Robust Reinforcement Learning in Continuous Control Tasks with Uncertainty Set Regularization

Anonymous Author(s)

Abstract: Reinforcement learning (RL) is recognized as lacking generalization and robustness under environmental perturbations, which excessively restricts its application for real-world robotics. Prior work claimed that adding regularization to the value function is equivalent to learning a robust policy with uncertain transitions. Although the regularization-robustness transformation is appealing for its simplicity and efficiency, it is still lacking in continuous control tasks. In this paper, we propose a new regularizer named Uncertainty Set Regularizer (USR), by formulating the uncertainty set on the parametric space of the transition function. To deal with unknown uncertainty sets, we further propose a novel adversarial approach to generate them based on the value function. We evaluate USR on the Real-world Reinforcement Learning (RWRL) benchmark and the Unitree A1 Robot, demonstrating improvements in the robust performance for perturbed testing environments and sim-to-real scenarios.

Keywords: Reinforcement Learning, Robustness, Robotics

1 Introduction

Reinforcement Learning (RL) is a powerful algorithmic paradigm used to solve sequential decision-making problems and has resulted in great success in various types of environments, e.g., mastering the game of Go [1], playing computer games [2] and operating smart grids [3, 4, 5]. The majority of these successes rely on an implicit assumption that the testing environment is identical to the training environment. However, this assumption is too strong for most realistic problems, such as controlling a robot. In more detail, there are several situations where mismatches might appear between training and testing environments in robotics: (1) Parameter Perturbations indicates that a large number of environmental parameters, e.g. temperature, friction factor could fluctuate after deployment and thus deviate from the training environment [6]; (2) System Identification estimates a transition function from limited experience. This estimation is biased compared with the real-world model [7]; (3) Sim-to-Real learns a policy in a simulated environment and performs on real robots for reasons of safety and efficiency [8]. The difference between simulated and realistic environments renders sim-to-real a challenging task.

In this paper, we aim to solve the mismatch between training environment and testing environment as a robust RL problem which regards training environment and testing environment as candidates in the uncertainty set of all possible environments. Robust Markov Decision Processes (Robust MDPs) [9, 10] is a common framework for analyzing the robustness of RL algorithms. Compared with the vanilla MDPs with a single transition model \( P(s'|s, a) \), Robust MDPs consider an uncertainty set of transition models \( \mathcal{P} = \{P\} \) to better describe the uncertain environments. This formulation is general enough to cover various scenarios for robot learning problems mentioned above.
Robust RL aims to learn a robust policy under the worst-case scenario for all transition models $P \in \mathcal{P}$. If the transition model $P$ is viewed as an adversarial agent and the uncertainty set $\mathcal{P}$ is its action space, one can reformulate Robust RL as a zero-sum game [11]. In general, solving such a problem is NP-hard [10]. Derman et al. [12], however, adopted the Legendre-Fenchel transform to avoid excessive mini-max computations by converting the minimization over the transition model to a regularization on the value function. Furthermore, it enjoys an additional advantage that it provides more possibilities to design novel regularizers for different types of transition uncertainties. The complexity of these value-function-based regularizers increases with the size of the state space, which leads to a nontrivial extension to continuous control tasks with infinitely large state space. This directly motivates the work of this paper.

We now summarize the contributions of this paper: (1) the robustness-regularization duality method is extended to continuous control tasks in parametric space; (2) the Uncertainty Set Regularizer (USR) on existing RL frameworks is proposed for learning robust policies; (3) the value function is learnt through an adversarial uncertainty set when the actual uncertainty set is unknown in some scenarios; (4) the USR is evaluated on the Real-world Reinforcement Learning (RWRL) benchmark, showing improvements for robust performances in perturbed testing environments with unknown uncertainty sets; (5) the sim-to-real performance of USR is verified through realistic experiments on the Unitree A1 robot.

