One Solution is Not All You Need: Few-Shot Extrapolation via Structured MaxEnt RL

Anonymous Authors[1]

Abstract

While reinforcement learning algorithms can learn effective policies for complex tasks, these policies are often brittle to even minor task variations, especially when variations are not explicitly provided during training. One natural approach to this problem is to train agents with manually specified variation in the training task or environment. However, this may be infeasible in practical situations. The key insight of this work is that learning *diverse* behaviors for accomplishing a task can directly lead to behavior that generalizes to varying environments, without needing to perform explicit perturbations during training. By identifying multiple solutions for the task in a single environment during training, our approach can generalize to new situations by abandoning solutions that are no longer effective and adopting those that are. We theoretically characterize a robustness set of environments that arises from our algorithm and empirically find that our diversity-driven approach can extrapolate to various changes in the environment and task.

1 Introduction

Deep reinforcement learning (RL) algorithms have demonstrated promising results on a variety of complex tasks, such as robotic manipulation (Levine et al., 2016; Gu et al., 2017) and strategy games (Mnih et al., 2013; Silver et al., 2017). Yet, these reinforcement learning agents are typically trained in just one environment, leading to performant but narrowly-specialized policies — policies that are optimal under the training conditions, but brittle to even small environment variations (Zhang et al., 2018). A natural approach to resolving this issue is to simply train the agent on a distribution of environments that correspond to variations of the training environment (Cobbe et al., 2018; Farebrother et al., 2018; Igl et al., 2019; Rajeswaran et al., 2016). These methods assume access to a set of user-specified training environments that capture the properties of the situations that the trained agent will encounter during evaluation. However, this assumption places a significant burden on the user to hand-specify all degrees of variation, or may produce poor generalization along the axes that are not varied sufficiently (Zhang et al., 2018).

One way of resolving this problem is to design algorithms that can automatically construct many variants of its training environment and optimize a policy over these variants. One can do so, for example, by training an adversary to perturb the agent (Pinto et al., 2017; Pattanaik et al., 2018). While promising, adversarial optimizations can be brittle, overly pessimistic about the test distribution, and compromise performance.

In contrast to both generalization and robustness approaches, humans do not need to practice a task under explicit perturbations in order to adapt to new situations. As a concrete example, consider the task of navigating through a forest with multiple possible paths. Traditional RL approaches may optimize for and memorize the shortest possible path, whereas a person will encounter, *and remember* many different paths during the learning process, including suboptimal paths that still reach the goal. While a single optimal policy would fail if the shortest path becomes unavailable, a repertoire of diverse policies would be robust even when a particular path is no longer successful. Concretely, practicing and remembering diverse solutions to a task can naturally lead to robustness. In this work, we consider how we might encourage reinforcement learning agents to do the same – learning a breadth of solutions to a task and remembering those solutions such that they can adaptively switch to a new solution when faced with a new environment.

The key contribution of this work is a framework for policy robustness by optimizing for diversity. Rather than training a single policy to be robust across a distribution of environments, we learn multiple policies such that these behaviors are *collectively* robust to a new distribution over MDPs. Critically, our approach can be used with only a *single training environment*, rather than requiring access to the entire set of

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

environments over which we wish to generalize. We theoretically characterize the set of environments over which we expect the policies learned by our method to generalize, and empirically find that our approach can learn policies that extrapolate over a variety of aspects of the environment, while also outperforming prior standard and robust reinforcement learning methods.

2 Preliminaries

The goal in a reinforcement learning problem is to optimize cumulative discounted reward in a Markov decision process (MDP) $\mathcal{M}$, defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(s_{t+1}|s_t, a_t)$ provides the transition dynamics, $\mathcal{R}(s_t, a_t)$ is a reward function, γ is a discount factor, and μ is an initial state distribution. A policy π defines a distribution over actions conditioned on the state, $\pi(a_t|s_t)$. Given a policy π, the probability density function of a particular trajectory $\tau = \{s_i, a_i, \}_{i=1}^{T}$ under policy π can be factorized as follows:

$$p(\tau) = \mu(s_0) \cdot \Pi_{t=0}^{T} \pi(a_t|s_t) \cdot \mathcal{P}(s_{t+1}|s_t, a_t).$$

The expected discounted sum of rewards of a policy π is is given by: $R_{\mathcal{M}}(\pi) = \mathbb{E}_{\tau\sim\pi}\left[R(\tau)\right] = \sum_t \gamma^t \mathcal{R}(s_t, a_t)$. The optimal policy $\pi^*_{\mathcal{M}}$ maximizes the return, $R_{\mathcal{M}}(\pi)$: $\pi^*_{\mathcal{M}} = \arg\max_\pi R_{\mathcal{M}}(\pi)$.

