When Is Generalizable Reinforcement Learning Tractable?

Dhruv Malik
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213

Yuanzhi Li
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213

Pradeep Ravikumar
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Agents trained by reinforcement learning (RL) often fail to generalize beyond the environment they were trained in, even when presented with new scenarios that seem similar to the training environment. We study the query complexity required to train RL agents that generalize to multiple environments. Intuitively, tractable generalization is only possible when the environments are similar or close in some sense. To capture this, we introduce *Weak Proximity*, a natural structural condition that requires the environments to have highly similar transition and reward functions and share a policy providing optimal value. Despite such shared structure, we prove that tractable generalization is impossible in the worst case. This holds even when each individual environment can be efficiently solved to obtain an optimal linear policy, and when the agent possesses a generative model. Our lower bound applies to the more complex task of representation learning for efficient generalization to multiple environments. On the positive side, we introduce *Strong Proximity*, a strengthened condition which we prove is sufficient for efficient generalization.

1 Introduction

Reinforcement learning (RL) is the dominant paradigm for sequential decision making in machine learning, and has achieved success in a variety of domains such as competitive gaming [33, 40] and robotic control [21, 22]. Despite this success, many issues prevent RL from being regularly used in the real world. For example, one typically trains and tests RL agents in the same environment. In such cases, an agent can memorize behavior that achieves high reward, without acquiring the true behavior that the system designer desires. This has raised concerns about RL agents overfitting to a single environment, instead of learning meaningful skills [17].

Indeed, a long line of work has noted the brittleness of RL agents: slight changes in the environment, such as those incurred by modeling or simulator design errors, or slight perturbations of the agent's trajectory, can lead to catastrophic declines in performance [36, 49, 23]. Furthermore, although RL agents can solve difficult tasks, they struggle to transfer the skills they learned in one task to perform well in a different but similar task [37, 48]. Yet, in the real world, it is reasonable to expect that RL agents will see scenarios that are at least mildly different from the specific scenarios they trained for.

Hence, a desirable property of RL agents is that of *generalization*, broadly defined as the ability to discern the correct notion of behavior and perform well in semantically similar environments. We focus on two popular generalization settings. The *Average Performance* setting assumes there is an

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

underlying distribution over the environments that an agent might encounter. The agent's goal is to perform well on average across this distribution [35, 34, 11]. The *Meta Reinforcement Learning* setting is closely related [20, 10, 37]. Here an agent first learns from a suite of training environments sampled from a distribution. Then at test time the agent must leverage this experience to adapt to a new environment sampled from the same distribution, via only a few queries in the new environment.

Of course, in full generality, both notions of generalization are impossible to achieve efficiently. This is especially true in the RL function approximation setting, where the *cardinality* of the state space is potentially infinite, and so we desire query complexity that scales (polynomially) with the *dimensionality* of the state space [13, 38, 14, 31, 46]. Hence, key to both lines of inquiry is the premise that the environments are structurally similar. For example, a robot may face the differing tasks of screwing a bottle cap and turning a doorknob, but both tasks involve turning the wrist [37]. The hope is that if the environments are sufficiently similar, then RL can exploit this structure to efficiently discover policies that generalize.

Yet, it remains unclear what kind of structure is necessary, and what it means for different environments to be close or similar. Motivated by this, we ask the following question:

What are the structural conditions on the environments that permit efficient generalization?

This question underlies the analysis of our paper. We focus on environments that share state-action spaces, since even this basic case is not well understood in the literature. Indeed, even in this simplified setting, efficient generalization can be highly non-trivial. We make the following contributions.

Our Contributions. We introduce *Weak Proximity*, a natural structural condition that is motivated by classical RL results, and requires the environments to have highly similar transition and reward functions and share optimal trajectories. We prove a statistical lower bound demonstrating that tractable generalization is impossible, despite this shared structure. This lower bound holds even when each individual environment can be efficiently solved to obtain an optimal linear policy, and when the agent possesses a generative model. Consequentially, we show that a classical metric for measuring the relative closeness of MDPs is not the right metric for modern RL generalization settings. Our lower bound implies that learning a state representation for the purpose of efficiently generalizing to multiple environments, is worst case sample inefficient — even when such a representation exists, the environments are ostensibly similar, and any single environment can be efficiently solved.

To provide a sufficient condition for efficient generalization, we introduce *Strong Proximity*. This structural condition strengthens Weak Proximity by additionally constraining the environments to share an optimal policy. We provide an algorithm which exploits Strong Proximity to provably and efficiently generalize, when the environments share deterministic transitions.

2 Related Work

Simulation Lemma. Many prior works define notions of statistical distance between Markov decision processes (MDPs), and measure the relative value of policies when deployed in different MDPs that are close under such metrics. The Simulation Lemma, which uses total variation distance between transitions and the absolute difference of rewards as this metric, is a well known formalization of this and has been very useful in classical prior work [26, 27, 6, 25, 1]. These works do not directly tackle generalization, but their analyses construct an approximate MDP that models the true MDP under the aforementioned metric. Solving this approximate MDP then corresponds to solving the true MDP. It is natural to ask whether this metric is useful for measuring the similarity of MDPs in modern RL generalization settings. We show this metric is not appropriate for the settings we study.

Transfer & Multitask Learning. There are varying formalisms of both settings, so we do not directly study them. However, they are broadly relevant, and we expect our theory to be useful for future studies of these settings. The works [32, 7, 24, 43] all study metrics for measuring variation between MDPs that are different from the metrics we study. A metric similar to the one used in the Simulation Lemma has also been studied [18], and we show that this is inappropriate for our settings.

Average Performance & Meta RL Settings. We directly study these two settings, which have seen much empirical work [35, 11, 12, 37, 48]. On the theoretical side, [5, 42] study an Average Performance setting where the agent receives a noisy observation in lieu of the actual state. We focus on the simpler setting where the agent knows its state. Recent works [16, 44] analyze the

MAML algorithm [20] in the context of Meta RL. In the worst case, their complexity bounds scale exponentially with the horizon, and they do not discuss structure which permits tractable Meta RL.

Representation Learning. A large body of work has focused on extracting a representation useful for a single MDP [19, 8, 30, 50]. Some works extend this to multiple MDPs [9, 3, 41], but they are about learning shared representations for MDPs that appear similar (but not from a sample efficiency perspective), while we formalize what it means for MDPs to be similar (in a sample efficient sense). Indeed, these works study the general case when the environments have distinct state spaces, but our lower bounds show generalization is non-trivial even when each MDP shares the same state space.

3 Problem Formulation

Notation & Preliminaries. Before describing our settings of interest, we establish notation and briefly review preliminaries. We always use M to denote a Markov decision process (MDP). Recall that an undiscounted finite horizon MDP is specified 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 distributions over states, a reward function R which maps state-action pairs to nonnegative real numbers, and a finite planning horizon H. We assume that the state-action pairs are featurized, so that $\mathcal{S} \times \mathcal{A} \subset \mathbb{R}^d$, and that $\|(s,a)\|_2 = 1$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. Any MDP we consider is undiscounted and has a finite action space, but could have an uncountable state space. If we need to refer to the transition or reward function of a specific MDP M, then we shall denote this via $\mathcal{T}_M$ or R_M. We will denote a distribution over MDPs as $\mathcal{D}$. We also assume that $\mathcal{S}$ can be partitioned into H different levels. This means that for each $s \in \mathcal{S}$ there exists a unique $h \in \{0, 1 \dots H-1\}$ 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. This assumption is without loss of generality, since we can always make the final coordinate of each state-action pair encode the number of timesteps that elapsed to reach the state. A "deterministic MDP" is one with deterministic transitions. For any MDP, we assume a single initial state s_0, which strengthens our lower bounds.

A policy maps each state to a corresponding distribution over actions, and shall typically be denoted by π. The total expected reward accumulated by policy π when initialized at state s in MDP M is given by $\mathbb{E}\left[\sum_{h=\text{level}(s)}^{H-1} R_M(s_h, a_h) \mid \pi\right]$ and will be denoted by $V_M^s(\pi)$. Here the expectation is over the trajectory $\{(s_h, a_h)\}_{h=\text{level}(s)}^{H-1}$ given that the first state in the trajectory is s. So $V_M^s(\pi)$ is the value of the policy π in MDP M with respect to (w.r.t) initial state s. Analogously, if a policy is parameterized by $\overline{\theta} = \{\theta_h\}_{h=0}^{H-1}$, then we denote it as $\pi(\overline{\theta})$, and the notation $V_M^s(\pi)$ is then replaced by $V_M^s(\overline{\theta})$. We assume that the cumulative reward collected by any trajectory from any initial state s in any MDP M is always bounded by 1. Hence the value of any policy in any MDP lies in the interval $[0, 1]$. $\mathbb{TV}(P, Q)$ denotes the total variation (TV) distance between probability distributions P and Q.

3.1 Problem Settings

Average Performance Setting. There is a fixed distribution $\mathcal{D}$ over a family of MDPs. One can sample MDPs from $\mathcal{D}$. The algorithm can query states in the sampled MDPs, to learn some common structure. The goal is to solve

$$\max_{\pi} \mathbb{E}_{M \sim \mathcal{D}} \left[V_M^{s_0}(\pi) \right]. \tag{1}$$

Meta Reinforcement Learning Setting. There is a fixed distribution $\mathcal{D}$ over a family of MDPs. At training time, one can sample MDPs from $\mathcal{D}$. The algorithm can query states in the sampled MDPs, to learn some common structure between all the MDPs. Then at test time, an MDP M_{test} is sampled from the same distribution $\mathcal{D}$. The goal of the algorithm is to learn a subroutine, which with non-trivial probability over the selection of M_{test}, can solve

$$\max_{\pi} V_{\mathrm{M}_{\text{test}}}^{s_0}(\pi), \tag{2}$$

significantly more efficiently than trying to solve M_{test} without having seen any training MDPs.

In both settings, "sampling an MDP" means drawing an MDP i.i.d from $\mathcal{D}$, so that the agent can then interact with it by performing trajectories in it. Note that in Eqs. (1) & (2), in full generality the initial state s_0 is random and depends on $M, \mathrm{M}_{\text{test}}$. We focus on the case when the MDPs supporting $\mathcal{D}$ share a state-action space, and hence share the same single initial s_0 since we assume a single

initial state for any MDP. While such assumptions are already strong, they only strengthen our lower bounds. Furthermore, it is necessary to understand this simpler setting, before looking at more complex scenarios. To the best of our knowledge, such a study has not appeared in prior work.

To solve the problems described by Eqs. (1) & (2), we need to define an appropriate query model for the algorithm. We consider two query models, the first of which is strictly stronger than the second.

Strong Query Model (SQM). Sampling an MDP from $\mathcal{D}$ incurs no cost. The agent has a generative model of any sampled MDP M. To interact with M, the agent inputs a state-action pair (s, a) of M into the model, and receives $R_M(s, a)$ and a state sampled from $\mathcal{T}_M(s, a)$. This incurs a query cost of one. The goal is to solve Eqs. (1) & (2) with total query cost that is at most polynomial in $|\mathcal{A}|, H, d$.

Weak Query Model (WQM). Sampling an MDP from $\mathcal{D}$ incurs a query cost of $q_{\mathcal{D}} \geq 1$. Within a sampled MDP M, the agent operates in the standard episodic RL setup. Concretely, during each episode the agent interacts with the MDP by starting from s_0, taking an action and observing the next state and reward, and repeating. Each action taken during an episode incurs a query cost of one. The goal is to solve Eqs. (1) & (2) with total query cost that is at most polynomial in $q_{\mathcal{D}}, |\mathcal{A}|, H, d$.

Note that under both SQM and WQM, we desire query cost that is polynomial in the dimension d of the state-action space, as opposed to the cardinality of the state space. This is standard for our function approximation setting [13, 38, 14, 31, 46], since the cardinality of the state space could be infinite. Also, we separate SQM and WQM because it is well known that different query models can lead to various subtleties in analysis and sample complexity guarantees [13, 38, 14, 31, 46]. The generative model that defines SQM assumes that we can simulate any state of our choice without performing a trajectory, which is unrealistic in practice, and is one of the strongest oracle models considered in prior literature [28, 4, 39, 14, 31, 2]. We shall present our lower bounds under SQM, which makes these results stronger, but shall present our upper bound under the natural and standard WQM.

Without any conditions on $\mathcal{D}$, the Average Performance & Meta RL settings can be intractable, even under SQM. This will occur if the MDPs supporting $\mathcal{D}$ do not share structure. This will also occur if any individual MDP cannot be solved efficiently. Nevertheless, in practice one often deals with MDPs which share meaningful structure [11, 37]. For instance, the transition distributions of the MDPs may be close in a suitable metric. Similarly, the reward functions of the MDPs might be close in an appropriate norm, or each MDP may share a set of optimal trajectories. And in practice, individual MDPs can usually be optimized efficiently [35, 48]. In such cases, it is reasonable to expect tractable generalization. We are interested in formalizing conditions that permit efficient generalization. We will particularly focus on conditions which capture shared structure of the MDPs and the tractability of individual MDPs. We now formally state the problem we consider throughout our paper.

Which conditions on $\mathcal{D}$ allow us to solve the Average Performance & Meta RL settings efficiently?

