Combining Pessimism with Optimism for Robust and Efficient Model-Based Deep Reinforcement Learning

Sebastian Curi [1] **Ilija Bogunovic** [1] **Andreas Krause** [1]

Abstract

In real-world tasks, reinforcement learning (RL) agents frequently encounter situations that are not present during training time. To ensure reliable performance, the RL agents need to exhibit *robustness* against worst-case situations. The robust RL framework addresses this challenge via a worst-case optimization between an agent and an adversary. Previous robust RL algorithms are either sample inefficient, lack robustness guarantees, or do not scale to large problems. We propose the *Robust Hallucinated Upper-Confidence* RL (`RH-UCRL`) algorithm to *provably solve this problem while attaining* near-optimal sample complexity guarantees. `RH-UCRL` is a model-based reinforcement learning (MBRL) algorithm that effectively distinguishes between epistemic and aleatoric uncertainty, and *efficiently* explores both the agent and adversary decision spaces during policy learning. We scale `RH-UCRL` to complex tasks via neural networks ensemble models as well as neural network policies. Experimentally, we demonstrate that `RH-UCRL` outperforms other robust deep RL algorithms in a variety of adversarial environments.

1. Introduction

A central challenge when deploying Reinforcement Learning (RL) agents in real environments is their robustness (Dulac-Arnold et al., 2019). As a motivating example, consider designing a braking system on an autonomous car. As this is a highly complex task, we want to *learn* a policy that performs this maneuver. Even if various real-world conditions can be simulated during the training time, it is infeasible to consider *all* possible ones such as road conditions, brightness, tire pressure, laden weight, or actuator wear, as these can all vary over time in potentially unpredictable ways. The main goal is then to learn a policy that *provably* brakes in a *robust* fashion so that, even if faced with new conditions, it performs reliably. Borrowing from the robust control perspective (e.g, as considered in $\mathcal{H}_\infty$ control Başar & Bernhard (2008)), we model robustness with an adversary that is allowed to perform the worst-case disturbance for the given policy.

While there are theoretical approaches for robust RL that offer sample complexity guarantees (see Section 2), the existing algorithms are highly impractical. On the other hand, there are empirically motivated heuristic approaches that lack provable robustness. We bridge this gap by proposing an algorithm that simultaneously enjoys rigorous theoretical guarantees, yet can be applied to complex tasks.

We develop the *Robust Hallucinated Upper-Confidence Reinforcement Learning* (`RH-UCRL`) algorithm for obtaining robust RL policies. It relies on a probabilistic model that can distinguish between epistemic uncertainty (that arises from the lack of data) and aleatoric uncertainty (stochastic noise) (Der Kiureghian & Ditlevsen, 2009). A key algorithmic principle behind `RH-UCRL` is *hallucination*: In particular, the *agent* hallucinates an additional control input to *maximize* an *optimistic* estimate of the robust performance whereas the *adversary* hallucinates an additional control input to *minimize* a *pessimistic* estimate of the robust performance. The amount of "hallucination" is limited by the epistemic uncertainty of the model and it decreases as the learning algorithm collects more data. In a number of experiments, we show that our algorithm exhibits robust performance and outperforms previous approaches on various popular RL benchmarks. In Figure 1, we demonstrate how `RH-UCRL` outperforms baselines in a pendulum swing-up control task. In particular, our main observations are: (i) Robust baselines specifically designed for different settings do not explore sufficiently and fail to find a swing-up maneuver; (ii) Non-robust algorithm finds the swing-up strategy, but as adversarial power increases its performance significantly deteriorates, whereas `RH-UCRL` exhibits higher robustness to *worst-case* perturbations.

Main contributions. We design `RH-UCRL`, the first practical *provably* robust RL algorithm that is: (i) *sample-efficient*, (ii) compatible with *deep models*, and (iii) simulator-free as

[1]Department of Computer Science, ETH Zurich, Switzerland. Correspondence to: Sebastian Curi <sebastian.curi@inf.ethz.ch>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

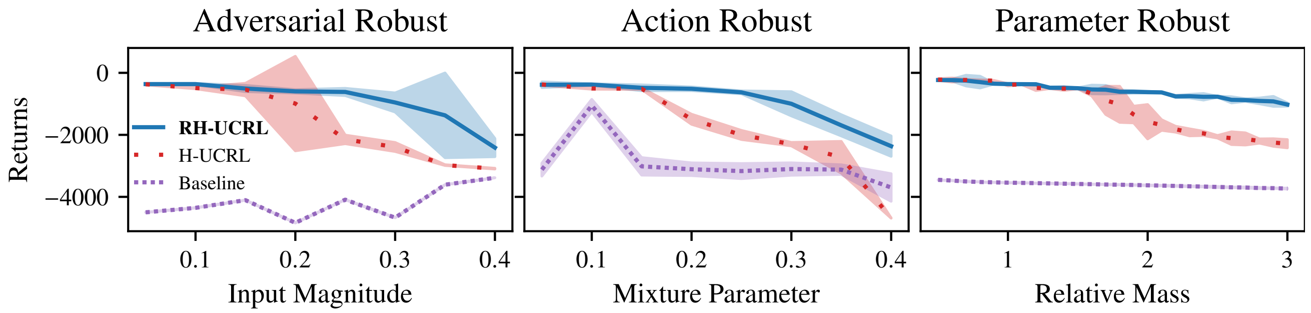

Figure 1. Performance of `RH-UCRL` (this work), `H-UCRL` (Curi et al., 2020a), and a baseline with no exploration in adversarial-, action-, and parameter-robust settings on a pendulum swing-up task. The baseline never learns a successful swing-up strategy due to insufficient exploration. `RH-UCRL` and `H-UCRL` learn a swing-up strategy but `RH-UCRL` outperforms `H-UCRL` as the perturbation increases.

it addresses exploration on a real system. We establish rigorous general sample-complexity and regret guarantees for our algorithm, and we specialize them to Gaussian Process models, hence obtaining sublinear robust regret guarantees. While previous robust RL works have focused on different individual settings, we gather and summarize them all for the first time, and show particular instantiations of our algorithm in each of them: (i) Adversarial-robust RL, (ii) Action-robust RL, and (iii) Parameter-robust RL. Finally, we provide experiments that include different environments and settings, and we empirically demonstrate that `RH-UCRL` outperforms or successfully competes with the state-of-the-art deep robust RL algorithms and other baselines.