## 2 Preliminaries

**Robust MDPs.** The mathematical framework of Robust MDPs [9, 10] extends regular MDPs in order to deal with uncertainty about the transition function. A Robust MDP can be formulated as a 6-tuple $\langle S, A, \mathcal{P}, r, \mu_0, \gamma \rangle$, where $S, A$ stand for the state and action space respectively, and $r(s, a) : S \times A \rightarrow \mathbb{R}$ stands for the reward function. Let $\Delta_S$ and $\Delta_A$ be the probability measure on $S$ and $A$ respectively. The initial state is sampled from an initial distribution $\mu_0 \in \Delta_S$, and the future rewards are discounted by the discount factor $\gamma \in [0, 1]$. The most important concept in robust MDPs is the uncertainty set $\mathcal{P} = \{ P(s'|s, a) : S \times A \rightarrow \Delta_S \}$ that controls the variation of transition $P$, compared with the stationary transition $P$ in regular MDPs. Let $\Pi = \{ \pi(a|s) : S \rightarrow \Delta_A \}$ be the policy space; the objective of Robust RL can then be formulated as a minimax problem,

$$J^* = \max_{\pi \in \Pi} \min_{P \in \mathcal{P}} \mathbb{E}_{\pi, P} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right].$$  \hspace{1cm} (1)

**Robust Bellman equation.** While Wiesemann et al. [10] has proved NP-hardness of this minimax problem with an arbitrary uncertainty set, most recent studies [13, 9, 14, 11, 15, 12, 16, 17, 18] approximate it by assuming a rectangular structure on the uncertainty set, i.e., $\mathcal{P} = \times_{(s,a) \in S \times A} \mathcal{P}_{sa}$, where $\mathcal{P}_{sa} = \{ P_{sa}(s') : \Delta_S \}$ denotes the local uncertainty of the transition at $(s, a)$. In other words, the variation of transition is independent at every $(s, a)$ pair. Under the assumption of a rectangular uncertainty set, the robust action value $Q^\pi(s, a)$ under policy $\pi$ must satisfy the following robust version of the Bellman equation [19] such that

$$Q^\pi(s, a) = r(s, a) + \min_{P_{sa} \in \mathcal{P}_{sa}} \sum_{s', a'} P_{sa}(s') \pi(a'|s') Q^\pi(s', a') = r(s, a) + \min_{P_{sa} \in \mathcal{P}_{sa}} \gamma \sum_{s'} P_{sa}(s') V^\pi(s').$$  \hspace{1cm} (2)

Nilim and Ghaoui [13] have shown that a robust Bellman operator admits a unique fixed point of Equation 2, the robust action value $Q^\pi(s, a)$.

**Robustness-regularization duality.** Solving the minimization problem in the RHS of Equation 2 can be further simplified by the Legendre-Fenchel transform [20]. For a function $f : X \rightarrow \mathbb{R}$, its convex conjugate is $f^*(x^*) := \text{sup} \{ x^* \cdot x - f(x) : x \in X \}$. Define $\delta_{P_{sa}}(P_{sa}) = 0$ if $P_{sa} \in \mathcal{P}_{sa}$ and $+\infty$ otherwise. Equation 2 can be transformed to its convex conjugate (refer to Derman et al. [12] for detailed derivation),

$$Q^\pi(s, a) = r(s, a) + \min_{P_{sa}} \gamma \sum_{s'} P_{sa}(s') V^\pi(s') + \delta_{\mathcal{P}_{sa}}(P_{sa}) = r(s, a) - \delta_{\mathcal{P}_{sa}}(-V^\pi(\cdot)).$$  \hspace{1cm} (3)
The transformation implies that the robustness condition on transitions can be equivalently expressed as a regularization constraint on the value function, referred to as the robustness-regularization duality. The duality can extensively reduce the cost of solving the minimization problem over infinite transition choices and thus is widely studied in the robust reinforcement learning research community [21, 22, 23].

As a special case, Derman et al. [12] considered a \( L_2 \) norm uncertainty set on transitions, i.e.,

\[
\mathbb{P}_{\text{sa}} = \{ P_{\text{sa}} + \alpha P_{\text{sa}} : \| P_{\text{sa}} \|_2 \leq 1 \},
\]