Latent-Conditioned Policies. In this work, we will consider policies conditioned on a latent variable. A latent-conditioned policy is described as $\pi(a|s, z)$ and is conditioned on a latent variable $z \in \mathbb{R}^d$. The latent variable z is drawn from a known distribution $z \sim p(Z)$. The probability of observing a trajectory τ under a latent-conditioned policy is $p(\tau) = \int_z p(\tau|z)p(z)$, where

$$p(\tau|z) = \mu(s_0) \cdot \Pi_{t=0}^{T} \pi(a_t|s_t, z) \cdot \mathcal{P}(s_{t+1}|s_t, a_t).$$

Mutual-Information in RL. In this work, we will maximize the mutual information between trajectories and latent variables. Estimating this quantity is difficult because computing marginal distributions over all possible trajectories, by integrating out z, is intractable. We can instead maximize a lower bound on the objective which consists of summing the mutual information between each state s_t in a trajectory τ and the latent variable z. It has been shown that a sum of the mutual information between states in τ, $s_1, \cdots, s_T$, and the latent variable z lower bounds the mutual information $I(\tau, z)$ (Jabri et al., 2019). Formally, $I(\tau; z) \geq \sum_{t=1}^{T} I(s_t; z)$.

Finally, we can lower-bound the mutual information between states and latent variables, as $I(S; Z) \geq \mathbb{E}_{z\sim p(z), s\sim\pi(z)}[\log q_\phi(z|s)] - \mathbb{E}_{z\sim p(z)}[\log p(z)]$ (Eysenbach et al., 2018), where the posterior $p(z|s)$ can be approximated with a learned discriminator $q_\phi(z|s)$.

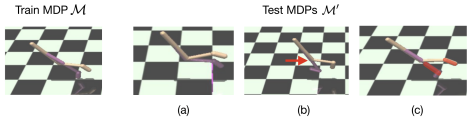

Figure 1: We evaluate SMERL on 3 types of environment perturbations: (a) the presence of an obstacle, (b) a force applied to one of the joints, and (c) motor failure at a subset of the joints.

3 Problem Statement: Few-Shot Robustness

In this paper, we aim to learn policies on a single training MDP that can generalize to perturbations of this MDP. In this section, we formalize this intuitive goal into a concrete problem statement that we call "few-shot robustness." During training, the algorithm collects samples from the (single) training MDP, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu)$. At test time, the agent is placed in a new test MDP $\mathcal{M}' = (\mathcal{S}, \mathcal{A}, \mathcal{P}', \mathcal{R}', \gamma, \mu)$, which belongs to a test set of MDPs S_{test}. Each MDP in this test set has identical state and action spaces as $\mathcal{M}$, but may have a different reward and transition function (see Figure 1). In Appendix A, we formally define the nature of the changes from training time to test time, which are guided by practical problems of interest, such as the navigation example described in Section 1. In the test MDP, the agent must acquire a policy that is optimal after only a handful of trials. Concretely, we refer to this protocol as *few-shot robustness*, where a trained agent is provided a budget of k episodes of interaction with the test MDP and must return a policy to be evaluated in this MDP. The final policy is evaluated in terms of its expected return in $\mathcal{M}'$.

4 Structured Maximum Entropy Reinforcement Learning

In this section, we present our approach for addressing the few-shot robustness problem defined in Section 3. We first present a concrete optimization problem that optimizes for few-shot robustness, then discuss how to transform this objective into a tractable form, and finally present a practical algorithm. Our algorithm, Structured Maximum Entropy Reinforcement Learning (SMERL), optimizes the approximate objective on a single training MDP.

4.1 Optimization with Multiple Policies

Our goal is to be able to learn policies on a single MDP that can achieve (near-)optimal return when executed on a test MDP in the set S_{test}. In order to maximize return on multiple possible test MDPs, we seek to learn a continuous (infinite) subspace or discrete (finite) subset of policies, which we denote as $\bar{\Pi}$. Then, given an MDP $\mathcal{M}' \in S_{\text{test}}$, we select the policy $\pi \in \bar{\Pi}$ that maximizes return $R(\mathcal{M}')$ on the test MDP. We wish to learn $\bar{\Pi}$ such that for any possible test MDP $\mathcal{M}' \in S_{\text{test}}$, there is always an effective policy $\pi \in \bar{\Pi}$.