2. Related Work

Robust RL. Robust Reinforcement Learning (Iyengar, 2005; Nilim & El Ghaoui, 2005; Wiesemann et al., 2013) typically uses the Zero-Sum Markov Games formalism introduced by Littman (1994); Littman & Szepesvári (1996). From a control engineering perspective, this is known as $\mathcal{H}_\infty$ control (Başar & Bernhard, 2008), and Bemporad et al. (2003) solve this problem for *known* linear systems where the optimal robust policy is an affine function of the state. For Markov Games with unknown systems, Lagoudakis & Parr (2002) introduce approximate value iteration whereas Tamar et al. (2014) and Perolat et al. (2015) present an approximate policy iteration scheme. Although these works present error propagation schemes restricted to the linear setting, they do not address the problem of exploration explicitly, and instead assume access to a sampling distribution that has sufficient state coverage. Zhang et al. (2020) propose a model-based algorithm for finding robust policies assuming access to a simulator that is able to sample at arbitrary state-action pairs. Finally, Bai & Jin (2020) recently introduce an algorithm that provably and efficiently outputs a policy, but it is limited to the tabular setting. Instead, our approach does not require a generative model and considers the full exploration problem in a finite horizon scenario. Furthermore, our algorithm does not require tabular nor linear function approximation and it is compatible with deep neural networks dynamical models.

Minimax Optimization Algorithms. Pinto et al. (2017) propose to solve the minimax optimization via a stochastic gradient descent approach for both players for *adversarial-robust* RL algorithms. Tessler et al. (2019) introduce *action-robust* RL and policy and value iteration algorithms to solve these problems. Rajeswaran et al. (2017) introduce the `EPOpt` to solve *parameter-robust* problems. Finally, Kamalaruban et al. (2020) propose to use a Stochastic-Gradient Langevin Dynamics algorithm to solve such problems via sampling instead of optimization. These algorithms are generally sample-inefficient as they do not explicitly explore but, given a model, they could be used to optimize a policy.

Provable Model-Based RL. Model-Based RL has better empirical sample-efficiency than model-free variants (Deisenroth & Rasmussen, 2011; Chua et al., 2018). Furthermore, the celebrated UCRL algorithm by Auer et al. (2009) has provable sample-efficiency guarantees, albeit being intractable for non-tabular environments. Recently, Curi et al. (2020a) instantiate the UCRL algorithm in continuous control problems using hallucinated control to explore. We build upon their `H-UCRL` algorithm and strictly generalize it to the robust RL setting. Our robust policy search strategy is inspired by the work in the related bandit setting with Gaussian Processes, where `Stable-OPT` (Bogunovic et al., 2018) combines pessimism with optimism to similarly generalize the non-robust `GP-UCB` (Srinivas et al., 2010) algorithm and provably discover robust designs.

3. Problem Statement

We consider a stochastic environment with states $\mathbf{s} \in \mathcal{S} \subseteq \mathbb{R}^p$, agent actions $\mathbf{a} \in \mathcal{A} \subset \mathbb{R}^q$, adversary actions $\bar{\mathbf{a}} \in \bar{\mathcal{A}} \subset \mathbb{R}^{\bar{q}}$, and *i.i.d.* additive transition noise vector $\boldsymbol{\omega}_h \in \mathbb{R}^p$. Both action sets are assumed to be compact, and the dynamics are given by:

$$\mathbf{s}_{h+1} = f(\mathbf{s}_h, \mathbf{a}_h, \bar{\mathbf{a}}_h) + \boldsymbol{\omega}_h \tag{1}$$

with $f\colon \mathcal{S} \times \mathcal{A} \times \bar{\mathcal{A}} \to \mathcal{S}$. We assume the true dynamics f are *unknown* and consider the episodic setting over a finite time horizon H. After every episode (i.e., every H time steps), the system is reset to a known state $\mathbf{s}_0$. In this work, we make the following assumptions regarding the stochastic environment and unknown dynamics:

Assumption 1. The dynamics f in Eq. (1) are L_f-Lipschitz continuous and, for all $h \in \{0, \dots, H-1\}$, the individual entries of the noise vector $\boldsymbol{\omega}_h$ are *i.i.d.* σ-sub-Gaussian.

At every time-step, the system returns a deterministic reward $r(\mathbf{s}_h, \mathbf{a}_h, \bar{\mathbf{a}}_h)$, where $r : \mathcal{S} \times \mathcal{A} \times \bar{\mathcal{A}} \to \mathbb{R}$ is assumed to be known to the agent. We consider time-homogeneous agent policies $\pi \in \Pi$, $\pi\colon \mathcal{S} \to \mathcal{A}$, that select actions according to $\mathbf{a}_h = \pi(\mathbf{s}_h)$. Similarly, we consider adversary policies $\bar{\pi} \in \bar{\Pi}$ on the common state space, i.e., $\bar{\pi}\colon \mathcal{S} \to \bar{\mathcal{A}}$, that select actions as $\bar{\mathbf{a}}_h = \bar{\pi}(\mathbf{s}_h)$. For the sake of simplicity, we omit straightforward extensions such as initial-state distributions, unknown reward functions, or time-indexed policies that can be adressed using standard techniques (Chowdhury & Gopalan, 2019). For now, we leave both Π and $\bar{\Pi}$ unspecified, but in Section 5, we parameterize them via neural networks.

The performance of a pair of policies $(\pi, \bar{\pi})$ on a given dynamical system $\tilde{f}$ is the episodic expected sum of returns:

$$J(\tilde{f}, \pi, \bar{\pi}) := \mathbb{E}_{\tau_{\tilde{f},\pi,\bar{\pi}}}\left[\sum_{h=0}^{H} r(\mathbf{s}_h, \mathbf{a}_h, \bar{\mathbf{a}}_h) \,\middle|\, \mathbf{s}_0\right], \tag{2}$$
$$\text{s.t.} \;\; \mathbf{s}_{h+1} = \tilde{f}(\mathbf{s}_h, \mathbf{a}_h, \bar{\mathbf{a}}_h) + \boldsymbol{\omega}_h,$$

where $\tau_{\tilde{f},\pi,\bar{\pi}} = \{(\mathbf{s}_{h-1}, \mathbf{a}_{h-1}, \bar{\mathbf{a}}_{h-1}), \mathbf{s}_h\}_{h=0}^{H}$ is a random trajectory induced by the stochastic noise $\boldsymbol{\omega}$, the dynamics $\tilde{f}$, and the policies π and $\bar{\pi}$.

We use $\pi^\star$ to denote the optimal deterministic *robust* policy from set Π in case of true dynamics f, i.e.,

$$\pi^\star \in \arg\max_{\pi \in \Pi} \min_{\bar{\pi} \in \bar{\Pi}} J(f, \pi, \bar{\pi}). \tag{3}$$

Even when the true system dynamics are known, finding a robust policy is generally a challenging task for arbitrary policy sets, reward and transition functions. In the rest, we make an assumption that Eq. (3) can be solved for a given dynamics, and in Section 4.3, we propose a concrete problem instantiation and algorithmic solution.