where \( P_{\text{sa}} \) is usually called the nominal transition model. It could represent prior knowledge on the transition model or a numerical value of the training environment. The uncertainty set denotes that the transition model is allowed to fluctuate around the nominal model with some degree \( \alpha \). Therefore, the corresponding Bellman equation in Equation 3 becomes \( Q^\pi(s, a) = r(s, a) + \gamma \sum_{s'} P_{\text{sa}}(s') V^\pi(s') - \alpha \| V^\pi(\cdot) \|_2 \). Similarly, the \( L_1 \) norm has also been used as uncertainty set on transitions [16], i.e., \( \mathbb{P}_{\text{sa}} = \{ P_{\text{sa}} + \alpha P_{\text{sa}} : \| P_{\text{sa}} \|_1 \leq 1 \}, \) and the Bellman equation becomes the form such that \( Q^\pi(s, a) = r(s, a) + \gamma \sum_{s'} P_{\text{sa}}(s') V^\pi(s') - \alpha \max_{s'} |V^\pi(s')| \). This robustness-regularization duality works well with finite state spaces but it’s not trivial to directly extend it to infinite state spaces since both \( \| V^\pi(\cdot) \|_2 \) and \( \max_{s'} |V^\pi(s')| \) regularizers need calculations on the infinite-dimensional vector \( V^\pi(\cdot) \). In this work, we extend this concept to continuous state space which is a critical characteristic in robotics, which we believe is a novelty with significance.

3 Uncertainty Set Regularized Robust Reinforcement Learning

Having introduced the robustness-regularization duality and the difficulties regarding its extension to continuous state space in Section 2, here, we will first present a novel extension to continuous state space with the uncertainty set defined on the parametric space of the transition function. We will then utilize this extension to derive a robust policy evaluation method that can be directly plugged into existing RL algorithms. Furthermore, to deal with the unknown uncertainty set, we propose an adversarial uncertainty set and visualize it in a simple moving-to-target task.

3.1 Uncertainty Set Regularized Robust Bellman Equation (USR-RBE)

For environments with continuous state spaces, the transition model \( P(s'|s, a; w) \) is usually represented as a parametric function \( P(s'|s, a; w) \), where \( w \) denotes the parameters of the transition function. Instead of defining the uncertainty set on the distribution space, we directly impose a perturbation on \( w \) within a set \( \Omega_w \). Consequently, the robust objective function (Equation 1) becomes

\[
J^* = \max_{\pi \in \Pi} \min_{w \in \Omega_w} \mathbb{E}_{\pi, P(s'|s, a; w)} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right].
\]

We further assume the parameter \( w \) fluctuates around a nominal parameter \( \tilde{w} \), such that \( w = \tilde{w} + \bar{w} \), with \( \bar{w} \) being a fixed parameter and \( \bar{w} \in \Omega_{\bar{w}} = \{ w - \bar{w} | w \in \Omega_w \} \) being the perturbation part. Inspired by Equation 3, we can derive a robust Bellman equation on the parametric space for continuous control problems as shown in Proposition 3.1.

**Proposition 3.1 (Uncertainty Set Regularized Robust Bellman Equation)** Suppose the uncertainty set of \( w \) is \( \Omega_w \) (i.e., the uncertainty set of \( \tilde{w} = w - \bar{w} \) is \( \Omega_{\tilde{w}} \)), the robust Bellman equation on parametric space can be represented as follows:

\[
Q^\pi(s, a) = r(s, a) + \gamma \int_{s'} P(s'|s, a; \tilde{w}) V^\pi(s') ds' - \gamma \int_{s'} \delta_{\Omega_{\bar{w}}} \left[ -\nabla_w P(s'|s, a; \bar{w}) V^\pi(s') \right] ds',
\]

where \( \delta_{\Omega_{\bar{w}}} (w) \) is the indicator function that equals 0 if \( w \in \Omega_w \) and \(+\infty\) otherwise, and \( \delta_{\Omega_{\tilde{w}}} (\bar{w}) \) is the convex dual function of \( \delta_{\Omega_{\bar{w}}} (w) \).

The proof is presented in Appendix A.1. Intuitively, Proposition 3.1 shows that ensuring robustness on parameter \( w \) can be transformed into an additional regularizer on action value \( Q^\pi(s, a) \) that relates to the product of the state value function \( V^\pi(s') \) and the derivative of the transition model \( \nabla_w P(s'|s, a; \bar{w}) \). Taking the \( L_2 \) uncertainty set (also used in Derman et al. [12]) as a special case,
i.e., \( \Omega_w = \{ \bar{w} + \alpha \hat{w} : \|\hat{w}\|_2 \leq 1 \} \), where \( \bar{w} \) stands for the parameter of the nominal transition model \( P(s'|s, a'; \bar{w}) \), the robust Bellman equation in Proposition 3.1 becomes

\[
Q^\pi(s, a) = r(s, a) + \gamma \int s' P(s'|s, a; \bar{w}) V^\pi(s') ds' - \alpha \int s' \| \nabla_w P(s'|s, a; \bar{w}) V^\pi(s') \|_2 ds'.
\] (5)

### 3.2 Uncertainty Set Regularized Robust Reinforcement Learning (USR-RRL)

To derive a practical robust RL algorithm with the proposed USR-RBE, we follow the policy iteration framework [24] commonly used in RL research.