Concretely, this gives rise to our formal training objective:

$$\Pi^* = \arg\max_{\bar{\Pi} \subset \Pi} \left[\sum_{\mathcal{M}' \in S_{\text{test}}} \left(\max_{\pi \in \bar{\Pi}} R_{\mathcal{M}'}(\pi) \right) \right]. \quad (1)$$

Our approach for maximizing the objective in Equation 1 is based on two insights. First, we represent the set $\bar{\Pi}$ using a latent variable policy $\pi(a|s,z)$. Such latent-conditioned policies can express multi-modal distributions. The latent variable can index different policies, making it possible to represent multiple behaviors with a single object. Second, we can produce diverse solutions to a task by encouraging the trajectories of different latent variables z to be distinct while still solving the task. An agent with a repertoire of such *distinct* latent policies can adopt a slightly sub-optimal solution if an optimal policy is no longer viable, or highly sub-optimal, in a test MDP. Concretely, we aim to maximize expected return while also producing unique trajectory distributions.

To encourage distinct trajectories for distinct z values, we introduce a diversity-inducing objective that encourages high mutual information between $p(Z)$, and the marginal trajectory distribution for the latent-conditioned policy $\pi(a|s,z)$. We optimize this objective subject to the constraint that each policy achieves return in $\mathcal{M}$ that is close to the optimal return. This optimization problem is:

$$\max_{\theta} \; I(\tau, z) \;\; \text{s.t.} \;\; \forall z, R_{\mathcal{M}}\left(\pi_\theta(\cdot|s,z)\right) \geq R_{\mathcal{M}}(\pi^*_{\mathcal{M}}(\cdot|s)) - \varepsilon, \quad (2)$$

where $\pi_\theta = \pi_\theta(\cdot|s;z)$, $\epsilon > 0$, and θ^* are parameters which maximize the objective. The constrained optimization in Equation 2 aims at learning a space of policies, indexed by the latent variable z, such that the set $\bar{\Pi} = \{\pi(\cdot|\cdot,z)|z \sim p(z)\}$ covers the space of possible policies $\pi(a|s)$ that induce near-optimal, long-term discounted return on the training MDP $\mathcal{M}$. The mutual information objective $I(\tau,z)$ enforces diversity among policies in $\bar{\Pi}$, but only when these policies are close to optimal.

4.2 The SMERL Optimization Problem

In order to tractably solve the optimization problem 2, we lower-bound the mutual information $I(\tau,z)$ by a sum of mutual information terms over individual states appearing in the trajectory τ, as discussed in Section 2. We then obtain the following surrogate, tractable optimization problem:

$$\max_{\theta} \sum_{t=1}^{T} I(s_t; z) \;\; \text{s.t.} \;\; \forall z, R_{\mathcal{M}}\left(\pi_\theta\right) \geq R_{\mathcal{M}}(\pi^*_{\mathcal{M}}) - \varepsilon. \quad (3)$$

Following the argument from (Eysenbach et al., 2018), we compute an unsupervised reward function from the mutual information between states and latent variables as $\tilde{r}(s,a) = \log q_\phi(z|s) - \log p(z)$, where $q_\phi(z|s)$ is a learned discriminator. Since the term $H(Z)$ encourages the distribution over the latent variables to have high entropy, we fix $p(z)$ to be uniform.

In order to satisfy the constraint in Equation 3 that $I(S;Z)$ is maximized *only when* the latent-conditioned policy achieves return $R_{\mathcal{M}}\left(\pi_\theta\right) \geq R_{\mathcal{M}}(\pi^*_{\mathcal{M}}) - \epsilon$, we only optimize the unsupervised reward when the environment return is within a pre-defined ϵ distance from the optimal return.

To this end, we optimize the sum of two quantities: **(1)** the discounted return obtained by executing a latent-conditioned policy in the MDP, $R_{\mathcal{M}}(\pi_\theta(\cdot|s,z))$, and **(2)** the discounted sum of unsupervised rewards $\sum_t \gamma^t \tilde{r}_t$, only if the policy's return satisfies the condition specified in Equation 3. Combining these components leads to the following optimization in practice ($\mathbb{1}_{[\cdot]}$ is the indicator function, $\alpha > 0$):

$$\max_{\theta} \mathbb{E}_{z \sim p(z)} \left[R_{\mathcal{M}}(\pi_\theta) + \alpha \mathbb{1}_{[R_{\mathcal{M}}(\pi_\theta) \geq R_{\mathcal{M}}(\pi^*_{\mathcal{M}}) - \varepsilon]} \sum_t \gamma^t \tilde{r}(s_t, a_t) \right] \quad (4)$$