Learning protocol. We consider the episodic setting in which, at every episode t, the learning algorithm selects both the agent's π_t and a fictitious adversary's $\bar{\pi}_t$ policies. The pair of policies (π_t, $\bar{\pi}_t$) is then deployed on the true

Algorithm 1 Robust Model-based Reinforcement Learning

```
1: Require: Calibrated dynamical model, reward function
   $r(\mathbf{s}_h, \mathbf{a}_h, \bar{\mathbf{a}}_h)$, horizon $H$, initial state $\mathbf{s}_0$
2: for $t = 1, 2, \dots$ do
3:    Select ($\pi_t$,$\bar{\pi}_t$) using the current dynamical model
4:    for $h = 1, \dots, H-1$ do
5:       $\mathbf{s}_{h,t} = f(\mathbf{s}_{h-1,t}, \pi_t(\mathbf{s}_{h-1,t}), \bar{\pi}_t(\mathbf{s}_{h-1,t})) + \boldsymbol{\omega}_{h,t}$
6:    end for
7:    Update statistical dynamical model with the $H$ observations $\{(\mathbf{s}_{h-1,t}, \mathbf{a}_{h-1,t}, \bar{\mathbf{a}}_{h-1,t}), \mathbf{s}_{h,t}\}_{h=0}^{H}$
8:    Reset the system to $\mathbf{s}_{0,t+1} = \mathbf{s}_0$
9: end for
```

system f, and a single realization of the trajectory $\tau_{f,\pi_t,\bar{\pi}_t}$ is observed and used to update the underlying statistical model. We summarize the general learning protocol in Algorithm 1. In the braking system example, this learning protocol implies that during training we are allowed to execute braking maneuvers as well as possible adversarial policies, e.g., changing the braking surface. The execution of both policies during training is crucial to guarantee robust performance: The learner can actively look for the *worst-case* adversarial policies that it might encounter during deployment and learn what to do when faced upon them.

Performance metric. For a small fixed $\epsilon > 0$, the goal is to output a robust policy π_T after T episodes such that:

$$\min_{\bar{\pi} \in \bar{\Pi}} J(f, \pi_T, \bar{\pi}) \geq \min_{\bar{\pi} \in \bar{\Pi}} J(f, \pi^\star, \bar{\pi}) - \epsilon, \tag{4}$$

where $\pi^\star$ is defined as in Eq. (3). Hence, we consider the task of near-optimal robust policy identification, but we note that one can also measure the performance in terms of the robust cumulative regret as discussed in Section 4.4. Thus, the goal is to output the agent's policy with near-optimal robust performance when facing its own *worst-case* adversary, and the adversary selects $\bar{\pi}$ *after* the agent selects π_T. Note that this is a stronger robustness notion than just considering the worst-case adversary of the optimal policy, since, by letting $\bar{\pi}^* \in \arg\min_{\bar{\pi} \in \bar{\Pi}} J(f, \pi^\star, \bar{\pi})$, we have $J(f, \pi_T, \bar{\pi}^*) \geq \min_{\bar{\pi} \in \bar{\Pi}} J(f, \pi_T, \bar{\pi})$.

Statistical Model. In this work, we take a *model-based reinforcement learning (MBRL)* approach. That is, to learn the dynamics and discover a near-optimal robust policy, we consider algorithms that model and sequentially learn about f from noisy state observations. The agent makes use of the observed data (collected within an episode) to simultaneously improve its estimate of the true dynamics f.

We use statistical estimation to probabilistically reason about dynamical models $\tilde{f}$ that are compatible with the observed data $\mathcal{D}_{1:t} = \left\{\tau_{f,\pi_{t'},\bar{\pi}_{t'}}\right\}_{t'=1}^{t}$. This can be done, e.g., by frequentist estimation of mean $\mu_t(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}})$ and confidence $\Sigma_t^2(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}})$ estimators, or by taking a Bayesian perspective and considering the posterior distribution $p(\tilde{f} \mid \mathcal{D}_{1:t})$ over dynamical models that leads to $\mu_t(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}}) = \mathbb{E}_{\tilde{f} \sim p(\tilde{f} \mid \mathcal{D}_{1:t})}[\tilde{f}(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}})]$ and $\Sigma_t^2(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}}) = \mathrm{Var}[\tilde{f}(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}})]$. In any case, we require the model to be *calibrated*:

Assumption 2 (Calibrated model)**.** The statistical model is *calibrated* w.r.t. f in Eq. (1), so that with $\sigma_t(\cdot) = \mathrm{diag}(\Sigma_t(\cdot))$ and a non-decreasing sequence of parameters $\{\beta_t\}_{t \geq 1} \in \mathbb{R}_{>0}$, each depending on $\delta \in (0, 1)$, it holds jointly for all $t \geq 1$ and $\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}} \in \mathcal{S} \times \mathcal{A} \times \bar{\mathcal{A}}$ that $|f(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}}) - \mu_{t-1}(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}})| \leq \beta_t \Sigma_{t-1}(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}})$ element-wise, with probability at least $1 - \delta$.

This assumption is important for exploration: if the model is not calibrated, then using the epistemic uncertainty of such a model will not *provably* guide exploration. For dynamics with finite norm in a known RKHS space, Assumption 2 is satisfied (Srinivas et al., 2010; Chowdhury & Gopalan, 2017). In case of neural network models, we can recalibrate one-step ahead predictions (Malik et al., 2019). Finally, we construct the set of plausible models at time t as $\mathcal{M}_t = \left\{\tilde{f} \mid |\tilde{f}(\cdot) - \mu_{t-1}(\cdot)| \leq \beta_t \Sigma_{t-1}(\cdot)\right\}$. By Assumption 2, we can guarantee that, with high probability, the true dynamics $f \in \mathcal{M}_t$, for all time t.

4. The Robust `H-UCRL` Algorithm (`RH-UCRL`)

We now develop our `RH-UCRL` algorithm that can be used in Algorithm 1 (at Line 3), for selecting policies π_t and $\bar{\pi}_t$. `RH-UCRL` takes the sequence of confidence parameters $\{\beta_t\}_{t \geq 1}$ from Assumption 2 as input. The main idea is to use our probabilistic model of f to *optimistically* select π_t and *pessimistically* select $\bar{\pi}_t$ w.r.t. all plausible dynamics.

4.1. Optimistic and Pessimistic Policy Evaluation

For any two policies π and $\bar{\pi}$, we provide the **(o)**ptimistic and **(p)**essimistic estimate of $J(f, \pi, \bar{\pi})$ at time t, and we denote them with $J_t^{(o)}(\pi, \bar{\pi})$ and $J_t^{(p)}(\pi, \bar{\pi})$, respectively. Such estimates are constructed considering the epistemic uncertainty in the dynamical model. For instance, the optimistic estimate is the maximum performance of a given policy, where the maximum is taken over the dynamical models in $\mathcal{M}_t$. In general, such optimization problem is intractable. However, we introduce an auxiliary function $\eta : \mathcal{S} \times \mathcal{A} \times \bar{\mathcal{A}} \to [-1, 1]^p$ and reparameterize the set of plausible models as $\tilde{f} = \mu_{t-1}(\cdot) + \beta_t \eta(\cdot) \Sigma_{t-1}(\cdot)$. Using this reparameterization, the optimistic estimate is given by:

$$J_t^{(o)}(\pi, \bar{\pi}) := \max_{\eta^{(o)}} J(f^{(o)}, \pi, \bar{\pi}), \tag{5a}$$

$$\text{s.t. } f^{(o)}(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}}) = \mu_{t-1}(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}}) + \beta_t \eta^{(o)}(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}}) \Sigma_{t-1}(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}}). \tag{5b}$$