For the policy evaluation procedure, the following theorem proposes an operator on action value \( Q^\pi \) and ensures its convergence to a unique fixed point by recursively running this operator.

**Theorem 3.2 (Convergence of Robust Policy Evaluation)** For any policy \( \pi \in \Delta_A \), the following operator \( T \) can reach a unique fixed point as the robust action value \( Q^\pi(s, a) \) if \( 0 \leq \gamma + \alpha \max_{s, a} \left| \int s' \| \nabla_w P(s'|s, a; \bar{w}) \|_2 ds' \right| \leq 1 \).

\[
TQ^\pi(s, a) = r(s, a) + \gamma \int s' P(s'|s, a; \bar{w}) V^\pi(s') ds' - \alpha \int s' \| \nabla_w P(s'|s, a; \bar{w}) V^\pi(s') \|_2 ds',
\]

where \( V^\pi(s') = \int s' \pi(a'|s') Q(s', a') da' \).

The proof can be found in Appendix A.2. Intuitively, this theorem indicates that one can acquire a robust action value given a certain policy and uncertainty set if the discount factor \( \gamma \) and the coefficient in the uncertainty set \( \alpha \) are properly set, which is satisfied in the practical implementations.

For the policy improvement procedure, all existing techniques (e.g. policy gradient methods) can be adopted to improve the current policy. By iterating the policy evaluation and improvement cycle, the policy will eventually converge to an equilibrium between optimality and robustness.

A practical concern on this algorithm is that calculating USR-RBE requires the knowledge of the transition model \( P(s'|s, a; \bar{w}) \). This naturally applies to model-based RL, as model-based RL learns a point estimate of the transition model \( P(s'|s, a; \bar{w}) \) by maximum likelihood approaches [25]. For model-free RL, we choose to construct a local Gaussian model with mean as parameters inspired by previous work [26]. Specifically, suppose one can access a triple (state \( s \), action \( a \) and next state \( x \)) from the experience replay buffer, a local transition model \( P(s'|s, a; \bar{w}) \) is constructed as a Gaussian distribution with mean \( x \) and covariance \( \Sigma \) (with \( \Sigma \) being a hyperparameter), i.e., the nominal parameter \( \bar{w} \) consists of \( (x, \Sigma) \). With this local transition model, we now have the full knowledge of \( P(s'|s, a; \bar{w}) \) and \( \nabla_w P(s'|s, a; \bar{w}) \), which allows us to calculate the RHS of Equation 5. To further approximate the integral calculation in Equation 5, we sample \( M \) points \( \{ s_1', s_2', ... , s_M' \} \) from the local transition model \( P(s'|s, a; \bar{w}) \) and use them to approximate the target action value by \( Q^\pi(s, a) \approx r(s, a) + \gamma \sum_{i=1}^M \left( V^\pi(s_i') - \alpha \| \nabla_w P(s_i'|s, a; \bar{w}) V^\pi(s_i) \|_2 / P(s_i'|s, a; \bar{w}) \right) \), where \( \bar{w} = (x, \Sigma) \). With this approximated target value, the Bellman operator is guaranteed to converge to the robust value of the current policy \( \pi \). Policy improvement can be further applied to learn a more robust policy. We explain how to incorporate USR-RBE into a classic RL algorithm Soft Actor Critic (SAC) [27] in Appendix A.3.