4.3 Practical Algorithm

We implement SMERL using soft actor-critic (SAC) (Haarnoja et al., 2018), but with a latent variable maximum entropy policy $\pi_\theta(a|s,z)$. The set of latent variables is chosen to be a fixed discrete set, Z, and we set $p(z)$ to be a categorical distribution over this set. At the beginning of each episode, a latent variable z is sampled from $p(Z)$ and the policy $\pi_\theta(\cdot|\cdot,z)$ is used to sample a trajectory. The transitions obtained, as well as the latent variable z, are stored in a replay buffer. When sampling states from the replay buffer, we compute the reward to optimize with SAC according to Equation 3 from Section 4.2:

$$r_{\text{SMERL}}(s_t,a_t) = r(s_t,a_t) + \alpha \mathbb{1}_{R_{\mathcal{M}}(\pi_\theta) \geq R_{\mathcal{M}}(\pi^*_{\mathcal{M}}) - \varepsilon} \tilde{r}(s_t,a_t). \quad (5)$$

For all states sampled from the replay buffer, we optimize the reward obtained from the environment r. For states in trajectories which achieve near-optimal return, the agent also receives unsupervised reward $\tilde{r}$, which is higher-valued when the agent visits states that are easy to discriminate, as measured by the likelihood of a discriminator $q_\phi(z|s)$. The discriminator is trained to infer the latent variable z from the states visited when executing that latent-conditioned policy. In order to measure whether $R_{\mathcal{M}}\left(\pi_\theta\right) \geq R_{\mathcal{M}}(\pi^*_{\mathcal{M}}) - \varepsilon$, we first train a baseline SAC agent on the environment, and treat the maximum return achieved by the trained SAC agent as the optimal return $R_{\mathcal{M}}(\pi^*_{\mathcal{M}})$.

The full training algorithm is described in Algorithm 1. Following the few-shot robustness evaluation protocol, given a budget of K episodes, each latent variable policy $\pi_\theta(\cdot|s,z)$ is executed in a test MDP $\mathcal{M}'$ for 1 episode. The policy which achieves the maximum sampled return is returned (see Algorithm 2).

Algorithm 1 SMERL: Training in training MDP $\mathcal{M}$

while *not converged* **do**
  Sample latent $z \sim p(z)$ and initial state $s_0 \sim \mu$.
  for $t \leftarrow 1$ **to** *steps_per_episode* **do**
    Sample action $a_t \sim \pi_\theta(\cdot|s_t, z_t)$.
    Step environment: $r_t, s_{t+1} \sim \mathcal{P}(r_t, s_{t+1}|s_t, a_t)$.
    Compute $q_\phi(z|s_{t+1})$ with discriminator.
    Let $\tilde{r}_t = \log q_\phi(z|s_{t+1}) - \log p(z)$.
  end for
  Compute $R_\mathcal{M}(\pi_\theta) = \sum_t r_t$.
  for $t \leftarrow 1$ **to** *steps_per_episode* **do**
    Compute reward r_{SMERL} according to Eq 5.
    Update θ to maximize r_{SMERL} with SAC.
    Update ϕ to maximize $\log q_\phi(z|s_t)$ with SGD.
  end for
end while

Algorithm 2 SMERL: Few-shot robustness evaluation in test MDP $\mathcal{M}'$

$R_{\text{MAX}} \leftarrow -\infty$
for $i \in \{1, 2, ..., K\}$ **do**
  Rollout policy $\pi_\theta(\cdot|s, z_i)$ in MDP $\mathcal{M}'$ for 1 episode and compute $R_{\mathcal{M}'}(\pi_\theta)$.
  $R_{\text{MAX}} \leftarrow \max(R_{\text{MAX}}, R_{\mathcal{M}'}(\pi_\theta))$
  $\pi_{\text{best}} \leftarrow \pi_\theta(\cdot|s, z_i)$
end for
Return π_{best}

5 Experimental Evaluation

The goal of our experimental evaluation is to test the central hypothesis of our work: does structured diversity-driven learning lead to policies that generalize to new MDPs? We also compare the performance of our method relative to prior approaches for generalizable and robust policy learning. To this end, we conduct experiments in two continuous control environments using the MuJoCo physics engine (Todorov et al., 2012): HalfCheetah-Goal and Walker2d-Velocity.

In HalfCheetah-Goal, the task is to navigate to a target goal location. In Walker-Velocity, the task is to move forward at a particular velocity. We perform evaluation in three types of test conditions: (1) an obstacle is present on the path to the goal, (2) a force is applied to one of the joints at a pre-specified small time interval (t_1, t_2) from time step $t_1 = 10$ to time step $t_2 = 15$, and (3) a subset of the motors fail for time intervals of varying lengths. For each test environment, we vary the amount of perturbation to measure the degree to which different algorithms are robust. We vary the height of the obstacle in (1), the magnitude of the force in (2), and the time interval length for which motor failure occurs in (3).

