additional assumption, we also provided a computationally tractable method.

In the supervised learning regime, we considered robust empirical risk minimization, when the loss function and uncertainty sets are convex. Under the assumption that these uncertainty sets are guaranteed to contain the data and are independent from data point to data point, we provided a practical framework based on CVXPY to specify and solve such robust empirical risk minimization problems. Notably, a user can specify problems involving very complicated uncertainty sets in just a few lines of code, in a convenient and natural fashion that directly follows the math.

We believe that there are a number of interesting directions for future work. In the sequential decision making regime, it would be interesting to see whether our notion of a small effective planning window can be integrated into problems where hierarchical planning is necessary. Are there practical problems where one can decompose the entire planning problem into a sequence of larger steps, where each individual step can be formalized and solved via our effective planning window framework? In the supervised learning regime, it would be interesting to see whether our work can be extended to kernelized settings. Is there a way to specify and solve the robust problem when the uncertainty sets are described by some featurization, in a manner that does not require the feature map to be explicitly computed, but instead only requires the specification of an appropriate kernel?

More broadly, we view this thesis as a stepping stone towards a foundational understanding of when machine learning systems can be successfully deployed in the real world. Our approach emphasized formalizing structure that theoretically permits efficient learning and decision making. While this approach is valuable, we believe that subsequent lines of inquiry that utilize this approach should augment it with other perspectives. In particular, the algorithms that exploit such formalized structure are not always very practical, even

though on paper they are guaranteed to be efficient. We believe that the most fruitful subsequent efforts will be those that first formalize structure and devise a theoretically justified algorithm, and then use the insights gleaned during this process to relax this provably efficient algorithm into one that is easily implementable, without compromising significantly on its key efficiency properties.

A Appendix for Section 2

A.1 Other Games Satisfying EPW

In this section, we shall verify that several other games besides Pong and Skiing satisfy the EPW condition. In Appendix A.1.1 we will verify this for the Atari games Tennis and Journey Escape. In Appendix A.1.2 we will verify this for the RL gaming benchmark CoinRun [CKH+19], which is more complex than Atari.

A.1.1 Atari Games

Tennis. This game is very similar to Pong, and is depicted in Figure 10. Here the agent controls the tennis player depicted at the top of the screen, who must hit the ball and prevent it from crossing its boundary. It plays against a player which hits the ball back according to a pre-specified stochastic decision rule (which is not trained). The agent loses if the ball crosses its own boundary, and wins if it hits the ball past the opponent's boundary.

The first two conditions of Generic Games are easy to verify. Note that the states in Tennis are raw images, so $\mathcal{F}$ is defined by any state where the ball has crossed the agent's boundary since this corresponds to the agent losing. It is known that Atari can be solved using a neural network policy [M+15], and this ensures that a policy class parameterized by neural networks is indeed complete.

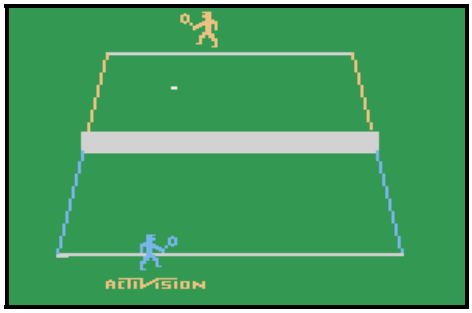


Figure 10: An image of the Atari Tennis game. The yellow player must move to hit the ball (the white dot) while playing against the opposing blue player.

To ensure that Tennis satisfies the third condition, we need to design an appropriate binary reward function. This is handled by redefining $\mathcal{F}$ to include any state $s \in \mathcal{S}_{H-1}$ where the ball has not crossed the opposing player's boundary. Then one can simply assign a reward of 1 to any state in $\mathcal{S}_{H-1} \setminus \mathcal{F}$, and 0 to all other states, as required by the Generic Game condition. Hence, playing optimally in this Generic Game framework ensures the ball has moved past the opponent's boundary, corresponding to winning the game.

We now verify that Tennis satisfies the EPW condition with a relatively small value of C. In Tennis, after the opposing player hits the ball, the agent must react to the trajectory

of the ball and adjust its position accordingly to hit it. If it takes too long to react before it starts adjusting its position, then it will be unable to reach the ball in time. More formally, assume that at timestep t the paddle has not lost the game and the ball is moving towards its boundary. At timestep t, the ball may be too close to the boundary, and so the agent will not not have enough time to move its player fast enough in order to reach the ball in time. However, at timestep $t - C$ the ball is further away from the boundary, so the agent has enough time to move its player appropriately in order to react, reach the ball and hit it back. So at timestep $t - C$ the agent lies in a safe state in $\mathcal{S}^\star$, since it has enough time to adjust its player and hit the ball back, and hence play optimally. Notably, if we let C' be the number of timesteps it takes for the ball to traverse from one end of the screen to the other, then $C \leq C'$. Hence, when H is large and the agent needs to control its player for many rounds, then C is a constant independent of H.

Journey Escape. This game is similar to Skiing, and is depicted in Figure 11. In this game, there are friendly and enemy objects. These objects come sequentially, and the agent must dodge enemy objects and collide with friendly objects.

The first two conditions of Generic Games are easy to verify. Note that the states in Journey Escape are raw images, so $\mathcal{F}$ is defined by any state where the agent has collided with an enemy object or missed a friendly object. It is known that Atari can be solved using a neural network policy [M+15], and this ensures that a policy class parameterized by neural networks is indeed complete.

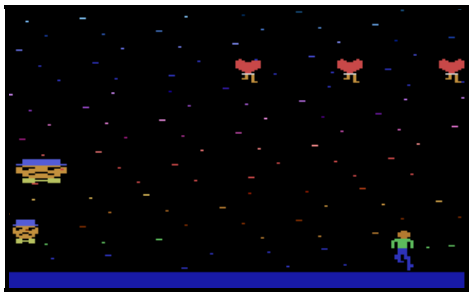


Figure 11: An image of the Atari Journey Escape game. The agent must avoid the enemy objects on screen to avoid receiving a penalty, and must collide with other friendly objects (not depicted) to increase the score.

To ensure that Journey Escape satisfies the third condition, we need to design an appropriate binary reward function. This is done by ensuring that $\mathcal{F}$ includes any state where the agent has collided with an enemy object or missed a friendly object, as described above. Then one can simply assign a reward of 1 to any state in $\mathcal{S}_{H-1} \setminus \mathcal{F}$, and 0 to all other states, as required by the Generic Game condition. Hence, playing optimally in this Generic Game framework ensures the agent has avoided all enemy objects while colliding with all friendly objects, corresponding to winning the game.

We now verify that Journey Escape satisfies the EPW condition with a relatively small

value of C. As the objects come towards the agent, it must react appropriately to adjust its position depending on whether the oncoming object is friendly or enemy. Let us focus on the enemy object case, since the friendly object case is symmetric. Formally, assume that at timestep t an enemy object is moving towards the agent. At timestep t, the object may be too close to the agent, and so the agent will not not have enough time to move away fast enough and get away from the enemy object. However, at timestep $t - C$ the agent is further away from the enemy object, so the agent has enough time to move away appropriately in order to react. So at timestep $t - C$ the agent lies in a safe state in $\mathcal{S}^\star$, since it has enough time to adjust its position and hence play optimally. Notably, if we let C' be the number of timesteps it takes for the agent to traverse from one end of the screen to the other, then $C \leq C'$. Hence, when H is large and the agent needs to play for many rounds, then C is a constant independent of H.

A.1.2 CoinRun

CoinRun is a recent RL gaming benchmark [CKH+19], and is depicted in Figure 12. In any CoinRun instance, an agent must move right and jump to avoid obstacles, which are sometimes randomly moving, until it arrives at a coin. It receives unit reward if it reaches the coin, and zero reward otherwise. If it collides with an obstacle then the game is over.

The first two conditions of Generic Games are easy to verify. Note that the states in CoinRun are raw images, so $\mathcal{F}$ is defined by any state where the agent has collided with an obstacle, as well as any state $s \in \mathcal{S}_{H-1}$ where the agent has not already reached the coin. It is known that CoinRun can be solved using a neural network policy [CKH+19], and this ensures that a policy class parameterized by neural networks is indeed complete.

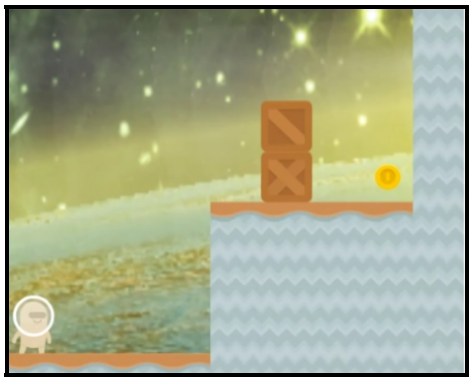


Figure 12: An image of the CoinRun game. The agent must move towards the coin while avoiding obstacles.

Note that the third condition of Generic Games is automatically satisfied by the reward function we described above. This is because the game already has a binary reward function, with unit reward for reaching the coin and completing the game, and zero reward otherwise.

To verify the EPW condition, note that the agent must react to obstacles which move towards it. Formally, assume that at timestep t an obstacle is moving towards the agent. At timestep t, the obstacle may be too close to the agent, and so the agent will not not have enough time to move away fast enough and get away from the obstacle. However, at timestep $t - C$ the agent is further away from the obstacle, so the agent has enough time to move away appropriately in order to react. So at timestep $t - C$ the agent lies in a safe state in $\mathcal{S}^\star$, since it has enough time to adjust its position and hence play optimally. In this game, C is a small constant, since it only takes a few timesteps for the agent to have enough time to move away from an oncoming obstacle.

A.2 Upper Bound Proof

In this section we will prove Theorem 1. First, we shall develop notation and state some helpful lemmas. We then present the proof of Theorem 1, and return to complete the proofs of the lemmas.

Recall that Algorithm 1 iteratively constructs a new $\overline{\theta}(t)$ after each timestep in its inner loop. So after t iterations of its inner loop, the algorithm has constructed $\overline{\theta}(t)$. For each $t \in \{0, 1 \dots H-2\}$ define the function $L_t : \Theta \to \mathbb{R}$ as follows:

$$L_t(\theta) = |\mathcal{A}|^{C+1} \cdot \mathbb{E}_{\tau \sim \pi(\overline{\theta}(t))} \left[\mathbb{I}_{s_{t+1+C} \in \mathcal{F}} \prod_{j=0}^{C} \pi_{s_{t+j}}^{a_{t+j}}(\theta) \right]$$

Here, the expectation is over the sampling of trajectories τ from policy $\pi(\overline{\theta}(t))$, where $\tau = \left\{ (s_h, a_h)_{h=0}^{H-1} \right\}$. So for each $t \in \{0, 1 \dots H-2\}$, the function L_t is associated with the quantity $\overline{\theta}(t)$ that has been constructed by Algorithm 1. Observe that the function $\widehat{L}_t$ defined in the inner loop of Algorithm 1 is the empirical version of L_t. Also recall the definition of $\widehat{\theta}_t$ from Eq. (2).

We now define the notation $\mathbb{P}_\theta(E \mid X)$ to denote the probability of event E occurring when using policy $\pi(\theta)$, conditioned on executing $\pi(\theta)$ from the state X. We now state a key lemma, which is essential to our proof of Theorem 1.

Lemma 1. *For any $(\mathcal{M}, \pi(\Theta))$ satisfying the EPW condition, and $\overline{\theta}(t)$ constructed by Algorithm 1 for any $t \in \{0, 1 \dots H-2\}$ when given sample size n, assume that $\mathbb{P}_{\overline{\theta}(t)}(s_t \in \mathcal{S}^\star \mid s_0) \geq 1 - \alpha$ for some $\alpha \in (0, 1)$. Then the event*

$$\mathbb{P}_{\overline{\theta}(t+1)}(s_{t+1} \notin \mathcal{S}^\star \mid s_0) \leq 2|\mathcal{A}|^{C+1} \sqrt{\frac{\log(2/\delta')}{n}} + \alpha$$

holds with probability at least $1-\delta'\left(1+16\sqrt{\frac{n}{\log(2/\delta')}}C\phi B\right)^k$.

Simply put, the lemma shows the following. Assume that up till timestep t, Algorithm 1 has computed a policy which arrives at a state in $\mathcal{S}^\star$ with high probability. Then at timestep $t+1$ it will compute a policy which arrives at a state in $\mathcal{S}^\star$ with only slightly worse probability. We shall return to prove this lemma in Appendix A.3.1. Let us now prove Theorem 1.

A.3 Proof Of Theorem 1

Note that by definition of a Generic Game, $V_{\mathcal{M}}^{s_0}(\pi(\theta^\star))=1$. Furthermore we have

$$\begin{aligned} V_{\mathcal{M}}^{s_0}(\pi(\overline{\theta})) &= \mathbb{P}_{\overline{\theta}}(s_{H-1}\in\mathcal{S}_{H-1}-\mathcal{F}\mid s_0) \\ &\geq \mathbb{P}_{\overline{\theta}}(s_{H-1}\in\mathcal{S}^\star\mid s_0) \end{aligned}$$

where the equality is by the definition of the binary reward function in Generic Games, and the inequality is since the MDP is partitioned into disjoint levels and additionally because $\mathcal{S}^\star\subseteq\mathcal{S}-\mathcal{F}$. It is hence to sufficient to show that Algorithm 1 returns $\overline{\theta}\equiv\overline{\theta}(H-1)$ satisfying

$$\mathbb{P}_{\overline{\theta}(H-1)}(s_{H-1}\in\mathcal{S}^\star\mid s_0)\geq 1-\epsilon \tag{13}$$

with probability at least $1-\delta$. We shall devote the remainder of the proof to this.

Let δ' be some real number in the interval $(0,1)$, whose precise value we will specify later. For each $t\in[H]$, let us define $\mathcal{E}_t$ to be the event that Algorithm 1 constructs $\overline{\theta}(t)$ satisfying

$$\mathbb{P}_{\overline{\theta}(t)}(s_t\notin\mathcal{S}^\star\mid s_0)\leq t\cdot 2|\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}}. \tag{14}$$

Let $\mathbb{P}_n$ denote the randomness of Algorithm 1, which manifests due to sampling of trajectories at each timestep of the inner loop of the algorithm. We claim that

$$\mathbb{P}_n(\cap_{j\leq t}\mathcal{E}_j)\geq 1-t\cdot\delta'\left(1+16\sqrt{\frac{n}{\log(2/\delta')}}C\phi B\right)^k \tag{15}$$

for each $t\in[H]$. We will prove this by strong induction, repeatedly using Lemma 1 and union bounding to obtain the desired estimate.

For the base case at timestep $t=0$, notice we trivially have $\mathbb{P}_{\overline{\theta}(0)}(s_0\in\mathcal{S}^\star\mid s_0)=1$, since $s_0\in\mathcal{S}^\star$ by definition of $\theta^\star$. This implies Eq. (14). In particular, we have $\mathbb{P}_n(\mathcal{E}_0)=1$, verifying Eq. (15) when $t=0$.

Now for the inductive step, assume that for some t we have $\mathbb{P}_n\left(\bigcap_{j\le t}\mathcal{E}_j\right) \ge 1 - t\cdot\delta'\left(1+16\sqrt{\frac{n}{\log(2/\delta')}}C\phi B\right)^k$. Then by conditioning on $\bigcap_{j\le t}\mathcal{E}_j$ and applying Lemma 1, we obtain that the event

$$\begin{aligned}\mathbb{P}_{\overline{\theta}(t+1)}(s_{t+1}\in\mathcal{S}^\star \mid s_0) &\ge 1 - 2|\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}} - t\cdot 2|\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}} \\ &= 1-(t+1)\cdot 2|\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}}\end{aligned}$$

holds with probability at least $1-\delta'\left(1+16\sqrt{\frac{n}{\log(2/\delta')}}C\phi B\right)^k$. Note that the above equation exactly matches Eq. (14), so conditioned on $\bigcap_{j\le t}\mathcal{E}_j$ we have shown $\mathbb{P}_n(\mathcal{E}_{t+1}) \ge 1-\delta'\left(1+16\sqrt{\frac{n}{\log(2/\delta')}}C\phi B\right)^k$. Applying a union bound, we have shown that

$$\mathbb{P}_n\left(\bigcap_{j\le t+1}\mathcal{E}_j\right) \ge 1-(t+1)\cdot\delta'\left(1+16\sqrt{\frac{n}{\log(2/\delta')}}C\phi B\right)^k,$$

which thus verifies Eq. (15) and hence the inductive step. We have therefore shown that Algorithm 1 constructs $\overline{\theta}(H-1)$ satisfying

$$\mathbb{P}_{\overline{\theta}(H-1)}(s_{H-1}\in\mathcal{S}^\star \mid s_0) \ge 1 - 2H|\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}}$$

with probability at least $1-H\delta'\left(1+16\sqrt{\frac{n}{\log(2/\delta')}}C\phi B\right)^k$. To show that the above equation exactly matches Eq. (13), we need only check that our choice of n yields the desired values of ϵ and δ.

To obtain our choice of n, we first set $2H|\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}} = \epsilon$. This yields

$$n = \frac{4H^2|\mathcal{A}|^{2C+2}}{\epsilon^2}\log\left(\frac{2}{\delta'}\right) \iff \sqrt{\frac{n}{\log(2/\delta')}} = \frac{2H|\mathcal{A}|^{C+1}}{\epsilon}. \tag{16}$$

Next, we make the substitution

$$\delta = H\delta'\left(1+16\sqrt{\frac{n}{\log(2/\delta')}}C\phi B\right)^k,$$

which results in

$$\delta = H\delta' \left(1 + \frac{32H|\mathcal{A}|^{C+1}}{\epsilon} C\phi B\right)^k,$$

implying that

$$\delta' = \delta \left(H \left(1 + \frac{32H|\mathcal{A}|^{C+1}}{\epsilon} C\phi B\right)^k\right)^{-1}.$$

Substituting the above expression for δ' into the first equivalence of Eq. (16), we finally get

$$n = \frac{4H^2|\mathcal{A}|^{2C+2}}{\epsilon^2} \left(\log\left(\frac{2H}{\delta}\right) + k \log\left(1 + \frac{32H|\mathcal{A}|^{C+1} C\phi B}{\epsilon}\right)\right),$$

and this completes the proof.

A.3.1 Proof Of Lemma 1

To facilitate our proof, we first state the following useful lemma. Recall the definition of $\widehat{\theta}_t$ from Eq. (2).

Lemma 2. *For any $(\mathcal{M}, \pi(\Theta))$ satisfying the EPW condition, and $\overline{\theta}(t)$ constructed by Algorithm 1 for any $t \in \{0, 1 \dots H-2\}$ when given sample size n, assume that $\mathbb{P}_{\overline{\theta}(t)}(s_t \in \mathcal{S}^\star \mid s_0) \geq 1 - \alpha$ for some $\alpha \in (0,1)$. Then the event*

$$\mathbb{E}_{s_t \sim \pi(\overline{\theta}(t))} \left[\mathbb{P}_{\widehat{\theta}_t}(s_{t+C+1} \in \mathcal{F} \mid s_t)\right] \leq 2|\mathcal{A}|^{C+1} \sqrt{\frac{\log(2/\delta')}{n}} + \alpha$$

holds with probability at least $1 - \delta' \left(1 + 16\sqrt{\frac{n}{\log(2/\delta')}} C\phi B\right)^k$.