Similarly, the pessimistic estimate is given by:

$$J_t^{(p)}(\pi, \bar{\pi}) := \min_{\eta^{(p)}} J(f^{(p)}, \pi, \bar{\pi}) \tag{6a}$$

$$\text{s.t. } f^{(p)}(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}}) = \mu_{t-1}(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}}) + \beta_t \eta^{(p)}(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}}) \Sigma_{t-1}(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}}). \tag{6b}$$

We note that $J_t^{(o)}$ and $J_t^{(p)}$ represent upper and lower bounds on the performance of the policies π, $\bar{\pi}$ in case of the true dynamics f. These estimates are computed by finding the most optimistic (pessimistic) dynamics compatible with the data. Note that the optimistic/pessimistic outcome is selected via decision variables $\eta^{(o)}/\eta^{(p)} : \mathcal{S} \times \mathcal{A} \times \bar{\mathcal{A}} \to [-1, 1]^p$, which are functions of the state as well as actions of both players. These select among all plausible outcomes of the dynamics bounded within the epistemic uncertainty over f. When the policies are fixed and clear from context, with slight abuse of notation we write $\eta(\mathbf{s}, \mathbf{a}, \bar{\mathbf{a}}) = \eta(\mathbf{s}, \pi(\mathbf{s}), \bar{\pi}(\mathbf{s})) = \eta(\mathbf{s})$. A crucial observation is that both Eqs. (5) and (6) can be viewed as two optimal control problems, where the decision variables $\eta^{(o)}/\eta^{(p)}$ are hallucinated control policies, whose effect is bounded by the model epistemic uncertainty. We can use optimal control algorithms (Camacho & Alba, 2013) to maximize/minimize the sum of rewards following the reparameterized dynamics $f^{(o)}/f^{(p)}$.

4.2. The `RH-UCRL` Algorithm

Given both the pessimistic and optimistic performance estimates from the previous section, we are now ready to state our algorithm. At each episode t, `RH-UCRL` selects the agent and adversary policies as follows:

$$\pi_t \in \arg\max_{\pi \in \Pi} \min_{\bar{\pi} \in \bar{\Pi}} J_t^{(o)}(\pi, \bar{\pi}), \tag{7a}$$

$$\bar{\pi}_t \in \arg\min_{\bar{\pi} \in \bar{\Pi}} J_t^{(p)}(\pi_t, \bar{\pi}). \tag{7b}$$

Thus, `RH-UCRL` selects the most optimistic robust policy for the agent player in Eq. (7a). The adversary player picks the most pessimistic policy given the selected agent policy in Eq. (7b). When the adversarial policy space $\bar{\Pi}$ is a singleton, `RH-UCRL` reduces to the `H-UCRL` algorithm.

Finally, after a total of T episodes, the algorithm outputs an agent policy $\hat{\pi}_T$ given by:

$$\hat{\pi}_T = \pi_{t^\star} \text{ s.t. } t^\star \in \arg\max_{t \in \{1, \ldots, T\}} J_t^{(p)}(\pi_t, \bar{\pi}_t). \tag{8}$$

There is no extra computational cost in identifying the output policy as $J_t^{(p)}(\pi_t, \bar{\pi}_t)$ is already computed by the learner in Eq. (7b) in every episode t. Thus, the algorithm simply returns the encountered agent policy with maximum pessimistic robust performance.

4.3. Practical Implementation

In this section, we describe the practical implementation of RH-UCRL using neural networks for the dynamical model as well as the true and hallucinated policies. We provide an implementation of the algorithm on https://github.com/sebascuri/rhucrl.

Model Learning. As used in PETS (Chua et al., 2018) and H-UCRL (Curi et al., 2020a), we use an ensemble of neural network as our model. In particular, each ensemble member receives as input the tuple $(\mathbf{s}_{h-1}, \mathbf{a}_{h-1}, \bar{\mathbf{a}}_{h-1})$ and the goal is to output a predictive distribution of $\mathbf{s}_h$. To this end, we train each ensemble member using type-II maximum likelihood estimation, using as targets the *difference* between two consequtive states, i.e., $\Delta_h = \mathbf{s}_h - \mathbf{s}_{h-1}$. Given a trained model that outputs $\mathcal{N}(\mu_{t-1}^{(i)}, \omega_{t-1}^{(i)})$ for each ensemble member i, we combine the predictions as a mixture of Gaussians. The mean prediction is $\mu_{t-1} = \frac{1}{N}\sum_{i=1}^{N} \mu_{t-1}^{(i)}$, the epistemic uncertainty is $\Sigma_{t-1}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(\mu_{t-1}^{(i)} - \mu_{t-1})(\mu_{t-1}^{(i)} - \mu_{t-1})^\top$, and the aleatoric uncertainty is $\omega_{t-1} = \frac{1}{N}\sum_{i=1}^{N}\omega_{t-1}^{(i)}$.

Policy Learning. To implement RH-UCRL, we parameterize π, $\bar{\pi}$, and η using neural network policies. We remark that $\bar{\pi}$ in the agent optimization (7a) and $\bar{\pi}$ in the adversary optimization (7b) are different so, for the sake of clarity, we call $\bar{\pi}'$ the policy in the agent optimization (7a). We approximate the finite-horizon RL problem with a discounted infinite-horizon problem using $\gamma = 1/(1-H)$ as discount factor. We use an actor-critic approach where we learn two separate critics, one for the optimistic performance in (7a) and the other one for the pessimistic performance in (7b) via fitted Q-iteration (Perolat et al., 2015; Antos et al., 2008). Finally, we do stochastic gradient ascent/descent using pathwise gradients through such learned critics (Mohamed et al., 2019; Silver et al., 2014). Namely, we compute the gradients of π, $\eta^{(o)}$, and $\bar{\pi}'$ through the learned optimistic critic, then we update π and $\eta^{(o)}$ via gradient ascent, whereas $\bar{\pi}'$ via gradient descent. Likewise, we compute the gradients of $\bar{\pi}$ and $\eta^{(p)}$ through the learned pessimistic critic for the fixed π and update both $\bar{\pi}$ and $\eta^{(p)}$ via gradient descent.

