When Is Generalizable Reinforcement Learning Tractable?

Anonymous Author(s)
Affiliation
Address
email

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 [32, 39] 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 [35, 48, 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 [36, 47]. 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 underlying distribution over the environments that an agent might encounter. The agent's goal is to perform well on average across this distribution [34, 33, 11]. The *Meta Reinforcement Learning* setting is closely related [20, 10, 36]. 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.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Of course, in full generality, both notions of generalization are impossible to achieve efficiently. 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 [36]. 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 [31, 7, 24, 42] 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 [34, 11, 12, 36, 47]. On the theoretical side, [5, 41] 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, 43] 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, 29, 49]. Some works extend this to multiple MDPs [9, 3, 40], 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 $h \in \{0, 1 \ldots H-1\}$ 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 $|\mathcal{A}|, H$.

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 $q_{\mathcal{D}}, |\mathcal{A}|, H$.

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, 37, 14, 30, 45]. 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, 38, 14, 30, 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, 36]. 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 [34, 47]. 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 [47], 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 $\mathcal{D}$.

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.

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. 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 $\pi(s) \in \mathrm{argmax}_{a \in \mathcal{A}} (s,a)^T \theta_h$ 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 $\pi_M^\star(s) \in \mathrm{argmax}_{a \in \mathcal{A}} (s,a)^T \theta_h^\star$.

(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.

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 $|\mathcal{A}|, H$.*

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 $\Omega(|\mathcal{A}|^H)$ 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 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, 44] 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 $\mathrm{M}_{\mathrm{test}}$ 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 $\mathrm{M}_{\mathrm{test}}$ 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*

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

in $o(|\mathcal{A}|H^k)$ queries, with probability at least $1/2$ over the selection of $\mathrm{M}_{\mathrm{test}}$ (and the randomness of the algorithm), then this algorithm must have required $\Omega(|\mathcal{A}|^H)$ 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 $\xi_{\mathrm{r}} = 0$, 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 $\xi_{\mathrm{r}} = 0$, $\xi_{\mathrm{tr}} = \Theta(\frac{1}{H})$ & $\alpha = 0$ and SIO with $\beta = 0$ & k, such that the following holds. Any (possibly randomized) algorithm requires $\Omega(|\mathcal{A}|^H)$ total query cost to find (with probability at least $1/2$ over the randomness of the algorithm) a policy π satisfying*

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

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 $\mathrm{M}_{\mathrm{test}}$ 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 $\xi_{\mathrm{r}} = 0$, $\xi_{\mathrm{tr}} = \Theta(\frac{1}{H})$ & $\alpha = 0$ and SIO with $\beta = 0$ & k, such that the following holds. If a (possibly randomized) algorithm at test time can identify π satisfying*

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

in $o(|\mathcal{A}|H^k)$ queries, with probability at least $1/2$ over the selection of $\mathrm{M}_{\mathrm{test}}$ (and the randomness of the algorithm), then this algorithm must have required $\Omega(|\mathcal{A}|^H)$ 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 [46, 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.

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 ... H − 1} to A
for t ∈ {0, 1 ... H − 1} do
    for i ∈ {1, 2 ... n} do
        Sample M_i ∼ D
        for a ∈ A do
            Begin a new episode in M_i at s_0
            if t > 0 then Execute action sequence {π(t')}_{0≤t'<t} to arrive at s_t end if
            Take action a to arrive at s' = T_{M_i}(s_t, a) and receive R_{M_i}(s_t, a)
            Query V̂ to obtain V̂^{s'}_{M_i} and store Q_{i,a} = R_{M_i}(s_t, a) + V̂^{s'}_{M_i}
        end for
    end for
    Store a_t ∈ argmax_{a'∈A}{(1/n) Σ_{i=1}^{n} Q_{i,a'}} and define π(t) = a_t
end for
return π
```

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 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 $q_{\mathcal{D}}, |\mathcal{A}|, H$.

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, 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.

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

[30] 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.

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

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

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

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

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

[42] 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.

[43] 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.

[44] 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.

[45] 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.

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

[47] 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.

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

[49] 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.