As mentioned above, there are two types of requirements. The first requirement should ensure that the MDPs are meaningfully similar. We formalize such conditions in Section 3.2. The second requirement should ensure that any individual MDP is efficiently solvable, else there is no hope to efficiently find policies that generalize for many MDPs. We formalize such properties in Section 3.3.

3.2 Strong & Weak Proximity

We now identify conditions that capture when the MDPs supporting $\mathcal{D}$ share meaningful structure. Since MDPs are defined in terms of rewards and transitions, it is very natural to impose conditions directly on the rewards and transitions. To this end, we state the following condition.

Condition 1 (Similar Rewards & Transitions) *The distribution $\mathcal{D}$ satisfies this condition with parameters $\xi_{\mathrm{r}}, \xi_{\mathrm{tr}} \geq 0$ when:*

(a) Each MDP supporting $\mathcal{D}$ shares the same state-action space $\mathcal{S} \times \mathcal{A}$.

(b) For all M_i, M_j supporting $\mathcal{D}$ and all $(s, a) \in \mathcal{S} \times \mathcal{A}$ we have $|R_{M_i}(s, a) - R_{M_j}(s, a)| \leq \xi_{\mathrm{r}}$.

(c) For all M_i, M_j supporting $\mathcal{D}$ and all $(s, a) \in \mathcal{S} \times \mathcal{A}$ we have $\mathbb{TV}(\mathcal{T}_{M_i}(s, a), \mathcal{T}_{M_j}(s, a)) \leq \xi_{\mathrm{tr}}$.

The parameters $\xi_{\mathrm{r}}, \xi_{\mathrm{tr}}$ naturally quantify the similarity of different MDPs. Conditions of this form are canonical and have yielded fruitful research in classical RL literature [26, 27, 6, 25, 1], in the guise of the Simulation Lemma (see Section 2). To concretize this condition with an example, consider a suite of simulated robotic goal reaching tasks [48], where the physics simulator is the same in each task, so the transitions are fixed and $\xi_{\mathrm{tr}} = 0$, but the goal location changes from task to task, implying that $\xi_{\mathrm{r}} > 0$. We now establish our Weak Proximity condition, which strictly strengthens Condition 1.

Condition 2 (Weak Proximity) *The distribution $\mathcal{D}$ satisfies Weak Proximity with parameters $\xi_{\mathrm{r}}, \xi_{\mathrm{tr}}, \alpha \geq 0$ when:*

(a) $\mathcal{D}$ satisfies Condition 1 with parameters $\xi_{\mathrm{r}}, \xi_{\mathrm{tr}} \geq 0$.

(b) There exists a deterministic policy $\pi^\star$ which for any MDP M satisfies $V_M^{s_0}(\pi^\star) \geq \max_{\pi'} V_M^{s_0}(\pi') - \alpha$.

Weak Proximity strengthens Condition 1 by additionally requiring (via part (b)) that there exists some policy $\pi^\star$ which provides α-suboptimal value for each MDP supporting $\mathcal{D}$. Intuitively, this condition implicitly constrains the MDPs to be similar, since there is a single policy which provides (nearly) optimal value, irrespective of the MDP it is deployed in. Furthermore, recall from Eqs. (1) & (2) that the objectives of the Average Performance & Meta RL settings are defined in terms of value w.r.t the initial state s_0. So it is natural to assume, as we do in part (b), that there is one policy which provides good value w.r.t s_0 for all MDPs. From an algorithmic perspective, this is helpful, because it ensures that we can restrict our search to those policies which perform well for many MDPs supporting $\mathcal{D}$.

To concretize this condition in our aforementioned example of simulated robotic goal reaching tasks [48], consider a suite of tasks where each task has multiple different equivalent goals (so the task is complete when the robot reaches any single one of these goals), but there is only one goal location that is shared and invariant across each task. The trajectory that leads to this goal location from s_0 defines a policy $\pi^\star$, such that for any task M we have $V_M^{s_0}(\pi^\star) = \max_{\pi'} V_M^{s_0}(\pi')$, implying that Weak Proximity is satisfied with $\alpha = 0$.

Although Condition 1 is natural and well motivated by classical RL literature, it (and Weak Proximity) may seem strong. This is because it requires that each MDP supporting $\mathcal{D}$ shares the same state space, which may not hold in practice. We stress that we will prove a lower bound under Weak Proximity, showing that efficient generalization is impossible even in the simpler regime of a shared state space.

We now present Strong Proximity, a condition which strictly strengthens Weak Proximity. We will later show that unlike its Weak counterpart, Strong Proximity indeed permits efficient generalization (when the environments are deterministic).

Condition 3 (Strong Proximity) *The distribution $\mathcal{D}$ satisfies Strong Proximity with parameters $\xi_{\mathrm{r}}, \xi_{\mathrm{tr}}, \alpha \geq 0$ when:*

(a) $\mathcal{D}$ satisfies Condition 1 with parameters $\xi_{\mathrm{r}}, \xi_{\mathrm{tr}} \geq 0$.

(b) There exists a deterministic policy $\pi^\star$ which is a near optimal policy for each MDP. Concretely, the policy $\pi^\star$ satisfies $V_M^s(\pi^\star) \geq \max_{\pi'} V_M^s(\pi') - \alpha$ for each state s and each MDP M.

Let us compare Weak with Strong Proximity. Part (a) remains identical. But Weak Proximity (b) only requires a shared policy which provides α-suboptimal value with respect to s_0. This is in contrast to the shared policy in part (b) of Strong Proximity, which provides α-suboptimal value for *any* state.

3.3 Tractability of Individual Optimization

As discussed previously, in order to efficiently solve Eqs. (1) & (2), we require the property that each individual MDP supporting $\mathcal{D}$ can be efficiently solved. It is natural to expect such a property to hold in practice. For instance, in the context of our earlier example of simulated robotic goal reaching tasks [48], any individual task can be efficiently solved via policy gradient methods. We now state two such properties, the first of which is strictly stronger than the second. Since these properties require a notion of query cost, we state both of them with reference to a generic query model QM, and when we later present our results we will instantiate QM to be either SQM or WQM. To avoid complicating notation in these statements, we assume in this subsection (as is our focus throughout

the paper) that all MDPs supporting $\mathcal{D}$ are defined on the same state-action space $\mathcal{S} \times \mathcal{A} \subset \mathbb{R}^d$. Recall that a linear policy π is parameterized by $\overline{\theta} = \{\theta_h\}_{h=0}^{H-1}$, where $\theta_h \in \mathbb{R}^d$ and $\|\theta_h\|_2 = 1$ for all $0 \leq h \leq H-1$, such that $\pi(s) \in \operatorname{argmax}_{a \in \mathcal{A}}(s,a)^T \theta_h$ for any $s \in \mathcal{S}_h$. Here $x^T y$ denotes the Euclidean inner product of $x, y \in \mathbb{R}^d$. We use $\pi_M^\star$ to denote an arbitrary deterministic optimal policy of MDP M.

Property 1 (Strong Individual Optimization (SIO)) *Let the query model be QM. The distribution $\mathcal{D}$ satisfies SIO with parameters $k > 0$ and $0 \leq \beta < 1/4$ when:*

(a) Any MDP M supporting $\mathcal{D}$ admits an optimal linear policy. Concretely, given any M, there exists $\overline{\theta^\star} = \{\theta_h^\star\}_{h=0}^{H-1}$ such that for every state $s \in \mathcal{S}_h$ we have $\pi_M^\star(s) \in \operatorname{argmax}_{a \in \mathcal{A}}(s,a)^T \theta_h^\star$.

(b) There exists a fixed and known algorithm, such that given any MDP M and any state s, this algorithm uses at most $\mathcal{O}(|\mathcal{A}|H^k)$ query cost (under QM) on M to identify (almost surely) a linear policy $\pi(\overline{\theta})$ parameterized by $\overline{\theta} = \{\theta_h\}_{h=0}^{H-1}$ which satisfies $\max_{\pi'} V_M^s(\pi') \geq V_M^s(\overline{\theta}) \geq \max_{\pi'} V_M^s(\pi') - \beta$. This algorithm then outputs $\pi(\overline{\theta})$ as well as $V_M^s(\overline{\theta})$.

Let us discuss this property. Part (a) requires that for any MDP supporting $\mathcal{D}$, there exists an optimal linear policy. Part (b) requires that the user has knowledge of an algorithm, which can efficiently find a linear policy providing β-suboptimal value from any input state s in any MDP M. The exponent k describes the (polynomially sized) complexity of this algorithm. We stated the SIO property with respect to a generic efficient algorithm, since MDPs with different structures can require different types of algorithms to solve efficiently. Nevertheless, in our lower bound construction, the algorithm we provide to satisfy SIO is extremely simple and natural. It is simply a greedy version of Monte Carlo Tree Search, which is extremely popular in practice [29, 40].

SIO is a fairly strong property, since it says that a linear policy is sufficient to optimize any individual MDP, whereas in practice one typically requires nonlinear neural network policies. SIO also heavily constrains each individual MDP supporting $\mathcal{D}$ to be efficiently solvable from any initial state. We stress that we will prove our *lower bounds* under SIO, which makes our result stronger. Meanwhile, we prove our *upper bounds* under the following property, which is significantly weaker than SIO.

Property 2 (Weak Individual Optimization (WIO)) *Let the query model be QM. The distribution $\mathcal{D}$ satisfies WIO with parameter $0 \leq \beta < 1/4$ when the following holds. There exists an oracle $\widehat{V}$, which takes as input a state s and MDP M, and outputs $\widehat{V}_M^s$ satisfying $\max_{\pi'} V_M^s(\pi') \geq \widehat{V}_M^s \geq \max_{\pi'} V_M^s(\pi') - \beta$, via query cost (under QM) on M that is polynomial in $|\mathcal{A}|, H, d$.*

WIO postulates the existence of an oracle $\widehat{V}$, which can efficiently approximate the optimal value that is achievable from an input state and MDP. To see that WIO is strictly weaker than SIO, simply note we can implement $\widehat{V}$ by running the algorithm described in part (b) of SIO. Note that in certain states, a user may use domain knowledge to implement $\widehat{V}$ without solving an entire RL problem. Also note that WIO does not place (arguably unrealistic) linearity restrictions on the MDPs supporting $\mathcal{D}$.

4 Main Results

We shall present our results in two subsections. In Section 4.1, we prove lower bounds which demonstrate that even under Weak Proximity, SQM and SIO, tractable generalization is worst case impossible. In Section 4.2, we prove that efficient generalization is possible under Strong Proximity, WQM and WIO, when the MDPs supporting $\mathcal{D}$ share a deterministic transition function.

4.1 Lower Bounds

Before stating our own results, we first state the following classical result which is known as the Simulation Lemma [26, 27, 6, 25, 1]. Recall that $\xi_{\mathrm{r}}, \xi_{\mathrm{tr}}$ are parameters used to satisfy Condition 1.

Lemma 1 *Consider any $\mathcal{D}$ satisfying Condition 1 with $\xi_{\mathrm{r}}, \xi_{\mathrm{tr}} \geq 0$. For any policy π and any M_1, M_2 supporting $\mathcal{D}$, we have that $|V_{M_1}^{s_0}(\pi) - V_{M_2}^{s_0}(\pi)| \leq \xi_{\mathrm{r}} H + \xi_{\mathrm{tr}} H$.*

This result is almost identical to the one given by [25], although there are some (minor) differences in assumptions so we provide a proof in Appendix D. This lemma shows that when $\mathcal{D}$ satisfies Condition 1 and $\xi_{\mathrm{r}}, \xi_{\mathrm{tr}}$ are each $o(\frac{1}{H})$, then efficient generalization is trivial, at least in problems where H is large and we want to optimize to within $o(1)$ tolerance. Concretely, take any M supporting $\mathcal{D}$ and use a standard RL method to find π which satisfies $V_M^{s_0}(\pi) \approx \max_{\pi'} V_M^{s_0}(\pi')$. Then Lemma 1 ensures $V_{M'}^{s_0}(\pi) \gtrsim \max_{\pi'} V_{M'}^{s_0}(\pi') - o(1)$ for any other MDP M' supporting $\mathcal{D}$. This implies $\mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi)\right] \gtrsim \max_{\pi'} \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi')\right] - o(1)$ and $V_{\mathrm{M_{test}}}^{s_0}(\pi) \gtrsim \max_{\pi'} V_{\mathrm{M_{test}}}^{s_0}(\pi') - o(1)$.

Since Weak Proximity implies Condition 1, Lemma 1 and all the above statements remain true when $\mathcal{D}$ satisfies Weak Proximity. Naturally then, in our settings it is only interesting to consider problems when at least one of either ξ_{r} or ξ_{tr} is $\Omega(\frac{1}{H})$. Our next result is a *lower bound* which shows that when $\xi_{\mathrm{r}} = \Theta(\frac{1}{H})$ and $\xi_{\mathrm{tr}} = 0$, then Weak Proximity is not sufficient to efficiently generalize in the Average Performance Setting. For the statement of this result, recall that $\xi_{\mathrm{r}}, \xi_{\mathrm{tr}}, \alpha$ are parameters used to satisfy Weak Proximity, while β, k are parameters used to satisfy SIO.