We will return to prove Lemma 2 in Appendix A.3.2. For now, let us return to the proof of Lemma 1. By the result of Lemma 2, the event

$$\mathbb{E}_{s_t \sim \pi(\overline{\theta}(t))} \left[\mathbb{P}_{\widehat{\theta}_t}(s_{t+C+1} \in \mathcal{F} \mid s_t)\right] \leq 2|\mathcal{A}|^{C+1} \sqrt{\frac{\log(2/\delta')}{n}} + \alpha$$

holds with probability at least $1 - \delta' \left(1 + 16\sqrt{\frac{n}{\log(2/\delta')}} C\phi B\right)^k$. Let us denote this event as

$\mathcal{E}$. Then on this event, we have

$$\begin{aligned}
\mathbb{E}_{s_t\sim\pi(\bar{\theta}(t))}\left[\mathbb{P}_{\widehat{\theta}_t}(s_{t+1}\notin\mathcal{S}^\star\mid s_t)\right] &= \mathbb{E}_{s_t\sim\pi(\bar{\theta}(t))}\Big[\mathbb{P}_{\widehat{\theta}_t}(s_{t+1}\notin\mathcal{S}^\star\wedge s_{t+C+1}\in\mathcal{F}\mid s_t)\\
&\qquad + \mathbb{P}_{\widehat{\theta}_t}(s_{t+1}\notin\mathcal{S}^\star\wedge s_{t+C+1}\notin\mathcal{F}\mid s_t)\Big]\\
&\overset{(i)}{\leq} \mathbb{E}_{s_t\sim\pi(\bar{\theta}(t))}\left[\mathbb{P}_{\widehat{\theta}_t}(s_{t+C+1}\in\mathcal{F}\mid s_t)+\mathbb{P}_{\widehat{\theta}_t}(s_{t+1}\notin\mathcal{S}^\star\wedge s_{t+C+1}\notin\mathcal{F}\mid s_t)\right]\\
&\overset{(ii)}{\leq} 2|\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}}+\alpha+\mathbb{E}_{s_t\sim\pi(\bar{\theta}(t))}\left[\mathbb{P}_{\widehat{\theta}_t}(s_{t+1}\notin\mathcal{S}^\star\wedge s_{t+C+1}\notin\mathcal{F}\mid s_t)\right]\\
&\overset{(iii)}{=} 2|\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}}+\alpha.
\end{aligned}$$

Here, step (i) is trivial as $\mathbb{P}(A\wedge B)\leq\mathbb{P}(A)$. Step (ii) follows from the definition of $\mathcal{E}$, and finally, step (iii) follows from the EPW condition and definition of C. It remains to note that

$$\mathbb{E}_{s_t\sim\pi(\bar{\theta}(t))}\left[\mathbb{P}_{\widehat{\theta}_t}(s_{t+1}\notin\mathcal{S}^\star\mid s_t)\right]=\mathbb{P}_{\bar{\theta}(t+1)}(s_{t+1}\notin\mathcal{S}^\star\mid s_0),$$

which follows directly from the definition $\bar{\theta}(t+1)=[\widehat{\theta}_0,\widehat{\theta}_1\ldots,\widehat{\theta}_{t-1},\widehat{\theta}_t,\theta_{\text{rand}},\ldots,\theta_{\text{rand}}]$ and the Law of Total Expectation.

A.3.2 Proof Of Lemma 2

To facilitate this proof, we first state the following two auxiliary lemmas.

Lemma 3. *For any $(\mathcal{M},\pi(\Theta))$ satisfying the EPW condition, and $\bar{\theta}(t)$ constructed by Algorithm 1 for any $t\in\{0,1\ldots H-2\}$, we have*

$$L_t(\theta)=\mathbb{E}_{s_t\sim\pi(\bar{\theta}(t))}\left[\mathbb{P}_\theta(s_{t+C+1}\in\mathcal{F}\mid s_t)\right],$$

where the expectation is taken with respect to the marginal distribution of s_t while sampling trajectories from $\pi(\bar{\theta})$.

Lemma 4. *For any $(\mathcal{M},\pi(\Theta))$ satisfying the EPW condition, and $\bar{\theta}(t)$ constructed by Algorithm 1 for any $t\in\{0,1\ldots H-2\}$, the functions L_t, $\widehat{L}_t$ are each Lipschitz with Lipschitz constant $|\mathcal{A}|^{C+1}(C+1)\phi$.*

We shall return to prove these lemmas in Appendices A.3.3 and A.3.4 respectively. Let us now return to the proof of Lemma 2. Recall from Lemma 3 that $\mathbb{E}_{s_t\sim\pi(\bar{\theta}(t))}\left[\mathbb{P}_{\widehat{\theta}_t}(s_{t+C+1}\in\mathcal{F}\mid s_t)\right]=L_t(\widehat{\theta}_t)$. So it is sufficient to show that the event

$$L_t(\widehat{\theta}_t)\leq 2|\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}}+\alpha$$

holds with probability at least $1-\delta'\left(1+16\sqrt{\frac{n}{\log(2/\delta')}}C\phi B\right)^k$, and we will devote the remainder of the proof to showing this. First, we use the characterization of L_t derived in Lemma 3 to show the helpful fact that

$$\begin{aligned} L_t(\theta^\star) &= \mathbb{E}_{s_t\sim\pi(\bar{\theta}(t))}\left[\mathbb{P}_{\theta^\star}(s_{t+C+1}\in\mathcal{F}\mid s_t)\right] \\ &= \mathbb{E}_{s_t\sim\pi(\bar{\theta}(t))}\left[\mathbb{P}_{\theta^\star}(s_{t+C+1}\in\mathcal{F}\mid s_t)\mathbb{I}_{s_t\in\mathcal{S}^\star}+\mathbb{P}_{\theta^\star}(s_{t+C+1}\in\mathcal{F}\mid s_t)\mathbb{I}_{s_t\notin\mathcal{S}^\star}\right] \\ &= \mathbb{E}_{s_t\sim\pi(\bar{\theta}(t))}\left[\mathbb{P}_{\theta^\star}(s_{t+C+1}\in\mathcal{F}\mid s_t)\mathbb{I}_{s_t\notin\mathcal{S}^\star}\right] \\ &\leq \alpha, \end{aligned} \tag{17}$$

where the final equality follows from the definition of $\mathcal{S}^\star$ and the inequality follows from the assumption that $\mathbb{P}_{\bar{\theta}(t)}(s_t\in\mathcal{S}^\star\mid s_0)\geq 1-\alpha$.

By the Regularity of $\pi(\Theta)$, we are guaranteed that Θ is contained in the Euclidean ball of radius B. For any $\gamma>0$, we use $\mathcal{N}(\gamma)$ to denote a minimal γ-covering of Θ. Recall that $\Theta\subset\mathbb{R}^k$. Also recall the standard fact [Ver18] that $|\mathcal{N}(\gamma)|\leq\left(1+\frac{2B}{\gamma}\right)^k$.

Now for any fixed $\theta\in\mathcal{N}(\gamma)$, we know from Hoeffding's inequality that the bound

$$|\widehat{L}_t(\theta)-L_t(\theta)|\leq\frac{|\mathcal{A}|^{C+1}}{2}\sqrt{\frac{\log(2/\delta')}{n}} \tag{18}$$

holds with probability at least $1-\delta'$. Hence, applying a union bound, we know that the above bound holds for every $\theta\in\mathcal{N}(\gamma)$ with probability at least $1-\delta'|\mathcal{N}(\gamma)|\geq 1-\delta'\left(1+\frac{2B}{\gamma}\right)^k$.

For any $\theta\in\Theta$, let θ_γ be an element of $\mathcal{N}(\gamma)$ such that $\|\theta-\theta_\gamma\|_2\leq\gamma$. We now argue that with probability at least $1-\delta'\left(1+\frac{2B}{\gamma}\right)^k$, any $\theta\in\Theta$ satisfies the bound

$$\begin{aligned} |\widehat{L}_t(\theta)-L_t(\theta)| &\overset{(i)}{\leq} |\widehat{L}_t(\theta)-\widehat{L}_t(\theta_\gamma)|+|\widehat{L}_t(\theta_\gamma)-L_t(\theta_\gamma)|+|L_t(\theta_\gamma)-L_t(\theta)| \\ &\overset{(ii)}{\leq} 2|\mathcal{A}|^{C+1}(C+1)\phi\gamma+|\widehat{L}_t(\theta_\gamma)-L_t(\theta_\gamma)| \\ &\overset{(iii)}{\leq} 2|\mathcal{A}|^{C+1}(C+1)\phi\gamma+\frac{|\mathcal{A}|^{C+1}}{2}\sqrt{\frac{\log(2/\delta')}{n}}, \end{aligned}$$

where step (i) follows from the triangle inequality, step (ii) is due to the Lipschitz property of $L_t,\widehat{L}_t$ we derived in Lemma 4, and step (iii) is due to Eq. (18). Now set $\gamma=\frac{1}{4(C+1)\phi}\sqrt{\frac{\log(2/\delta')}{n}}$. Then the bound

$$|\widehat{L}_t(\theta)-L_t(\theta)|\leq|\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}} \tag{19}$$

holds for uniformly for each $\theta \in \Theta$ with probability at least $1 - \delta' \left(1 + 16\sqrt{\frac{n}{\log(2/\delta')}} C\phi B\right)^k$. Let $\mathcal{E}$ denote the event that Eq. (19) holds uniformly for each $\theta \in \Theta$.

We now use this uniform bound to control the quantity $L_t(\widehat{\theta}_t)$. Concretely, on the event $\mathcal{E}$ we have that

$$\begin{aligned} L_t(\widehat{\theta}_t) &= L_t(\widehat{\theta}_t) - \widehat{L}_t(\widehat{\theta}_t) + \widehat{L}_t(\widehat{\theta}_t) - \widehat{L}_t(\theta^\star) + \widehat{L}_t(\theta^\star) \\ &\overset{(iv)}{\leq} L_t(\widehat{\theta}_t) - \widehat{L}_t(\widehat{\theta}_t) + \widehat{L}_t(\theta^\star) \\ &\overset{(v)}{\leq} |\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}} + \widehat{L}_t(\theta^\star), \end{aligned}$$

where step (iv) follows from the definition $\widehat{\theta}_t \in \operatorname{argmin}_\theta \widehat{L}_t(\theta)$, and step (v) follows from Eq. (19). To obtain control on $L_t(\widehat{\theta}_t)$, it remains to bound $\widehat{L}_t(\theta^\star)$. Simply observe that on the event $\mathcal{E}$ we have

$$\begin{aligned} \widehat{L}_t(\theta^\star) &= \widehat{L}_t(\theta^\star) - L_t(\theta^\star) + L_t(\theta^\star) \\ &\overset{(vi)}{\leq} |\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}} + L_t(\theta^\star) \\ &\overset{(vii)}{\leq} |\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}} + \alpha. \end{aligned}$$

Steps (vi) and (vii) follow from Eq. (19) and Eq. (17) respectively. Putting the previous two equations together and recalling the definition of $\mathcal{E}$, we have demonstrated that the bound

$$L_t(\widehat{\theta}_t) \leq 2|\mathcal{A}|^{C+1}\sqrt{\frac{\log(2/\delta')}{n}} + \alpha$$

holds with probability at least $1 - \delta' \left(1 + 16\sqrt{\frac{n}{\log(2/\delta')}} C\phi B\right)^k$. As argued earlier, this is sufficient to complete the proof.

A.3.3 Proof Of Lemma 3

Recall that notation $\tau = \left\{(s_h, a_h)_{h=0}^{H-1}\right\}$. Also recall from Algorithm 1 that at timestep t onwards, $\pi(\overline{\theta}(t))$ executes θ_{rand}, implying it selects actions uniformly at random regardless

of the state. This fact allows us to decompose $L_t(\theta)$ as follows

$$\begin{aligned}
L_t(\theta) &= |\mathcal{A}|^{C+1} \cdot \mathbb{E}_{\tau\sim\pi(\bar{\theta}(t))}\left[\mathbb{I}_{s_{t+1+C}\in\mathcal{F}}\prod_{j=0}^{C}\pi_{s_{t+j}}^{a_{t+j}}(\theta)\right] \\
&= |\mathcal{A}|^{C+1} \cdot \mathbb{E}_{s_t\sim\pi(\bar{\theta}(t))}\left[\mathbb{E}_{\pi(\theta_{\text{rand}})}\left(\mathbb{I}_{s_{t+1+C}\in\mathcal{F}}\prod_{j=0}^{C}\pi_{s_{t+j}}^{a_{t+j}}(\theta)\;\middle|\; s_t\right)\right] \\
&= \mathbb{E}_{s_t\sim\pi(\bar{\theta}(t))}\left[\mathbb{E}_{\pi(\theta)}\left(\mathbb{I}_{s_{t+1+C}\in\mathcal{F}}\;\middle|\; s_t\right)\right] \\
&= \mathbb{E}_{s_t\sim\pi(\bar{\theta}(t))}\left[\mathbb{P}_\theta(s_{t+C+1}\in\mathcal{F}\mid s_t)\right].
\end{aligned}$$

This completes the proof.

A.3.4 Proof Of Lemma 4

We first note that the product of $m \geq 2$ functions $\{f_i\}_{i=1}^m$ which are bounded by 1 and Lipschitz continuous with constant L is also Lipschitz continuous with constant mL. This can be proved by induction. Consider the base case when $m = 2$. For any x, y in the domain, we have

$$\begin{aligned}
|f_1(x)f_2(x) - f_1(y)f_2(y)| &= |f_1(x)f_2(x) - f_1(x)f_2(y) + f_1(x)f_2(y) - f_1(y)f_2(y)| \\
&\overset{(i)}{\leq} |f_1(x)|\,|f_2(x) - f_2(y)| + |f_2(y)|\,|f_1(x) - f_2(y)| \\
&\overset{(ii)}{\leq} |f_2(x) - f_2(y)| + |f_1(x) - f_1(y)| \\
&\overset{(iii)}{\leq} 2L\|x - y\|_2,
\end{aligned}$$

where in steps (i), (ii) and (iii) we have used the triangle inequality, the fact that f_1, f_2 are bounded by 1 and the Lipschitz continuity of f_1, f_2 respectively.

Now assume that $g_{[k]} = f_1 \dots f_k$ is Lipschitz continuous with constant kL for some $k \geq 2$. Following the same steps from above, we see that $g_{[k]}f_{k+1}$ is $(k+1)L$-Lipschitz. This completes the proof of the fact.

We now prove the statement of the lemma. Let θ, θ' be two distinct policy parameters. For any $h \in [H]$, we have

$$\left|L_h(\theta) - L_h(\theta')\right| = |\mathcal{A}|^{C+1} \cdot \left|\mathbb{E}_{\tau\sim\pi(\bar{\theta})}\left[\mathbb{I}_{s_{t+1+C}\in\mathcal{F}}\prod_{j=0}^{C}\pi_{s_{t+j}}^{a_{t+j}}(\theta)\right] - \mathbb{E}_{\tau\sim\pi(\bar{\theta})}\left[\mathbb{I}_{s_{t+1+C}\in\mathcal{F}}\prod_{j=0}^{C}\pi_{s_{t+j}}^{a_{t+j}}(\theta')\right]\right|$$

$$
\begin{aligned}
&= |\mathcal{A}|^{C+1} \cdot \left| \mathbb{E}_{\tau\sim\pi(\bar{\theta})} \left[\mathbb{I}_{s_{t+1+C}\in\mathcal{F}} \left(\prod_{j=0}^{C} \pi_{s_{t+j}}^{a_{t+j}}(\theta) - \prod_{j=0}^{C} \pi_{s_{t+j}}^{a_{t+j}}(\theta') \right) \right] \right| \\
&\overset{(iv)}{\leq} |\mathcal{A}|^{C+1} \cdot \mathbb{E}_{\tau\sim\pi(\bar{\theta})} \left[\mathbb{I}_{s_{t+1+C}\in\mathcal{F}} \left(\left| \prod_{j=0}^{C} \pi_{s_{t+j}}^{a_{t+j}}(\theta) - \prod_{j=0}^{C} \pi_{s_{t+j}}^{a_{t+j}}(\theta') \right| \right) \right] \\
&\overset{(v)}{\leq} |\mathcal{A}|^{C+1} \cdot \mathbb{E}_{\tau\sim\pi(\bar{\theta})} \left[\mathbb{I}_{s_{t+1+C}\in\mathcal{F}} \cdot \left((C+1)\phi \|\theta - \theta'\|_2 \right) \right] \\
&\leq |\mathcal{A}|^{C+1} (C+1)\phi \|\theta - \theta'\|_2.
\end{aligned}
$$

Step (iv) is due to Jensen's inequality, and step (v) is due the Lipschitz continuity of a product of Lipschitz continuous functions bounded by 1 which was shown earlier. Since the functions are policy probabilities, they are bounded by 1, and are also ϕ-Lipschitz due to the Regularity of $\pi(\Theta)$.

Analogously, we can show the Lipschitz continuity of $\widehat{L}_h$ for any $h \in \{0, 1 \ldots, H-2\}$, by replacing the expectation with the empirical average over trajectory samples, and this completes the proof of the Lemma.

A.4 Lower Bound Proof Sketch

As discussed earlier, this result follows almost directly from the results of Du et al. [DKWY20], and so we only sketch the proof. First we note it is well known that softmax linear policies are Lipschitz [AKLM20]. In our proof, we use Θ as the scaled unit ball in $\mathbb{R}^k$, where the scaling factor is polynomial in $H, \mathcal{A}$. Hence the policy class $\pi(\Theta)$ is indeed Regular. For the construction of $\mathcal{M}$, we use the same construction that was given in the proof of Theorem 4.1 in [DKWY20]. Recall this construction is defined by a horizon H MDP $\mathcal{M}$ whose states, actions and transitions are defined by a binary tree with H levels. There is a single state on the final level with unit reward, and all the other states in the tree have zero reward. To cast this construction in our Generic Game framework, we only need to make a slight modification. For each state on the penultimate level whose child does not have reward, modify it transitions so that taking any action from here deterministically exits the MDP. Then discard each of the now unreachable states on the final level, so the final level only contains a state with unit reward. The set $\mathcal{F}$ is precisely defined by the states on the penultimate level of the tree from where taking any action exits the MDP. The binary rewards property is true by definition. And the complete policy class property is true directly by the proof of [DKWY20]. Note here, that since we have a bounded Θ and are using a softmax linear policy class $\pi(\Theta)$, the θ^* does not lead to $\mathcal{S}_{H-1} - \mathcal{F}$ almost surely. However, since B is polynomial in $H, \mathcal{A}$, using θ^* will lead to $\mathcal{S}_{H-1} - \mathcal{F}$ with probability exponentially large in H. Our main Theorem 1 easily handles this.