5.1 Can SMERL Quickly Generalize To Extrapolated Environments?

We study whether these policies are robust to various test conditions in continuous-control problems. We compare SMERL to standard maximum-entropy RL (SAC), an approach that learns multiple diverse policies but does not maximize a reward signal from the environment (DIAYN), and a robust RL method (RARL). By comparing to SAC and DIAYN, we aim to test the importance of learning diverse policies and of ensuring near-optimal return, for achieving robustness. By comparing to RARL, we aim to understand how SMERL compares to adversarial optimization approaches.

We report results in Figure 2. On all types of test perturbations, we find that SMERL is consistently as robust or more robust than SAC and DIAYN. When the perturbation amount is small, the performance of SAC and SMERL is approximately the same. As the perturbation magnitude increases (e.g. obstacle height, force magnitude, or agent's mass), the performance of SAC quickly drops while SMERL is still able to achieve near-optimal return. With large perturbation magnitudes, all methods fail to complete the task on most test environments, as we expect. RARL is robust to the force and motor failure perturbations but cannot solve the task when an obstacle is present. Since DIAYN is trained independently of task reward, it only occasionally solves the task and otherwise produces policies that perform structured diverse behavior but don't achieve near-optimal return. SMERL balances the task reward and DIAYN reward to achieve few-shot robustness, since it only adds the DIAYN reward when the latent policies are near-optimal.

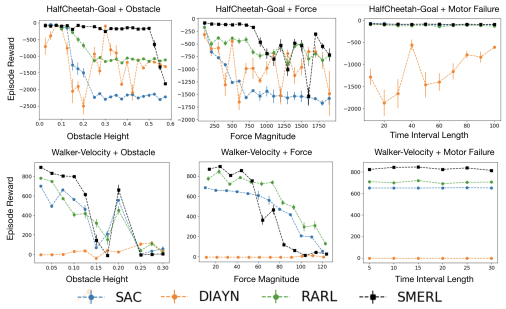

Figure 2: Figure comparing the robustness of SAC, DIAYN, RARL, and SMERL to 3 types of perturbations, on HalfCheetah-Goal and Walker-Velocity. SMERL is more consistently robust to environment perturbations than other maximum entropy, diversity seeking, and robust RL methods.

6 Conclusion

In this paper, we present a robust RL algorithm, SMERL, for learning RL policies that can extrapolate to out-of-distribution test conditions with only a small number of trials. The core idea underlying SMERL is that we can learn a set of policies that finds multiple diverse solutions to a single task.

References

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Farebrother, J., Machado, M. C., and Bowling, M. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396. IEEE, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Igl, M., Ciosek, K., Li, Y., Tschiatschek, S., Zhang, C., Devlin, S., and Hofmann, K. Generalization in reinforcement learning with selective noise injection and information bottleneck. In *Advances in Neural Information Processing Systems*, pp. 13956–13968, 2019.

Jabri, A., Hsu, K., Gupta, A., Eysenbach, B., Levine, S., and Finn, C. Unsupervised curricula for visual meta-reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 10519–10530, 2019.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., and Chowdhary, G. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2040–2042. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2817–2826. JMLR. org, 2017.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Zhang, C., Vinyals, O., Munos, R., and Bengio, S. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

Appendices

A Analysis of SMERL

We now provide a theoretical analysis of SMERL. We show how the tractable objective shown in Equation 4 can be derived out of the optimization problem in Equation 1 for particular choices of robustness sets S_{test} of MDPs. Our analysis in divided into three parts. First, we define our choice of MDP robustness set. We then provide a reduction of this set over MDPs to a robustness set over policies. Finally, we show that an optimal solution of our tractable objective is indeed optimal for this policy robustness set under certain assumptions.

A.1 Robustness Sets of MDPs and Policies

Following our problem definition in Section 2, our robustness sets will be defined over MDPs $\mathcal{M}'$, which correspond to versions of the training MDP $\mathcal{M}$ with altered reward or dynamics. For the purpose of this discussion, we limit ourselves to discrete state and action spaces. Drawing inspiration from the navigation example in Section 1, we now define our choice of robustness set, which we will later connect to the set of MDPs to which we can expect SMERL to generalize. Hence, we define the MDP robustness set as:

Definition 1. *Given a training MDP $\mathcal{M}$ and $\epsilon > 0$, the MDP robustness set, $S_{\mathcal{M},\epsilon}$, is the set of all MDPs $\mathcal{M}'$ which satisfy two properties:*

$(1) R_{\mathcal{M}}(\pi^*_{\mathcal{M}}) - R_{\mathcal{M}}(\pi^*_{\mathcal{M}'}) \leq \epsilon$