Theorem 1 *Let the query model be SQM. For any $k \geq 3$, there exists $\mathcal{D}$ satisfying Weak Proximity with $\xi_{\mathrm{r}} = \Theta(\frac{1}{H})$, $\xi_{\mathrm{tr}} = 0$ & $\alpha = 0$ and SIO with $\beta = 0$ & k, such that the MDPs supporting $\mathcal{D}$ are deterministic and the following holds. Any (possibly randomized) algorithm requires $\Omega\left(\min\left\{|\,\mathcal{A}\,|^H, 2^d\right\}\right)$ total query cost to find (with probability at least $1/2$ over the randomness of the algorithm) a policy π satisfying*

$$\mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi)\right] \geq \max_{\text{linear policy } \pi'} \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi')\right] - 1/4.$$

We defer the proof to Appendix A.1. Let us discuss this theorem, which is stated for the Average Performance Setting, when the MDPs supporting $\mathcal{D}$ all share a deterministic transition function. Recall that SQM is the stronger query model we consider, which strengthens this lower bound, and trivially implies a lower bound for when WQM is the query model. Also recall that SIO is the stronger individual optimization property that we consider, and it ensures that the user can efficiently find a linear policy providing optimal value w.r.t any initial state for any individual MDP, since $\beta = 0$. Moreover, Weak Proximity (b) ensures that each MDP supporting $\mathcal{D}$ shares a policy that provides optimal value (w.r.t s_0), since $\alpha = 0$. And Weak Proximity (a) *explicitly* requires that the reward functions are (non-trivially) close, in the sense defined by Condition 1, because $\xi_{\mathrm{r}} = \Theta(\frac{1}{H})$. Despite this significant structure, the theorem demonstrates that one can still require an exponential query cost to find a policy that is nearly as good as the best *linear* policy (which is of course easier than finding the best generic policy). Note that this lower bound holds with $\alpha = \beta = \xi_{\mathrm{tr}} = 0$, and so implies a lower bound for when any of $\alpha, \beta, \xi_{\mathrm{tr}}$ are strictly positive. As we discuss at the end of Section 4.1, Theorem 1 (and its forthcoming corollaries) immediately applies to the task of learning a feature mapping which maps similar states to the same vector, for the purpose of efficiently solving Average Performance and Meta RL settings.

We note that in the construction used to prove the lower bound of Theorem 1, the algorithm we provide to satisfy the SIO property is extremely simple and natural. It is simply a greedy version of Monte Carlo Tree Search, which is extremely popular in practice [29, 40].

Let us provide some intuition for our proof of Theorem 1. In the $|\mathcal{A}|$-ary tree hard instance used in our proof, there are $\Omega(|\,\mathcal{A}\,|^H)$ possible trajectories. The fact that $\xi_{\mathrm{r}} = \Theta(\frac{1}{H})$ allows us enough degrees of freedom to hide the policy that generalizes across $\mathcal{D}$, so that identifying it requires querying each of the $\Omega(|\,\mathcal{A}\,|^H)$ trajectories. We leverage recent techniques [14, 45] to construct a suitable featurization of the state-action space, that is expressive enough to allow for efficiently finding an optimal linear policy for any single MDP, but does not leak any further information.

A similar result holds for the Meta RL setting. Recall that by SIO (b), the user has access to an algorithm which can solve any $\mathrm{M_{test}}$ at test time in $\mathcal{O}(|\,\mathcal{A}\,|H^k)$ queries, *even if it does no training*. So it only makes sense to train, if one can use this training to solve $\mathrm{M_{test}}$ in $o(|\,\mathcal{A}\,|H^k)$ queries. The following corollary to Theorem 1 demonstrates that this may require exponential query cost during training time. Its proof is presented in Appendix A.2.

Corollary 1 *Let the query model be SQM. For any $k \geq 3$, there exists $\mathcal{D}$ satisfying Weak Proximity with $\xi_{\mathrm{r}} = \Theta(\frac{1}{H})$, $\xi_{\mathrm{tr}} = 0$ & $\alpha = 0$ and SIO with $\beta = 0$ & k, such that the MDPs supporting $\mathcal{D}$ are deterministic and the following holds. If a (possibly randomized) algorithm at test time can identify π satisfying*

$$V_{\mathrm{M_{test}}}^{s_0}(\pi) \geq \max_{\text{linear policy } \pi'} V_{\mathrm{M_{test}}}^{s_0}(\pi') - 1/4$$

in $o(|\mathcal{A}|H^k)$ queries, with probability at least $1/2$ over the selection of M_{test} (and the randomness of the algorithm), then this algorithm must have required $\Omega\left(\min\left\{|\mathcal{A}|^H, 2^d\right\}\right)$ total query cost at training time.

So far we have presented results for when the MDPs supporting $\mathcal{D}$ share a deterministic transition function but have (slightly) varying rewards. For the remainder of Section 4.1, we present analogous results for when the MDPs share a reward function but have (slightly) varying transitions, again under both SIO and SQM. Recall from our discussion of Lemma 1 that when $\xi_{\text{r}} = 0$, it is only interesting to consider problems when ξ_{tr} is $\Omega(\frac{1}{H})$. Unfortunately, our next result is a corollary of Theorem 1 which shows that efficiently solving the Average Performance Setting is impossible in this regime.

Corollary 2 *Let the query model be SQM. For any $k \geq 3$, there exists $\mathcal{D}$ satisfying Weak Proximity with $\xi_{\text{r}} = 0$, $\xi_{\text{tr}} = \Theta(\frac{1}{H})$ & $\alpha = 0$ and SIO with $\beta = 0$ & k, such that the following holds. Any (possibly randomized) algorithm requires $\Omega\left(\min\left\{|\mathcal{A}|^H, 2^d\right\}\right)$ total query cost to find (with probability at least $1/2$ over the randomness of the algorithm) a policy π satisfying*

$$\mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi)\right] \geq \max_{\text{linear policy } \pi'} \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi')\right] - 1/4.$$

We defer the proof to Appendix A.3. We recall the discussion of Theorem 1, and note that the same discussion applies here, after swapping ξ_{tr} with ξ_{r}. An analogous result holds for the Meta RL setting. As we discussed before presenting Corollary 1, it only makes sense to train, if one can use this training in order to solve M_{test} in $o(|\mathcal{A}|H^k)$ queries. The following result shows that this is impossible without exponential total query cost at training time. Its proof is presented in Appendix A.4.

Corollary 3 *Let the query model be SQM. For any $k \geq 3$, there exists $\mathcal{D}$ satisfying Weak Proximity with $\xi_{\text{r}} = 0$, $\xi_{\text{tr}} = \Theta(\frac{1}{H})$ & $\alpha = 0$ and SIO with $\beta = 0$ & k, such that the following holds. If a (possibly randomized) algorithm at test time can identify π satisfying*

$$V_{\mathrm{M}_{\text{test}}}^{s_0}(\pi) \geq \max_{\text{linear policy } \pi'} V_{\mathrm{M}_{\text{test}}}^{s_0}(\pi') - 1/4$$

in $o(|\mathcal{A}|H^k)$ queries, with probability at least $1/2$ over the selection of M_{test} (and the randomness of the algorithm), then this algorithm must have required $\Omega\left(\min\left\{|\mathcal{A}|^H, 2^d\right\}\right)$ total query cost at training time.

In conjunction with Lemma 1, the results of Theorem 1 and Corollaries 1, 2 & 3 suggest that the classical (and quite natural) way of measuring variation in MDPs using Condition 1 is not the right metric for the modern Average Performance & Meta RL settings. When both ξ_{r} and ξ_{tr} are $o(\frac{1}{H})$, then these settings are trivially solvable. But when either ξ_{r} or ξ_{tr} is $\Theta(\frac{1}{H})$ then these settings become exponentially hard, even under the additional Weak Proximity condition as well as SIO & SQM.

Note that Theorem 1 and Corollaries 1, 2 & 3 all hold in the setting where each MDP supporting $\mathcal{D}$ shares a state-action space. So these lower bounds immediately apply to more complex settings where the MDPs are defined on disjoint state-action spaces, and where learning an appropriate representation is necessary. Indeed, it is popular in practice to learn a feature mapping which maps similar states to the same vector. Our results show that if such a mapping enables efficient solution of the Average Performance & Meta RL settings, then learning the mapping itself is worst case inefficient.

4.2 Upper Bound

We now show that Strong Proximity permits efficient generalization when the MDPs supporting $\mathcal{D}$ share deterministic transitions. While this setting is restricted, we study it because our Theorem 1 shows that even this setting can be worst case inefficient under strong assumptions. Furthermore, past literature on even traditional RL with a single MDP has often focused on the deterministic setting [47, 15]. Notably, to prove our upper bound we only require the weaker WQM and weaker WIO. Our method is defined in Algorithm 1. It exploits Strong Proximity, which requires the existence of a policy which provides optimal value for each MDP from *any* given initial state, even though the objectives in Eqs. (1) & (2) are defined only in terms of value w.r.t s_0.

Let us describe Algorithm 1. It represents policy π as a vector which stores one action for each timestep in $\{0, 1 \ldots H-1\}$. It initializes arbitrary π and incrementally updates it at each timestep

Algorithm 1 Inputs: horizon length H, distribution $\mathcal{D}$, sample size n, oracle $\widehat{V}$ as defined in WIO

Initialize π as an arbitrary function from $\{0, 1 \dots H-1\}$ to $\mathcal{A}$
for $t \in \{0, 1 \dots H-1\}$ **do**
  for $i \in \{1, 2 \dots n\}$ **do**
    Sample $M_i \sim \mathcal{D}$
    for $a \in \mathcal{A}$ **do**
      Begin a new episode in M_i at s_0
      if $t > 0$ **then** Execute action sequence $\{\pi(t')\}_{0 \le t' < t}$ to arrive at s_t **end if**
      Take action a to arrive at $s' = \mathcal{T}_{M_i}(s_t, a)$ and receive $R_{M_i}(s_t, a)$
      Query $\widehat{V}$ to obtain $\widehat{V}^{s'}_{M_i}$ and store $Q_{i,a} = R_{M_i}(s_t, a) + \widehat{V}^{s'}_{M_i}$
    end for
  end for
  Store $a_t \in \operatorname{argmax}_{a' \in \mathcal{A}} \{\frac{1}{n} \sum_{i=1}^{n} Q_{i,a'}\}$ and define $\pi(t) = a_t$
end for
return π

t. At the beginning of any timestep $t > 0$, π has been constructed to play the action $\pi(t') = a_{t'}$ at each timestep $t' < t$. The algorithm then executes $\{\pi(t')\}_{0 \le t' < t}$ to arrive at s_t. Crucially, due to the assumption of a shared state-action space and shared deterministic transitions, the state s_t is fully determined by π and does *not* depend on the particular M_i. Exploiting WIO, the method queries $\widehat{V}$ to estimate the value in M_i of each child state of s_t. Averaging this estimated value over $\{M_i\}_{i=1}^n$ yields an estimate of the expected value (over the randomness in $\mathcal{D}$) of each action at s_t. Finally, the algorithm picks the action a_t with the highest estimated value, and updates π to play a_t at timestep t. This algorithm operates in the standard RL framework and falls under the purview of WQM.

The following result provides a performance guarantee for Algorithm 1. Recall that $\alpha, \xi_{\mathrm{tr}}, \xi_{\mathrm{r}}$ are parameters used to satisfy Strong Proximity and β is a parameter used to satisfy WIO.

Theorem 2 *Let the query model be WQM. Consider any $\mathcal{D}$ satisfying WIO with $\beta \ge 0$ and Strong Proximity with $\xi_{\mathrm{tr}} = 0$ and any $\alpha, \xi_{\mathrm{r}} \ge 0$, such that the MDPs supporting $\mathcal{D}$ are deterministic. Fix $\epsilon, \delta > 0$, and let π be the output of Algorithm 1 when run with $n = \frac{H^2}{\epsilon^2} \log\left(\frac{2H|\mathcal{A}|}{\delta}\right)$ samples. Then with probability at least $1 - \delta$, we are guaranteed that*

$$\mathbb{E}_{M \sim \mathcal{D}}\left[V_M^{s_0}(\pi)\right] \ge \max_{\pi'} \mathbb{E}_{M \sim \mathcal{D}}\left[V_M^{s_0}(\pi')\right] - \epsilon - 3\,\alpha\,H - 3\,\beta\,H.$$

Hence the total query cost under WQM required to achieve this guarantee is polynomial in $q_{\mathcal{D}}, |\mathcal{A}|, H, d$.

We defer the proof to Appendix B. A few comments are in order. First, note that Theorem 2 directly provides a guarantee for the Average Performance setting. It also provides a guarantee for the Meta RL setting, since the π found by Algorithm 1 will on average perform well for $\mathrm{M}_{\mathrm{test}}$, and the user can use π to warm start any finetuning or adaptation at test time. Second, the specified value of n depends only on quantities that are either known a priori or chosen by the user. This makes Algorithm 1 *parameter free* — the user does not need to know the values of $\alpha, \beta, \xi_{\mathrm{r}}, \xi_{\mathrm{tr}}$ to run this method.

Third, note Theorem 2 holds under WIO. By contrast, Weak Proximity was insufficient for efficient generalization even when paired with SIO. This suggests that a condition that is both necessary and sufficient for efficient generalization lies somewhere between Weak and Strong Proximity — assuming, of course, that we do not assume an individual optimization property that is even stronger than SIO. Indeed, SIO is already quite strong, since SIO says that a linear policy is sufficient to optimize any individual MDP, but in practice one typically employs nonlinear neural network policies.