It remains to show the existence of $f : \mathbb{R}^d \to \mathbb{R}$ where f is a linear combination of two neurons, and $V^\star_{\mathcal{M}}(s) = f(s)$ for each $s \in \mathcal{S}$. Again, this follows almost immediately from the proof provided by [DKWY20]. Recall that for d sufficiently large, their proof demonstrates the existence of $\theta^{\star\star}$ such that $s^T\theta^{\star\star} = 1$ if $V^\star_{\mathcal{M}}(s) = 1$ and $s^T\theta^{\star\star} \leq 0.25$ if $V^\star_{\mathcal{M}}(s) = 0$. It remains to observe that the function f defined as

$$f(x) = \text{ReLU}(2x^T\theta^{\star\star} - 1) - \text{ReLU}(-2x^T\theta^{\star\star} - 1)$$

exactly satisfies the claim.

B Appendix for Section 3

B.1 Analysis of Algorithm 2

In this section, we analyze the complete policy regret of Algorithm 2, and prove Theorem 1. Before we formally prove Theorem 1, we first state below two key lemmas, that will be useful throughout. The first lemma shows an equivalence between tallying bandit problems and Markov decision processes (MDPs) [SB18]. The second lemma verifies that the $N_{xy}(\pi)$ quantity defined in Algorithm 2 is well defined. We additionally introduce new quantities μ and $\pi^\star$ which will be useful for our proofs. With this outline in mind, let us begin the analysis.

Lemma 1. *Any (m, g, h)-tallying bandit problem can be equivalently expressed as a finite horizon Markov decision process (MDP).*

The proof of this Lemma 1 is given in Appendix B.10. For the sake of brevity, we have not stated the explicit details of the reduction (for instance, the definition of state space or transition function of the corresponding MDP) in the statement of this lemma. Nevertheless, these details are readily found in the proof.
Recall that in Section 3.4.1, to facilitate the definition of Algorithm 2 we defined the quantity $N_{xy}(\pi)$ via the following procedure. Execute π for $n+1$ periods so that we have played the action sequence $\pi_1, \pi_2 \dots \pi_{n\sqrt{T}}, \pi_{n\sqrt{T}+1} \dots \pi_{(n+1)\sqrt{T}}$. Then use this action sequence to define

$$N_{xy}(\pi) = \frac{1}{\sqrt{T}} \sum_{t=n\sqrt{T}+1}^{(n+1)\sqrt{T}} \mathbb{I}(\pi_t = x) \cdot \mathbb{I}\left(y = \sum_{t'=\max\{1, t-m+1\}}^{t} \mathbb{I}(\pi_{t'} = x) \right).$$

The next lemma shows that $N_{xy}(\pi)$ is always well defined if $1 \le n \le \sqrt{T} - 1$ and $m \le \sqrt{T}$.

Lemma 2. *Consider any (m, g, h)-tallying bandit problem where $m \le \sqrt{T}$, and fix any $\sqrt{T}$-cyclic policy π, any $x \in \mathcal{X}$ and any $y \in \{1, 2 \dots m\}$. When defined via the aforementioned procedure, the quantity $N_{xy}(\pi)$ is well defined for any $1 \le n \le \sqrt{T} - 1$. Furthermore, $N_{xy}(\pi)$ is independent of any action sequence that was played before π was executed for $n+1$ periods.*

The proof of this Lemma 2 is deferred to Appendix B.9. Now for each $\sqrt{T}$-cyclic policy π, we define the quantity $\mu(\pi)$ as follows

$$\mu(\pi) = \sum_{(x,y) \in \mathcal{X} \times \{1,2 \dots m\}} N_{xy}(\pi) h_x(y).$$

Via this definition and the result of Lemma 2, it is immediate that when $m \leq \sqrt{T}$, if we play an arbitrary action sequence and then execute π for $n+1$ periods, then the (expected) cumulative loss experienced in the final period (i.e., the $(n+1)$th period) is $\mu(\pi)\sqrt{T}$. We use this notion of μ to define the policy $\pi^\star$ as

$$\pi^\star \in \operatorname*{argmin}_{\pi \in A_1} \mu(\pi).$$

Recall for this definition that A_1 as defined in Algorithm 2 is the set of all $\sqrt{T}$-cyclic policies. With these definitions in hand, we are now in a position to formally prove Theorem 1.

B.2 Proof of Theorem 1

First, we note that if $m > \sqrt{T}$, then the statement of the theorem is trivially true since the complete policy regret is always upper bounded by T. Hence, for the remainder of the proof it suffices to assume that $m \leq \sqrt{T}$. For any policy π, which is a length T deterministic sequence of actions, let $\ell_t(\pi)$ denote the expected loss suffered at timestep t while playing π. Our next lemma shows that for any tallying bandit problem, the loss suffered by the optimal policy (i.e., the policy in $\mathcal{X}^T$ that experiences the minimum cumulative expected loss) can be well approximated by $T\mu(\pi^\star)$.

Lemma 3. *Given any (m, g, h)-tallying bandit problem, let $\pi^{\star\star}$ denote an optimal policy in $\mathcal{X}^T$. Then*

$$T\mu(\pi^\star) - \sum_{t=1}^{T} \ell_t(\pi^{\star\star}) \leq (m+1)\sqrt{T}.$$

The proof of Lemma 3 is deferred to Appendix B.8. Now let ℓ^s denote the loss experienced in epoch $s \in \{1, 2 \dots S\}$ of Algorithm 2. The following lemma bounds the cumulative loss of Algorithm 2 relative to $T\mu(\pi^\star)$.

Lemma 4. *Assume that $m \leq \sqrt{T}$. With probability at least $1-\delta$, the total loss of Algorithm 2 relative to $T\mu(\pi^\star)$ can be upper bounded as*

$$\sum_{s=1}^{S} \ell^s - T\mu(\pi^\star) \leq Km\sqrt{T}\left(5\log_2\left(\frac{\sqrt{T}}{4Km}+1\right) + 400\sqrt{\log\left(\frac{2Km\log(T)}{\delta}\right)}\right).$$

The proof of this Lemma 4 is provided in Appendix B.3. With the results of Lemma 3 and Lemma 4 in hand, we now utilize them to prove Theorem 1 as follows. Note that the

complete policy regret $\mathcal{R}^{\mathrm{cp}}$ of Algorithm 2 satisfies

$$\begin{aligned}
\mathcal{R}^{\mathrm{cp}} &= \sum_{s=1}^{S} \ell^s - \sum_{t=1}^{T} \ell_t(\pi^{\star\star}) \\
&= \sum_{s=1}^{S} \ell^s - T\mu(\pi^\star) + T\mu(\pi^\star) - \sum_{t=1}^{T} \ell_t(\pi^{\star\star}) \\
&\leq Km\sqrt{T}\left(5\log_2\left(\frac{\sqrt{T}}{4Km}+1\right) + 400\sqrt{\log\left(\frac{2Km\log(T)}{\delta}\right)}\right) + (m+1)\sqrt{T} \\
&\leq 1200Km\sqrt{T}\left(\sqrt{\log\left(2Km\log(T)/\delta\right)} + \log_2\left(\sqrt{T}/(2Km)\right)\right).
\end{aligned}$$

This completes the proof of Theorem 1. ■

B.3 Proof of Lemma 4

To facilitate the proof, we require the following critical lemma, which bounds the loss incurred by Algorithm 2 in each epoch $s \in \{1, 2 \dots S\}$. For the statement of the following lemma, note that completing any epoch $s \in \{1, 2 \dots S\}$ takes a total of $T_s = 2n_s Km\sqrt{T}$ timesteps.

Lemma 5. *Assume that $m \leq \sqrt{T}$. With probability at least $1-\delta$, we have simultaneously for each epoch $s \in \{2, 3 \dots S\}$ that the total loss relative to $T_s\mu(\pi^\star)$ is bounded as*

$$\ell^s - T_s\mu(\pi^\star) \leq Km\left(\sqrt{T} + 8n_s\sqrt{T}C_{s-1}\right).$$

The proof of this Lemma 5 is provided in Appendix B.4. Observe that by the result of Lemma 5, we are guaranteed with probability at least $1-\delta$ that

$$\begin{aligned}
\sum_{s=1}^{S} \ell^s - T\mu(\pi^\star) &= \sum_{s=1}^{S} \left(\ell^s - T_s\mu(\pi^\star)\right) \\
&\leq 4Km\sqrt{T} + \sum_{s=2}^{S} \left(\ell^s - T_s\mu(\pi^\star)\right) \\
&\leq 4Km\sqrt{T} + \sum_{s=2}^{S} Km\left(\sqrt{T} + 8n_s\sqrt{T}C_{s-1}\right) \\
&\leq 5SKm\sqrt{T} + 8Km\sum_{s=2}^{S} n_s\sqrt{T}C_{s-1}.
\end{aligned} \tag{20}$$

Now substituting in the definitions

$$n_s = 2^s \text{ and } C_{s-1} = \sqrt{\frac{32Km}{n_{s-1}\sqrt{T}} \log\left(\frac{2KmS}{\delta}\right)} = \sqrt{\frac{64Km}{n_s\sqrt{T}} \log\left(\frac{2KmS}{\delta}\right)},$$

which were provided in Algorithm 2, into the final term on the RHS of Eq. (20) yields that

$$\begin{aligned}
8Km\sum_{s=2}^{S} n_s\sqrt{T}C_{s-1} &= 8Km\sum_{s=2}^{S} n_s\sqrt{T}\sqrt{\frac{64Km}{n_s\sqrt{T}} \log\left(\frac{2KmS}{\delta}\right)} \\
&= 64K^{1.5}m^{1.5}\sqrt{\log\left(\frac{2KmS}{\delta}\right)}\sum_{s=2}^{S} n_s\sqrt{T}\sqrt{\frac{1}{n_s\sqrt{T}}} \\
&= 64K^{1.5}m^{1.5}\sqrt{\log\left(\frac{2KmS}{\delta}\right)}T^{1/4}\sum_{s=2}^{S}\sqrt{n_s} \\
&= 64K^{1.5}m^{1.5}\sqrt{\log\left(\frac{2KmS}{\delta}\right)}T^{1/4}\sum_{s=2}^{S} 2^{s/2} \\
&\le 400K^{1.5}m^{1.5}\sqrt{\log\left(\frac{2KmS}{\delta}\right)}T^{1/4}2^{S/2}.
\end{aligned}$$

Finally, we recall the definition of $S = \log_2\left(\frac{\sqrt{T}}{4Km} + 1\right)$ to observe that

$$\begin{aligned}
8Km\sum_{s=2}^{S} n_s\sqrt{T}C_{s-1} &\le 400K^{1.5}m^{1.5}\sqrt{\log\left(\frac{2KmS}{\delta}\right)}T^{1/4}2^{S/2} \\
&= 400K^{1.5}m^{1.5}\sqrt{\log\left(\frac{2KmS}{\delta}\right)}T^{1/4}\sqrt{\left(\frac{\sqrt{T}}{4Km} + 1\right)} \\
&\le 400K^{1.5}m^{1.5}\sqrt{\log\left(\frac{2KmS}{\delta}\right)}T^{1/4}\frac{T^{1/4}}{\sqrt{Km}} \\
&= 400Km\sqrt{\log\left(\frac{2KmS}{\delta}\right)}\sqrt{T} \\
&\le 400Km\sqrt{\log\left(\frac{2Km\log(T)}{\delta}\right)}\sqrt{T}
\end{aligned} \tag{21}$$

Combining Eq. (20) with Eq. (21) yields the result. ■

B.4 Proof of Lemma 5

To facilitate the proof, we leverage the following critical lemma, which bounds the gap of the average value μ of policies in A_s versus $\mu(\pi^\star)$.

Lemma 6. *Assume that $m \leq \sqrt{T}$. The event*

$$\cap_{s=1}^{S} \cap_{\pi \in A_s} \left\{ \mu(\pi) - \mu(\pi^\star) \leq 4C_{s-1} \right\},$$

occurs with probability at least $1 - \delta$.

The proof of this Lemma 6 is provided in Appendix C.1.4. Let us now return to the main proof. For ease in notation, let ℓ^{sxy} denote the total loss experienced in epoch s of Algorithm 2 while executing the policy π_{sxy} for $2n_s$ epochs. Hence we have $\ell^s = \sum_{(x,y) \in \mathcal{X} \times \{1,2 \dots m\}} \ell^{sxy}$.
Note that within a single epoch $s > 1$, for each $(x, y) \in \mathcal{X} \times \{1, 2 \dots m\}$ we execute the policy π_{sxy} for $2n_s$ periods, where each period takes $\sqrt{T}$ timesteps. Recall the fact that when $m \leq \sqrt{T}$, if we play an arbitrary action sequence and then execute π_{sxy} for $n + 1$ periods for $n \geq 1$, then the (expected) cumulative loss experienced in the final period (i.e., the $(n + 1)$th period) is $\mu(\pi_{sxy})\sqrt{T}$. In particular, this fact implies that if we execute π_{sxy} for $2n_s$ periods, then the total loss experienced ℓ^{sxy} during these $2n_s$ periods is upper bounded by

$$\ell^{sxy} \leq \mu(\pi_{sxy})\sqrt{T}(2n_s - 1) + \sqrt{T} \leq 2n_s\mu(\pi_{sxy})\sqrt{T} + \sqrt{T}.$$

Hence we can use Lemma 6 to upper bound

$$\begin{aligned} \ell^{sxy} - 2n_s\mu(\pi^\star)\sqrt{T} &\leq 2n_s\mu(\pi_{sxy})\sqrt{T} + \sqrt{T} - 2n_s\mu(\pi^\star)\sqrt{T} \\ &= \sqrt{T} + 2n_s\sqrt{T}\left(\mu(\pi_{sxy}) - \mu(\pi^\star)\right) \\ &\leq \sqrt{T} + 8n_s\sqrt{T}C_{s-1}. \end{aligned}$$

This bound holds uniformly for each $(x, y) \in \mathcal{X} \times \{1, 2 \dots m\}$, and hence we have that

$$\ell^s - T_s\mu(\pi^\star) = \sum_{(x,y) \in \mathcal{X} \times \{1,2 \dots m\}} \left(\ell^{sxy} - 2n_s\mu(\pi^\star)\sqrt{T} \right) \leq Km \left(\sqrt{T} + 8n_s\sqrt{T}C_{s-1} \right).$$

This completes the proof. ■

B.5 Proof of Lemma 6

To facilitate the proof, we require the following two critical helper results. The first result bounds the error incurred when estimating $\mu(\pi)$ via the stochastic realizations

$\left\{\{\widetilde{h}_x(y)_{s,k}\}_{k=1}^{n_s N_{xy}(\pi_{sxy})\sqrt{T}}\right\}_{(x,y)\in\mathcal{X}\times\{1,2\ldots m\}}$. The second result shows that while running Algorithm 2, which is based on successive elimination of inferior policies over epochs $s \in \{1, 2 \ldots S\}$, at any epoch s we never eliminate $\pi^\star$ from our set A_s of feasible policies.

Lemma 7. *Assume that $m \leq \sqrt{T}$. Fix any $s \in \{1, 2 \ldots S\}$, and let B_s denote the event that for all $\pi \in A_s$ we simultaneously have that*

$$|\widehat{\mu}_s(\pi) - \mu(\pi)| \leq C_s.$$

Then B_s occurs with probability at least $1 - \delta/S$.

Lemma 8. *Assume that $m \leq \sqrt{T}$. The event $\cap_{s=1}^S B_s$, where the event B_s is defined in Lemma 7, implies the event that*

$$\pi^\star \in \cap_{s=1}^S A_s \text{ and } \cap_{s=1}^S \left\{0 \leq \widehat{\mu}_s(\pi^\star) - \widehat{\mu}_s(\widehat{\pi}_s) \leq 2C_s\right\}.$$

The proofs of Lemma 7 and Lemma 8 are provided in Appendix B.6 and Appendix B.7 respectively.
Let us now return to the proof. By the result of Lemma 7 and a union bound, the event $\cap_{s=1}^S B_s$ occurs with probability at least $1 - \delta$. Furthermore, the result of Lemma 8 shows that the event $\cap_{s=1}^S B_s$ implies the event

$$\pi^\star \in \cap_{s=1}^S A_s. \tag{22}$$

So on the event $\cap_{s=1}^S B_s$, note that for any s and any $\pi \in A_s$ we have

$$\begin{aligned}
\mu(\pi) - \mu(\pi^\star) &\overset{(i)}{\leq} \widehat{\mu}_{s-1}(\pi) - \mu(\pi^\star) + C_{s-1} \\
&\overset{(ii)}{\leq} \widehat{\mu}_{s-1}(\widehat{\pi}_{s-1}) - \mu(\pi^\star) + 3C_{s-1} \\
&\overset{(iii)}{\leq} \widehat{\mu}_{s-1}(\pi^\star) - \mu(\pi^\star) + 3C_{s-1} \\
&\overset{(iv)}{\leq} \mu(\pi^\star) - \mu(\pi^\star) + 4C_{s-1} \\
&= 4C_{s-1},
\end{aligned}$$

where step (i) follows from Lemma 7, step (ii) follows from the definition of A_s and the fact that $\pi \in A_s$, step (iii) follows from the definition of $\widehat{\pi}_{s-1}$ and Eq. (22), and step (iv) follows again from Lemma 7 and Eq. (22). This completes the proof. ■