4.4. Theoretical Analysis

In this section, we theoretically analyze the performance of the RH-UCRL algorithm. First, we use the notion of *robust cumulative regret*[1]

$$R_T = \sum_{t=1}^{T} \min_{\bar{\pi}\in\bar{\Pi}} J(f, \pi^\star, \bar{\pi}) - \min_{\bar{\pi}\in\bar{\Pi}} J(f, \pi_t, \bar{\pi}),$$

[1]Similar notions of robust cumulative regret have been analyzed before in bandit optimization (see, e.g., Kirschner et al., 2020).

which measures the difference in performance between the optimal robust policy and the sequence of agent's policies $\{\pi_1, \dots, \pi_T\}$ selected at every episode in Eq. (7a). Below (see Theorem 1) we establish that RH-UCRL achieves sublinear regret, i.e., $R_T/T \to 0$ for $T \to \infty$. In addition to the robust regret notion, we also analyze the recommendation rule of RH-UCRL via Eq. (8), and the number of episodes T required to output a near-optimal robust policy (see Corollary 1). We start by analyzing a general robust model-based RL framework, and later on, we demonstrate the utility of the obtained results by specializing them to the important special case of Gaussian Process dynamics models. We defer all the proofs from this section to Appendix B. Before stating our main theoretical results, we introduce some additional assumptions:

Assumption 3 (Lipschitz continuity)**.** At every episode t, the functions Σ_t, any agent's and adversary's policies $\pi_t \in \Pi$, $\bar{\pi}_t \in \bar{\Pi}$, and the reward $r(\cdot,\cdot,\cdot)$ are Lipschitz continuous with respective constants L_σ, L_π, $L_{\bar{\pi}}$ and L_r.

The previous assumption is mild and has been used in non-robust model-based RL, see, e.g., Curi et al. (2020a), where it is noted that neural networks with Lipschitz-continuous non-linearities (or GPs with Lipschitz continuous kernels) output Lipschitz-continuous predictions. Furthermore, the policy classes Π and $\bar{\Pi}$, as well as the reward functions, are typically known and designed in a way that is compatible with the previous assumption.

Both the robust regret and sample complexity rates that we analyze depend on the difficulty of learning the underlying statistical model. Models that are easy to learn typically require fewer samples and allow algorithms to make better decisions sooner. To express the difficulty of learning the imposed calibrated model class, we use the following model-based complexity measure:

$$\Gamma_T := \max_{\tilde{\mathcal{D}}_{1:T}} \sum_{t=1}^{T} \sum_{(\mathbf{s},\mathbf{a},\bar{\mathbf{a}})\in\tilde{\mathcal{D}}_t} \|\Sigma_{t-1}(\mathbf{s},\mathbf{a},\bar{\mathbf{a}})\|_2^2 \tag{9}$$

where each $\tilde{\mathcal{D}}_t \subset \{\mathcal{S}\times\mathcal{A}\times\bar{\mathcal{A}}\}^H$. This quantity has a worst-case flavor as it considers the data (collected during T episodes by any algorithm) that lead to maximal total predictive uncertainty of the model. For the special case of RKHS/GP dynamics models, we show below that this quantity can be effectively bounded, and the bound is sublinear (in the number of episodes T) for most commonly used kernel functions.

General results. Now, we can state the main result of this section. In the following theorem, we bound the robust cumulative regret incurred by the policies from Eq. (7a).

Theorem 1. *Under Assumptions 1 to 3, let* $C = (1 + L_f + 2L_\sigma)(1 + L_\pi^2 + L_{\bar{\pi}}^2)^{1/2}$ *and let* $\mathbf{s}_{t,h} \in \mathcal{S}$, $\mathbf{a}_{t,h} \in \mathcal{A}$, $\bar{\mathbf{a}}_{t,h} \in \bar{\mathcal{A}}$

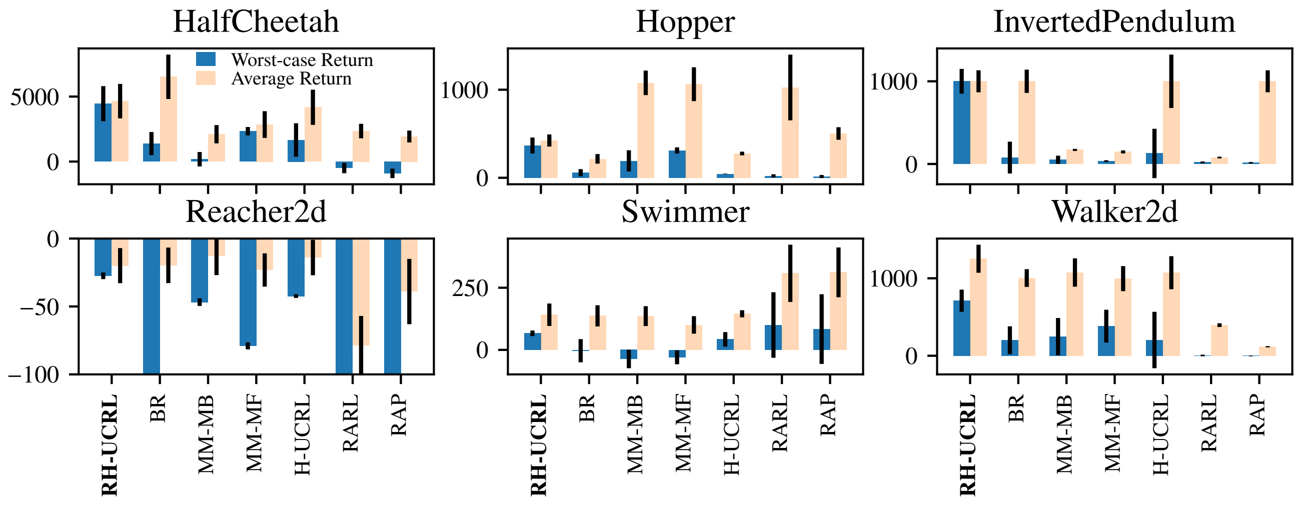

Figure 2. Worst-case and average return of different algorithms in the Adversarial-Robust Setting in Mujoco tasks. RH−UCRL outperforms the other algorithms in terms of worst-case return. The non-robust baseline, H−UCRL, has good average performance but poor worst-case performance (e.g., Inverted Pendulum). The deep robust RL baselines have worse sample complexity and often underperform. Our ablations are also non-robust, since exploration of both *agent* and *adversary* is crucial here to achieve robust performance.

for all $t, h > 0$. Then, for any fixed $H \geq 1$, with probability at least $1 - \delta$, the robust cumulative regret of RH−UCRL *is upper bounded by:*

$$R_T = \mathcal{O}\Big(L_r C^H \beta_T^H H^{3/2} \sqrt{T \Gamma_T}\Big).$$

This regret bound shows that RH−UCRL achieves sublinear robust regret when $\beta_T^H \sqrt{\Gamma_T} = o(\sqrt{T})$. Below, we show a concrete example of GP models where this is indeed the case. The obtained bound also depends on the Lipschitz constants from Assumption 3, as well as the episode length H that we assume is constant. The dependency of the regret bound on the problem dimension is hidden in Γ_T, while β_T depends also on δ (see Assumption 2).

Next, we characterize the number of episodes (samples) required by RH−UCRL to output ϵ-optimal robust policy. Our analysis upper bounds the optimal robust performance according to the confidence bounds from Assumption 2, but also addresses the challenge of characterizing the impact of exploring different adversary policies in Eq. (7b).