Finally, observe that ξ_{r} does not appear in the error bound. So ξ_{r} can be arbitrarily large, and Theorem 2 requires no *explicit* conditions on the reward functions of the MDPs supporting $\mathcal{D}$, as in the sense of Condition 1. Instead, the *implicit* reward structure induced by the shared nearly optimal policy required by Strong Proximity is sufficient. Comparing this observation with the result of Theorem 1 suggests that the classical explicit constraints on rewards and transitions is not appropriate for modern RL generalization settings. Instead, implicit constraints of the sort afforded by Strong Proximity offer a more fine grained characterization of when efficient generalization is possible.

Recall from Theorem 1 that a lower bound holds for Weak Proximity and SIO even with $\alpha = \beta = 0$. However, Strong Proximity and WIO provide enough structure that the error bound of Theorem 2 can tolerate $\alpha, \beta \geq 0$. But these α, β terms in the error bound of Theorem 2 scale linearly with H. It is natural to question whether this scaling is due to a suboptimality of Algorithm 1 or looseness in our analysis. We provide a partial answer to this question in Appendix C, where we prove that the dependency on β given in the result of Theorem 2 is tight to within a logarithmic factor in H.

5 Discussion

In this paper, we studied the design of RL agents that generalize. We proved that efficient generalization is worst case impossible, even under structural conditions like Weak Proximity and strong assumptions on the query model and tractability of individual MDPs. This result extends to the task of learning representations for the purpose of efficient generalization. On the positive side, we provided Strong Proximity, which permits efficient generalization, even under mild assumptions on the query model and individual tractability. Our analysis highlights that classical metrics for measuring similarity of MDPs are inappropriate for modern RL. It also suggests that a condition which is both necessary and sufficient for efficient generalization lies between Weak & Strong Proximity — unless we make (arguably unreasonable) assumptions on the tractability of individual MDPs.

Negative Societal Impacts. Our work is theoretical, and we do not foresee any direct societal impacts, at least in the short term. In the long term, our work may increase the technological feasibility of developing agents that can be deployed in society. In this scenario, a bad actor may deploy harmful, malicious agents. This must be prevented by properly understanding the technology (which our work aims to do), and working with policy makers to prevent bad actors from accessing it.

Limitations of Our Work. The primary limitation of our work is that our upper bound has limited applicability. It holds only when the MDPs share a state-action space, and when the MDPs are determinstic, which is very restrictive in practice. Our rationale for working in this restricted setting was due to our lower bounds, which show that even this toy setting can be worst case inefficient, and because it is necessary to understand the toy setting before looking at more complex scenarios. Nevertheless, our upper bound is several steps removed from the practice of RL. It is best interpreted as a preliminary sufficient condition for when efficient generalization is possible, albeit in a toy setting, and is far from conclusive on this matter.

Future Work. Note that our upper bound might apply if we are a priori given a feature mapping which maps similar states of different MDPs to the same state space. For example, in self driving, learning to drive in different countries might be difficult because the images of traffic signs are different. But if a known feature map extracts the underlying meaning of these signs, then our upper bound could conceivably apply. The key direction for future work, is how to learn such a feature mapping efficiently, while ensuring that it is still useful for generalization.

Acknowledgments and Disclosure of Funding

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE1745016. We further acknowledge the support of NSF via RI 2007517 and IIS-1909816. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] P. Abbeel and A. Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2005.

[2] A. Agarwal, S. Kakade, and L. F. Yang. Model-based reinforcement learning with a generative model is minimax optimal. In *Proceedings of the Conference on Learning Theory*, 2020.

[3] R. Agarwal, M. C. Machado, P. S. Castro, and M. G. Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2021.

[4] M. G. Azar, R. Munos, and H. J. Kappen. On the sample complexity of reinforcement learning with a generative model. In *Proceedings of the International Conference on Machine Learning*, 2012.

[5] M. Bertran, N. Martinez, M. Phielipp, and G. Sapiro. Instance-based generalization in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

[6] R. I. Brafman and M. Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2003.

[7] E. Brunskill and L. Li. Sample complexity of multi-task reinforcement learning. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2013.

[8] P. S. Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[9] P. S. Castro and D. Precup. Using bisimulation for policy transfer in mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010.

[10] I. Clavera, A. Nagabandi, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019.

[11] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman. Quantifying generalization in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2019.

[12] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2020.

[13] S. S. Du, Y. Luo, R. Wang, and H. Zhang. Provably efficient q-learning with function approximation via distribution shift error checking oracle. In *Advances in Neural Information Processing Systems*, 2019.

[14] S. S. Du, S. M. Kakade, R. Wang, and L. F. Yang. Is a good representation sufficient for sample efficient reinforcement learning? In *International Conference on Learning Representations*, 2020.

[15] S. S. Du, J. D. Lee, G. Mahajan, and R. 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.

[16] A. Fallah, A. Mokhtari, and A. Ozdaglar. Provably convergent policy gradient methods for model-agnostic meta-reinforcement learning. *arXiv preprint arxiv:2002.05135*, 2020.

[17] J. Farebrother, M. C. Machado, and M. Bowling. Generalization and regularization in DQN. *arXiv preprint arxiv:1810.00123*, 2018.

[18] F. Feng, W. Yin, and L. F. Yang. Does knowledge transfer always help to learn a better policy? *arXiv preprint arxiv:1912.02986*, 2019.

[19] N. Ferns, P. Panangaden, and D. Precup. Metrics for finite markov decision processes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2004.

[20] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning*, 2017.

[21] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2017.

[22] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine. Composable deep reinforcement learning for robotic manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2018.

[23] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[24] N. Jiang. PAC reinforcement learning with an imperfect model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[25] S. Kakade, M. Kearns, and J. Langford. Exploration in metric state spaces. In *Proceedings of the International Conference on Machine Learning*, 2003.

[26] M. Kearns and D. Koller. Efficient reinforcement learning in factored mdps. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999.

[27] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49:209–232, 2002.

[28] M. Kearns, Y. Mansour, and A. Y. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999.

[29] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the European Conference on Machine Learning*, 2006.

[30] C. L. Lan, M. G. Bellemare, and P. S. Castro. Metrics and continuity in reinforcement learning. *arXiv preprint arxiv:2102.01514*, 2021.

[31] T. Lattimore, C. Szepesvari, and G. 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.

[32] A. Lazaric and M. Ghavamzadeh. Bayesian multi-task reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2010.

[33] V. Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[34] A. Nichol, V. Pfau, C. Hesse, O. Klimov, and J. Schulman. Gotta learn fast: A new benchmark for generalization in RL. *arXiv preprint arxiv:1804.03720*, 2018.

[35] C. Packer, K. Gao, J. Kos, P. Krähenbühl, V. Koltun, and D. Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arxiv:1810.12282*, 2018.

[36] A. Rajeswaran, K. Lowrey, E. V. Todorov, and S. M. Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*. 2017.

[37] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the International Conference on Machine Learning*, 2019.

[38] B. V. Roy and S. Dong. Comments on the du-kakade-wang-yang lower bounds. *arXiv preprint arxiv:1911.07910*, 2019.

[39] A. Sidford, M. Wang, X. Wu, L. Yang, and Y. Ye. Near-optimal time and sample complexities for solving markov decision processes with a generative model. In *Advances in Neural Information Processing Systems*, 2018.

[40] D. Silver et al. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[41] A. Sonar, V. Pacelli, and A. Majumdar. Invariant policy optimization: Towards stronger generalization in reinforcement learning. *arXiv preprint arxiv:2006.01096*, 2020.

[42] X. Song, Y. Jiang, S. Tu, Y. Du, and B. Neyshabur. Observational overfitting in reinforcement learning. In *International Conference on Learning Representations*, 2020.

[43] H. Wang, S. Zheng, C. Xiong, and R. Socher. On the generalization gap in reparameterizable reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2019.

[44] L. Wang, Q. Cai, Z. Yang, and Z. Wang. On the global optimality of model-agnostic meta-learning. In *Proceedings of the International Conference on Machine Learning*, 2020.

[45] R. Wang, S. S. Du, L. F. Yang, and R. Salakhutdinov. On reward-free reinforcement learning with linear function approximation. In *Advances in Neural Information Processing Systems*. 2020.

[46] G. Weisz, P. Amortila, B. Janzer, Y. Abbasi-Yadkori, N. Jiang, and C. Szepesvàri. On query-efficient planning in mdps under linear realizability of the optimal state-value function. *arXiv preprint arxiv:2102.02049*, 2021.

[47] Z. Wen and B. Van Roy. Efficient exploration and value function generalization in deterministic systems. In *Advances in Neural Information Processing Systems*, 2013.

[48] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*, 2019.

[49] A. Zhang, N. Ballas, and J. Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arxiv:1806.07937*, 2018.

[50] A. Zhang, R. T. McAllister, R. Calandra, Y. Gal, and S. Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] .
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] .
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 4 for the complete statements of all the theoretical results, see the beginning of Section 3 for notation and preliminaries, and see Sections 3.2 & 3.3 for the Conditions and Properties that are fundamental to our results.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendices A, B, C, & D.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A] No experiments were run.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A] No experiments were run.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] No experiments were run.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A] No experiments were run.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A] No such assets were used or created.
   (b) Did you mention the license of the assets? [N/A] No such assets were used or created.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A] No such assets were used or created.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] No such assets were used or created.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] No such assets were used or created.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We did not crowdsource or conduct research with human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] We did not crowdsource or conduct research with human subjects.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We did not crowdsource or conduct research with human subjects.

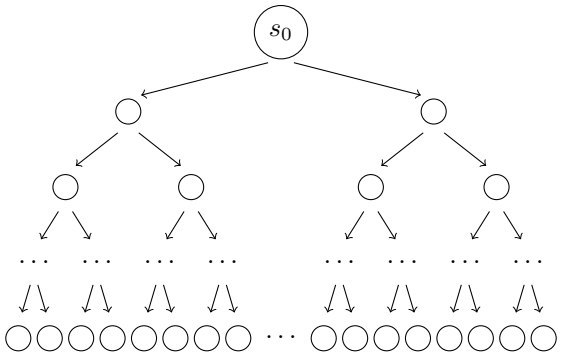

Figure 1: An illustration of the generic binary tree structure used to define the MDPs that support the $\mathcal{D}$ constructed in the proof of Theorem 1.

A Lower Bound Proof Details

In this section, we will provide proofs of Theorem 1 and Corollaries 1, 2 & 3. For ease in presentation, we shall assume throughout that the action space $\mathcal{A}$ for each MDP contains two actions, which we denote a_1 and a_2. It is easy to extend the proofs to the case when there are many actions. We will often use the notation s_1 and s_2 to denote the child states of a state s when taking actions a_1 and a_2 respectively. We shall also use π^*_M to denote an optimal policy for MDP M. Whenever we require H to be sufficiently large for an algebraic argument to go through, we shall assume so.

A.1 Proof Of Theorem 1

Construction Details. Our construction of $\mathcal{D}$ will consist of MDPs whose shared state and action spaces are generically defined by a binary tree. The structure of the binary tree is depicted in Figure 1. The tree is of length H, each node in the tree will define a different state and the edges connecting two nodes define the actions. In this fashion, each state has two actions, which we denote a_1 and a_2, and taking either action leads deterministically to the corresponding child. Taking any action from states on the final level of the tree exits the MDP. The state s_0 is the root of the tree. We thus have defined the shared state and action space for MDPs supporting $\mathcal{D}$, and have also defined the shared and deterministic transition dynamics. So the MDPs are all deterministic by construction. Note that to verify Weak Proximity (a), the construction so far implies that the state-action space is shared and the transitions satisfy $\xi_{\text{tr}} = 0$, although we have not yet defined the reward structure so cannot say anything about ξ_{r}.

To complete the definition of $\mathcal{D}$, we must complete the definition of each individual MDP supporting $\mathcal{D}$ by defining a reward function for each MDP. We shall do so via the procedure described below. But before that, fix a state s^* on level H, which is selected by sampling uniformly at random from the set of states on level H. Note that the location of s^* will be kept hidden from the user, although we will define $\mathcal{D}$ with reference to a fixed s^*.

Note that there are $2^{\frac{H}{2}-1} - 1$ states s on level $\frac{H}{2}$ such that the subtree rooted at s does not contain s^*. Also note that each subtree rooted at such an s has $2^{\frac{H}{2}-1}$ states on level H. If one were to "view" the final level of a binary tree on paper, as in Figure 1, there is a natural ordering of the states on the final level from left to right. In each subtree rooted at such an s, consider the ith state in this ordering of the states on the final level of that subtree, and denote this ith state as $x_{i,s}$. Then define $\mathcal{S}_i = \cup_s x_{i,s}$. There are a total of $2^{\frac{H}{2}-1}$ such sets $\mathcal{S}_i$. We will construct $2^{\frac{H}{2}-1}$ MDPs total to support the distribution $\mathcal{D}$, one for each $\mathcal{S}_i$. Note that for $i \neq j$, we have $\mathcal{S}_i \cap \mathcal{S}_j = \varnothing$. Note also that $\mathcal{S}_i$ never contains a state which is also in the subtree rooted at level $\frac{H}{2}$ that contains s^*.