B.6 Proof of Lemma 7

For the proof of this lemma, it is useful to define the quantity C_{sxy} as

$$C_{sxy} = \sqrt{\frac{32}{n_s N_{xy}(\pi_{sxy})\sqrt{T}} \log\left(\frac{2KmS}{\delta}\right)},$$

for each $(s, x, y) \in \{1, 2 \dots S\} \times \mathcal{X} \times \{1, 2 \dots m\}$. Fix any $(x, y) \in \mathcal{X} \times \{1, 2 \dots m\}$. Note that by Hoeffding's bound [Hoe63], we are guaranteed that the event

$$\left| h_x(y) - \frac{1}{n_s N_{xy}(\pi_{sxy})\sqrt{T}} \sum_{k=1}^{n_s N_{xy}(\pi_{sxy})\sqrt{T}} \widetilde{h}_x(y)_{s,k} \right| \leq C_{sxy}, \tag{23}$$

occurs with probability at least $1 - \delta/(KmS)$. A union bound then ensures that the above event occurs simultaneously for all $(x, y) \in \mathcal{X} \times \{1, 2 \dots m\}$ with probability at least $1 - \delta/S$. We now claim that this (simultaneous) event is a subset of B, which is sufficient to complete the proof.
To establish the claim, note that on this event, we are guaranteed for any $\pi \in A_s$ that

$$\begin{aligned}
&|\mu(\pi) - \widehat{\mu}_s(\pi)| \\
&= \left| \sum_{(x,y) \in \mathcal{X} \times \{1,2 \dots m\}} N_{xy}(\pi) h_x(y) - \sum_{(x,y) \in \mathcal{X} \times \{1,2 \dots m\}} N_{xy}(\pi) \frac{1}{n_s N_{xy}(\pi_{sxy})\sqrt{T}} \sum_{k=1}^{n_s N_{xy}(\pi_{sxy})\sqrt{T}} \widetilde{h}_x(y)_{s,k} \right| \\
&= \left| \sum_{(x,y) \in \mathcal{X} \times \{1,2 \dots m\}} N_{xy}(\pi) \left(h_x(y) - \frac{1}{n_s N_{xy}(\pi_{sxy})\sqrt{T}} \sum_{k=1}^{n_s N_{xy}(\pi_{sxy})\sqrt{T}} \widetilde{h}_x(y)_{s,k} \right) \right| \\
&\leq \sum_{(x,y) \in \mathcal{X} \times \{1,2 \dots m\}} N_{xy}(\pi) C_{sxy},
\end{aligned}$$

where the final step follows from the triangle inequality and Eq. (23). Continuing the

above, we have that

$$\begin{aligned}
|\mu(\pi) - \widehat{\mu}_s(\pi)| &\leq \sum_{(x,y)\in\mathcal{X}\times\{1,2\ldots m\}} N_{xy}(\pi) C_{sxy} \\
&= \sqrt{\frac{32}{n_s\sqrt{T}} \log\left(\frac{2KmS}{\delta}\right)} \sum_{(x,y)\in\mathcal{X}\times\{1,2\ldots m\}} \frac{N_{xy}(\pi)}{\sqrt{N_{xy}(\pi_{sxy})}} \\
&\overset{(i)}{\leq} \sqrt{\frac{32}{n_s\sqrt{T}} \log\left(\frac{2KmS}{\delta}\right)} \sum_{(x,y)\in\mathcal{X}\times\{1,2\ldots m\}} \sqrt{N_{xy}(\pi)} \\
&\overset{(ii)}{\leq} \sqrt{Km}\sqrt{\frac{32}{n_s\sqrt{T}} \log\left(\frac{2KmS}{\delta}\right)} \\
&= C_s,
\end{aligned}$$

where step (i) follows from the fact that $N_{xy}(\pi_{sxy}) \geq N_{xy}(\pi)$ by its definition in Algorithm 2, and step (ii) follows from the fact that $N_{xy}(\pi) \in [0,1]$, that $\sum_{(x,y)\in\mathcal{X}\times\{1,2\ldots m\}} N_{xy}(\pi) = 1$ as well as the Cauchy-Schwarz inequality. This establishes the claim and hence completes the proof. ■

B.7 Proof of Lemma 8

Assume that the event $\cap_{s'=1}^{S} B_{s'}$ is true. On this event, we prove the lemma by induction on s. First we demonstrate the base case of $s = 1$, which is that $\pi^\star \in A_1$ and $0 \leq \widehat{\mu}_1(\pi^\star) - \widehat{\mu}_1(\widehat{\pi}_1) \leq 2C_1$. Then for the inductive step we show that if the event $\pi^\star \in A_{s-1}$ and $0 \leq \widehat{\mu}_{s-1}(\pi^\star) - \widehat{\mu}_{s-1}(\widehat{\pi}_{s-1}) \leq 2C_{s-1}$ occurs, then we also have that the event

$$\pi^\star \in A_s \text{ and } 0 \leq \widehat{\mu}_s(\pi^\star) - \widehat{\mu}_s(\widehat{\pi}_s) \leq 2C_s,$$

is also true.
For the base case, note that by definition we are guaranteed $\pi^\star \in A_1$. And by the definition of $\widehat{\pi}_1$, we know that $0 \leq \widehat{\mu}_1(\pi^\star) - \widehat{\mu}_1(\widehat{\pi}_1)$. Furthermore, recalling the definition of the event B_1 in Lemma 7, on the event B_1 we have that

$$\widehat{\mu}_1(\pi^\star) - \mu(\pi^\star) \leq C_1 \text{ and } \mu(\widehat{\pi}_1) - \widehat{\mu}_1(\widehat{\pi}_1) \leq C_1.$$

Putting these equations together and using the fact that $\mu(\pi^\star) \leq \mu(\widehat{\pi}_1)$ ensures that

$$\widehat{\mu}_1(\pi^\star) - \widehat{\mu}_1(\widehat{\pi}_1) \leq 2C_1.$$

This verifies the base case.

For the inductive step, assume that $\pi^\star \in A_{s-1}$ and $0 \le \widehat{\mu}_{s-1}(\pi^\star) - \widehat{\mu}_{s-1}(\widehat{\pi}_{s-1}) \le 2C_{s-1}$ occurs. Then the definition of A_s and the inductive hypothesis directly imply that $\pi^\star \in A_s$. Hence, it is true by definition of $\widehat{\pi}_s$ that $0 \le \widehat{\mu}_s(\pi^\star) - \widehat{\mu}_s(\widehat{\pi}_s)$. Then recalling the definition of the event B_s in Lemma 7, on the event B_s we have that

$$\widehat{\mu}_s(\pi^\star) - \mu(\pi^\star) \le C_s \text{ and } \mu(\widehat{\pi}_s) - \widehat{\mu}_s(\widehat{\pi}_s) \le C_s.$$

Putting these equations together and using the fact that $\mu(\pi^\star) \le \mu(\widehat{\pi}_s)$ ensures that

$$\widehat{\mu}_s(\pi^\star) - \widehat{\mu}_s(\widehat{\pi}_s) \le 2C_s.$$

This verifies the inductive step. As argued earlier, this is sufficient to complete the proof. ■

B.8 Proof of Lemma 3

Note that since $\mu(\pi^\star) \in [0,1]$, the statement is trivial for $m \ge \sqrt{T}$. Hence, assume for the remainder of the proof that $m < \sqrt{T}$. There exists some $k \in \{0, 1 \dots \sqrt{T} - 1\}$ such that

$$\sum_{t=k\sqrt{T}+1}^{(k+1)\sqrt{T}} \ell_t(\pi^{\star\star}) \le \frac{1}{\sqrt{T}} \sum_{t=1}^{T} \ell_t(\pi^{\star\star}).$$

Define the $\sqrt{T}$-cyclic policy π by letting $\pi_t = \pi^{\star\star}_{k\sqrt{T}+t}$ for all $1 \le t \le \sqrt{T}$. Note that via the MDP characterization provided in Lemma 1, we can equivalently think of the tallying bandit problem as some MDP that we denote M. The proof of Lemma 1 shows that regardless of the state we start at, either playing $\pi_1, \pi_2 \dots \pi_m$ or playing $\pi^{\star\star}_{k\sqrt{T}+1}, \pi^{\star\star}_{k\sqrt{T}+2} \dots \pi^{\star\star}_{k\sqrt{T}+m}$ leads to the same state in M. From that state, either playing $\pi_{m+1}, \pi_{m+2} \dots \pi_{\sqrt{T}}$ or playing $\pi^{\star\star}_{k\sqrt{T}+m+1}, \pi^{\star\star}_{k\sqrt{T}+m+2} \dots \pi^{\star\star}_{(k+1)\sqrt{T}}$ leads to the same sequence of states, and hence the same sequence of (expected) losses. Hence we have shown that after a single first execution of π, we can bound the loss as

$$\sum_{t=1}^{\sqrt{T}} \ell_k(\pi) \le m + \sum_{t=k\sqrt{T}+1}^{(k+1)\sqrt{T}} \ell_k(\pi^{\star\star}) \le m + \frac{1}{\sqrt{T}} \sum_{t=1}^{T} \ell_t(\pi^{\star\star}).$$

Repeating this argument for $\sqrt{T}$ executions of π, we have shown that

$$\sum_{t=1}^{T} \ell_k(\pi) \le m\sqrt{T} + \sqrt{T} \sum_{t=k\sqrt{T}+1}^{(k+1)\sqrt{T}} \ell_k(\pi^{\star\star}) \le m\sqrt{T} + \sum_{t=1}^{T} \ell_t(\pi^{\star\star}). \tag{24}$$

Now, we observe that $T\mu(\pi) \le \sum_{t=1}^{T} \ell_k(\pi) + \sqrt{T}$, where we used the fact that when $m \le \sqrt{T}$, if we play an arbitrary action sequence and then execute π for $n+1$ periods for

$n \geq 1$, then the (expected) cumulative loss experienced in the final period (i.e., the $(n+1)$th period) is $\mu(\pi)\sqrt{T}$. Finally, we note that $\mu(\pi^\star) \leq \mu(\pi)$ by definition, and so we have that

$$T\mu(\pi^\star) \leq T\mu(\pi) \leq \sum_{t=1}^{T} \ell_k(\pi) + \sqrt{T},$$

which combined with Eq. (24) implies that

$$T\mu(\pi^\star) - \sum_{t=1}^{T} \ell_t(\pi^{\star\star}) \leq (m+1)\sqrt{T}.$$

This completes the proof. ■

B.9 Proof of Lemma 2

To prove this result, we leverage the MDP characterization of tallying bandits provided by Lemma 1. Let M denote the MDP corresponding to the given (m, g, h)-tallying bandit problem. Note that by the proof of Lemma 1, and by the assumption that $m \leq \sqrt{T}$, after executing π for $n \geq 1$ periods we have arrived at the state $(\pi_{n\sqrt{T}-m+1} \dots \pi_{n\sqrt{T}})$. And since π is $\sqrt{T}$-cyclic we are guaranteed that

$$(\pi_{n\sqrt{T}-m+1} \dots \pi_{n\sqrt{T}}) = (\pi_{\sqrt{T}-m+1} \dots \pi_{\sqrt{T}}).$$

So if we execute π for one more period (i.e., the $(n+1)$th period) from this starting state, then regardless of n we will observe an identical sequence of states, since the transition function of the MDP M is deterministic. Hence, the quantity

$$\sum_{t=n\sqrt{T}+1}^{(n+1)\sqrt{T}} \mathbb{I}(\pi_t = x) \cdot \mathbb{I}\left(y = \sum_{t'=\max\{1,t-m+1\}}^{t} \mathbb{I}(\pi_{t'} = x)\right),$$

used to define $N_{xy}(\pi)$ is independent of n, ensuring that $N_{xy}(\pi)$ is well defined. It remains to establish the claim that $N_{xy}(\pi)$ is independent of the action sequence that was played before π was executed for $n+1$ periods. Denote this prior action sequence as $a_{1:k}$ for any finite value of k. Note that regardless of what $a_{1:k}$ is, after we execute π for n periods we still arrive at the state

$$(\pi_{n\sqrt{T}-m+1} \dots \pi_{n\sqrt{T}}) = (\pi_{\sqrt{T}-m+1} \dots \pi_{\sqrt{T}}),$$

in the MDP M. Then if we execute π for one more period (i.e., the $(n+1)$th period) from this starting state, then regardless of $a_{1:k}$ we will observe an identical sequence of states, since the transition function of the MDP M is deterministic. This ensures that $N_{xy}(\pi)$ is well defined. ■

B.10 Proof of Lemma 1

Given an (m, g, h)-tallying bandit problem, for ease in notation let the (finite) action set $\mathcal{X}$ be denoted as $\{1, 2 \dots K\}$. To define the MDP M, let its state space be $(\{0\} \cup \mathcal{X})^m$ and let its action space be $\mathcal{X}$. Let the initial state be the length m vector $(0, 0 \dots 0)$. For each state $s = (s_1, s_2 \dots s_m)$ and action i, define the deterministic transition function $\mathcal{T}_M : (\{0\} \cup \mathcal{X})^m \times \mathcal{X} \to (\{0\} \cup \mathcal{X})^m$ of the MDP as

$$\mathcal{T}_M(s, i) = (s_2, s_2 \dots s_{m-1}, s_m, i).$$

Let the (finite) horizon of the MDP be the time horizon T of the tallying bandit problem. Finally, define for each state s the (expected) reward function $R_M : (\{0\} \cup \mathcal{X})^m \to [0, 1]$ of the MDP as

$$R_M(s) = 1 - h_{s_m} \left(\sum_{t=1}^{m} \mathbb{I}(s_t = s_m) \right).$$

It is immediate the taking actions in the tallying bandit problem corresponds to taking actions in the state space of this MDP. ■

B.11 Proof of Theorem 2

First note that as an immediate consequence of Proposition 1, we must have $\mathbb{E}\left[\mathcal{R}^{\mathrm{cp}}\right] \geq mK/128$. So for the remainder of the proof, we focus on showing that $\mathbb{E}\left[\mathcal{R}^{\mathrm{cp}}\right] \geq c\sqrt{mKT}$ for some numerical constant $c > 0$. Also note that due to the result of Proposition 1, we can assume for this proof that $m \leq T/100$, because the complete policy regret scales linearly with T in the regime that $m > T/100$.

At a high level, our proof will proceed via a reduction to best arm identification in stochastic multi armed bandit problems [Sli19]. Roughly speaking, we will show the existence of an (m, g, h)-tallying bandit problem, such that minimizing the complete policy regret in this problem is at least as hard as identifying the best arm in a stochastic multi armed bandit problem with $\widetilde{\Theta}(mK)$ arms.

To this end, we first recall Lemma 1, which was originally stated at the beginning of Appendix B.1 and proved in Appendix B.10. We have restated it here for convenience since it shall be useful for our proof of Proposition 1.

Lemma 1. *Any (m, g, h)-tallying bandit problem can be equivalently expressed as a finite horizon Markov decision process (MDP).*

Now, construct an (m, g, h)-tallying bandit problem using the following procedure, where we assume that m is at least some sufficiently large universal constant. Sample $(x^\star, y^\star)$

uniformly at random from $\mathcal{X} \times \{23m/24, 23m/24 + 1 \dots m\}$. Define $h_x(y) = 1/2$ for each $(x, y) \in \mathcal{X} \times \{1, 2 \dots m\}$ such that $(x, y) \neq (x^\star, y^\star)$. Also define $h_{x^\star}(y^\star) = 1/2 - \epsilon$ for some $\epsilon \in (0, 1)$ to be specified later.

We now define the stochastic bandit feedback model for this tallying bandit problem as follows. When the player plays action x, and this action x has been played a total of y times in the past m timesteps (including the current timestep), then the player receives as feedback a Bernoulli random variable with mean $h_x(y)$. It is immediate that this feedback model meets the criteria outlined in Definition 4.

Let us now upper bound the cumulative loss incurred by the optimal policy in this tallying bandit problem. To do so, consider the policy π, which is a length T sequence of actions, that we define as follows. Choose some $x \in \mathcal{X}$ such that $x \neq x^\star$. We define π to choose action $x^\star$ for $y^\star$ timesteps and then choose action x for $m - y^\star$ timesteps, and then repeat this length m sequence over and over. Recalling the definition of a $\sqrt{T}$-cyclic policy that was stated in Section 3.4.1, we can analogously say that π is an m-cyclic policy, such that within each period of length m it plays $x^\star$ for $y^\star$ times and then plays x for $m - y^\star$ times. By Lemma 1, there exists an MDP M that is equivalent to the constructed tallying bandit problem. Recalling the characterization of this MDP M provided in the proof of Lemma 1, let us understand the sequence of states in M that we arrive at when we follow π. In the first m timesteps, π plays $x^\star$ for $y^\star$ times and then plays x for $m - y^\star$ times, and so we arrive at the state

$$(x^\star, x^\star \dots x^\star, x, x \dots x) \equiv (x^\star)^{y^\star} \times x^{m-y^\star}. \tag{25}$$

Then for the next $y^\star$ timesteps, π plays $x^\star$ repeatedly. This leads to the progression of states given by

$$\begin{gathered} (x^\star)^{y^\star - 1} \times x^{m-y^\star} \times (x^\star) \\ (x^\star)^{y^\star - 2} \times x^{m-y^\star} \times (x^\star)^2 \\ (x^\star)^{y^\star - 3} \times x^{m-y^\star} \times (x^\star)^3 \\ \vdots \\ (x^\star) \times x^{m-y^\star} \times (x^\star)^{y^\star - 1} \\ x^{m-y^\star} \times (x^\star)^{y^\star}. \end{gathered} \tag{26}$$

Then for the next $m - y^\star$ timesteps, it plays x, so that the state after these timesteps is

$$(x^\star, x^\star \dots x^\star, x, x \dots x) \equiv (x^\star)^{y^\star} \times x^{m-y^\star},$$

and we have arrived back at the state listed in Eq. (25).

Let us now use this insight to bound the expected loss incurred by following π. Critically, for every period of m timesteps after the very first period, Eq. (26) shows that we observe (stochastic instantiations of) the loss value $h_{x^\star}(y^\star)$ for exactly $y^\star \in \{23m/24, 23m/24 + 1 \dots m\}$ timesteps. We hence upper bound the expected

cumulative loss incurred by π as

$$\begin{aligned}(1/2)m + ((1/2-\epsilon)y^\star + (1/2)(m-y^\star))\frac{T-m}{m} &= m/2 + (-\epsilon y^\star + (1/2)m)\frac{T-m}{m} \\ &= -\epsilon y^\star \frac{T-m}{m} + \frac{T-m}{2} + m/2 \\ &\leq -\epsilon \frac{23m}{24}\frac{T-m}{m} + T/2 \\ &= -\epsilon \frac{23(T-m)}{24} + T/2 \\ &= T/2 - \epsilon 23T/24 + 23m\epsilon/24.\end{aligned} \tag{27}$$

Let us now consider the performance of any algorithm which attempts to solve this tallying bandit problem. Instead of the algorithm operating in the usual oracle model, where it might need m actions to arrive at a state that it deems beneficial, let us strengthen the algorithm by equipping it with a generative model. Concretely, we strengthen the algorithm so that at any timestep, it can query any state in M and receive a stochastic instantiation of the $h_x(y)$ loss value corresponding to that state. Given this generative model, it is immediate that an algorithm that is attempting to minimize its cumulative loss (or equivalently, minimize its complete policy regret), is attempting to maximize the number of times it queries states whose loss value is $h_{x^\star}(y^\star)$.
Hence, we can interpret this algorithm as running best arm identification on a classical stochastic multi armed bandit problem with $mK/24$ arms. Note that the number of arms in this stochastic multi armed bandit problem is $mK/24$, since $(x^\star, y^\star)$ was sampled uniformly at random from a set of cardinality $mK/24$. The remainder of the argument closely follows that of Slivkins [Sli19]. So pick $\epsilon = \sqrt{cmK/T}$, for some numerical constant $c > 0$ whose precise value can be found in the proofs of Corollary 2.9 and Theorem 2.10 of Slivkins [Sli19]. These two results show that for each timestep less than or equal to T, with probability at least $1/12$ the algorithm does not select a state whose loss value is $h_{x^\star}(y^\star)$. In particular, this means that the expected cumulative loss of this algorithm is at least

$$(1/12)T/2 + (11/12)T(1/2-\epsilon) = T/24 + 11T/24 - 11\epsilon T/12 = T/2 - 11\epsilon T/12. \tag{28}$$

Putting together Eq. (27) and Eq. (28), we hence have that the expected complete policy regret of this algorithm is lower bounded as

$$\begin{aligned}\mathbb{E}\left[\mathcal{R}^{\mathrm{cp}}\right] &\geq T/2 - 11\epsilon T/12 - (T/2 - \epsilon 23T/24 + 23m\epsilon/24) \\ &= -11\epsilon T/12 + \epsilon 23T/24 - 23m\epsilon/24 \\ &= \epsilon T/24 - 23m\epsilon/24 \\ &\geq c\sqrt{mKT},\end{aligned}$$

where in the final step we used our assumption that $m \leq T/100$, substituted in the definition of $\epsilon = \sqrt{cmK/T}$ and redefined the value of the numerical constant c. This completes the proof. ■

B.12 Proof of Proposition 1

First, we recall Lemma 1, which was originally stated at the beginning of Appendix B.1 and proved in Appendix B.10. We have restated it here for convenience since it shall be useful for our proof of Proposition 1.