(2) *The trajectory distribution of $\pi^*_{\mathcal{M}'}$ is the same on $\mathcal{M}$ and $\mathcal{M}'$.*

Intuitively, the set $S_{\mathcal{M},\epsilon}$ consists of all MDPs for which the optimal policy $\pi^*_{\mathcal{M}'}$ achieves a return *on the training MDP* that is close to the return achieved by its own optimal policy, $\pi^*_{\mathcal{M}}$. Additionally, the optimal policy of $\mathcal{M}'$ must produce the same trajectory distribution in $\mathcal{M}'$ as in $\mathcal{M}$. These properties are motivated by practical situations, where a perturbation to a training MDP, such as an obstacle blocking an agent's path, allows different policies in the training MDP to be optimal on the test MDP. This perturbation creates a test MDP $\mathcal{M}'$ whose optimal policy achieves return close to the optimal policy of $\mathcal{M}$ since it takes only a slightly longer path to the goal, and that path is traversed by the same policy in the original MDP $\mathcal{M}$. Given this intuition, the MDP robustness set $S_{\mathcal{M},\epsilon}$ will be the set that we use for the test set of MDPs S_{test} in Equation 1 in our upcoming derivation.

While we wish to generalize to MDPs in the MDP robustness set, in our training protocol an RL agent has access to only a single training MDP. It is not possible directly optimize over the set of test MDPs, and SMERL instead optimizes over policies in the training MDP. In order to analyze the connection between the policies learned by SMERL and robustness to test MDPs, we consider a related robustness set, defined in terms of sub-optimal policies on the training MDP:

Definition 2. *Given a training MDP $\mathcal{M}$ and $\epsilon > 0$, the policy robustness set, $S_{\pi^*_{\mathcal{M}},\epsilon}$ is defined as*

$$S_{\pi^*_{\mathcal{M}},\epsilon} = \{\pi \mid R_{\mathcal{M}}(\pi^*_{\mathcal{M}}) - R_{\mathcal{M}}(\pi) \leq \epsilon \text{ and } \pi \text{ is a deterministic policy } \}.$$

The policy robustness set consists of all policies which achieve return close to the optimal return of the training MDP. Since the optimal policies of the MDP robustness set also satisfy this condition, intuitively, $S_{\pi^*_{\mathcal{M}},\epsilon}$ encompasses the optimal policies for MDPs from $S_{\mathcal{M},\epsilon}$.

Next, we formalize this intuition, and in Sec. A.3 show how this convenient relationship can replace the optimization over $S_{\mathcal{M},\epsilon}$ in Eq. 1 with an optimization over policies, as performed by SMERL.

A.2 Connecting MDP Robustness Sets with Policy Robustness Sets

Every policy in $S_{\pi^*_{\mathcal{M}},\epsilon}$ is optimal for some MDP in $S_{\mathcal{M},\epsilon}$. Thus, if an agent can learn all policies in $S_{\pi^*_{\mathcal{M}},\epsilon}$, then we can guarantee the ability to perform optimally in each and every possible MDP that can be encountered at test time. In order to formally prove this intuition, we provide a set of two containment results. Proofs can be found in Appendix B.

Proposition 1. *For each MDP $\mathcal{M}'$ in the MDP robustness set $S_{\mathcal{M},\epsilon}$, $\pi^*_{\mathcal{M}'}$ exists in the policy robustness set $S_{\pi^*_{\mathcal{M}},\epsilon}$.*

Proposition 2. *Given an MDP $\mathcal{M}$ and each policy π in the policy robustness set $S_{\pi^*_{\mathcal{M}},\epsilon}$, there exists an MDP $\mathcal{M}' = (\mathcal{S}, \mathcal{A}, \bar{\mathcal{P}}, \bar{\mathcal{R}}, \gamma, \mu)$ such that $\mathcal{M}' \in S_{\mathcal{M},\epsilon}$ and $\pi = \pi^*_{\mathcal{M}'}$.*

We next use this connection between $S_{\pi^*_{\mathcal{M}},\epsilon}$ an $S_{\mathcal{M},\epsilon}$ to verify that SMERL indeed finds a solution to our formal training objective (Equation 1).

A.3 Optimizing the Robustness Objective

Now that we have shown that any policy in $S_{\pi^*_{\mathcal{M}},\epsilon}$ is optimal in some MDP in $S_{\mathcal{M},\epsilon}$, we now show how this relation can be utilized to simplify the objective in Equation 1. Finally, we show that this simplification naturally leads to the trajectory-centric mutual information objective. We first introduce a modified training objective below in Equation 6, and then show in Proposition 3 that under some mild conditions, the solution obtained by optimizing Equation 6 matches the solution obtained by solving Equation 1:

$$\Pi^* = \arg\max_{\bar{\Pi} \subset \Pi} \min_{\hat{\pi} \in S_{\pi^*_{\mathcal{M}},\epsilon}} [\max_{\pi \in \bar{\Pi}} \mathbb{E}_{\tau \sim \hat{\pi}} \log p(\tau|\pi)]. \quad (6)$$

Proofs from this section can be found in Appendix B.