Define $\epsilon = \frac{1}{\frac{H}{2} - \log(H^k) - 1}$, and note that ϵ is $\Theta(\frac{1}{H})$. For each $\mathcal{S}_i$, we will define the MDP M_i by defining R_{M_i} as follows. Fix an $\mathcal{S}_i$. For each $s \in \mathcal{S}_i$, let s_p denote the unique ancestor of s on level

$\frac{H}{2} + \log(H^k) + 1$ of the tree. On the path connecting s_p to s, assign each state-action pair a reward of ϵ. So the total reward accumulated by following the path from the root through s_p to s and taking either action from s is $\epsilon(\frac{H}{2} - \log(H^k) - 1) = 1$. Also, consider the unique ancestor s_p^* of s^* on level $\frac{H}{2} + \log(H^k) + 1$, and assign each state-action pair along the path s_p^* to s^* a reward of ϵ. So the total reward accumulated by following the path from the root to s^* and taking either action from s^* is $\epsilon(\frac{H}{2} - \log(H^k) - 1) = 1$. Any other state-action pair in the tree is assigned zero reward. This completes the definition of R_{M_i}, and hence the definition of M_i, except that we have not yet featurized the state-action space.

Perform this procedure for each of the $2^{\frac{H}{2}-1}$ sets $\mathcal{S}_i$, to obtain a set which contains $2^{\frac{H}{2}-1}$ such MDPs. Once we featurize the state-action space in the fashion described below, the definition of $\mathcal{D}$ is completed by assigning the uniform distribution to this set. The key behind this construction, is that for any MDP supporting $\mathcal{D}$, each subtree rooted at level $\frac{H}{2}$ contains a single path which provides the optimal unit value. But there is only one path providing unit value that is shared by each MDP, this is the path from the root to s^*. Note that for any M_i, all of the subtrees rooted at states on level $\frac{H}{2}$ are identical, with the exception of the subtree containing s^*.

Let us now discuss how to featurize the state-action space. Note that each MDP is defined on a common state-action space that has at most 2^H states. Sample vectors $\{z_1, z_2 \dots z_{2^H}\}$ i.i.d from the spherical measure on the surface of the unit sphere that sits in $d-2$ dimensions. By the results of [14], we can pick d to be at most a polynomial of H, while ensuring that $\|z_i\|_2 = 1$ and $|z_i^T z_j| \leq \frac{1}{50}$ for all $i \neq j$. Take any ordering of the states $\{s_i\}$ indexed by i, and assign the following features

$$\phi(s_i, a_1) = [z_i, 1, 0] \text{ and } \phi(s_i, a_2) = [0, 1, 1].$$

Rescaling the features to have unit norm is trivial, so we work with these since they make the computations more apparent. In the above, it appears that $\phi(s_i, a_2)$ is the same for each i, but this is without loss of generality since we can always add a dummy coordinate that makes them all unique. Crucially note that the features do *not* depend on the reward structure of the MDP, since they are completely agnostic to the choice of rewards. Hence they do not leak any information about the rewards.

Verifying Weak Proximity (a). We have already checked above that the state-action space is shared and $\xi_{\text{tr}} = 0$. To see that ξ_{r} is $\Theta(\frac{1}{H})$, simply note that any state in any MDP supporting $\mathcal{D}$ has reward either 0 or ϵ, and ϵ is $\Theta(\frac{1}{H})$. This verifies Weak Proximity (a).

Verifying Weak Proximity (b). Define π^* to be the deterministic policy which prescribes the path leading from the root to s^*, and at states not along this path it prescribes an arbitrary action. It is then immediate from our above arguments that for any M, $V_M^{s_0}(\pi^*) = \max_{\pi'} V_M^{s_0}(\pi') = 1$, so Weak Proximity (b) is satisfied with parameter $\alpha = 0$.

Verifying SIO (a). Fix any M. The key to verifying this is the observation that all subtrees rooted at level $H/2$ are identical (in terms of reward structure, not features), except for the one that contains s^*. In the first $H/2 - 1$ levels one can use an arbitrary policy. So now consider any level h that is greater than or equal to $H/2$. Inside the subtree rooted at $H/2$ that contains s^*, there is a unique state s on this level that is an ancestor of s^*. Let us denote z to be the spherical measure random variable that was used to define the feature map for this state, so that $\phi(s, a_1) = [z, 1, 0]$ and $\phi(s, a_2) = [0, 1, 1]$.

Now observe that within the subtree rooted at $H/2$ that contains s^*, the only state from where one can achieve nonzero reward (on the level h we are considering) is s. So the policy at the other states in this subtree does not matter. Similarly, in each of the other subtrees rooted at $H/2$, there is a unique state s' which can yield nonzero reward (on the level h we are considering). Furthermore, since these other subtrees are identical, the optimal policy within one subtree works for another. To construct our $\theta \equiv \theta_h^\star$ for this level h, we use this information to claim that there are four cases to consider.

First, consider the case when $\pi^*_M(s) = a_1$ and $\pi^*_M(s') = a_2$ for any s' in another subtree which can yield nonzero reward. Let $\theta = [z, 0, 1/2]$. Then we have that

$$\phi(s, a_1)^T\theta = [z, 1, 0]^T[z, 0, 1/2] = 1{+}0{+}0 = 1 \text{ and } \phi(s, a_2)^T\theta = [0, 1, 1]^T[z, 0, 1/2] = 0{+}0{+}1/2 = 1/2,$$

which implies that we pick a_1 at s if we follow the linear policy. But

$$\phi(s', a_1)^T\theta = [z', 1, 0]^T[z, 0, 1/2] \leq \frac{1}{50}{+}0{+}0 = \frac{1}{50} \text{ and } \phi(s', a_2)^T\theta = [0, 1, 1]^T[z, 0, 1/2] = 0{+}0{+}1/2 = 1/2,$$

which implies that we pick a_2 at s' if we follow the linear policy θ. Note that this argument holds for *any* s' which can yield nonzero reward lying in *any* of the subtrees that do not contain s^*, and also recall the optimal policy does not change when looking at different subtrees not containing s^*.

Second, consider the case when $\pi^*_M(s) = a_2$ and $\pi^*_M(s') = a_1$ for any s' in another subtree which can yield nonzero reward. Let $\theta = -[z, 0, 1/2]$. Since we have simply negated the above case, the result holds.

Third, consider the case when $\pi^*_M(s) = a_1$ and $\pi^*_M(s') = a_1$ for any s' in another subtree which can yield nonzero reward. Let $\theta = [0, 1, -1/2]$. Then the result follows immediately because $\phi(s'', a_1)^T\theta = 1$ but $\phi(s'', a_2)^T\theta = 1/2$ for every state s'' in the tree.

Fouth, consider the case when $\pi^*_M(s) = a_2$ and $\pi^*_M(s') = a_2$ for any s' in another subtree which can yield nonzero reward. Let $\theta = -[0, 1, -1/2]$. Then the result follows immediately because we have simply negated the above case.

It remains to rescale the features and θ to ensure they all have unit norm.

Verifying SIO (b). Consider the following algorithm, which takes as input an MDP M_i that is known to support $\mathcal{D}$, but it knows nothing else about M_i (in particular it does not know which $\mathcal{S}_i$ was used to define M_i). It of course does not know anything about the location of s^* other than the fact that s^* was sampled uniformly at random to define $\mathcal{D}$. It has also not been allowed to query any other MDPs supporting $\mathcal{D}$. The algorithm begins by picking an arbitrary state s' on level $\frac{H}{2}$. It then queries each state on level $\log(H^k) + 1$ of the subtree rooted at s' (this is level $\frac{H}{2} + \log(H^k) + 1$ of the entire tree). There is a single state s_p on this level which has an action providing reward ϵ, and all other states will give reward zero. When it finds the state s_p on this level providing reward ϵ, it stores the path leading to this state s_p. It then queries each of the two child states of s_p to identify which of these child states lies along the optimal path, which is doable since exactly one of these child states has an action which provides reward ϵ. It greedily takes the action required to arrive at this child, and stores this action. It then repeats this greedy procedure from the child $\Theta(H)$ times until it reaches the final level of the tree. In this manner, if it was given MDP M_i as input then it identifies a path from the root through s' and s_p (or perhaps s^*_p) to either s^* or some state s on level H which satisfies $s \in \mathcal{S}_i$. Let π be the deterministic policy which prescribes this path, and prescribes arbitrary actions for states not along this path, then it is clear $V^{s_0}_{M_i}(\pi) = \max_{\pi'} V^{s_0}_{M_i}(\pi') = 1$. Note also that this same algorithm also immediately can be used to find π satisfying $V^{a}_{M_i}(\pi) = \max_{\pi'} V^{a}_{M_i}(\pi')$ for any initial state a. To see this, note that if a is an ancestor of s^* or some $s \in \mathcal{S}_i$, then the described algorithm will identify this and find an appropriate path. If a is not an ancestor of either of these, then the value of this state is zero and the same algorithm can be used to certify this. The sample complexity of this algorithm is the number of queries it took at the beginning to identify s_p in the subtree rooted at s'. Note that there are $2^{\log(H^k)+1}$ states on level $\log(H^k) + 1$ of the subtree rooted at s', so the sample complexity of this algorithm is $\mathcal{O}(H^k)$. To convert the policy π found by the algorithm into a linear policy $\overline{\theta} = \{\theta_h\}_{h=0}^{H-1}$, simply note that we can set $\theta_h = [0, 1, -\frac{1}{2}]$ if π recommends a_1 while following its path at timestep h, and set $\theta_h = [0, -1, \frac{1}{2}]$ if π recommends a_2 while following its path at timestep h. Then rescale θ_h to ensure it has unit norm. This verifies SIO (b) with parameter $\beta = 0$.

With this construction of $\mathcal{D}$ in hand, we return to the proof of Theorem 1, during which we shall also prove the claim we made above about how s^* cannot be identified in a polynomial number

of samples. A basic computation reveals that any path through the tree which does not end in s^*, has value (in expectation over $\mathcal{D}$ w.r.t. s_0) at most $\frac{2}{H}$. However, the path through the tree which ends in s^* has value (in expectation over $\mathcal{D}$ w.r.t. s_0) precisely one. The optimal policy (in terms of value in expectation over $\mathcal{D}$ w.r.t. s_0) is hence clearly the deterministic policy which prescribes following the path through the tree that ends in s^*. Hence, any algorithm which can find a policy π satisfying $\mathbb{E}_{M\sim\mathcal{D}}[V_M^{s_0}(\pi)] \geq \max_{\pi'} \mathbb{E}_{M\sim\mathcal{D}}[V_M^{s_0}(\pi')] - \frac{1}{4}$ must be able to identify s^* with non-trivial probability. So to show that finding such a π requires $\Omega(2^H)$ queries, it is sufficient to show that identifying s^* requires $\Omega(2^H)$ queries.

This is immediate from the nature of our construction. Simply observe that any algorithm must query in the subtree rooted at level $\frac{H}{2}$ that contains s^* at least once in order to identify s^*. However, for any MDP there are $2^{\frac{H}{2}-1}$ subtrees on level $\frac{H}{2}$ and all but one of them are identical. Of course, identifying the correct subtree with non-trivial probability requires $\Omega(2^H)$ total queries. Note that we crucially used here the fact that the features do *not* depend on the reward structure of the MDP, and hence do not leak any information about the rewards. Note also that our argument holds when the agent has access to a generative model, since it can transition to any state to query it. Moreover, the difficulty here comes from querying states, and not from sampling MDPs. So the result holds under SQM. Finally, we note that the policy prescribing the path through s^* can easily be expressed as a linear policy. This completes the proof.

A.2 Proof Of Corollary 1

We use the same $\mathcal{D}$ that was constructed in the proof of Theorem 1. Before proceeding with the proof of Corollary 1, we note the following key fact. If we sample M from $\mathcal{D}$, but do not query any MDP supporting $\mathcal{D}$ beforehand, then any algorithm that can find (possibly nonlinear) π satisfying $V_M^{s_0}(\pi) \geq \max_{\pi'} V_M^{s_0}(\pi') - 1/4$, with probability at least $1/2$ over the selection of M, requires $\Omega(H^k)$ query cost (under QM) on M. This fact demonstrates that the algorithm described in SIO (b) is minimax optimal, since *any* procedure will need the same complexity to solve an MDP sampled from $\mathcal{D}$, assuming it has not already queried other MDPs beforehand. In particular, any algorithm which can solve $\mathrm{M}_{\mathrm{test}}$ at test time in $o(H^k)$ queries must rely on querying during training time.

To prove this fact, note that any algorithm (that is optimal to within the constant $\frac{1}{4}$) must discover a path that has $\Omega(H)$ state-action pairs intersecting with a path that leads either to s^* or to a state $s \in \mathcal{S}_i$. Of course, discovering such a path is equivalent to identifying s^* or the s_p corresponding to $s \in \mathcal{S}_i$. Assume for now the claim that we cannot identify s^* with a polynomial number of samples. Then we need only show that identifying an s_p corresponding to $s \in \mathcal{S}_i$ requires $\Omega(H^k)$ queries. But this is immediate, since all we know is that M_i was sampled from $\mathcal{D}$, and each subtree (that doesn't contain s^*) rooted at level $\frac{H}{2}$ is identical. So without loss of generality take any subtree (not containing s^*) rooted at level $\frac{H}{2}$, then a priori our prior for the location of s_p in that subtree is exactly the uniform distribution over states on level $\log(H^k)+1$ of that subtree. So we must query at least half of the $2^{\log(H^k)+1} = \Omega(H^k)$ states on level $\log(H^k)+1$ of that subtree to identify s_p with non-trivial probability. Conditioned on our claim that it is not easy to identify s^*, the claimed fact. Note that this argument uses the fact that we can directly query states on level $\log(H^k)+1$ of that subtree, and so holds under a generative model. Note that we crucially used here the fact that the features do *not* depend on the reward structure of the MDP, and hence do not leak any information about the rewards.