Lemma 1. *Any (m, g, h)-tallying bandit problem can be equivalently expressed as a finite horizon Markov decision process (MDP).*

We now formally define the algorithm $\mathbb{ALG}_{\text{det}}$ below in Algorithm 4. Subsequently, we prove the upper bound in Proposition 1, and then we prove the lower bound in Proposition 1.

Algorithm 4 $\mathbb{ALG}_{\text{det}}$

Require: memory capacity m, time horizon T

for $x \in \mathcal{X}$ **do**
  for $y \in \{1, 2 \dots m\}$ **do**
    Choose action x.
    Observe and store $h_x(y)$.
  end for
end for
Plan (offline) an optimal policy $\overline{\pi} = \{\overline{\pi}_k\}_{k=mK+1}^{T}$ to play for remaining $T - mK$ timesteps.
Choose actions according to $\overline{\pi}$ for remaining $T - mK$ timesteps.

B.13 Proof of Upper Bound in Proposition 1

First note that the offline planning step in Algorithm 4 is statistically (although perhaps not computationally) feasible, since the player has full information about the loss functions once it stores $h_x(y)$ for each $(x, y) \in \mathcal{X} \times \{1, 2 \dots m\}$.
Let $\pi^{\star\star}$ denote the optimal policy for the problem, so that $\pi^{\star\star} \in \mathcal{X}^T$ is a length T sequence of actions such that following this sequence achieves the minimum loss. Let $\ell_t(\pi^{\star\star})$ denote the loss incurred at the tth timestep while playing the action sequence $\{\pi_t^{\star\star}\}_{t=1}^{T}$. Let $\ell(\overline{\pi})$ denote the total loss incurred by the final step of Algorithm 4 which starts playing $\overline{\pi}$ at timestep $mK + 1$.
To complete the proof, we will leverage the MDP characterization of tallying bandit problems that was established in the proof of Lemma 1. Let M denote the equivalent MDP for this tallying bandit problem. Observe that $\overline{\pi}$ is the optimal policy for the remaining $T - mK$ timesteps assuming that $\overline{\pi}$ is initialized at the initial state $s = (K, K \dots K)$ in

this MDP M. Also note that playing the action sequence $\{\pi_t^{\star\star}\}_{t=mK+1}^{T}$ from the state s leads to a cumulative loss that is upper bounded by

$$m + \sum_{t=m(K+1)+1}^{T} \ell_t(\pi^{\star\star}),$$

which follows because regardless of the initial state, we arrive at state $s' = (\pi_{mK+1}^{\star\star}, \pi_{mK+2}^{\star\star} \dots \pi_{m(K+1)}^{\star\star})$ in M after playing the length m action sequence $\{\pi_t^{\star\star}\}_{t=mK+1}^{m(K+1)}$. From state s', the losses we experience when playing $\{\pi_t^{\star\star}\}_{t=m(K+1)+1}^{T}$ at each timestep $t \geq m(K+1)+1$ are exactly $\ell_t(\pi^{\star\star})$. The optimality of $\overline{\pi}$ hence implies that

$$\ell(\overline{\pi}) \leq m + \sum_{t=m(K+1)+1}^{T} \ell_t(\pi^{\star\star}). \tag{29}$$

Hence, by naively upper bounding the loss incurred in the first mK timesteps, the complete policy regret of Algorithm 4 is upper bounded via Eq. (29) as

$$\begin{aligned}
\mathcal{R}^{\mathrm{cp}} &\leq mK + \ell(\overline{\pi}) - \sum_{t=mK+1}^{T} \ell_t(\pi^{\star\star}) \\
&\leq mK + m + \sum_{t=m(K+1)+1}^{T} \ell_t(\pi^{\star\star}) - \sum_{t=mK+1}^{T} \ell_t(\pi^{\star\star}) \\
&= mK + m - \sum_{t=mK+1}^{m(K+1)} \ell_t(\pi^{\star\star}) \\
&\leq (m+1)K.
\end{aligned}$$

This verifies the upper bound in Proposition 1. ■

B.14 Proof of Lower Bound in Proposition 1

Construct an (m, g, h)-tallying bandit problem via the following procedure. Sample an action $x^\star$ uniformly at random from $\mathcal{X}$, and keep its identity hidden from the user. Define the functions $\{h_x\}_{x \in \mathcal{X}}$ as

$$h_{x^\star}(y) = \begin{cases} 1 \text{ if } y < m \\ 0 \text{ if } y = m \end{cases} \quad \text{and } h_x = 1 \text{ if } x \neq x^\star.$$

It is immediate that the optimal policy always plays the action $x^\star$, and its cumulative loss is precisely $m - 1$. Meanwhile, to obtain zero loss at any timestep, the player must identify

$x^\star$. Note that to identify whether a certain action x equals $x^\star$, the player must play x for m consecutive times, in order to receive the feedback $h_x(m)$. Since there are K actions, and identifying whether an action is the correct one requires m queries, a standard counting argument [DKWY20] reveals that (in expectation over a possibly randomized strategy) the player makes at least $mK/64$ queries before observing $h_{x^\star}(m)$. Hence, the (expected) complete policy regret of the (possibly randomized) player is lower bounded as

$$\mathbb{E}\left[\mathcal{R}^{\mathrm{cp}}\right] \geq mK/64 - (m-1) \geq mK/128,$$

where we assume that K is larger than some numerical constant. This verifies the lower bound in Proposition 1. ■

C Appendix for Section 4

C.1 Analysis of Algorithm 3

In this section, we analyze the complete policy regret of Algorithm 3, and prove Theorem 3. As discussed in Section 4.3, our analysis is overall rather standard, although we require a careful choice of parameters to ensure optimal dependencies in the final result. Thus, many of the computations closely follow those of Malik et al. [MLS22], but we nevertheless provide the entire argument for the sake of completeness. Before we formally prove Theorem 3, we first introduce the function $\mu : \mathcal{X} \to [0, 1]$ which will be useful for our proofs. For any action $x \in \mathcal{X}$ let us define $\mu(x)$ as

$$\mu(x) = h_x(\|w_x\|_1),$$

With this definition in hand, we are now in a position to formally prove Theorem 3.

C.1.1 Proof of Theorem 3

For any policy π, which is a length T deterministic sequence of actions, let $\ell_t(\pi)$ denote the expected loss suffered at timestep t while playing π. Define the policy $\pi^\star$ as

$$\pi^\star \in \operatorname*{argmin}_{\pi \in \mathcal{X}^T} \sum_{t=1}^{T} \ell_t(\pi),$$

so that $\pi^\star$ is an optimal policy (i.e., a policy that suffers the minimum cumulative expected loss). Note that the definition of an (m, w, h)-weighted tallying bandit ensures that for any timestep t, there exists an action $x \in \mathcal{X}$ and $y \in \{1\} \times \{0, 1\}^{m-1}$ such that $\ell_t(\pi^\star) = h_x(w_x^\top y)$. Thus, the α-REO condition ensures that

$$\mu(x^\star) = h_{x^\star}(\|w_{x^\star}\|_1) \leq h_x(w_x^\top y) + \alpha = \ell_t(\pi^\star) + \alpha.$$

In particular, this implies that

$$\sum_{t=1}^{T} \ell_t(\pi^\star) \geq T\mu(x^\star) - \alpha T. \tag{30}$$

Now let ℓ^s denote the loss experienced in epoch $s \in \{1, 2 \dots S\}$ of Algorithm 3. The following lemma bounds the cumulative loss of Algorithm 3 relative to $T\mu(x^\star)$.

Lemma 9. *With probability at least $1-\delta$, the total loss of Algorithm 3 relative to $T\mu(x^\star)$ can be upper bounded as*

$$\sum_{s=1}^{S} \ell^s - T\mu(x^\star) \leq 4KM + Km\log(T) + 800\sqrt{KT\log\left(\frac{2K\log(T)}{\delta}\right)}.$$

The proof of this Lemma 9 is provided in Appendix C.1.2. With the result of Lemma 9 in hand, we now utilize it to prove Theorem 3 as follows. Note via Eq. (30) and Lemma 9 that the complete policy regret $\mathcal{R}^{\text{cp}}$ of Algorithm 3 satisfies

$$\begin{aligned}
\mathcal{R}^{\text{cp}} &= \sum_{s=1}^{S} \ell^s - \sum_{t=1}^{T} \ell_t(\pi^\star) \\
&\leq \sum_{s=1}^{S} \ell^s - T\mu(x^\star) + \alpha T \\
&\leq 4KM + Km\log(T) + 800\sqrt{KT\log\left(\frac{2K\log(T)}{\delta}\right)} + \alpha T.
\end{aligned}$$

This completes the proof of Theorem 3. ■

C.1.2 Proof of Lemma 9

To facilitate the proof, we require the following critical lemma, which bounds the loss incurred by Algorithm 3 in each epoch $s \in \{1, 2 \dots S\}$. For the statement of the following lemma, note that completing any epoch $s \in \{1, 2 \dots S\}$ takes a total of $T_s = 2|A_s|n_s$ timesteps.

Lemma 10. *With probability at least $1-\delta$, we have simultaneously for each epoch $s \in \{2, 3 \dots S\}$ that the total loss relative to $T_s\mu(x^\star)$ is bounded as*

$$\ell^s - T_s\mu(x^\star) \leq |A_s|\left(m + 4(2n_s - m)C_{s-1}\right).$$

The proof of this Lemma 10 is provided in Appendix C.1.3. Observe that by the result of

Lemma 10, we are guaranteed with probability at least $1-\delta$ that

$$\begin{aligned}\sum_{s=1}^{S} \ell^s - T\mu(x^\star) &= \sum_{s=1}^{S} \left(\ell^s - T_s\mu(x^\star)\right) \\ &\leq 4KM + \sum_{s=2}^{S} \left(\ell^s - T_s\mu(x^\star)\right) \\ &\leq 4KM + \sum_{s=2}^{S} |A_s| \left(m + 4(2n_s - m)C_{s-1}\right) \\ &\leq 4KM + SKm + 8\sum_{s=2}^{S} |A_s| n_s C_{s-1}.\end{aligned} \tag{31}$$

Recall the definitions $n_s = KM2^s/|A_s|$ and $T_s = 2|A_s|n_s$ provided in Algorithm 3. Also note that

$$C_{s-1} = \sqrt{\frac{32}{n_{s-1}} \log\left(\frac{2KS}{\delta}\right)} = \sqrt{\frac{32}{n_s|A_s|/(2|A_{s-1}|)} \log\left(\frac{2KS}{\delta}\right)} = \sqrt{\frac{64|A_{s-1}|}{n_s|A_s|} \log\left(\frac{2KS}{\delta}\right)}.$$

Substituting the above relations into the final term on the RHS of Eq. (31), we get that

$$\begin{aligned}8\sum_{s=2}^{S} |A_s| n_s C_{s-1} &= 8\sum_{s=2}^{S} |A_s| n_s \sqrt{\frac{64|A_{s-1}|}{n_s|A_s|} \log\left(\frac{2KS}{\delta}\right)} \\ &= 8\sqrt{\log\left(\frac{2KS}{\delta}\right)} \sum_{s=2}^{S} |A_s| n_s \sqrt{\frac{64|A_{s-1}|}{n_s|A_s|}} \\ &= 8\sqrt{\log\left(\frac{2KS}{\delta}\right)} \sum_{s=2}^{S} |A_s| \sqrt{n_s} \sqrt{\frac{64|A_{s-1}|}{|A_s|}} \\ &= 8\sqrt{\log\left(\frac{2KS}{\delta}\right)} \sum_{s=2}^{S} |A_s| \sqrt{\frac{KM2^s}{|A_s|}} \sqrt{\frac{64|A_{s-1}|}{|A_s|}} \\ &= 8\sqrt{\log\left(\frac{2KS}{\delta}\right)} \sum_{s=2}^{S} \sqrt{KM2^s} \sqrt{64|A_{s-1}|} \\ &\leq 8\sqrt{\log\left(\frac{2KS}{\delta}\right)} \sum_{s=2}^{S} K\sqrt{M2^s} \sqrt{64} \\ &\leq 800\sqrt{\log\left(\frac{2KS}{\delta}\right)} K\sqrt{M}2^{S/2}.\end{aligned}$$

Now recall from Algorithm 3 the definition of $S = \log_2\left(\frac{T}{4KM} + 1\right)$. Substituting this into the equation above, we get that

$$\begin{aligned} 8\sum_{s=2}^{S} |A_s| n_s C_{s-1} &\leq 800\sqrt{\log\left(\frac{2KS}{\delta}\right)} K\sqrt{M} 2^{S/2} \\ &= 800\sqrt{\log\left(\frac{2KS}{\delta}\right)} K\sqrt{M}\sqrt{\frac{T}{4KM} + 1} \\ &\leq 800\sqrt{\log\left(\frac{2KS}{\delta}\right)} K\sqrt{M}\sqrt{\frac{T}{KM}} \\ &= 800\sqrt{\log\left(\frac{2KS}{\delta}\right)} \sqrt{K}\sqrt{T}. \end{aligned} \tag{32}$$

Combining Eq. (31) with Eq. (32) and using the upper bound $S \leq \log(T)$ yields the result. ■

C.1.3 Proof of Lemma 10

To facilitate the proof, we leverage the following critical lemma, which bounds the gap of the value $\mu(x)$ of each action $x \in A_s$ versus $\mu(x^\star)$.

Lemma 11. *The event*

$$\cap_{s=2}^{S} \cap_{x \in A_s} \left\{\mu(x) - \mu(x^\star) \leq 4C_{s-1}\right\},$$

occurs with probability at least $1 - \delta$.

The proof of this Lemma 11 is provided in Appendix C.1.4. Let us now return to the main proof. For any epoch $s > 1$ and any action $x \in A_s$, let ℓ^s denote the total loss experienced in epoch s of Algorithm 3 while executing the action x for $2n_s$ times. Hence we have $\ell^s = \sum_{x \in A_s} \ell^{sx}$.
Note that within a single epoch $s > 1$, for each $x \in A_s$ we execute x for $2n_s$ times. For the latter $2n_s - m$ times that x is executed, action x has been played m times in the previous m timesteps. Hence, for the latter $2n_s - m$ times that x is executed, the expected loss of playing the action x is $h_x(m) = \mu(x)$. Thus, we have that

$$\ell^{sx} \leq m + (2n_s - m)\mu(x).$$

Hence, for each $s > 1$ and $x \in A_s$, we can use Lemma 11 to upper bound

$$\begin{aligned}\ell^{sx} - 2n_s\mu(x^\star) &\leq m + (2n_s - m)\mu(x) - 2n_s\mu(x^\star) + m\mu(x^\star) \\ &\leq m + (2n_s - m)\left(\mu(x) - \mu(x^\star)\right) \\ &\leq m + 4(2n_s - m)C_{s-1}.\end{aligned}$$

This bound holds uniformly for each $x \in A_s$. Recalling that $T_s = 2|A_s|n_s$, we hence have that

$$\begin{aligned}\ell^s - T_s\mu(x^\star) &= \sum_{x \in A_s} \ell^{sx} - 2|A_s|n_s\mu(x^\star) \\ &= \sum_{x \in A_s} \left(\ell^{sx} - 2n_s\mu(x^\star)\right) \\ &\leq \sum_{x \in A_s} \left(m + 4(2n_s - m)C_{s-1}\right) \\ &= |A_s|\left(m + 4(2n_s - m)C_{s-1}\right).\end{aligned}$$

This completes the proof. ■

C.1.4 Proof of Lemma 11

To facilitate the proof, we require the following two critical helper results. The first result bounds the error incurred when estimating $\mu(x)$ via the stochastic realizations $\{\widetilde{h}_x(m)_{s,k}\}$. The second result shows that while running Algorithm 3, which is based on successive elimination of inferior actions over epochs $s \in \{1, 2 \dots S\}$, at any epoch s we never eliminate $x^\star$ from our set A_s of feasible actions.

Lemma 12. *Fix any $s \in \{1, 2 \dots S\}$, and let B_s denote the event that for all actions $x \in A_s$ we simultaneously have that*

$$|\widehat{\mu}_s(x) - \mu(x)| \leq C_s.$$

Then B_s occurs with probability at least $1 - \delta/S$.

Lemma 13. *The event $\cap_{s=1}^S B_s$, where the event B_s is defined in Lemma 12, implies the event that*

$$x^\star \in \cap_{s=1}^S A_s \text{ and } \cap_{s=1}^S \left\{0 \leq \widehat{\mu}_s(x^\star) - \widehat{\mu}_s(\widehat{x}_s) \leq 2C_s\right\}.$$

The proofs of Lemma 12 and Lemma 13 are provided in Appendix C.1.5 and Appendix C.1.6 respectively.