Proposition 3. *The solution to the objective in Equation 1 is the same as the solution to the objective in Equation 6 when* $S_{test} = S_{\mathcal{M},\epsilon}$.

Finally, we now show that the set of policies obtained by optimizing Equation 6 is the same as the set of solutions obtained by the SMERL mutual information objective (Equation 2).

Proposition 4. *[Informal] With usual notation and for a sufficiently large number of latent variables, the set of policies* Π^* *that result from solving the optimization problem in Equation 6 is equivalent to the set of policies* $\pi_{\theta^*,z}$ *that result from solving the optimization problem in Equation 2.*

A more formal theorem statement and its proof are in Appendix B. Propositions 3 and 4 connect the solutions to the optimization problems in Equation 1 and Equation 2, for a specific instantiation of S_{test}. Our results in this section suggest that the general paradigm of diversity-driven learning is effective for robustness when the test MDPs satisfy certain properties. In Section 5, we will empirically measure SMERL's robustness when optimizing the SMERL objective on practical problems where the test perturbations satisfy these conditions.

B Proofs

Proposition 1. *For each MDP* $\mathcal{M}'$ *in the MDP robustness set* $S_{\mathcal{M},\epsilon}$, $\pi^*_{\mathcal{M}'}$ *exists in the policy robustness set* $S_{\pi^*_{\mathcal{M}},\epsilon}$.

Proof. This result follows by the definition of the set $S_{\mathcal{M},\epsilon}$. □

Proposition 2. *Given an MDP* $\mathcal{M}$ *and each policy* π *in the policy robustness set* $S_{\pi^*_{\mathcal{M}},\epsilon}$, *there exists an MDP* $\mathcal{M}' = (\mathcal{S}, \mathcal{A}, \bar{\mathcal{P}}, \bar{\mathcal{R}}, \gamma, \mu)$ *such that* $\mathcal{M}' \in S_{\mathcal{M},\epsilon}$ *and* $\pi = \pi^*_{\mathcal{M}'}$.

Proof. This argument can be shown by first noting that the value of any policy, π, in an MDP can be written as, $Q^\pi(P) = (I - \gamma P^\pi)^{-1}[R]$. Now, for any given policy $\pi \in S_{\pi^*_{\mathcal{M}},\epsilon}$, we show that we can modify the dynamics to P' such that $Q^\pi(P') \geq Q^{\pi'}(P')$, for all other policies π'. Such a dynamics P' always exists for any policy π, since for any optimal policy π' in the original MDP with transition dynamics P, we can re-write $P \cdot \pi$ as $P \cdot \pi' = P \cdot \pi' \cdot \frac{\pi}{\pi'}$ and by modifying the transition dynamics, as $P' = P \cdot \frac{\pi}{\pi'}$. With this transformation, π' is optimal in this modified MDP with dynamics P'. □

Proposition 3. *The solution to the objective in Equation 1 is the same as the solution to the objective in Equation 6 when* $S_{test} = S_{\mathcal{M},\epsilon}$.

Proof. We first note that for any optimal policy $\pi^*_{\mathcal{M}'}$ of an MDP $\mathcal{M}' \in S_{\mathcal{M},\epsilon}$, the trajectory distribution in the original MDP, $p_{\mathcal{M}}(\tau|\pi^*_{\mathcal{M}'})$, is the same as the trajectory distribution in the perturbed MDP, $\mathcal{M}'$, $p_{\mathcal{M}'}(\tau|\pi^*_{\mathcal{M}'})$. Now, we aim to learn a policy π that obtains the same trajectory distribution as $\pi^*_{\mathcal{M}'}$ in $\mathcal{M}'$, which is also the trajectory distribution of $\pi^*_{\mathcal{M}'}$ in $\mathcal{M}$. Further, we know that the policy $\pi^*_{\mathcal{M}'}$ is contained in the policy robustness set, , our learned policies, $\bar{\Pi}$, should be able to match the trajectory distribution of any policy $\pi' \in S_{\pi^*_{\mathcal{M}},\epsilon}$, thereby, ensuring that the trajectory distribution of at least one policy $\pi \in \bar{\Pi}$ is close to the trajectory distribution of π'. The objective in Equation 6 precisely uses this connection – it searches for a set of policies, $\bar{\Pi}$, such that at least one policy $\pi' \in \bar{\Pi}$ can closely match the trajectory distribution of a given policy $\pi^*_{\mathcal{M}'} \in S_{\pi^*_{\mathcal{M}},\epsilon}$, which also turns out to be the set of optimal policies for $\mathcal{M}' \in S_{\mathcal{M},\epsilon}$. Moreover, this trajectory matching can be performed directly in the original MDP, $\mathcal{M}$, since optimal policies for $\mathcal{M}'$ admit the same trajectory distribution in both $\mathcal{M}$ and $\mathcal{M}'$. □