We now return to the proof of Corollary 1. For any M_i, there are two types of paths that are optimal, the first is through s^* and the others are through the states lying in $\mathcal{S}_i$. Note that since $\mathrm{M}_{\mathrm{test}} \sim \mathcal{D}$ at test time, the location of the optimal paths in $\mathrm{M}_{\mathrm{test}}$ that do not intersect s^* are sampled uniformly at random. A nearly identical argument to the one used in the proof of Theorem 1, and the proof of the preceding fact, shows that identifying any of these optimal paths that are sampled uniformly at random requires $\Omega(H^k)$ queries. The only difference is that we condition on the event that $\mathrm{M}_{\mathrm{test}}$ is not the same as any of the MDPs queried during training, which occurs with high probability when we are only allowed a polynomial number of queries during training time. Note that even after conditioning on this event, finding an optimal path (which does not go through s^*) of $\mathrm{M}_{\mathrm{test}}$ requires

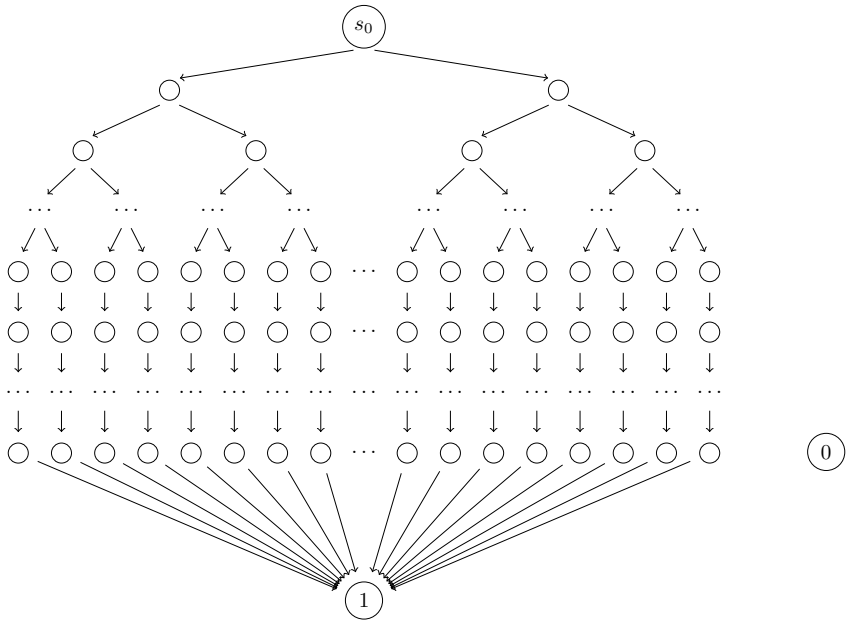

Figure 2: An illustration of the generic structure used to define the MDPs that support the $\mathcal{D}$ constructed in the proof of Corollary 2. Observe that the first half of the structure is a tree, while the second half comprises of linear sequences of states.

$\Omega(H^k)$ queries. This is simply because conditioning on the aforementioned event only changes the probability of the location of such a path by a value which is exponentially small in H, assuming again that we used polynomial in H number of queries during training. So we cannot perform any inference to reduce the size of the set of feasible locations of an optimal path (which does not go through s^*). Then we can essentially repeat the same argument used in the proof of Theorem 1. Hence, if an algorithm hopes to solve $\mathrm{M}_{\mathrm{test}}$ at test time in $o(H^k)$ queries, then during training time it must narrow the possible locations of s^* to a set whose cardinality is polynomial in H. But by the proof of Theorem 1, this would require $\Omega(2^H)$ queries.

A.3 Proof Of Corollary 2

Construction Details. The $\mathcal{D}$ that we construct here is very similar in spirit to the $\mathcal{D}$ that was constructed in the proof of Theorem 1. $\mathcal{D}$ will be supported by MDPs whose shared state and action spaces all share the same generic structure. We depict this generic structure in Figure 2. For the first $\frac{H}{2}$ levels, the structure is defined by a binary tree, where the nodes in the tree represent states. Each state in the tree has two actions. For the next $\frac{H}{2}$ levels, there are numerous linear sequences of states of length $\frac{H}{2}$. Each such sequence emanates from a corresponding state that is a leaf of the tree. These sequences all end in a common state, which we denote (1). Each state in the sequence has a single action. The state s_0 is the root of the tree. There is also a state (0), which a priori is disconnected from the remainder of the structure. This implies that there is a shared state-action space. For each MDP M supporting $\mathcal{D}$, we will have that R_M maps (1) to one, and maps all other states including (0) to zero. Taking any action from (0) or (1) always exits the MDP for any MDP supporting $\mathcal{D}$. A priori, the transition function in this generic structure is deterministic. And it is "natural" in the sense that taking either action from a state in the tree leads to the corresponding child, and taking the action when in the linear sequence of states leads to the next state in the sequence. Of course, we will modify this "a priori natural" transition function in different ways for each MDP.

Let $\mathcal{F}_h$ denote the set of states at level h of this generic structure that we have outlined above. Before constructing $\mathcal{D}$, first select a state s^* uniformly at random from the $2^{H/2-1}$ states in $\mathcal{F}_{H/2}$. As in the

proof of Theorem 1, s^* is hidden from the user.

Note that there are $2^{\frac{H}{4}-1}-1$ states s on level $\frac{H}{4}$ such that the subtree rooted at s does not contain s^*. Also note that each subtree rooted at such an s has $2^{\frac{H}{4}-1}$ states on level $\frac{H}{2}$ of the entire structure. If one were to "view" the final level of a binary tree on paper, there is a natural ordering of the states on the final level from left to right. In each subtree rooted at such an s such that subtree does not contain s^*, consider the ith state in this ordering of the states on the final level of that subtree (which is the level $\frac{H}{2}$ of the entire structure). Denote this ith state as $x_{i,s}$. Then define $\mathcal{S}_i = \cup_s x_{i,s}$. There are a total of $2^{\frac{H}{4}-1}$ such sets $\mathcal{S}_i$. We will construct $2^{\frac{H}{4}-1}$ MDPs total to support the distribution $\mathcal{D}$, one for each $\mathcal{S}_i$. Note that for $i \neq j$, we have $\mathcal{S}_i \cap \mathcal{S}_j = \varnothing$. Note also that $\mathcal{S}_i$ never contains a state which is also in the subtree rooted at level $\frac{H}{4}$ that contains s^*.

To define each MDP M that supports $\mathcal{D}$, it is sufficient to define the transition function $\mathcal{T}_M$ for M, since we have already defined the shared state-action space and the shared reward function. Similar to the proof of Theorem 1, for each $\mathcal{S}_i$ construct an MDP M_i as follows:

1. Consider the linear portion of M_i that lies below any state s' satisfying $s' \notin \mathcal{S}_i$ and $s' \neq s^*$. Recall that a priori, the generic structure for any MDP was deterministic, which implies when you are at a state in this linear portion then taking the action deterministically leads to the next state in the sequence. We now modify this so that for any state-action pair along the linear sequence below s', taking the action takes you to the child with probability $1 - {}^{10}/_{H}$, and makes you jump out to state $\textcircled{0}$ with probability ${}^{10}/_{H}$. Do this for all s' satisfying $s' \notin \mathcal{S}_i$ and $s' \neq s^*$. Leave the linear sequence of states below s^* and any $s \in \mathcal{S}_i$ unchanged, so that the transitions in these two linear sequences remain deterministic.
2. For each $s \in \mathcal{S}_i$, there is a unique path connecting state s to its unique ancestor on level $\frac{H}{4}$. This path defines a sequence of state-action pairs that leads one from level $\frac{H}{4}$ to s. Again recall that a priori, the generic structure for any MDP was deterministic, which implies that taking one of these actions while in the tree deterministically leads you to the corresponding child. Modify the transitions for these state-action pairs along this path of length $\frac{H}{4}$, so that when you take an action, then with probability $1 - {}^{1}/_{H^{k-1}}$ the action leads to the child, and with probability ${}^{1}/_{H^{k-1}}$ it takes you directly to state $\textcircled{1}$. Do this for each $s \in \mathcal{S}_i$. Similarly, consider the path between s^* and its unique ancestor on level $\frac{H}{4}$. Modify the transitions for the state-action pairs along this path so that when you take an action, then with probability $1 - {}^{1}/_{H^{k-1}}$ the action leads to the child, and with probability ${}^{1}/_{H^{k-1}}$ it takes you directly to state $\textcircled{1}$.

We have thus defined $\mathcal{T}_{M_i}$ for each M_i, where each M_i is identified by a particular $\mathcal{S}_i$. This defines a set containing a total of $2^{\frac{H}{4}-1}$ different MDPs M_i. To complete the definition of $\mathcal{D}$, simply consider the uniform distribution over the set of M_i that we have created. We use a featurization that is identical to the one used in Theorem 1 for the binary tree portion of our construction here, and arbtirary features for the linear portion since there is only one action to take here.

Verifying Weak Proximity (a). We have already established the state-action space for each MDP is shared. Note that by definition each MDP shares a reward function, since $\textcircled{1}$ always provides unit reward and each other state provides zero reward, regardless of the MDP. So we have $\xi_{\mathrm{r}} = 0$. Then observe that when we modified the transitions at states, we changed them by making them lead to states $\textcircled{0}$ or $\textcircled{1}$ with probability at most $\max\{\frac{1}{H^k}, \frac{10}{H}\}$. It remains to note the ℓ_1 characterization of the total variation distance and that $k \geq 3$, implying that $\xi_{\mathrm{tr}} = \Theta(\frac{1}{H})$. This verifies Weak Proximity (a).

Verifying Weak Proximity (b). Observe that following the unique path leading from the root through state s^* till $\textcircled{1}$ always provides value 1, regardless of the MDP. This is because if we take the actions corresponding to this path, then either we hop out to $\textcircled{1}$ while we are in the tree, or we

reach the linear sequence from where we deterministically arrive at (1). Hence this path is always optimal. In turn, the policy which prescribes this path, while doing something arbitrary at states not along this path, always provides optimal value. So Weak Proximity (b) is satisfied with parameter $\alpha = 0$.

Verifying SIO (a). The proof of this is identical to the one in Theorem 1.

Verifying SIO (b). To verify this, sample $M_i \sim \mathcal{D}$ and consider the following greedy algorithm. Note that s^* is not revealed a priori, nor do we know the locations of the states in $\mathcal{S}_i$, and the only information that the user has is that this MDP supports $\mathcal{D}$. Otherwise this problem trivially does not require any samples to solve. Start at any arbitrary state on level $\frac{H}{4}$ and sample the left action $\mathcal{O}(H^{k-1})$ times and the right action $\mathcal{O}(H^{k-1})$ times. With high probability, one of these actions will lead to the state (1) at least once. And of course with unit probability the other action will not lead to (1). Simply take the action that has led to (1) at least once, and repeat this procedure at the next state. Repeating this procedure until you reach level $H/2$ and union bounding guarantees that with high probability we find a path that leads to either s^* or a state $s \in \mathcal{S}_i$. From here on, one can deterministically follow the linear sequence of states to arrive at (1). The policy that prescribes this path then provides the optimal (unit) value for that MDP, so $\beta = 0$. The total sample complexity of this method is $\Theta(H^{k-1} \times H) = \Theta(H^k)$. Again, note that this argument holds under a generative model as defined in SQM, since we can plug in whatever state we want, and receive as feedback from the generative model the next state sampled from the transition process. Note also that in similar fashion to Theorem 1, this same algorithm can be used to identify a policy achieving optimal value with respect to any initial state in the tree. To convert the policy found by the algorithm into a linear policy, we use the same technique as in Theorem 1.

We now complete the proof of Corollary 2. A basic computation shows that for $k \geq 3$ and sufficiently large H, the value (in expectation over $\mathcal{D}$) of any path is at most $\frac{1}{10}$. But the value of the path (in expectation over $\mathcal{D}$) through s^* till (1) is one. Hence any algorithm which can find π satisfying $\mathbb{E}_{M\sim\mathcal{D}}[V_M^{s_0}(\pi)] \geq \max_{\pi'} \mathbb{E}_{M\sim\mathcal{D}}[V_M^{s_0}(\pi')] - \frac{1}{4}$ must be able to identify s^* with non-trivial probability. So to show that finding such a π requires $\Omega(2^H)$ queries, it is sufficient to show that identifying s^* requires $\Omega(2^H)$ queries. This is done via identical arguments to the ones used to prove Theorem 1.

A.4 Proof Of Corollary 3

The proof of this corollary is basically identical to the proof of Corollary 1. Briefly, if an algorithm can solve $\mathrm{M}_{\mathrm{test}}$ with a number of queries at test time that is strictly fewer than $\Omega(H^k)$, then at training time it must have narrowed the possible locations of s^* to a set whose cardinality is polynomial in H. By the proofs of Theorem 1 and Corollary 2, this requires $\Omega(2^H)$ queries during training time.