Let us now return to the proof. By the result of Lemma 12 and a union bound, the event $\cap_{s=1}^{S} B_s$ occurs with probability at least $1-\delta$. Furthermore, the result of Lemma 13 shows that the event $\cap_{s=1}^{S} B_s$ implies the event

$$x^\star \in \cap_{s=1}^{S} A_s. \tag{33}$$

So on the event $\cap_{s=1}^{S} B_s$, note that for any $s > 1$ and any action $x \in A_s$ we have

$$\begin{aligned}
\mu(x) - \mu(x^\star) &\overset{(i)}{\leq} \widehat{\mu}_{s-1}(x) - \mu(x^\star) + C_{s-1} \\
&\overset{(ii)}{\leq} \widehat{\mu}_{s-1}(\widehat{x}_{s-1}) - \mu(x^\star) + 3C_{s-1} \\
&\overset{(iii)}{\leq} \widehat{\mu}_{s-1}(x^\star) - \mu(x^\star) + 3C_{s-1} \\
&\overset{(iv)}{\leq} \mu(x^\star) - \mu(x^\star) + 4C_{s-1} \\
&= 4C_{s-1},
\end{aligned}$$

where step (i) follows from Lemma 12, step (ii) follows from the definition of A_s and the fact that $x \in A_s$, step (iii) follows from the definition of $\widehat{x}_{s-1}$ and Eq. (33), and step (iv) follows again from Lemma 12 and Eq. (33). This completes the proof. ■

C.1.5 Proof of Lemma 12

Fix any $x \in \mathcal{X}$. Recalling the definition of C_s provided in Algorithm 3, Hoeffding's bound [Hoe63] ensures that the event

$$|\widehat{\mu}_s(x) - \mu(x)| = \left| \mu(x) - \frac{1}{n_s} \sum_{k=1}^{n_s} \widetilde{h}_x(m)_{s,k} \right| \leq \sqrt{\frac{32}{n_s} \log\left(\frac{2KS}{\delta}\right)} = C_s, \tag{34}$$

occurs with probability at least $1 - \delta/(KS)$. Since $|A_s| \leq K$, a union bound then ensures that the above event occurs simultaneously for all $x \in A_s$ with probability at least $1 - \delta/S$. ■

C.1.6 Proof of Lemma 13

Assume that the event $\cap_{s'=1}^{S} B_{s'}$ is true. On this event, we prove the lemma by induction on s. First we demonstrate the base case of $s = 1$, which is that $x^\star \in A_1$ and $0 \leq \widehat{\mu}_1(x^\star) - \widehat{\mu}_1(\widehat{x}_1) \leq 2C_1$. Then for the inductive step we show that if the event $x^\star \in A_{s-1}$ and $0 \leq \widehat{\mu}_{s-1}(x^\star) - \widehat{\mu}_{s-1}(\widehat{x}_{s-1}) \leq 2C_{s-1}$ occurs, then we also have that the event

$$x^\star \in A_s \text{ and } 0 \leq \widehat{\mu}_s(x^\star) - \widehat{\mu}_s(\widehat{x}_s) \leq 2C_s,$$

is also true.
For the base case, note that by definition we are guaranteed $x^\star \in A_1$. And by the definition of $\widehat{x}_1$, we know that $0 \leq \widehat{\mu}_1(x^\star) - \widehat{\mu}_1(\widehat{x}_1)$. Furthermore, recalling the definition of the event B_1 in Lemma 12, on the event B_1 we have that

$$\widehat{\mu}_1(x^\star) - \mu(x^\star) \leq C_1 \text{ and } \mu(\widehat{x}_1) - \widehat{\mu}_1(\widehat{x}_1) \leq C_1.$$

Putting these equations together and using the fact that $\mu(x^\star) \leq \mu(\widehat{x}_1)$ ensures that

$$\widehat{\mu}_1(x^\star) - \widehat{\mu}_1(\widehat{x}_1) \leq 2C_1.$$

This verifies the base case.
For the inductive step, assume that $x^\star \in A_{s-1}$ and $0 \leq \widehat{\mu}_{s-1}(x^\star) - \widehat{\mu}_{s-1}(\widehat{x}_{s-1}) \leq 2C_{s-1}$ occurs. Then the definition of A_s and the inductive hypothesis directly imply that $x^\star \in A_s$. Hence, it is true by definition of $\widehat{x}_s$ that $0 \leq \widehat{\mu}_s(x^\star) - \widehat{\mu}_s(\widehat{x}_s)$. Then recalling the definition of the event B_s in Lemma 12, on the event B_s we have that

$$\widehat{\mu}_s(x^\star) - \mu(x^\star) \leq C_s \text{ and } \mu(\widehat{x}_s) - \widehat{\mu}_s(\widehat{x}_s) \leq C_s.$$

Putting these equations together and using the fact that $\mu(x^\star) \leq \mu(\widehat{x}_s)$ ensures that

$$\widehat{\mu}_s(x^\star) - \widehat{\mu}_s(\widehat{x}_s) \leq 2C_s.$$

This verifies the inductive step. As argued earlier, this is sufficient to complete the proof. ■

C.2 Proof of Theorem 4

Assume for the sake of contradiction that the statement is false. Then there exists some ϵ satisfying the given conditions and some function f, such that $\overline{\mathcal{A}}_{\epsilon,f}$ is not empty. This implies the existence of an algorithm $\mathcal{A}$, such that when it is given as input any positive integers T, K, M with $M \leq T$, the algorithm $\mathcal{A}$ satisfies that

$$\mathbb{E}[\mathcal{R}^{\text{cp}}(\mathcal{A}, \mathsf{tb})] \leq \min\left\{T/4, f(m_{\mathsf{tb}}, K)\left(T^{1-\epsilon_1} + T^{\epsilon_3} M^{1-\epsilon_2}\right)\right\} \text{ for all } \mathsf{tb} \in \mathsf{UTB}_{T,M,K}. \quad (35)$$

If $\mathcal{A}$ was a randomized algorithm, then this implies the existence of a deterministic algorithm with the same property. So we can assume without loss of generality that $\mathcal{A}$ is deterministic.
Fix some integer $K \geq 2$. Pick some sufficiently large T, M such that the following conditions hold simultaneously

$$M < T/4 \text{ and } f(1, K)\left(T^{1-\epsilon_1} + T^{\epsilon_3} M^{1-\epsilon_2}\right) < M/2. \quad (36)$$

To see these conditions are simultaneously feasible, recall that $\epsilon_1 \in (0,1)$ and $0 \leq \epsilon_3 < \epsilon_2 < 1$. Let $\gamma = \min\{\epsilon_1, \epsilon_2 - \epsilon_3\} > 0$. So if we choose $M = T^{1-\gamma/2}$, then since this M satisfies $M = T^{1-\gamma/2} < T$, we have that

$$\begin{aligned} f(1,K)\left(T^{1-\epsilon_1} + T^{\epsilon_3} M^{1-\epsilon_2}\right) &< f(1,K)\left(T^{1-\epsilon_1} + T^{\epsilon_3} T^{1-\epsilon_2}\right) \\ &= f(1,K)\left(T^{1-\epsilon_1} + T^{1-(\epsilon_2-\epsilon_3)}\right) \\ &\leq 2f(1,K)T^{1-\gamma}. \end{aligned}$$

So for sufficiently large T, we have for this choice of $M = T^{1-\gamma/2}$ that $M < T/4$ and also that $f(1,K)\left(T^{1-\epsilon_1} + T^{\epsilon_3} M^{1-\epsilon_2}\right) < M/2$. This shows that Eq. (36) is feasible.
We will now define two unweighted tallying bandit problems, each of which have K actions. Recall that in an unweighted tallying bandit problem with memory capacity m, the loss associated with playing an action at a given timestep is fully defined by the number of times that action was played in the last m timesteps. Concretely, assume that in some unweighted tallying bandit problem tb, we play action x on the current timestep, and the total number of times it has been played in the last m timesteps (including the current timestep) is $1 \leq y \leq m$. Then there exists a function $h_{\mathsf{tb},x} : \{1, 2 \dots m\} \to [0,1]$, such that denote the loss associated with playing this action is given by $h_{\mathsf{tb},x}(y)$. We will use this notation to instantiate the forthcoming unweighted tallying bandit problems.
With this notation in hand, let us instantiate the unweighted tallying bandit problem tb_A with memory length $m_{\mathsf{tb}_A} = 1$ as follows. For action x_1, we have that $h_{\mathsf{tb}_A,x_1} = 1/2$. For action x_2, we have that $h_{\mathsf{tb}_A,x_2} = 1$. And for every other action x, let $h_{\mathsf{tb}_A,x} = 1$. We say that whenever the player plays action x, the player almost surely observes $h_{\mathsf{tb}_A,x}(1)$. Notice that since $m_{\mathsf{tb}_A} = 1$, and there is no stochasticity in the observation of losses when we play any action, tb_A is indeed a deterministic multi-armed bandit problem.
Now, we instantiate the tallying bandit problem tb_B with memory length $m_{\mathsf{tb}_B} = M$ as follows. For action x_1 we define $h_{\mathsf{tb}_B,x_1} = 1/2$. For action x_2 we define

$$h_{\mathsf{tb}_B,x_2}(y) = \begin{cases} 1 \text{ if } 1 \leq y < M \\ 0 \text{ if } y = M \end{cases}.$$

And for every other action x, we define $h_{\mathsf{tb}_B,x} = 1$. Once again, we enforce that there is no stochasticity in the player's observation of losses when the player plays an action. So the feedback model in tb_B is deterministic.
Let the horizon length for problems tb_A and tb_B be the T chosen as per Eq. (36). Note also that both problem instances satisfy REO with parameter $\alpha = 0$, and so $\mathsf{tb}_A, \mathsf{tb}_B \in \mathsf{UTB}_{T,M,K}$. So via our assumption of the determinism of $\mathcal{A}$, via Eq. (35) and via the fact that $m_{\mathsf{tb}_A} = 1$, we have that

$$\mathcal{R}^{\mathrm{cp}}(\mathcal{A}, \mathsf{tb}_A) \leq f(m_{\mathsf{tb}_A}, K)\left(T^{1-\epsilon_1} + T^{\epsilon_3} M^{1-\epsilon_2}\right) = f(1,K)\left(T^{1-\epsilon_1} + T^{\epsilon_3} M^{1-\epsilon_2}\right) < M/2, \tag{37}$$

where the final inequality follows due to Eq. (36). And again by our assumption of the determinism of $\mathcal{A}$ and via Eq. (35), we have that

$$\mathcal{R}^{\text{cp}}(\mathcal{A}, \mathtt{tb}_B) \leq T/4. \tag{38}$$

When $\mathcal{A}$ is run on problem $\mathtt{tb}_A$, there are 2 cases. Either $\mathcal{A}$ plays x_2 for M times in a row (at some point in its execution for T timesteps while solving $\mathtt{tb}_A$) or it does not.
Consider the first case, where $\mathcal{A}$ plays x_2 for M times in a row on problem $\mathtt{tb}_A$. Then we have that $\mathcal{R}^{\text{cp}}(\mathcal{A}, \mathtt{tb}_A) \geq M/2$. This is because the optimal strategy for $\mathtt{tb}_A$ always plays x_1 on each timestep, and so any timestep where $\mathcal{A}$ plays action $x \neq x_1$ will add $h_{\mathtt{tb}_A, x} - h_{\mathtt{tb}_A, x_1} = 1 - 1/2 = 1/2$ to the CPR of $\mathcal{A}$. This is a contradiction to Eq. (37).
Consider the second case, where $\mathcal{A}$ never plays x_2 for M times in a row on problem $\mathtt{tb}_A$. Note that the deterministic algorithm $\mathcal{A}$ can be viewed as a length T sequence of functions, where the tth function maps the past $t-1$ action choices and loss observations to the action played at timestep t. Also note that the observed loss of a playing an action in $\mathtt{tb}_B$ is different from playing the same action in $\mathtt{tb}_A$, if and only if that action was x_2 and it was played M times in the prior M timesteps (including the current timestep).
Thus, since $\mathcal{A}$ never plays x_2 for M times in a row on problem $\mathtt{tb}_A$, it sees the *identical* sequence of loss outputs when it is deployed on $\mathtt{tb}_B$, and hence makes the identical sequence of actions as it would have if deployed in $\mathtt{tb}_A$, which in turn implies that it never plays x_2 for M times in a row on problem $\mathtt{tb}_B$. Since $M < T/4$ via Eq. (36), and playing any action $x \neq x_2$ will always yield loss at least $1/2$, the strategy that always plays x_2 is optimal and has cumulative loss of $M - 1$. So, since Eq. (36) implies that

$$T/2 - (M-1) > T/2 - M > T/2 - T/4 = T/4,$$

we have that

$$\mathcal{R}^{\text{cp}}(\mathcal{A}, \mathtt{tb}_B) \geq T/2 - (M-1) > T/4.$$

This is a contradiction to Eq. (38).
In either case, we have arrived at a contradiction. Hence, we have shown that for each ϵ satisfying the given conditions and each function f, the corresponding set $\overline{\mathcal{A}}_{\epsilon,f}$ is the empty set. This completes the proof. ■

C.3 Proof of Proposition 2

In this section, we provide a formal proof of Proposition 2. Let $\mathcal{X} = \{x_1, x_2\}$. Let w be defined componentwise as $w_i = \frac{1}{2^i}$ for each $1 \leq i \leq m$. Set $w_x = w$ for each $x \in \mathcal{X}$. Now sample a bit string $y^\star$ uniformly at random from $\{0,1\}^{m-1}$, whose identity is kept hidden from the player. Define $h_{x_1} = 1$ and define

$$h_{x_2}(w_{x_2}^\top y^{t,x_2,m}) = h_{x_2}(w^\top y^{t,x_2,m}) = \begin{cases} 1 \text{ if } w^\top y^{t,x_2,m} \neq w^\top (1, y^\star) \\ 0 \text{ if } w^\top y^{t,x_2,m} = w^\top (1, y^\star) \end{cases}.$$

We assume that there is no stochasticity in the loss feedback experienced by the player. This defines an (m, w, h)-weighted tallying bandit game. For ease in notation in the sequel, we also define $v \in \mathcal{X}^{m-1}$ as

$$v_i = \begin{cases} x_2 \text{ if } y^\star_i = 1 \\ x_1 \text{ if } y^\star_i = 0 \end{cases}.$$

We claim that with our choice of w, if $y \neq y' \in \{1\} \times \{0,1\}^{m-1}$ then $w^\top y \neq w^\top y'$. We defer the formal proof of this claim for now, and use this claim to complete the proof of Proposition 2. Critically, the claim implies that we incur non-unit loss at timestep t if and only if we play action x_2 at timestep t and have $y^{t,x_2,m} = (1, y^\star)$. Equivalently, we incur non-unit loss at timestep t if and only if our action sequence for the timesteps $t, t-1 \dots t-m$ is (x_2, v). Thus, the policy that cyclically plays $v_{m-1}, v_{m-2} \dots v_1, x_2$ incurs a loss of zero at least once every m timesteps. Meanwhile, identifying the optimal policy is at least as hard as playing the action sequence (x_2, v), which in turn is at least as hard as identifying $y^\star$.

A standard "needle in the haystack" argument [DKWY20] then shows that identifying $y^\star$ requires $\widetilde{\Omega}\left(2^m\right)$ timesteps. In turn, since the cyclic policy $v_{m-1}, v_{m-2} \dots v_1, x_2$ incurs a loss of zero at least once every m timesteps, this implies that the expected CPR $\mathbb{E}\left[\mathcal{R}^{\text{cp}}\right]$ of any (possibly randomized) algorithm is lower bounded by $\widetilde{\Omega}\left(\min\{2^m, T\}/m\right)$, where the expectation is over the (possible) randomization of the algorithm as well as the sampling of $y^\star$.

Let us now return to prove our claim that if $y \neq y' \in \{1\} \times \{0,1\}^{m-1}$ then $w^\top y \neq w^\top y'$. Assume for the sake of contradiction that $y \neq y'$ but $w^\top y = w^\top y'$. Let $J \subseteq \{2, 3 \dots m\}$ be the set of coordinates that y, y' differ, and let $j^\star = \min J$. Note that J is non-empty by assumption, and so $j^\star$ is well defined. Assume without loss of generality that $y_{j^\star} - y'_{j^\star} = 1$ (the case when $y_{j^\star} - y'_{j^\star} = -1$ is completely symmetric). Then observe that

$$\begin{aligned} 0 &= w^\top (y - y') \\ &= \sum_{j=1}^{m} w_j (y_j - y'_j) \\ &= \sum_{j \in J} w_j (y_j - y'_j) \\ &= w_{j^\star} + \sum_{j \neq j^\star \in J} w_j (y_j - y'_j). \end{aligned} \tag{39}$$

We can now lower bound Eq. (39) as

$$\begin{aligned} 0 &= w_{j^\star} + \sum_{j \neq j^\star \in J} w_j (y_j - y'_j) \\ &\geq w_{j^\star} - \sum_{j \neq j^\star \in J} w_j |y_j - y'_j| \\ &= w_{j^\star} - \sum_{j \neq j^\star \in J} w_j \\ &\geq w_{j^\star} - \sum_{j=j^\star+1}^{m} w_j. \end{aligned} \tag{40}$$

Now substituting in our choice of w into Eq. (40), we find that

$$0 \geq w_{j^\star} - \sum_{j=j^\star+1}^{m} w_j = \frac{1}{2^{j^\star}} - \sum_{j=j^\star+1}^{m} \frac{1}{2^j} = \frac{1}{2^{j^\star}} \left(1 - \sum_{j=1}^{m-j^\star} \frac{1}{2^j} \right) > 0,$$

which of course is a contradiction. This proves our claim that if $y \neq y' \in \{1\} \times \{0,1\}^{m-1}$ then $w^\top y \neq w^\top y'$. ■

C.4 Extended Numerical Results & Details

C.4.1 Unweighted Tallying Bandit

In this section, we first present additional experimental results for the unweighted tallying bandit problem, where the loss functions are identical to those described in Section 4.4.1. Hence, this problem satisfies 0-REO. Here, we vary the values of m, K, M, and plot the CPR of each method over time. The results are shown in Figure 13. In each case, we observe that SE outperforms the baselines. These results also show that the performance of Algorithm 3 is robust to using a conservative upper bound M on m.

We now present results for a different unweighted tallying bandit problem, where α-REO is satisfied with $\alpha > 0$. For each action $x \in \mathcal{X}$, we define $w_x = \vec{1}/(4m)$. We fix an action $x^\star \in \mathcal{X}$ and a different action $x^{\star\star} \in \mathcal{X}$, and then define the loss functions $\{h_x\}_{x \in \mathcal{X}}$ as

$$h_x(y^{t,x,m}) = \begin{cases} 1 - w_x^\top y^{t,x,m} - 0.15 \text{ if } x = x^\star, y^{t,x,m} = \vec{1} \\ 1 - w_x^\top y^{t,x,m} - \frac{m-1}{2m} - 0.2 \text{ if } x = x^{\star\star}, y^{t,x,m} = (1,0,0\ldots0) \\ 1 - w_x^\top y^{t,x,m} \text{ otherwise} \end{cases}.$$

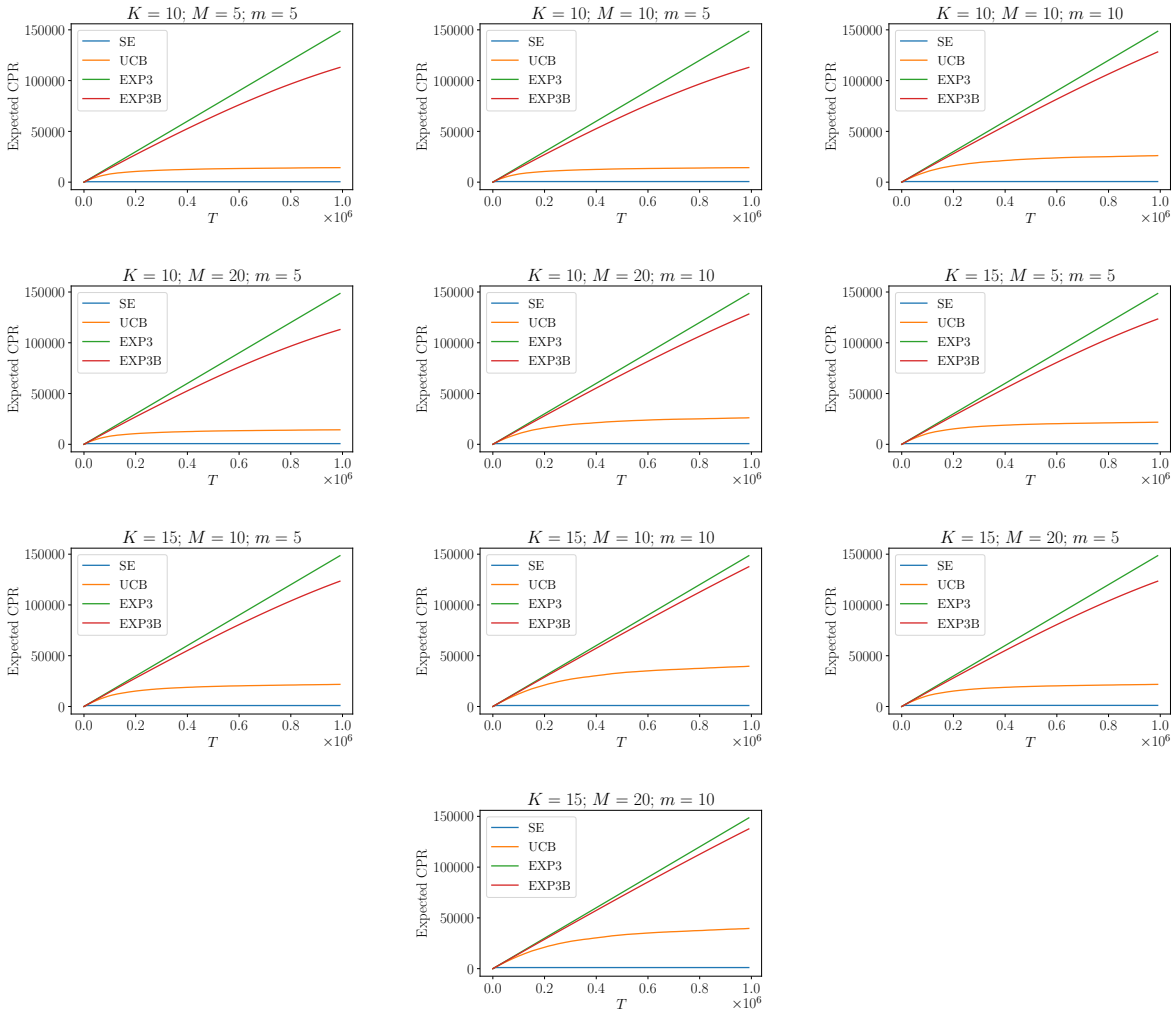


Figure 13: We plot the expected CPR of each algorithm, when deployed on the unweighted tallying bandit problem described in Section 4.4.1, with varying values of m, K, M. In all plots, each datapoint is obtained by averaging over 20 problem instances, and the shaded region depicts ± 1 standard error around the mean.