Corollary 1. *Consider the assumptions and setup of Theorem 1, and suppose that*

$$\frac{T}{\beta_T^{2H} \Gamma_T} \geq \frac{16 L_r^2 H^3 C^{2H}}{\epsilon^2}, \tag{10}$$

for some fixed $\epsilon > 0$ and $H \geq 1$. Then, with probability at least $1 - \delta$ after T episodes, RH−UCRL *achieves:*

$$\min_{\bar{\pi} \in \bar{\Pi}} J(f, \hat{\pi}_T, \bar{\pi}) \geq \min_{\bar{\pi} \in \bar{\Pi}} J(f, \pi^\star, \bar{\pi}) - \epsilon, \tag{11}$$

where $\hat{\pi}_T$ is the output of RH−UCRL*, reported according to Eq. (8), and $\pi^\star$ is the optimal robust policy given in Eq. (3).*

Gaussian Process Models. We specialize the regret bound obtained in Theorem 1 to the case of Gaussian Process (GP) models. GPs are popular statistical models that are frequently used to model unknown dynamics (Deisenroth & Rasmussen, 2011; Kamthe & Deisenroth, 2018; Curi et al., 2020b). These models are very expressive due to a versatility of possible kernel functions, and can naturally differentiate between aleatoric noise and epistemic uncertainty. Moreover, GPs are known to be provably well-calibrated when the unknown dynamics f are B_f-smooth as measured by the GP kernel.

In Appendix C, we recall the GP maximum information gain (MIG) which is a kernel-dependent quantity (first introduced by Srinivas et al. (2010)), that is frequently used in various GP optimization works to characterize complexity of learning a GP model. Sublinear upper bounds for MIG are known (c.f. Srinivas et al. (2010)) for most popularly used kernels (e.g., linear, squared-exponential, etc.), as well as for their compositions, e.g., additive kernels (Krause & Ong, 2011). We recall the known results and use MIG to express β_T and upper bound Γ_T in Theorem 1. For example, when we use independent GP models with either (i) linear or (ii) squared-exponential kernels, for every component, we obtain the following *sublinear* (in T) regret bounds $O(H^{3/2} p \left[(p+q+\bar{q}) \ln(pTH)\right]^{(H+1)/2} \sqrt{T})$ and $O(H^{3/2} p \left[\ln(pTH)\right]^{(p+q+\bar{q})(H+1)/2} \sqrt{T})$, respectively.

Finally, we note that the previously used MIG bounds require $\mathcal{S}$ to be compact, which does not hold under the considered noise model in Assumption 3. By bounding the domain w.h.p., Curi et al. (2020a) show that this only increases the

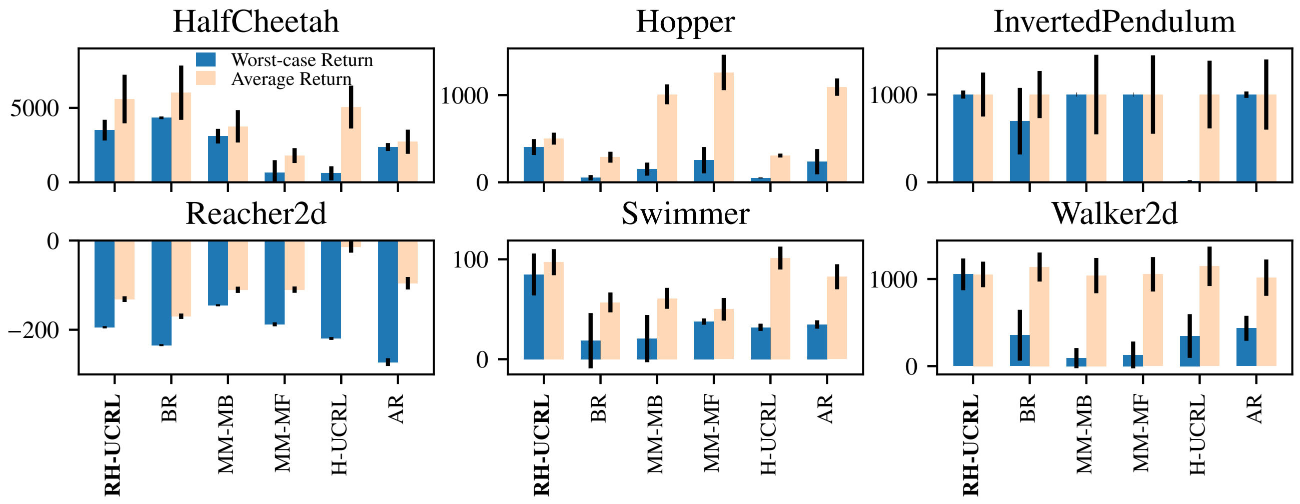

Figure 3. Average and worst-case return of different algorithms in the Noisy Action-Robust Setting in Mujoco tasks. `RH-UCRL` mostly outperforms other algorithms in terms of worst-case return. The non-robust baseline, `H-UCRL`, has good average performance but has an extreme drop in worst-case performance. Overall, the ablations perform better here than in the Adversarial-Robust setting.

MIG bounds (e.g., in case of the squared-exponential kernel) by at most a polylog(T) factor.

5. Robust RL Applications and Experiments

We now discuss concrete instantiations of `RH-UCRL` for three important robust RL scenarios: (i) *adversarial-robustness*, (ii) *action-robustness*, and (iii) *parameter-robustness*. In all of the above scenarios, we experimentally demonstrate that `RH-UCRL` outperforms or successfully competes with the state-of-the-art variants designed specifically for these settings. In Appendix D, we present experimental details and additional results.

Experimental Environments. We use the Mujoco suite (Todorov et al., 2012) to demonstrate the effectiveness of our algorithms in all the considered robust-RL settings. In particular, we use the Half Cheetah, Hopper, Inverted Pendulum, Reacher, Swimmer, and Walker robots.

Baselines. Besides the specific algorithms designed for each setting, we use `H-UCRL` (Curi et al., 2020a) as a non-robust baseline and three ablations derived from `RH-UCRL`, namely `MiniMax`, `MiniMaxMF` and `BestResponse`. The `MiniMax` algorithm is:

$$(\pi_t, \bar{\pi}_t) \in \arg\max_{\pi\in\Pi} \min_{\bar{\pi}\in\bar{\Pi}} J_t^{(e)}(\pi, \bar{\pi}), \tag{12}$$

where $J_t^{(e)}(\pi, \bar{\pi})$ corresponds to the *expected* performance, where the expectation is taken with respect to the aleatoric and epistemic uncertainty, i.e., none of the players actively explore. Next, the `MiniMaxMF` algorithm is a model-free implementation of Eq. (12) that uses SAC (Haarnoja et al., 2018) as the optimizer for each player. The `BestResponse` algorithm is:

$$\pi_t \in \arg\max_{\pi\in\Pi} \min_{\bar{\pi}\in\bar{\Pi}} J_t^{(o)}(\pi, \bar{\pi}), \tag{13a}$$

$$\bar{\pi}_t \in \arg\min_{\bar{\pi}\in\bar{\Pi}} J_t^{(e)}(\pi_t, \bar{\pi}). \tag{13b}$$

Thus, the agent is the same as in `RH-UCRL`, whereas the adversary simply plays the best-response to the agent's policy and does not perform exploration with pessimism. The goal of `BestResponse` is to analyze if exploration of the adversary through pessimism is empirically important, of `MaxiMin-MB` is to analyze if any exploration is empirically important, and of `MaxiMin-MF` is to analyze if using a model of the dynamics is beneficial.