Proposition 4. *[Informal] With usual notation and for a sufficiently large number of latent variables, the set of policies* Π^* *that result from solving the optimization problem in Equation 6 is equivalent to the set of policies* $\pi_{\theta^*,z}$ *that result from solving the optimization problem in Equation 2.*

Proof. If the subset $\bar{\Pi}$ were allowed to have infinite size, then the choice of $\bar{\Pi}$ which optimizes the optimization problem shown in Equation 6 is equal to $S_{\pi^*_{\mathcal{M}},\epsilon}$. Now, when the cardinality of the set is limited, we seek a set of policies, $\bar{\Pi}$ that can most efficiently cover the space

For the second optimization problem, satisfying the condition $\forall z \in \bar{Z}\ R_{\mathcal{M}}(\pi_z) \geq R_{\mathcal{M}}(\pi^*_{\mathcal{M}}) - \epsilon$ would give us a set of K policies $\{\pi_z | z \in \bar{Z}\}$, each of which lie in the set $S_{\pi^*_{\mathcal{M}},\epsilon}$. To ensure that each of the policies is deterministic, we minimize $H(\tau|Z)$. Since the transition function $\mathcal{P}$ is deterministic, if $H(\tau|Z)$ is minimized, then each π_z is deterministic. We also want the trajectories induced by the policies to be different from each other so that we get all the policies in the set $S_{\pi^*_{\mathcal{M}},\epsilon}$. This is achieved by maximizing the entropy of all the trajectories $H(\tau)$. Thus, the solution to the second optimization problem is also equal to the set $S_{\pi^*_{\mathcal{M}},\epsilon}$.

$$I(\tau; Z) = H(\tau) - H(\tau|Z) \tag{7}$$

□

C Experimental Setup

To estimate the optimal return value that SMERL requires, we train SAC on a single training environment using 3 seeds for 3 million environment steps. We then select a value of ϵ

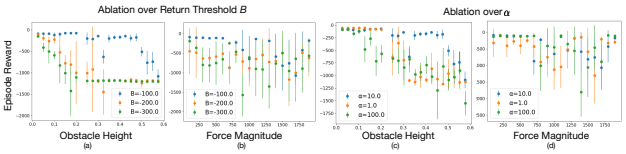

Figure 3: On HalfCheetah-Goal, we study the effects on performance when (a) varying B in obstacles test environments, (b) varying B in the force test environments, (c) varying α in the obstacles test environments, and (d) varying α in the force test environments. In (a) and (b), $\alpha = 1.0$, and in (c) and (d), $B = -100.0$. We plot the mean performance and standard error bars over 5 runs of the best latent-conditioned policy on each test environment. SMERL is more sensitive to hyperparameter settings in the obstacles test environments as compared to the force test environments.

for SMERL to set a return threshold above which the unsupervised reward is added. For SMERL, SAC, and DIAYN, we learn $|Z| = 5$ latent-conditioned policies, and follow our few-shot evaluation protocol described in Section 3 with budget $k = 5$. Specifically, we run every latent-conditioned policy once and select the policy which achieves the highest return on that environment; we then run the selected policies 5 times and compute the averaged performance. All results are reported using 3 random seeds. While the policies learned on the train MDP are stochastic, during evaluation, we select the mean action (SAC, DIAYN, and SMERL). This does not make the performance worse for any of the approaches.

C.1 Hyperparameter Sensitivity Analysis

On HalfCheetah-Goal, we examine the effect of varying B, the environment return threshold above which we add the unsupervised objective, and α, the weight by which the unsupervised reward is multiplied, on the evaluation performance of SMERL. We perform this hyperparameter study on two test environments: HalfCheetah-Goal with an obstacle and HalfCheetah-Goal with a force applied to one of the joints. We find that the robustness of SMERL is sensitive to the choice of α: $\alpha = 10.0$ results in policies that are more robust to the environment perturbations. In contrast, the relationship between the value of B and evaluation performance is inconsistent between the obstacles perturbation and the force perturbation test environments.