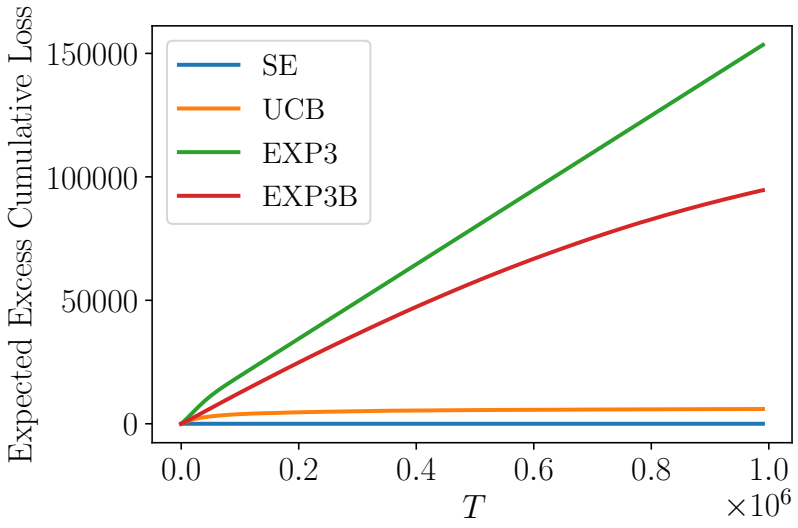


Figure 14: We plot as a function of time the expected cumulative loss of each algorithm in excess of that of SE, on the unweighted tallying bandit instance described in Appendix C.4.1, with $m = 4, K = 5, M = 4$. Note this instance satisfies α-REO with $\alpha = 0.1125$. Each datapoint is obtained by averaging over 20 problem instances, and the shaded region depicts ± 1 standard error around the mean.

A numerical computation reveals that this unweighted tallying bandit problem instance satisfies α-REO with $\alpha = \max\{0, -0.2 + \frac{2m-3}{4m}\}$. We empirically study the performance of Algorithm 3 relative to the baselines on this problem instance, with a choice of $m = 4, K = 5, M = 4$. Since the optimal policy in this problem is not obvious, the CPR is difficult to compute. So in lieu of the CPR, we plot the expected cumulative loss of each algorithm in excess of SE's loss (hence the CPR at any time is obtained by applying a constant shift to each algorithm's excess loss). The results are shown in Figure 14, where we observe that our method outperforms all others.

C.4.2 Weighted Tallying Bandit

Here, we describe the loss functions that were used to define the WTB problem instance described in Section 4.4.2. For each action $x \in \mathcal{X}$, we define w_x in the following fashion. First we define the vector $v \in \mathbb{R}^m$ coordinate wise by setting its ith coordinate as $v_i = 1/2^i$. Then we set $w_x = v/(2\|v\|_1)$ for each $x \in \mathcal{X}$. We fix an action $x^\star \in \mathcal{X}$, and then define the

loss functions $\{h_x\}_{x\in\mathcal{X}}$ as

$$h_x(y^{t,x,m}) = \begin{cases} 1 - w_x^\top y^{t,x,m} - 0.15 \text{ if } x = x^\star, y^{t,x,m} = \vec{1} \\ 1 - w_x^\top y^{t,x,m} \text{ otherwise} \end{cases}.$$

Hence, this weighted tallying bandit problem satisfies 0-REO.

C.4.3 Simulated F1 Tournament

In this section, we provide details of our F1 lap time dataset, data processing and probabilistic model fitting, as well as criteria used to define meaningful WTB problem instances. We conclude this section with extended results for the simulated F1 tournament.

Dataset & Data Processing
As discussed in 4.4.4, we make use of pre-first-pit-stop F1 lap time data from 1950-2022 [Rao22] to learn a probabilistic lap time model for various drivers and races, which we then use to define WTB instances. This dataset consists of race data from 1120 races and 858 drivers. Note that in each race, a different subset of drivers participated in the race. For the purposes of this paper, each entry in the dataset is a tuple of the form (`driver ID`, `race ID`, `Lap Times`), where `driver ID` $\in \{1,\dots,858\}$, `race ID` $\in \{1,\dots,1120\}$, and `Lap Times` is a list of tuples, where each tuple of the form (`Lap Time`, `Pit Stop`). `Lap Time` denotes the official recorded time in seconds for the driver to complete a lap during the F1 tournament, and `Pit Stop` is a binary-valued variable indicating whether the lap included a pit stop.

In order to make lap times comparable across races, all lap times in this dataset have been normalized such that they lie in the interval [0,1]. Specifically, if the raw lap time for driver i's k^{th} lap in race j is $\ell_{i,j,k}$, then the normalized lap time is given by $\tau_{i,j,k} := \frac{\ell_{i,j,k} - \min_{w,x} \ell_{w,j,x}}{\max_{y,z} \ell_{y,j,z} - \min_{w,x} \ell_{w,j,x}}$.

Additionally, for each race j, we filter lap time data to include eligible drivers that have sufficient data to justify fitting a model. Within a fixed race j, a driver is considered eligible if the dataset includes at least 8 consecutive lap time datapoints until their first pit stop. After the set of eligible drivers has been determined for race j, we shorten the sequence of all drivers' lap time data in race j to match the length of the shortest sequence in race j. For example, if there are 3 eligible drivers in race j, where driver a takes 8 laps until taking a pit stop, driver b takes 9 laps, and driver c takes 10 laps, then we only make use of the first 8 lap times of drivers b and c when learning their lap time model.

Probabilistic Model Fitting

We use our normalized lap time data to fit a probabilistic model of lap times for each eligible driver-race pair with maximum likelihood. In particular, we assume that for driver i in race j, the k^{th} normalized lap time, which is denoted $\tau_{i,j,k}$, has distribution

$$\mathcal{N}\left(\beta_{i,j}\exp(-k\alpha_{i,j}) - |\gamma_{i,j}|k,\ \sigma_{i,j}^2\right).$$

Thus, for each driver-race (i,j) pair, our formulation involves fitting 4 parameters $(\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j}, \sigma_{i,j})$. We use all normalized pre-first-pit-stop data (at least 8 datapoints) to fit these parameters using maximum likelihood with `scipy.optimize.curve_fit`, which leverages the Levenberg-Marquardt algorithm as implemented in MINPACK [V+20].

At a high level, our probabilistic model stipulates that for each driver i and race j, the lap times decrease exponentially at first and then linearly. More concretely, as the lap index k increases, the expected lap time of the driver i in race j decreases at a rate determined by $\alpha_{i,j}$ (with initial condition $\beta_{i,j}$), but it never drops below $-|\gamma_{i,j}|k$ (our k values are small, and so the lap times plateau to a positive number). The variance of the lap times for driver i in race j is $\sigma_{i,j}$ (note that this quantity is independent of k). Figure 15 shows normalized lap time data and the learned probabilistic model for 9 races (the probabilistic model for the 10th race is given in Figure 7a). By inspection, the probabilistic model is a reasonable approximation for the distribution of (sequential) lap times.

Constructing Meaningful Tallying Bandit Instances

In order to test SE, we further filter the drivers such that the driver lap time models define a meaningfully challenging bandit problem. Specifically, we define an eligible TB driver pair to be two drivers competing in the same race such that both of their terminal mean lap times are difficult to discriminate. In particular, denote the terminal expected lap time for driver i (resp. i') with μ_i (resp. $\mu_{i'}$). Then, drivers i and i' are considered to define an eligible TB driver pair if the following conditions hold:

- driver i and i' both compete in the same race j,
- $\mu_i \in [\mu_{i'} - \sigma_{i',j}^2, \mu_{i'} + \sigma_{i',j}^2]$,
- $\mu_{i'} \in [\mu_i - \sigma_{i,j}^2, \mu_i + \sigma_{i,j}^2]$.

Note these conditions essentially imply that the problem is at least as hard as distinguishing the means of two Gaussians that are "close" to each other. Ten eligible TB driver pairs meet these criteria, which we use to define 10 TB instances. Specifically, the lap time model of each driver in the pair for race j defines the loss in a TB instance, with each driver corresponding to an arm. The (stochastic) instantaneous loss at timestep t for 'playing' driver i in race j depends on the number of times driver i has completed a lap in

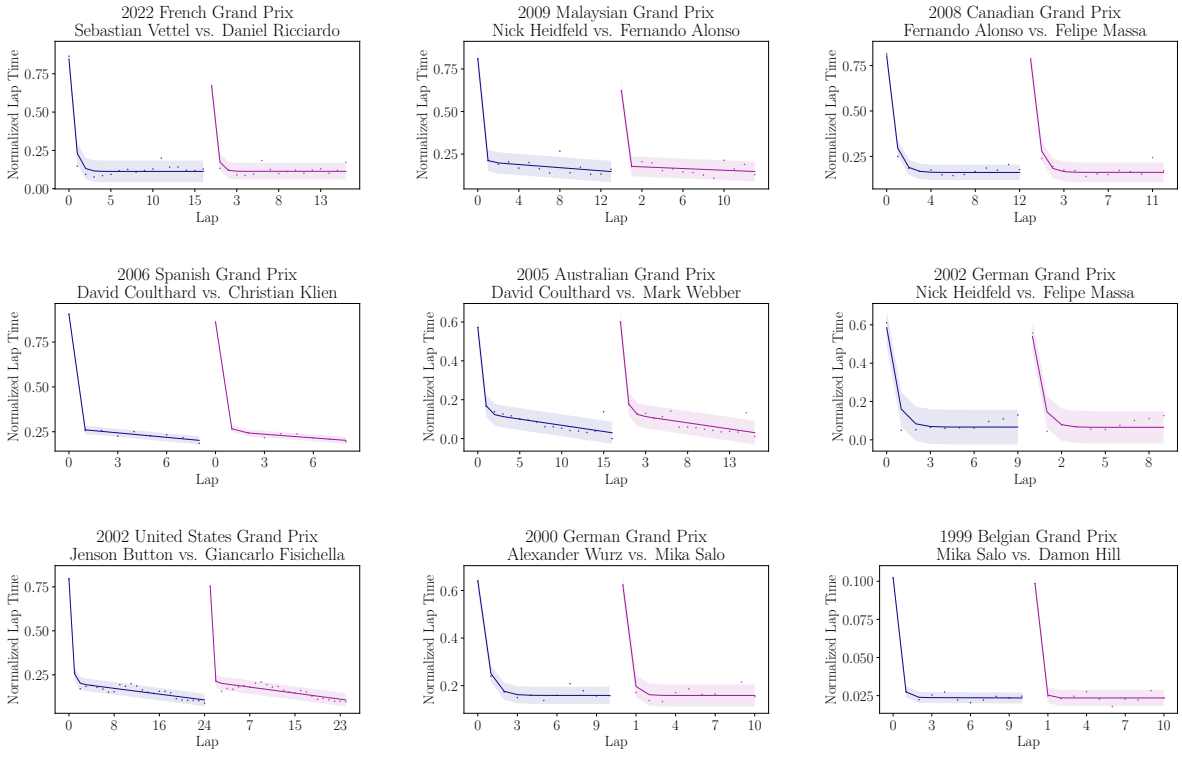


Figure 15: In each plot, we select a driver pair and a particular race, and we depict our fitted probabilistic lap time model. In each plot, our probabilistic model parameterizes a distribution over lap times for each lap index (note that different plots have different lap indices). The solid line depicts the mean of this distribution, for each lap index. The shaded region contains ± 2 standard deviations of this distribution, centered around the distribution's mean. The dotted points are the actual lap times (we note that these almost always lie within 2 standard deviations of the mean). Note that all the lap times are normalized, so that each lap time lies in the interval $[0, 1]$.

race j in the previous m timesteps, or equivalently (the tally) denoted k. This instantaneous loss is thus $\tau_{i,j,k} \sim \mathcal{N}(\beta_{i,j} \exp(-k\alpha_{i,j}) - |\gamma_{i,j}|k, \sigma_{i,j}^2)$.

Extended Results

We now show results for all 10 instances of our simulated F1 tournament, where each instance is constructed in similar fashion to the instance constructed in Section 4.4.4. We select 10 different driver pairs and races in which they competed. For each driver pair and race, we utilize F1 lap time data [Rao22] to fit a probabilistic lap time model as discussed above. In Figure 15, we illustrate our probabilistic models of lap times for each of the selected driver pairs, and show that their lap times tend to decrease as the lap index increases.

We use the probabilistic models depicted in Figure 15 to simulate 9 instances of this tournament with $K = 2$ and m equaling the number of lap indices in the appropriate plot. The results for the 10th race were given in Section 4.4.4. For each instance, we maintain a tally of the number of times each driver in that instance was chosen in the prior m timesteps. The loss associated with picking a driver is governed by the distribution parameterized by our fitted probabilistic model. In particular, if we pick driver x and we have picked them y times in the last m timesteps, then the instantaneous loss is sampled from the distribution parameterized by our fitted probabilistic model for driver x at lap index y. The two drivers for each instance are chosen such that their calibrated performance is difficult to distinguish, as discussed above. Note that in this setting, one has $o(T)$ CPR if and only if one plays the worse driver $o(T)$ many times. In Figure 16, we plot each method's CPR over time for each of the 10 instances of this tournament, showing in each case that SE outperforms the baselines.

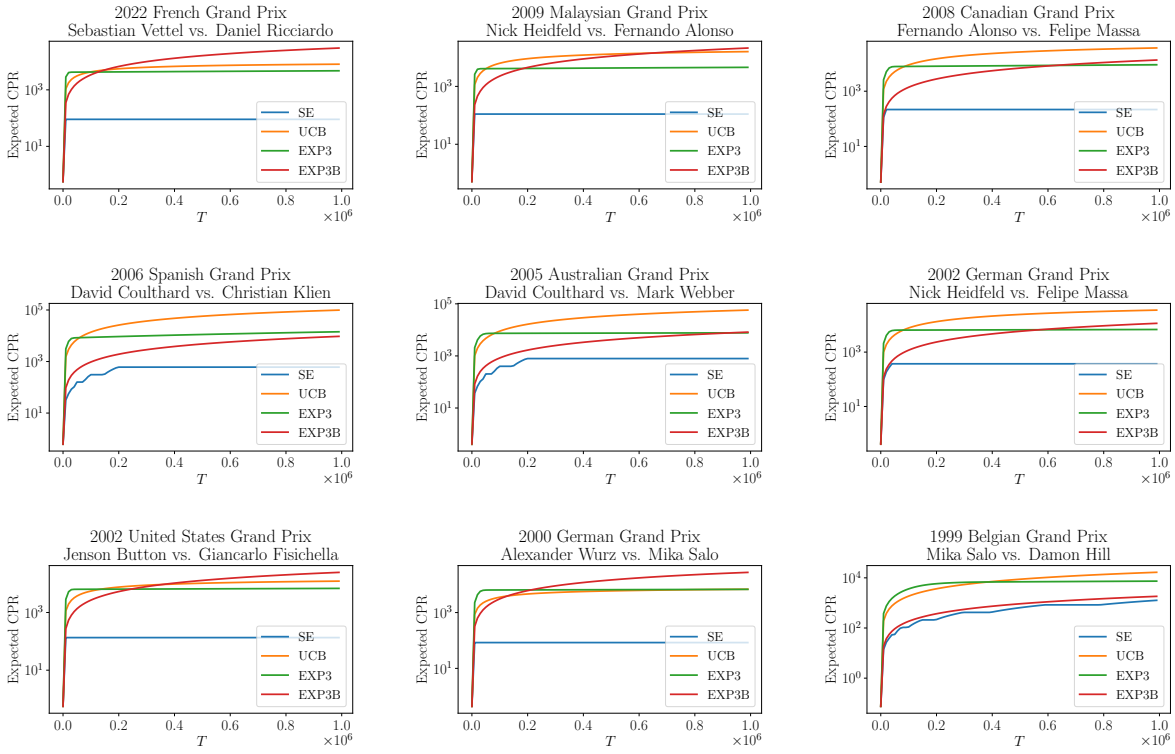


Figure 16: For each plot in Figure 15, we simulate a WTB instance and compare the performance of the various algorithms. We depict as a function of time the expected CPR of each algorithm. Data is obtained by averaging over 20 problem instances, each with $K = 2$ and $T = 10^6$. The shaded region depicts ± 1 standard error around the mean. The value m used in a plot is the length of the lap index in the corresponding plot in Figure 15. We set M to equal m.