B Upper Bound Proof Details

In this section, we will provide a formal proof of Theorem 2. For ease in presentation, we shall assume throughout that the action space $\mathcal{A}$ for each MDP contains two actions, which we denote a_1 and a_2. It is easy to extend the proofs to the case when there are many actions. We will often use the notation s_1 and s_2 to denote the child states of a state s when taking actions a_1 and a_2 respectively. We shall also use π^*_M to denote an optimal policy for MDP M. Before proving Theorem 2, we shall state two helpful lemmas. Recall the definition of π^* from Strong Proximity (b).

Lemma 2 *Consider any $\mathcal{D}$ satisfying WIO with $\beta \geq 0$ and Strong Proximity with $\xi_{\mathrm{tr}} = 0$ and any $\alpha, \xi_{\mathrm{r}} \geq 0$, such that the MDPs supporting $\mathcal{D}$ are deterministic. Run Algorithm 1 with $n \geq 1$ samples, and assume at timestep t we are at state s_t such that $\pi^*(s_t) = a_1$. We are guaranteed that the event*

$$\frac{1}{n}\sum_{i=1}^{n} Q_{i,a_1} \geq \frac{1}{n}\sum_{i=1}^{n} Q_{i,a_2} - \alpha - \beta$$

occurs almost surely. The symmetric statement for $\pi^*(s_t) = a_2$ is also true.

Lemma 3 *Consider any $\mathcal{D}$ satisfying WIO with $\beta \geq 0$ and Strong Proximity with $\xi_{\text{tr}} = 0$ and any $\alpha, \xi_{\text{r}} \geq 0$, such that the MDPs supporting $\mathcal{D}$ are deterministic. Run Algorithm 1 with $n = \frac{H^2}{\epsilon^2} \log\left(\frac{2H|\mathcal{A}|}{\delta}\right)$ samples, and assume at timestep t we are at state $s \equiv s_t$. Then the event*

$$\left|\frac{1}{n}\sum_{i=1}^{n} Q_{i,a_1} - \mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a_1) + V_M^{s_1}(\pi_M^*)\right]\right| \leq \beta + \frac{\epsilon}{2H}$$

$$\textit{and } \left|\frac{1}{n}\sum_{i=1}^{n} Q_{i,a_2} - \mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a_2) + V_M^{s_2}(\pi_M^*)\right]\right| \leq \beta + \frac{\epsilon}{2H}$$

occurs with probability at least $1 - \frac{\delta}{H}$.

The proofs of Lemmas 2 and 3 can be found in Appendix B.2 and Appendix B.3 respectively. With these lemmas in hand, we now turn to the proof of Theorem 2.

B.1 Proof Of Theorem 2

First observe that by Strong Proximity (b), we have

$$\mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi^*)\right] \geq \mathbb{E}_{M\sim\mathcal{D}}\left[\max_{\pi_M} V_M^{s_0}(\pi_M)\right] - \alpha \geq \max_{\pi'} \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi')\right] - \alpha\,.$$

Hence to prove the theorem, it is sufficient to prove that

$$\mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi)\right] \geq \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi^*)\right] - \epsilon - 2\,\alpha\,H - 3\,\beta\,H,$$

and we shall devote the remainder of the proof to this.

Recall that the policy constructed by Algorithm 1 is represented as a vector of length H, which stores an action for each timestep. We use the terminology "algorithm recommends an action at a timestep" to mean that at that timestep, the algorithm stores that action in the policy that it is constructing. Our proof rests on a key claim, which is that while following Algorithm 1, at each timestep the algorithm recommends an action whose suboptimality (in expectation over all MDPs) relative to the other action is at most $\frac{\epsilon}{H} + 2\,\alpha + 3\,\beta$. Concretely, assume the algorithm recommends an action a at state s which transports us to s'. We claim that with high probability at least $1 - \frac{\delta}{H}$,

$$\mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a) + V_M^{s'}(\pi^*)\right] \geq \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s}(\pi^*)\right] - \frac{\epsilon}{H} - 2\,\alpha - 3\,\beta\,. \tag{3}$$

First, we argue why this claim in Eq. (3) is sufficient to prove the theorem. Assume this claim to be true, and denote s_h to be the state achieved by the algorithm after h timesteps and a_h to be the action recommended by the algorithm at timestep h. Then applying a union bound over H timesteps, with high probability at least $1 - \delta$ we are guaranteed that

$$\begin{aligned}
\mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_H}(\pi^*) + \sum_{h=0}^{H-1} R_M(s_h,a_h)\right] &= \mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s_{H-1},a_{H-1}) + V_M^{s_H}(\pi^*)\right] + \mathbb{E}_{M\sim\mathcal{D}}\left[\sum_{h=0}^{H-2} R_M(s_h,a_h)\right] \\
&\geq \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_{H-1}}(\pi^*) + \sum_{h=0}^{H-2} R_M(s_h,a_h)\right] - \frac{\epsilon}{H} - 2\,\alpha - 3\,\beta \\
&\geq \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_{H-2}}(\pi^*) + \sum_{h=0}^{H-3} R_M(s_h,a_h)\right] - \frac{2\epsilon}{H} - 4\,\alpha - 6\,\beta \\
&\geq \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi^*)\right] - \frac{\epsilon H}{H} - 2\,\alpha\,H - 3\,\beta\,H \\
&= \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi^*)\right] - \epsilon - 2\,\alpha\,H - 3\,\beta\,H.
\end{aligned}$$

By assumption, the MDPs supporting $\mathcal{D}$ have shared deterministic transitions and a common state-action space $\mathcal{S}\times\mathcal{A}$, and hence the above calculations remain valid. So we have found a sequence of actions $\{a_h\}_{h=0}^{H-1}$, which defines a path through the $\mathcal{S}\times\mathcal{A}$ and enables us to arrive at state $s_H \in \mathcal{S}$

with the above property. Of course $V_M^{s_H}(\pi^*)$ is just trivially zero, since the planning horizon is H. So the path $\{a_h\}_{h=0}^{H-1}$ we have found, which defines a deterministic policy denoted by π, satisfies

$$\begin{aligned}\mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi)\right] &= \mathbb{E}_{M\sim\mathcal{D}}\left[\sum_{h=0}^{H-1} R_M(s_h,a_h)\right] \\ &= \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_H}(\pi^*) + \sum_{h=0}^{H-1} R_M(s_h,a_h)\right] \\ &\geq \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi^*)\right] - \epsilon - 2\,\alpha\,H - 3\,\beta\,H,\end{aligned}$$

which exactly proves the theorem.

Hence it is sufficient to prove the claim in Eq. (3), and we shall devote the remainder to proving this claim. Assume that while running the algorithm we are at some state s. Recall the notation that s_1 and s_2 are the child states of s when taking actions a_1 and a_2 respectively, and recall π_M^* denotes an optimal policy for MDP M. By the result of Lemma 3, we have with probability at least $1-\frac{\delta}{H}$ that

$$\begin{aligned}&\left|\frac{1}{n}\sum_{i=1}^{n} Q_{i,a_1} - \mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a_1) + V_M^{s_1}(\pi_M^*)\right]\right| \leq \beta + \frac{\epsilon}{2H} \\ \text{and } &\left|\frac{1}{n}\sum_{i=1}^{n} Q_{i,a_2} - \mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a_2) + V_M^{s_2}(\pi_M^*)\right]\right| \leq \beta + \frac{\epsilon}{2H}.\end{aligned} \tag{4}$$

We condition on this event to verify the claim in Eq. (3).

Now assume WLOG that $\pi^*(s) = a_1$, since the case when $\pi^*(s) = a_2$ is entirely symmetric. By the principle of Bellman optimality, this ensures that for any MDP M we have $R_M(s,a_1) + V_M^{s_1}(\pi^*) = V_M^s(\pi^*)$. Furthermore by the result of Lemma 2,

$$\frac{1}{n}\sum_{i=1}^{n} Q_{i,a_1} \geq \frac{1}{n}\sum_{i=1}^{n} Q_{i,a_2} - \alpha - \beta\,. \tag{5}$$

We now consider two cases. For the first case, assume $\frac{1}{n}\sum_{i=1}^{n} Q_{i,a_1} > \frac{1}{n}\sum_{i=1}^{n} Q_{i,a_2}$. Then the algorithm recommends action a_1 and transports us to state s_1. Recall again by our WLOG assumption that for any MDP M, we have $R_M(s,a_1) + V_M^{s_1}(\pi^*) = V_M^s(\pi^*)$. So we are guaranteed that

$$\mathbb{E}\left[R_M(s,a_1) + V_M^{s_1}(\pi^*)\right] = \mathbb{E}\left[V_M^s(\pi^*)\right],$$

and so the claim in Eq. (3) is trivially shown to be true in this case.

For the second case, assume $\frac{1}{n}\sum_{i=1}^{n} Q_{i,a_1} \leq \frac{1}{n}\sum_{i=1}^{n} Q_{i,a_2}$. Then the algorithm will recommend action a_2 and transport us to state s_2. But note that by the bound Eq. (5), we have

$$\left|\frac{1}{n}\sum_{i=1}^{n} Q_{i,a_1} - \frac{1}{n}\sum_{i=1}^{n} Q_{i,a_2}\right| \leq \alpha + \beta\,.$$

Combining this with the bound Eq. (4), and the triangle inequality, we have

$$\left|\mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a_1) + V_M^{s_1}(\pi_M^*)\right] - \mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a_2) + V_M^{s_2}(\pi_M^*)\right]\right| \leq \alpha + 3\,\beta + \frac{\epsilon}{H}.$$

Hence we have

$$\begin{aligned}\mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a_2) + V_M^{s_2}(\pi_M^*)\right] &\geq \mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a_1) + V_M^{s_1}(\pi_M^*)\right] - \alpha - 3\,\beta - \frac{\epsilon}{H} \\ &\geq \mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a_1) + V_M^{s_1}(\pi^*)\right] - \alpha - 3\,\beta - \frac{\epsilon}{H} \\ &= \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^s(\pi^*)\right] - \alpha - 3\,\beta - \frac{\epsilon}{H},\end{aligned} \tag{6}$$

where the second inequality follows from the optimality of π^*_M for M and the equality follows from our WLOG assumption and the principle of Bellman optimality. Of course Strong Proximity (b) also guarantees that for any MDP M we have $V_M^{s_2}(\pi^*) \geq V_M^{s_2}(\pi^*_M) - \alpha$. We use this to upper bound the LHS of Eq. (6) and obtain

$$\mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a_2) + V_M^{s_2}(\pi^*)\right] \geq \mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a_2) + V_M^{s_2}(\pi^*_M)\right] - \alpha \geq \mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s}(\pi^*)\right] - 2\,\alpha - 3\,\beta - \frac{\epsilon}{H}.$$

This exactly demonstrates the claim in Eq. (3), which as argued earlier, is sufficient to complete the proof of the theorem.

B.2 Proof Of Lemma 2

Let $s \equiv s_t$ and recall that s_1 and s_2 denote the child states obtained by taking actions a_1 and a_2 respectively from state s. The assumption $\pi^*(s) = a_1$ is WLOG, since the case when $\pi^*(s) = a_2$ is entirely symmetric. Now consider any MDP M and recall π^*_M denotes an optimal policy for MDP M. There are two cases to consider.

For the first case, assume there exists π^*_M such that $\pi^*_M(s) = a_1$. Then,

$$R_M(s,a_1) + V_M^{s_1}(\pi^*_M) = V_M^{s}(\pi^*_M) \geq R_M(s,a_2) + V_M^{s_2}(\pi^*_M) \geq R_M(s,a_2) + V_M^{s_2}(\pi^*_M) - \alpha,$$

where we used the principle of Bellman optimality.

For the second case, assume there only exists π^*_M such that $\pi^*_M(s) = a_2$. Then,

$$\begin{aligned} R_M(s,a_1) + V_M^{s_1}(\pi^*_M) &\geq R_M(s,a_1) + V_M^{s_1}(\pi^*) \\ &= V_M^{s}(\pi^*) \\ &\geq V_M^{s}(\pi^*_M) - \alpha \\ &= R_M(s,a_2) + V_M^{s_2}(\pi^*_M) - \alpha, \end{aligned}$$

where the first inequality follows from the optimality of π^*_M for M, the equalities follow from the principle of Bellman optimality as well as the WLOG assumption that $\pi^*(s) = a_1$, and the second inequality follows from Strong Proximity (b).

So in either case we are guaranteed that

$$R_M(s,a_1) + V_M^{s_1}(\pi^*_M) \geq R_M(s,a_2) + V_M^{s_2}(\pi^*_M) - \alpha\,.$$

Recall that for action a leading to state s' from state s, we have $Q_{i,a} = R_{M_i}(s,a) + \widehat{V}_{M_i}^{s'}$ by definition. We are then guaranteed by WIO that

$$R_{M_i}(s,a) + V_{M_i}^{s'}(\pi^*_{M_i}) \geq Q_{i,a} \geq R_{M_i}(s,a) + V_{M_i}^{s'}(\pi^*_{M_i}) - \beta\,.$$

Hence for any M_i we must have

$$\begin{aligned} Q_{i,a_1} &= R_{M_i}(s,a_1) + \widehat{V}_{M_i}^{s_1} \\ &\geq R_{M_i}(s,a_1) + V_{M_i}^{s_1}(\pi^*_{M_i}) - \beta \\ &\geq R_{M_i}(s,a_2) + V_{M}^{s_2}(\pi^*_{M_i}) - \alpha - \beta \\ &\geq R_{M_i}(s,a_2) + \widehat{V}_{M_i}^{s_2} - \alpha - \beta \\ &= Q_{i,a_2} - \alpha - \beta\,. \end{aligned}$$

Averaging each side of the above inequality over n completes the proof.