### 3.3 Adversarial Uncertainty Set

The proposed method in Section 3.2 relies on the prior knowledge of the uncertainty set of the parametric space. The \( L_p \) norm uncertainty set is most widely used in the Robust RL and robust optimal control literature. However, such a fixed uncertainty set may not sufficiently adapt to various perturbation types. The \( L_p \) norm uncertainty set with its larger region can result in an over-conservative policy, while the one with a smaller region may lead to a risky policy. In this section, we learn an adversarial uncertainty set through the agent’s policy and value function.
Generating the Adversarial Uncertainty Set. The basic idea of the adversarial uncertainty set is to provide a broader uncertainty range to parameters that are more sensitive to the value function, which is naturally measured by the derivative. The agent learning on such an adversarial uncertainty set is easier to adapt to the various perturbation types of parameters. We generate the adversarial uncertainty set in 5 steps as illustrated in Figure 1, (1) sample next state \( s' \) according to the distribution \( P(\cdot | s, a; \bar{w}) \), given the current state \( s \) and action \( a \); (2) forward pass by calculating the state value \( V(s') \) at next state \( s' \); (3) backward pass by using the reparameterization trick [28] to compute the derivative \( g(\bar{w}) = \nabla_w V(s'; \bar{w}) \); (4) normalize the derivative by \( d(\bar{w}) = g(\bar{w})/\sum_i w_i^2 \); (5) generate the adversarial uncertainty set \( \Omega_w = \{ \bar{w} + \alpha \bar{w} : \| \bar{w} / d(\bar{w}) \|_2 \leq 1 \} \). The pseudo-code to generate the adversarial uncertainty set is explained in Algorithm 2.

Characteristics of the Adversarial Uncertainty Set. To further investigate the characteristics of the adversarial uncertainty set, we visualize it in a simple moving-to-target task: controlling a particle to move towards a target point \( e \) (Figure 2.a). The two-dimensional state \( s \) informs the position of the agent, and the two-dimensional action \( a = (a_1, a_2) \left( \|a\|_2 = 1 \right) \) controls the force in two directions. The environmental parameter \( w = (w_1, w_2) \) represents the contact friction in two directions respectively. The transition function is expressed as \( s' \sim N(s + (a_1w_1, a_2w_2), \Sigma) \), and the reward is defined by the progress of the distance to the target point minus a time cost: \( r(s, a, s') = d(s, e) - d(s', e) - 2 \). The nominal value \( \bar{w} = (1, 1) \) (Figure 2.b) indicates the equal friction factor in two directions for the training environment. It is easy to conjecture that the optimal action is pointing towards the target point, and the optimal value function \( V^*(s) = -d(s, e) \).

We visualize \( L_2, L_1 \) and adversarial uncertainty set of the contact friction \( w \) in Figure 2.c,d,e respectively, at a specific state \( s = (4, 3) \) and the optimal action \( a^* = (-0.8, -0.6) \). \( L_2 \) and \( L_1 \) uncertainty set satisfy \( (w_1^2 + w_2^2)^{0.5} \leq 1 \) and \( |w_1| + |w_2| \leq 1 \) respectively. Adversarial uncertainty set is calculated by following the generation procedure in Section 3.3, that the normalized derivative \( d(\bar{w}) = [0.8, 0.6]^T \) and adversarial uncertainty set is \( (w_1^2/0.64 + w_2^2/0.36)^{0.5} \leq 1 \), as an ellipse in Figure 2.e. Compared with the \( L_2 \) uncertainty set, the adversarial uncertainty set extends the perturbation range of the horizontal dimensional parameter since it is more sensitive to the final return. As a result, the agent learning to resist such an uncertainty set is expected to generally perform well on unseen perturbation types, which will be verified in experiments in the next section.

4 Experiments

In this section, we provide experimental results on the Real-world Reinforcement Learning (RWRL) benchmark [29], to validate the effectiveness of the proposed regularizing USR for resisting perturbations in the environment. Besides, we apply USR on a sim-to-real task to show its potential on real-world robots.

Figure 1: Illustration of the steps for generating the adversarial uncertainty set.

Figure 2: Illustration of different types of uncertainty sets to investigate their characteristics.
### 4.1 Experimental Setups

**Task Description.** RWRL, whose back-end is the Mujoco environment [30], is a continuous control benchmark consisting of real-world challenges for RL algorithms. Using this benchmark, we will evaluate the proposed algorithm regarding the robustness of the learned policy in physical environments with perturbations of parameters of the state equations (dynamics). In more detail, we first train a policy through interaction with the nominal environments (i.e., the environments without any perturbations), and then test the policy in the environments with perturbations within a range over relevant physical parameters. In this paper, we conduct experiments on six tasks: cartpole_balance, cartpole.swingup, walker_stand, walker.walk, quadruped.walk, quadruped.run, with growing complexity in state and action space. More details about the specifications of tasks are shown in Appendix B.1. The perturbed variables and their value ranges can be found in Table 2.