D Appendix for Section 5

In this section, we present a helper function that automatically converts problems of the form (9) to problems of the form (11) and (12). This helper function requires the user to pass as inputs the CVXPY variables, constraints and loss function that define (9), and is presented below. The code is available at

https://github.com/cvxgrp/rerm_code.

```python
from typing import Callable, List
import cvxpy as cp
```

```python
import numpy as np
from cvxpy.transforms.suppfunc import SuppFunc

def form_rerm(
    f: Callable,
    y: np.ndarray,
    theta: cp.Variable,
    theta_constraints: List[cp.Constraint],
    xs: List[cp.Variable],
    x_constraints: List[List[cp.Constraint]],
    mode: str
):
    """
    Args:
        f: a convex function
        y: a vector of length n
        theta: a CVXPY variable
        theta_constraints: a list of constraints on theta
        xs: a list of n scalar CVXPY variables
        x_constraints: a list of n lists of constraints on xs
        mode: "non_increasing" or "non_decreasing_sym_abs"

    Returns:
        A CVXPY problem instance of the robust ERM
        problem.
    """
    n = len(xs)
    assert theta.ndim <= 1
    assert n == len(x_constraints) == len(y)

    z = cp.Variable(n)
    obj = 0.0
    constraints = theta_constraints

    for i in range(n):
        obj += f(z[i])
        G = SuppFunc(xs[i], x_constraints[i])
        if mode == "non_increasing":
            constraints += [
                G(-theta) + y[i] <= -z[i],
            ]
```

```
        elif mode == "non_decreasing_sym_abs":
            constraints += [
                G(theta) - y[i] <= z[i],
                G(-theta) + y[i] <= z[i],
            ]
        else:
            raise NotImplementedError

    prob = cp.Problem(cp.Minimize(obj), constraints)
    return prob
```

We emphasize that the loss function f provided by the user is not verified to have the claimed curvature properties specified in the mode argument. It is impossible to verify this in general, so the user must be careful to provide a loss function that has the correct curvature properties. We also mention that the user is not limited to pass in a loss function f that is a CVXPY atom. Instead, the user has the flexibility to create a loss function that is a composition of several CVXPY atoms, as follows.

```
def f(x: cp.Variable):
    """
    An example non-decreasing convex function of the magnitude of x.
    This is how a user can specify a arbitrary convex function.
    """
    return cp.square(cp.power(cp.abs(x), 1.5))
```

References

[AB10] Jean-Yves Audibert and Sébastien Bubeck. Best arm identification in multi-armed bandits. In *Conference on Learning Theory*, 2010.

[ABGK22] Pranjal Awasthi, Kush Bhatia, Sreenivas Gollapudi, and Kostas Kollias. Congested bandits: Optimal routing via short-term resets, 2022.

[ACBF02] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002.

[ACBFS02] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[Ada61] Jack A. Adams. The second facet of forgetting: A review of warmup decrement. *Psychological Bulletin*, 58:257–273, 1961.

[ADMM18] Raman Arora, Michael Dinitz, Teodor V. Marinov, and Mehryar Mohri. Policy regret in repeated games. In *Advances in Neural Information Processing Systems*, 2018.

[ADT12] Raman Arora, Ofer Dekel, and Ambuj Tewari. Online bandit learning against an adaptive adversary: From regret to policy regret. In *International Conference on Machine Learning*, 2012.

[AFG22] Alireza Aghasi, MohammadJavad Feizollahi, and Saeed Ghadimi. Rigid: Robust linear regression with missing data, 2022.

[AHK+14] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *Proceedings of the International Conference on Machine Learning*, 2014.

[AHR08] Jacob Abernethy, Elad Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *Conference on Learning Theory*, 2008.

[AKKS20] Alekh Agarwal, Sham Kakade, Akshay Krishnamurthy, and Wen Sun. Flambe: Structural complexity and representation learning of low rank mdps. In *Advances in Neural Information Processing Systems*, 2020.

[AKLM20] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in markov

decision processes. In *Proceedings of the Conference on Learning Theory*, 2020.

[AL20] Zeyuan Allen-Zhu and Yuanzhi Li. Feature purification: How adversarial training performs robust deep learning. *arXiv preprint arXiv:2005.10190*, 2020.

[ALS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *ICML*, 2019. Full version available at http://arxiv.org/abs/1811.03962.

[AMK13] Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J. Kappen. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine Learning*, 91(3):325–349, 2013.

[Ans95] Mark H. Anshel. Examining warm-up decrement as a function of interpolated open and closed motor tasks: Implications for practice strategies. *Journal of Sports Sciences*, 13:247–256, 1995.

[AOM17] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, 2017.

[AW93] Mark H. Anshel and Craig A. Wrisberg. Reducing warm-up decrement in the performance of the tennis serve. *Journal of Sport & Exercise Psychology*, 15(3):290–303, 1993.

[AZL19] Zeyuan Allen-Zhu and Yuanzhi Li. What can resnet learn efficiently, going beyond kernels? In *Advances in Neural Information Processing Systems*, 2019.

[AZL20] Zeyuan Allen-Zhu and Yuanzhi Li. Backward feature correction: How deep learning performs deep learning. *arXiv preprint arXiv:2001.04413*, 2020.

[AZLS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *Proceedings of the International Conference on Machine Learning*, 2019.

[B+19] Christopher Berner et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arxiv:1912.06680*, 2019.

[BBC11] Dimitris Bertsimas, David Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

[BF16] Djallel Bouneffouf and Raphael Féraud. Multi-armed bandit problem with known trend. *Neurocomputing*, 205(C):16–21, 2016.

[BGZ14] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In *Advances in Neural Information Processing Systems*, 2014.

[BNVB13] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[BPK+20] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the Atari human benchmark. In *Proceedings of the International Conference on Machine Learning*, 2020.

[BSSS19] Soumya Basu, Rajat Sen, Sujay Sanghavi, and Sanjay Shakkottai. Blocking bandits. In *Advances in Neural Information Processing Systems*, 2019.

[BTEGN09] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton university press, 2009.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[CBDS13] Nicolò Cesa-Bianchi, Ofer Dekel, and Ohad Shamir. Online learning with switching costs and other adaptive adversaries. In *Advances in Neural Information Processing Systems*, 2013.

[CCB20] Leonardo Cella and Nicoló Cesa-Bianchi. Stochastic bandits with delay-dependent payoffs. In *International Conference on Artificial Intelligence and Statistics*, pages 1168–1177, 2020.

[CDK+20] Corinna Cortes, Giulia DeSalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Discrepancy-based algorithms for non-stationary rested bandits. *arXiv preprint arXiv:1710.10657*, 2020.

[CKH+19] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2019.

[CLA+03] Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, and John Riedl. Is seeing believing? how recommender system interfaces affect users' opinions. In *Conference on Human Factors in Computing Systems*, 2003.

[CM07] Antonio G. Chessa and Jaap M. J. Murre. A neurocognitive model of advertisement content and brand name recall. *Marketing Science*, 26(1):130–141, 2007.

[DB16] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[DDKP14] Ofer Dekel, Jian Ding, Tomer Koren, and Yuval Peres. Bandits with switching costs: $T^{2/3}$ regret. In *Symposium on Theory of Computing*, 2014.

[DKJ+19] Simon Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudik, and John Langford. Provably efficient RL with rich observations via latent state decoding. In *Proceedings of the International Conference on Machine Learning*, 2019.

[DKWY20] Simon S. Du, Sham M. Kakade, Ruosong Wang, and Lin F. Yang. Is a good representation sufficient for sample efficient reinforcement learning? In *International Conference on Learning Representations*, 2020.

[DLL+19] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *Proceedings of the International Conference on Machine Learning*, 2019.

[DLMW20] Simon S. Du, Jason D. Lee, Gaurav Mahajan, and Ruosong Wang. Agnostic q-learning with function approximation in deterministic systems: Near-optimal bounds on approximation error and sample complexity. In *Advances in Neural Information Processing Systems*, 2020.

[DLWZ19] Simon S Du, Yuping Luo, Ruosong Wang, and Hanrui Zhang. Provably efficient q-learning with function approximation via distribution shift error checking oracle. In *Advances in Neural Information Processing Systems*, 2019.

[DLY+20] Kefan Dong, Yuping Luo, Tianhe Yu, Chelsea Finn, and Tengyu Ma. On the expressivity of neural networks for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2020.

[DMM+19] Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. On the sample complexity of the linear quadratic regulator. *Foundations of Computational Mathematics*, pages 1–47, 2019.

[DPWZ20] Kefan Dong, Jian Peng, Yining Wang, and Yuan Zhou. Root-n-regret for learning in markov decision processes with function approximation and low bellman rank. In *Proceedings of the Conference on Learning Theory*, 2020.

[DSL+18] Bo Dai, Albert Shaw, Lihong Li, Lin Xiao, Niao He, Zhen Liu, Jianshu Chen, and Le Song. SBEED: Convergent reinforcement learning with nonlinear

function approximation. In *Proceedings of the International Conference on Machine Learning*, 2018.

[EDMM02] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. PAC bounds for multi-armed bandit and markov decision processes. In *Conference on Computational Learning Theory*, 2002.

[EGL97] Laurent El Ghaoui and Hervé Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on matrix analysis and applications*, 18(4):1035–1064, 1997.

[FGKM18] Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *Proceedings of the International Conference on Machine Learning*, 2018.

[FKM05] Abraham D. Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Symposium on Discrete Algorithms*, 2005.

[FRH+19] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *Advances in Neural Information Processing Systems*, 2019.

[GM11] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *International Conference on Algorithmic Learning Theory*, 2011.

[GN21] Kristóf Gyódi and Łukasz Nawaro. Determinants of airbnb prices in european cities: A spatial econometrics approach. *Tourism Management*, 86:104319, 2021.

[HK12] Elad Hazan and Satyen Kale. Online submodular minimization. *Journal of Machine Learning Research*, 13(93):2903–2922, 2012.

[HKR16] Hoda Heidari, Michael Kearns, and Aaron Roth. Tight policy regret bounds for improving and decaying bandits. In *International Joint Conference on Artificial Intelligence*, 2016.

[HMMA+17] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. In *Advances in Neural Information Processing Systems*, 2017.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[IGS22] Conor Igoe, Ramina Ghods, and Jeff Schneider. Multi-agent active search: A reinforcement learning approach. *IEEE Robotics and Automation Letters*, 7(2):754–761, 2022.

[JAZBJ18] Chi Jin, Zeyuan Allen-Zhu, Sébastien Bubeck, and Michael I Jordan. Is Q-learning provably efficient? In *Advances in Neural Information Processing Systems*, 2018.

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[JKA+17] Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E. Schapire. Contextual decision processes with low Bellman rank are PAC-learnable. In *Proceedings of the International Conference on Machine Learning*, 2017.

[JLM21] Chi Jin, Qinghua Liu, and Sobhan Miryoosefi. Bellman eluder dimension: New rich classes of rl problems, and sample-efficient algorithms, 2021.

[JYWJ20] Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Proceedings of the Conference on Learning Theory*, 2020.

[KI18] R. Kleinberg and N. Immorlica. Recharging bandits. In *IEEE Symposium on Foundations of Computer Science*, 2018.

[KKL03] Sham Kakade, Michael Kearns, and John Langford. Exploration in metric state spaces. In *International Conference on Machine Learning*, 2003.

[Kla80] Roberta L. Klatzky. *Human memory: structures and processes*. Freeman, 1980.

[KMN99] Michael Kearns, Yishay Mansour, and Andrew Y. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *International Joint Conference on Artificial Intelligence*, 1999.

[LC18] Andrea Locatelli and Alexandra Carpentier. Adaptivity to smoothness in x-armed bandits. In *Conference On Learning Theory*, 2018.

[LCCG21] Pierre Laforgue, Giulia Clerici, Nicolò Cesa-Bianchi, and Ran Gilad-Bachrach. Break your bandit routine with LSD rewards: a last switch dependent analysis of satiation and seasonality. *arXiv preprint arXiv:2110.11819*, 2021.

[LCM17] Nir Levine, Koby Crammer, and Shie Mannor. Rotting bandits. In *Advances in Neural Information Processing Systems*, 2017.

[LFDA16] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17:1334–1373, 2016.

[LHK21] David Lindner, Hoda Heidari, and Andreas Krause. Addressing the long-term impact of ML decisions via policy regret. In *International Joint Conference on Artificial Intelligence*, 2021.

[LL18a] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, 2018.

[LL18b] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, 2018.

[LML+24] Eric Luxenberg, Dhruv Malik, Yuanzhi Li, Aarti Singh, and Stephen Boyd. Specifying and solving robust empirical risk minimization problems using CVXPY. *Journal of Optimization Theory and Applications*, 202:1158–1168, 2024.

[LMZ20] Yuanzhi Li, Tengyu Ma, and Hongyang R. Zhang. Learning over-parametrized two-layer neural networks beyond ntk. In *Proceedings of the Conference on Learning Theory*, 2020.

[LR85] T.L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

[LSW20] Tor Lattimore, Csaba Szepesvari, and Gellert Weisz. Learning with good feature representations in bandits and in RL with a generative model. In *Proceedings of the International Conference on Machine Learning*, 2020.

[M+15] Volodymyr Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[MHKL20] Dipendra Misra, Mikael Henaff, Akshay Krishnamurthy, and John Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2020.

[MILS23] Dhruv Malik, Conor Igoe, Yuanzhi Li, and Aarti Singh. Weighted tallying bandits: Overcoming intractability via repeated exposure optimality. In *International Conference on Machine Learning*, 2023.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*. 2013.

[MLR21] Dhruv Malik, Yuanzhi Li, and Pradeep Ravikumar. When is generalizable reinforcement learning tractable? *arXiv preprint arxiv:2101.00300*, 2021.

[MLS22] Dhruv Malik, Yuanzhi Li, and Aarti Singh. Complete policy regret bounds for tallying bandits. In *Conference on Learning Theory*, 2022.

[MOSW02] N. Merhav, E. Ordentlich, G. Seroussi, and M.J. Weinberger. On sequential strategies for loss functions with memory. *IEEE Transactions on Information Theory*, 48(7):1947–1958, 2002.

[MPB+20] Dhruv Malik, Ashwin Pananjady, Kush Bhatia, Koulik Khamaru, Peter L Bartlett, and Martin J Wainwright. Derivative-free methods for policy optimization: Guarantees for linear quadratic systems. *Journal of Machine Learning Research*, 21:1–51, 2020.

[MPSL21] Dhruv Malik, Aldo Pacchiano, Vishwak Srinivasan, and Yuanzhi Li. Sample efficient reinforcement learning in continuous state spaces: A perspective beyond linearity. In *International Conference on Machine Learning*, 2021.

[MTPR22] Alberto Maria Metelli, Francesco Trovò, Matteo Pirola, and Marcello Restelli. Stochastic rising bandits. In *International Conference on Machine Learning*, 2022.

[MY18] Mehryar Mohri and Scott Yang. Competing with automata-based expert sequences. In *International Conference on Artificial Intelligence and Statistics*, 2018.

[PBG19] Ciara Pike-Burke and Steffen Grunewalder. Recovering bandits. In *Advances in Neural Information Processing Systems*, 2019.

[PMR+20] Ashwin Phatak, Utkarsh Mujumdar, Robert Rein, Fabian Wunderlich, Marc Garnica, and Daniel Memmert. Better with each throw — a study on calibration and warm-up decrement of real-time consecutive basketball free throws in elite nba athletes. *German Journal of Exercise and Sport Research*, 50:273–279, 2020.

[PW21] A. Pananjady and M. J. Wainwright. Instance-dependent ℓ_∞-bounds for policy evaluation in tabular reinforcement learning. *IEEE Transactions on Information Theory*, 67:566–585, 2021.

[Rao22] Rohan Rao. Formula 1 world championship (1950 - 2022). https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020, 2022.

[RVC16] Timothy J. Ricker, Evie Vergauwe, and Nelson Cowan. Decay theory of immediate memory: From brown (1958) to today (2014). *Quarterly Journal of Experimental Psychology*, 69(10):1969–1995, 2016.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

[SBS06] Pannagadatta K. Shivaswamy, Chiranjib Bhattacharyya, and Alexander J. Smola. Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*, 7(47):1283–1314, 2006.

[SGR16] Ayan Sinha, David F Gleich, and Karthik Ramani. Deconvolving feedback loops in recommender systems. In *Advances in Neural Information Processing Systems*, 2016.

[SJ19] Max Simchowitz and Kevin G Jamieson. Non-asymptotic gap-dependent regret bounds for tabular mdps. In *Advances in Neural Information Processing Systems*. 2019.

[SLA+15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning*, 2015.

[SLB23] Philipp Schiele, Eric Luxenberg, and Stephen Boyd. Disciplined saddle programming, 2023.

[SLC+19] Julien Seznec, Andrea Locatelli, Alexandra Carpentier, Alessandro Lazaric, and Michal Valko. Rotting bandits are no harder than stochastic ones. In *International Conference on Artificial Intelligence and Statistics*, 2019.

[Sli19] Aleksandrs Slivkins. Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning*, 12(1-2):1–286, 2019.

[SMLV20] Julien Seznec, Pierre Menard, Alessandro Lazaric, and Michal Valko. A single algorithm for both restless and rested rotting bandits. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[SMSM00] Richard S Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000.

[TL12] Cem Tekin and Mingyan Liu. Online learning of rested and restless bandits. *IEEE Transactions on Information Theory*, 58(8):5588–5611, 2012.

[V+20] Pauli Virtanen et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[Ver18] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[VRD19] Benjamin Van Roy and Shi Dong. Comments on the du-kakade-wang-yang lower bounds. *arXiv preprint arxiv:1911.07910*, 2019.

[WAJ+21] Gellert Weisz, Philip Amortila, Barnabas Janzer, Yasin Abbasi-Yadkori, Nan Jiang, and Csaba Szepesvari. On query-efficient planning in mdps under linear realizability of the optimal state-value function. In *Proceedings of the Conference on Learning Theory*, 2021.

[WAS21] Gellert Weisz, Philip Amortila, and Csaba Szepesvari. Exponential lower bounds for planning in mdps with linearly-realizable optimal action-value functions. In *Proceedings of the International Conference on Algorithmic Learning Theory*, 2021.

[WDYK20] Ruosong Wang, Simon S Du, Lin Yang, and Sham Kakade. Is long horizon RL more difficult than short horizon RL? In *Advances in Neural Information Processing Systems*, 2020.

[WHFM20] Fabian Wunderlich, Herbert Heuer, Philip Furley, and Daniel Memmert. A serial-position curve in high-performance darts: The effect of visuomotor calibration on throwing accuracy. *Psychological Research*, 84:2057–2064, 2020.

[Whi81] Peter Whittle. Arm-Acquiring Bandits. *The Annals of Probability*, 9(2):284–292, 1981.

[WSY20] Ruosong Wang, Russ R Salakhutdinov, and Lin Yang. Reinforcement learning with general value function approximation: Provably efficient approach via bounded eluder dimension. In *Advances in Neural Information Processing Systems*, 2020.

[WVR13] Zheng Wen and Benjamin Van Roy. Efficient exploration and value function generalization in deterministic systems. In *Advances in Neural Information Processing Systems*, 2013.

[WWDK21] Yining Wang, Ruosong Wang, Simon S. Du, and Akshay Krishnamurthy. Optimism in reinforcement learning with generalized linear function

approximation. In *International Conference on Learning Representations*, 2021.

[XCM08] Huan Xu, Constantine Caramanis, and Shie Mannor. Robust regression and lasso. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

[XCM09] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10(51):1485–1510, 2009.

[YW19] Lin Yang and Mengdi Wang. Sample-optimal parametric q-learning using linearly additive features. In *Proceedings of the International Conference on Machine Learning*, 2019.

[YW20] Lin Yang and Mengdi Wang. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound. In *Proceedings of the International Conference on Machine Learning*, 2020.