5.1. Adversarial-Robust Reinforcement Learning

This setting is the most general one that we also consider in Section 4. The agent and the adversary can have distinct action spaces, which can also be seen as a particular instance of multi-agent RL with two competing agents. In the braking system motivating example, this can be used to model an adversarial state-dependent friction coefficient, e.g., icy roads. Having good robust performance in this setting implies braking robustly even with changing conditions.

The deep robust RL algorithms that we compare with are `RARL` (Pinto et al., 2017) and `RAP` (Vinitsky et al., 2020), and we use the adversarial action space proposed by Pinto et al. (2017). We train all algorithms for 200 episodes except for `RARL` and `RAP`; since they are on-policy algorithms, we train them for 1000 episodes. To evaluate *robust performance* (recall Eq. (4)), we freeze the output policy and train only its adversary by using `SAC` for 200 episodes.

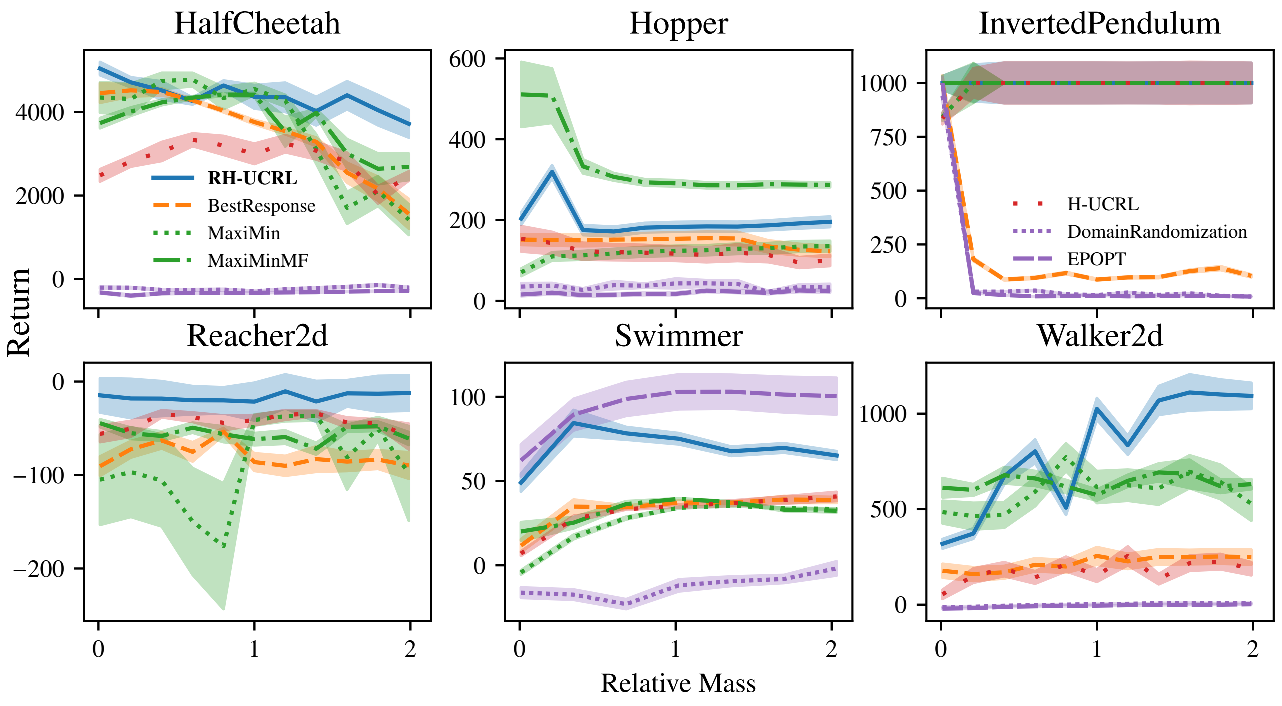

Figure 4. Returns of different algorithms in the Parameter-Robust Setting in Mujoco tasks for different masses during evaluation. Although `RH-UCRL` optimizes for the worst-case relative mass in this setting, it also performs well over different value of mass parameters.

In Figure 2, we show the *worst-case* and *average* returns on the different environments. In terms of *average* performance, there is no algorithm that performs better than others in all of the environments. On the other hand, comparing *worst-case* performance, `RH-UCRL` clearly outperforms the robust ablations, deep robust RL and non-robust baselines. For example, in the Inverted Pendulum stabilization task, `RH-UCRL` is the *only* algorithm that discovers a robust policy while all other algorithms severely fail. `BestResponse` and `RAP` manage to learn a policy that stabilizes the pendulum even when they learn with an adversary. However, when facing a *worst-case* adversary, they fail to complete the task.

Comparing `RH-UCRL` with non-robust `H-UCRL`, we see that in most environments it has comparable or better *worst-case* and *average* performance. This indicates that `RH-UCRL` is not only robust, but using an adversary during training practically helps with exploration. Pinto et al. (2017) also report similar findings regarding robust training. Comparing `RH-UCRL` with the ablations, we see that `RH-UCRL` achieves higher robust performance. From here, we conclude that exploring with both the agent and the adversary during training is crucial to achieve high robust performance in this setting. Finally, we see that both `RARL` and `RAP` have poor robust performance when trained for 1000 episodes, which demonstrates their sample inefficiency.

5.2. Action-Robust Reinforcement Learning

Tessler et al. (2019) introduce the action-robust setting, where both the agent and the adversary share the action space $\mathcal{A}$ and jointly execute a single action in the environment. This is useful, e.g., to model robustness to changes in the actuator dynamics, e.g., due to tire wear or incorrect pressure in a braking system. The action is sampled from a mixture policy $\mathbf{a}^{\text{mix}} \sim \pi^{\text{mix}} = \Upsilon_\alpha(\pi, \bar{\pi})$, where $\alpha \in [0,1]$ is a known parameter that controls the mixture proportion. One example of the mixture policy is the noisy-robust setting, in which $\Upsilon_\alpha(\pi, \bar{\pi}) = (1-\alpha)\pi + \alpha\bar{\pi}$. Another example is the noisy-robust setting, in which $\Upsilon_\alpha = \pi$ with probability $(1-\alpha)$ and $\Upsilon_\alpha = \bar{\pi}$ with probability α. The system evolves according to $\mathbf{s}_{h+1} = f'(\mathbf{s}_h, \mathbf{a}_h^{\text{mix}}) + \boldsymbol{\omega}_h$.