B.3 Proof Of Lemma 3

Note that for action a leading to state s' from state s, we have $Q_{i,a} = R_{M_i}(s,a) + \widehat{V}_{M_i}^{s'}$ by definition. We are then guaranteed by WIO that

$$R_{M_i}(s,a) + V_{M_i}^{s'}(\pi^*_{M_i}) \geq Q_{i,a} \geq R_{M_i}(s,a) + V_{M_i}^{s'}(\pi^*_{M_i}) - \beta\,.$$

Hence we must have

$$\frac{1}{n}\sum_{i=1}^{n}(R_{M_i}(s,a)+V_{M_i}^{s'}(\pi_{M_i}^*)) \geq \frac{1}{n}\sum_{i=1}^{n}Q_{i,a} \geq \frac{1}{n}\sum_{i=1}^{n}(R_{M_i}(s,a)+V_{M_i}^{s'}(\pi_{M_i}^*)) - \beta\,.$$

By Hoeffding's bound and our choice of n, we are guaranteed that with probability at least $1-\frac{\delta}{H}$ that

$$\left|\frac{1}{n}\sum_{i=1}^{n}(R_{M_i}(s,a_1)+V_{M_i}^{s_1}(\pi_{M_i}^*)) - \mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a_1)+V_M^{s_1}(\pi_M^*)\right]\right| \leq \frac{\epsilon}{2H}$$

$$\text{and } \left|\frac{1}{n}\sum_{i=1}^{n}(R_{M_i}(s,a_2)+V_{M_i}^{s_2}(\pi_{M_i}^*)) - \mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a_2)+V_M^{s_2}(\pi_M^*)\right]\right| \leq \frac{\epsilon}{2H}.$$

So for action a leading to state s' from state s, we can combine the previous two equations via the triangle inequality to obtain

$$\begin{aligned}
&\left|\frac{1}{n}\sum_{i=1}^{n}Q_{i,a} - \mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a)+V_M^{s'}(\pi_M^*)\right]\right| \\
\leq &\left|\frac{1}{n}\sum_{i=1}^{n}(R_{M_i}(s,a)+V_{M_i}^{s'}(\pi_{M_i}^*)) - \frac{1}{n}\sum_{i=1}^{n}Q_{i,a}\right| \\
&+\left|\frac{1}{n}\sum_{i=1}^{n}(R_{M_i}(s,a)+V_{M_i}^{s'}(\pi_{M_i}^*)) - \mathbb{E}_{M\sim\mathcal{D}}\left[R_M(s,a)+V_M^{s'}(\pi_M^*)\right]\right| \\
\leq &\,\beta+\frac{\epsilon}{2H}.
\end{aligned}$$

This completes the proof.

C Near Tightness Of Theorem 2

As discussed in Section 4.2, the α, β terms in the error bound of Theorem 2 scale linearly with H. So when either α or β is $\Omega(\frac{1}{H})$, then our bound becomes vacuous. It is natural to question whether this unfortunate scaling is due to a possible suboptimality of Algorithm 1 or some looseness in our analysis. The following result provides a (partial) answer to this question.

Proposition 1 *Let n be the total query cost that any algorithm is allowed to use under WQM. For any $\beta \geq 0$, there exists $\mathcal{D}$ satisfying WIO with β and Strong Proximity with $\xi_{\mathrm{r}} = \xi_{\mathrm{tr}} = \alpha = 0$, such that the MDPs supporting $\mathcal{D}$ are deterministic and the following holds. Any (possibly randomized) algorithm will output (with probability at least $\frac{1}{2}$) a policy π satisfying*

$$\mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi)\right] \leq \max_{\pi'}\mathbb{E}_{M\sim\mathcal{D}}\left[V_M^{s_0}(\pi')\right] - \frac{\beta\,H}{\log(50n)}.$$

This result demonstrates that the dependency on β given in the result of Theorem 2 is tight to within a logarithmic factor in H, and cannot be improved beyond this logarithmic factor by a better algorithm or sharper analysis. It remains unclear whether the dependence on α is tight. Isolating this is more difficult, since in any construction α and β become inherently intertwined, and it is unclear to us how to separate the dependencies on these two distinct parameters. We believe this is an interesting direction for future work. Intuitively the dependency on α seems tight — roughly speaking it seems natural that if we are planning over H timesteps, while using the structure induced by π^* to guide our actions, then the suboptimality of α at each timestep leads to a total error blowup of $\alpha\,H$. Nevertheless, a formal proof (or otherwise) would be an interesting development. We now turn to proving Proposition 1.

Proof. Fix any $\beta \geq 0$. It is sufficient to construct a single MDP M with deterministic transitions, and construct an oracle $\widehat{V}$ satisfying WIO with parameter β, and show that any algorithm will output a policy π that satisfies

$$V_M^{s_0}(\pi) \leq \max_{\pi'} V_M^{s_0}(\pi) - \frac{\beta H}{\log(50n)},$$

with probability at least $\frac{1}{2}$. This is because we can define $\mathcal{D}$ to be the point mass on M, so that Strong Proximity (a) and (b) are satisfied trivially with $\xi_{\mathrm{r}} = \xi_{\mathrm{tr}} = \alpha = 0$. We will define M as a binary tree, where the states are nodes in the tree, and the (deterministic) actions are described by the edges connecting nodes, which immediately ensures that M is deterministic. We will construct the following MDP M by chunking the levels of the tree into blocks. Each block has length $\log(50n)$. To facilitate the definition of the reward function used to define M, we will first assign each state its optimal value, i.e. the value that one could get by following the optimal policy from that state. This will naturally allow us to later define rewards. We will assign these optimal values of the states in a sequential fashion, by considering each subtree rooted at some state on level $h_k = k \log(50n)$, where k is a nonnegative integer.

Start by considering level $h_0 = 0$. Then on level $h_1 = \log(50n)$, pick a single state uniformly at random, denote it s_{h_1,s_0}, and assign it value that satisfies $V^{s_{h_1,s_0}}(\pi^*) = V^{s_0}(\pi^*)$. We will call this state the special state for level h_1. Let all other states s on level h_1 have value $V^s(\pi^*) = V^{s_0}(\pi^*) - \beta$.

Recursively define the values of the leaves below this level in an identical fashion. To be more concrete, do the following procedure for each state s on level $h_1 = \log(50n)$. Consider all the states in the subtree of s that lie on level $h_2 = 2\log(50n)$. Pick a single one of these states uniformly at random, denote it $s_{h_2,s}$, and assign it value that satisfies

$$V^{s_{h_2,s}}(\pi^*) = V^s(\pi^*).$$

We will call this state one of the special states for level h_2. And for all other states $s' \neq s_{h_2,s}$ in the subtree of s that lie on level $h_2 = 2\log(50n)$, assign them each value that satisfies

$$V^{s'}(\pi^*) = V^s(\pi^*) - \beta.$$

In this fashion, we can assign values for every state that lies on a level $k\log(50n)$ where k is a nonnegative integer. We assume without loss of generality that $\log(50n)$ evenly divides H. Of course, this immediately defines the values for all states in the tree. To actually define rewards which satisfies the structure induced by the values, fix $V_M^{s_0}(\pi^*)$ to be any number, and then simply assign rewards to the leaves of the tree which agree with the assigned values of those leaves. The reward is zero for all states in the tree that are not leaves.

We now define the oracle $\widehat{V}$. For any state s in the levels h satisfying $h_0 \leq h \leq h_1$, let $\widehat{V}(s) = V^{s_0}(\pi^*) - \beta$. Similarly, consider any state s on level h_1, and now consider any state $s' \neq s$ in the subtree rooted at s such that level$(s') \leq h_2$. Then let $\widehat{V}(s') = V^s(\pi^*) - \beta$. Recursively repeat this until we have defined $\widehat{V}$ for each state in the tree. By definition, we have defined $\widehat{V}$ so that it satisfies WIO with parameter β.

With this construction in hand, the proof is now straightforward to complete. It is sufficient to prove this in the case when an algorithm returns a deterministic policy (or path), since any stochastic policy is a randomization over deterministic paths. We claim that with probability at least $\frac{1}{2}$, the path outputted by the algorithm never intersects *any* of the special states in the *entire* tree. It is sufficient to prove this, because this implies that the path loses β value for a total of $\frac{H}{\log(50n)}$ times, implying that its value is at most

$$V_M^{s_0}(\pi^*) - \frac{\beta H}{\log(50n)}.$$

By symmetry, it is clear that any algorithm which can identify even a single special state anywhere in the tree with non-trivial probability, can be used to identify the special state on level h_1. Hence, it is sufficient to prove that there is no algorithm which can identify the special state on level h_1 with non-trivial probability. Strengthen the query model (slightly) so that querying a state will return whether it is the special state on level h_1. But now, observe that we are only allowed to query n states total. And to identify the special state on level h_1, the algorithm must query most of the states on level h_1, which is a total of

$$2^{\log(50n)} = 50n$$

states. This implies that it cannot identify the special state on level h_1 with probability at least $\frac{1}{2}$. As discussed above, this is sufficient to complete the proof. $\square$

D Auxiliary Proof Details

In this section, we prove Lemma 1, relying heavily on the treatment given in [25]. Note that in our setting, we only use this result in the context of our lower bounds, and all our lower bounds have finite state-action spaces. So it is sufficient to prove the result assuming that the state-action space is finite.

Let τ denote a trajectory, and let $\mathbb{P}_1^\pi(\tau), \mathbb{P}_2^\pi(\tau)$ denote the probabilities of taking τ in MDPs M_1, M_2 respectively. Let $R_1(\tau), R_2(\tau)$ denote the rewards obtained by following τ in MDPs M_1, M_2 respectively. The penultimate step of the proof of Lemma 4.3 in [25] shows that

$$\sum_\tau |\mathbb{P}_1^\pi(\tau) - \mathbb{P}_2^\pi(\tau)| \leq \xi_{\mathrm{tr}}\, H. \tag{7}$$

The definition of ξ_{r} and a straightforward application of the triangle inequality reveals that

$$|R_1(\tau) - R_2(\tau)| \leq \sum_{t=0,(s_t,a_t)\in\tau}^{H-1} |R_{M_1}(s_t,a_t) - R_{M_2}(s_t,a_t)| \leq \xi_{\mathrm{r}}\, H. \tag{8}$$

Again using the triangle inequality, we obtain that

$$\begin{aligned}
\left|V_{M_1}^{s_0}(\pi) - V_{M_2}^{s_0}(\pi)\right| &= \left|\sum_\tau \left(\mathbb{P}_1^\pi(\tau)R_1(\tau) - \mathbb{P}_2^\pi(\tau)R_2(\tau)\right)\right| \\
&\leq \left|\sum_\tau \left(\mathbb{P}_1^\pi(\tau)R_1(\tau) - \mathbb{P}_1^\pi(\tau)R_2(\tau)\right)\right| + \left|\sum_\tau \left(\mathbb{P}_1^\pi(\tau)R_2(\tau) - \mathbb{P}_2^\pi(\tau)R_2(\tau)\right)\right| \\
&\leq \sum_\tau \mathbb{P}_1^\pi(\tau)\,|R_1(\tau) - R_2(\tau)| + \left|\sum_\tau \left(\mathbb{P}_1^\pi(\tau)R_2(\tau) - \mathbb{P}_2^\pi(\tau)R_2(\tau)\right)\right| \\
&\leq \xi_{\mathrm{r}}\, H + \left|\sum_\tau \left(\mathbb{P}_1^\pi(\tau)R_2(\tau) - \mathbb{P}_2^\pi(\tau)R_2(\tau)\right)\right|,
\end{aligned}$$

where we used the result of Eq. (8). Furthermore,

$$\begin{aligned}
\left|V_{M_1}^{s_0}(\pi) - V_{M_2}^{s_0}(\pi)\right| &\leq \xi_{\mathrm{r}}\, H + \left|\sum_\tau \left(\mathbb{P}_1^\pi(\tau)R_2(\tau) - \mathbb{P}_2^\pi(\tau)R_2(\tau)\right)\right| \\
&\leq \xi_{\mathrm{r}}\, H + \sum_\tau |\mathbb{P}_1^\pi(\tau) - \mathbb{P}_2^\pi(\tau)|\,|R_2(\tau)| \\
&\leq \xi_{\mathrm{r}}\, H + \sum_\tau |\mathbb{P}_1^\pi(\tau) - \mathbb{P}_2^\pi(\tau)| \\
&\leq \xi_{\mathrm{r}}\, H + \xi_{\mathrm{tr}}\, H,
\end{aligned}$$

where we used the result of Eq. (7) and also our assumption that the reward of any trajectory is always upper bounded by one. This completes the proof.