**Evaluation metric.** A severe problem for Robust RL research is the lack of a standard metric to evaluate policy robustness. To resolve this obstacle, we define a new robustness evaluation metric which we call Robust-AUC to calculate the area under the curve of the return with respect to the perturbed physical variables, in analogy to the definition of regular AUC [31]. More specifically, a trained policy $\pi$ is evaluated in an environment with perturbed variable $P$ whose values $v$ change in the range $[v_{\min}, v_{\max}]$ and achieves different returns $r$. Then, these two sets of data are employed to draw a parameter-return curve $C(v, r)$ to describe the relationship between returns $r$ and perturbed values $v$. We define the relative area under this curve as Robust-AUC such that

$$\text{Robust-AUC} = \frac{\text{Area}(C(v, r))}{v_{\max} - v_{\min}}.$$  

Compared to the vanilla AUC, Robust-AUC describes the correlation between returns and the perturbed physical variables, which can sensitively reflect the response of a learning procedure (to yield a policy) to unseen perturbations, i.e., the robustness. We further explain the practical implementations to calculate Robust-AUC of the experiments in Appendix B.2.

**Baselines and Implementation of Proposed Methods.** We first compare USR with a standard version of Soft Actor Critic (SAC) [27], which stands for the category of algorithms without regularizers (None-Reg). Another category of baselines is to directly impose $L_p$ regularization on the parameters of the value function ($L_1$-Reg, $L_2$-Reg) [32], which is a common way to improve the generalization of function approximation but without consideration of the environmental perturbations; For a fixed uncertainty set as introduced in Section 4.2, we first compare USR with standard version of Soft Actor Critic (SAC) [27], which stands for the category of algorithms without regularizers (None-Reg). Another category of baselines is to directly impose $L_p$ regularization on the parameters of the value function ($L_1$-Reg, $L_2$-Reg) [32], which is a common way to improve the generalization of function approximation but without consideration of the environmental perturbations; For a fixed uncertainty set as introduced in Section 3.2, we implement two types of uncertainty sets on transitions, $L_2$-USR and $L_1$-USR, which can be viewed as an extension of Derman et al. [12] and Wang and Zou [16] for continuous control tasks respectively; finally, we also evaluate the adversarial uncertainty set (Section 3.3), denoted as Adv-USR. We conclude all model structures and hyperparameters in Appendix B.3 - B.4.

### 4.2 Main Results

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Variables</th>
<th>None-Reg</th>
<th>L1-Reg</th>
<th>L2-Reg</th>
<th>L1-USR</th>
<th>L2-USR</th>
<th>Adv-USR</th>
</tr>
</thead>
<tbody>
<tr>
<td>cartpole.swingup</td>
<td>pole_length</td>
<td>393.41 (21.95)</td>
<td>319.73 (23.84)</td>
<td>368.16 (6.33)</td>
<td>444.93 (12.77)</td>
<td>424.48 (2.27)</td>
<td>430.91 (2.35)</td>
</tr>
<tr>
<td></td>
<td>pole_mass</td>
<td>155.25 (4.79)</td>
<td>96.85 (4.32)</td>
<td>131.35 (12.42)</td>
<td>175.28 (3.33)</td>
<td>159.81 (2.41)</td>
<td>193.13 (2.27)</td>
</tr>
<tr>
<td></td>
<td>joint_damping</td>
<td>137.20 (4.55)</td>
<td>140.16 (0.58)</td>
<td>165.01 (0.16)</td>
<td>164.21 (0.48)</td>
<td>169.88 (2.68)</td>
<td>170.39 (0.76)</td>
</tr>
<tr>
<td></td>
<td>slider_damping</td>
<td>783.76 (9.14)</td>
<td>775.73 (16.50)</td>
<td>797.59 (5.52)</td>
<td>793.55 (5.02)</td>
<td>781.02 (5.80)</td>
<td>819.32 (3.00)</td>
</tr>
<tr>
<td></td>
<td>average_rank</td>
<td>4.5</td>
<td>5.75</td>
<td>3.75</td>
<td>2.5</td>
<td>3.25</td>
<td>1.25</td>
</tr>
<tr>
<td>walker_stand</td>
<td>thigh_length</td>
<td>487.02 (40.50)</td>
<td>461.95 (50.03)</td>
<td>497.38 (43.98)</td>
<td>488.71 (42.42)</td>
<td>511.16 (49.84)</td>
<td>505.88 (50.22)</td>
</tr>
<tr>
<td></td>
<td>torso_length</td>
<td>614.06 (41.10)</td>
<td>586.16 (68.84)</td>
<td>586.20 (44.31)</td>
<td>598.02 (36.17)</td>
<td>610.93 (38.47)</td>
<td>632.53 (46.47)</td>
</tr>
<tr>
<td></td>
<td>joint_damping</td>
<td>607.24 (115.28)</td>
<td>387.89 (109.53)</td>
<td>443.82 (63.32)</td>
<td>389.77 (76.96)</td>
<td>527.87 (116.75)</td>
<td>514.77 (126.00)</td>
</tr>
<tr>
<td></td>
<td>contact_friction</td>
<td>946.74 (22.20)</td>
<td>947.24 (29.16)</td>
<td>941.92 (22.21)</td>
<td>943.11 (21.97)</td>
<td>940.73 (20.89)</td>
<td>943.69 (16.02)</td>
</tr>
<tr>
<td></td>
<td>average_rank</td>
<td>2.5</td>
<td>4.25</td>
<td>4.25</td>
<td>2.5</td>
<td>3.00</td>
<td>2.25</td>
</tr>
<tr>
<td>quadruped.walk</td>
<td>shank_length</td>
<td>492.55 (124.44)</td>
<td>406.07 (90.41)</td>
<td>503.13 (106.50)</td>
<td>540.39 (126.22)</td>
<td>564.60 (135.49)</td>
<td>571.85 (64.99)</td>
</tr>
<tr>
<td></td>
<td>torso_density</td>
<td>471.45 (99.14)</td>
<td>600.86 (45.21)</td>
<td>526.22 (79.11)</td>
<td>442.05 (70.73)</td>
<td>472.80 (63.52)</td>
<td>602.09 (55.36)</td>
</tr>
<tr>
<td></td>
<td>joint_damping</td>
<td>675.95 (67.2)</td>
<td>71.54 (91.84)</td>
<td>704.56 (65.64)</td>
<td>762.50 (79.76)</td>
<td>658.17 (112.67)</td>
<td>785.11 (40.79)</td>
</tr>
<tr>
<td></td>
<td>contact_friction</td>
<td>683.80 (135.80)</td>
<td>906.92 (100.19)</td>
<td>770.44 (158.42)</td>
<td>777.40 (106.04)</td>
<td>767.80 (109.00)</td>
<td>909.73 (21.24)</td>
</tr>
<tr>
<td></td>
<td>average_rank</td>
<td>5.25</td>
<td>3.50</td>
<td>3.00</td>
<td>3.75</td>
<td>4.25</td>
<td>1.25</td>
</tr>
</tbody>
</table>
We show the Robust-AUC and its significance value of cartpole.swingup, walker_stand, quadruped_walk in Table 1. Due to page limits, the results of other tasks are presented in Appendix C.1. In addition to calculating Robust-AUC under different perturbations, we also rank all algorithms and report the average rank as an overall robustness performance of each task. Notably, L1-Reg and L2-Reg do not improve on the robustness, and even impair the performance in comparison with the None-Regularized agent on simple domains (cartpole and walker). In contrast, we observe that both L2-USR and L1-USR can outperform the default version under certain perturbations (e.g. L1-USR in cartpole.swingup for pole_length, L2-USR in walker_stand for thigh_length); they are, however, not effective for all scenarios. We argue that the underlying reason could be that the fixed shape of the uncertainty set cannot adapt to all perturbed cases. This is supported by the fact that Adv-USR achieves the best average rank among all perturbed scenarios, showing the best zero-shot generalization performance in continuous control tasks. For complex tasks like quadruped_run, it is surprising that L1-Reg can achieve competitive results compared with Adv-USR but with slightly larger uncertainty on the Robust-AUC, probably because the sparsity by L1 regularization can reduce redundant features. We also compare the computational cost of all algorithms both empirically and theoretically in Appendix C.4. It is concluded that Adv-USR can improve the robustness in most cases without increasing too much computational burden. More limitations on the computational time are discussed in Section 7. A video (quadruped_walk.mov in supplementary material) is provided as an intuitive demonstration of the effect of the Adv-USR when a walking quadruped is perturbed by different contact frictions. We also carry out two additional testing scenarios to imitate the perturbation in real: all parameters deviate from the nominal values simultaneously (Appendix C.2) and the perturbed value follows a random walk during the testing episode (Appendix C.3). Adv-USR consistently performs best and is well-adapted to different perturbations.

4.3 Study on Sim-to-real Robotic Task

In this section, we further investigate the robustness of Adv-USR on a sim-to-real robotic task. Sim-to-real is a commonly adopted setup to apply RL algorithms on real robots, where the agent is first trained in simulation and then transferred to real robots. Unlike the environmental setup in Section 4.1 with additional perturbations during the testing phases, sim-to-real inherently possesses a mismatch between training and testing environments potentially due to: (1) the simulator possesses a simplified dynamics model and suffers from accumulated error [33] and (2) there are significant differences between simulators and real hardware in robot’s parameters, such as a quadruped example in Table 5. As a result, this setup is an ideal testbed and practical application of the proposed robust RL algorithm.

Specifically, we use the Unitree A1 robot [34] and the Bullet simulator [35] as the platform for sim-to-real transfer. The agents learn standing and locomotion in simulations and directly perform on real robots without adaption [36]. Since other baselines cannot generalize well even in pure simulated RWRL environments, we only compare SAC agents with and without Adv-USR method. Most previous works [37, 38] utilize domain randomization (DR) techniques [39] to deal with sim-to-real mismatches. DR requires training on multiple randomly initialized simulated instances with diverse environmental parameters, expecting the policy to be generalized in the testing environment (real robot). In contrast, Adv-USR only requires training on single nominal parameters, which tremendously improves the efficiency and feasibility. Detailed setup of the sim-to-real task is described in Appendix B.5.

Figure 3: Episode returns of all algorithms on sim-to-real task.

(a) Standing
(b) Locomotion
We run 50 testing trials per baseline and report the episodic returns in Figure 3 and an explanatory video (a1.mp4) in the supplementary material. Both agents succeed to learn a nearly optimal policy in simulation for both tasks. For the standing task, the agent with Adv-USR maintains its performance on real robots while the other agent fails to achieve the standing position. For the more complex locomotion task, Adv-USR still performs better than None-Reg on real robots, however, the performance deteriorates compared with simulation. To alleviate the extreme sim-to-real difference, combining Adv-USR and other sim-to-real techniques could be a more powerful strategy, which would be verified in further work.

5 Related Work

Robust Reinforcement Learning (Robust RL) has recently become a popular topic [9, 29, 40, 41], due to its effectiveness in tackling perturbations. Besides the transition perturbation in this paper, there are other branches relating to action, state and reward. Early works in Robust RL concentrated on action space perturbations. Pinto et al. [42] first proposed an adversarial agent perturbing the action of the principle agent, training both alternately in a mini-max style. This adversarial agent in action space always leads to over-conservative behaviours. Reward perturbation has proved to be converted to transition perturbation by augmenting the reward value in the state [22]. In this paper, we further transform state perturbation to transition perturbation in Appendix E, thus covered by our proposed method.

Sim-to-Real is a key research topic in robot learning. Compared to the Robust RL problem, it aims to learn a robust policy from simulations for generalization in real-world environments. Domain randomization is a common approach to ease this mismatch in sim-to-real problems [39, 8]. However, Mankowitz et al. [15] has demonstrated that it actually optimizes the average case of the environment rather than the worst-case scenario (as seen in our research), which fails to perform robustly during testing. More recent active domain randomization methods [43] resolve this flaw by automatically selecting difficult environments during the training process. The idea of learning an adversarial uncertainty set considered in this paper can be seen as a strategy to actively search for more valuable environments for training. We discuss different types of uncertainty sets and the relation of Robust RL, Bayesian RL and Adaptive RL approaches more fully in Appendix D.

6 Conclusion

In this paper, we adopt the robustness-regularization duality method to design new regularizers in continuous control problems to improve the robustness and generalization of RL algorithms. Furthermore, to deal with unknown uncertainty sets, we design an adversarial uncertainty set according to the learned action state value function and incorporate it into a new regularizer. The proposed method shows great promise regarding generalization and robustness under environmental perturbations in simulated and realistic robotic tasks. Moreover, it does not require training in multiple diverse environments or fine-tuning in testing environments, which makes it an efficient and valuable add-on to RL for robot learning.

7 Limitations

The computational cost of Adv-USR is acceptable for the local Gaussian model in the experiments, but is a critical factor when applying it to more complex dynamics with millions of parameters (i.e. common in current model-based RL research [44]). Therefore, methods to automatically detect important variables could be required. Finally, it is desirable to combine Adv-USR with domain randomization in more sophisticated robotic tasks.
References