Besides the previous baselines, we compare to `AR-DDPG` (Tessler et al., 2019), and show the results of the experiment in Figure 3. Here, `RH-UCRL` is also comparable or better than the baselines in terms of *average* and *worst-case* returns. However, the ablations perform better than in the adversarial-robust setting. This is possibly due to the agent and adversary sharing the action space: The agent injects "enough" exploration to successfully learn both policies.

5.3. Parameter-Robust Reinforcement Learning

The goal in this setting is to be robust to changes in parameters, such as mass or friction, that can occur between training and test time. Being robust to a fixed parameter is equivalent to considering a stateless adversary policy in the RH-UCRL algorithm (7), i.e., $\bar{\Pi} : \varnothing \to \mathcal{A}$. Common benchmarks in this setting are domain randomization (Peng et al., 2018; Tobin et al., 2017) and EP-OPT (Rajeswaran et al., 2017). The former randomizes the parameters in the simulation and uses the *average* over these parameters as a surrogate of the maximum. The latter also randomizes the parameters but considers the CVaR as a surrogate of the maximum. As they are on-policy procedures, we train them using data for 1000 episodes. Finally, we evaluate the policies in different environments by varying the corresponding mass parameters.

We show the results of this setting in Figure 4. Although RH-UCRL optimizes for the worst-case parameter, it performs well over different mass parameter values, and, except in the Walker environment, its performance remains robust and nearly constant for different values of the mass parameter. H-UCRL is trained with nominal mass only (relative mass = 1), and it suffers in performance when varying the mass. This is most notable in the Half Cheetah environment (see Fig. 4). The robust variants, instead, can alter the mass during training and often perform better than H-UCRL. A particular case happens with the BestResponse algorithm in the Inverted Pendulum, where the adversary is greedy and so it swiftly chooses a small mass and never changes it during training. The agent learns only for this small mass and, when evaluated with different ones, it performs poorly. We also observe that in the Hopper, the MaxiMin-MF outperforms the MaxiMin-MB. The reason for this might be due to early stopping of the environment, as it is possible that the transitions collected in 200 episodes are not sufficient for learning the model, but allow for learning a policy in a model-free way.

6. Conclusion

We introduced the RH-UCRL algorithm, a practical algorithm for deep model-based robust RL. It uses optimistic and pessimistic estimates of the robust performance to efficiently explore both the agent and fictitious adversary decision spaces during policy learning. We showed that RH-UCRL is provably robust and we established sample-complexity and regret guarantees. We instantiated our algorithm in important robust-RL settings such as adversarial-robust RL, parameter-robust RL, and action-robust RL. Empirically, RH-UCRL outperforms state-of-the-art deep robust RL algorithms. Perhaps surprisingly, its discovered robust policies often attain better non-robust performance than the ones found by non-robust algorithms, indicating benefits of RH-UCRL for exploration.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation program grant agreement No 815943, ETH Zürich Postdoctoral Fellowship 19-2 FEL-47, and from the Swiss National Science Foundation under NCCR Automation under grant agreement 51NF40 180545.

References

Antos, A., Szepesvári, C., and Munos, R. Fitted Q-iteration in continuous action-space MDPs. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 9–16, 2008.

Auer, P., Jaksch, T., and Ortner, R. Near-optimal regret bounds for reinforcement learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 89–96, 2009.

Bai, Y. and Jin, C. Provable self-play algorithms for competitive reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 551–560, 2020.

Başar, T. and Bernhard, P. *H-infinity optimal control and related minimax design problems: a dynamic game approach*. Springer Science & Business Media, 2008.

Bemporad, A., Borrelli, F., and Morari, M. Min-max control of constrained uncertain discrete-time linear systems. *IEEE Transactions on automatic control*, 48(9):1600–1606, 2003.

Bogunovic, I., Scarlett, J., Jegelka, S., and Cevher, V. Adversarially robust optimization with Gaussian processes. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 5760–5770, 2018.

Camacho, E. F. and Alba, C. B. *Model predictive control*. Springer science & business media, 2013.

Chowdhury, S. R. and Gopalan, A. On kernelized multi-armed bandits. In *International Conference on Machine Learning (ICML)*, pp. 844–853, 2017.

Chowdhury, S. R. and Gopalan, A. Online learning in kernelized Markov decision processes. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 3197–3205, 2019.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 4754–4765, 2018.

Cover, T. M. and Thomas, J. A. Entropy, relative entropy and mutual information. *Elements of information theory*, 2(1):12–13, 1991.

Curi, S., Berkenkamp, F., and Krause, A. Efficient model-based reinforcement learning through optimistic policy search and planning. *Conference on Neural Information Processing Systems (NeurIPS)*, 33, 2020a.

Curi, S., Melchior, S., Berkenkamp, F., and Krause, A. Structured variational inference in partially observable unstable gaussian process state space models. In *Learning for Dynamics and Control*, pp. 147–157, 2020b.

Deisenroth, M. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on machine learning (ICML)*, pp. 465–472, 2011.

Der Kiureghian, A. and Ditlevsen, O. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

Desautels, T., Krause, A., and Burdick, J. W. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *Journal of Machine Learning Research (JMLR)*, 15:3873–3923, 2014.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Durand, A., Maillard, O.-A., and Pineau, J. Streaming kernel regression with provably adaptive mean, variance, and regularization. *The Journal of Machine Learning Research (JMLR)*, 19(1):650–683, 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pp. 1861–1870, 2018.

Iyengar, G. N. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

Kamalaruban, P., Huang, Y.-T., Hsieh, Y.-P., Rolland, P., Shi, C., and Cevher, V. Robust reinforcement learning via adversarial training with Langevin dynamics. *Conference on Neural Information Processing Systems (NeurIPS)*, 33, 2020.

Kamthe, S. and Deisenroth, M. Data-efficient reinforcement learning with probabilistic model predictive control. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1701–1710, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kirschner, J., Bogunovic, I., Jegelka, S., and Krause, A. Distributionally robust Bayesian optimization. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 2174–2184, 2020.

Krause, A. and Ong, C. S. Contextual Gaussian process bandit optimization. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 2447–2455, 2011.

Lagoudakis, M. G. and Parr, R. Value function approximation in zero-sum Markov games. In *Conference on Uncertainty in artificial intelligence (UAI)*, pp. 283–292, 2002.

Littman, M. L. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings*, pp. 157–163. Elsevier, 1994.

Littman, M. L. and Szepesvári, C. A generalized reinforcement-learning model: Convergence and applications. In *International Conference on Machine Learning (ICML)*, pp. 310–318, 1996.

Malik, A., Kuleshov, V., Song, J., Nemer, D., Seymour, H., and Ermon, S. Calibrated model-based deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 4314—4323, 2019.

Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. Monte Carlo gradient estimation in machine learning. *arXiv preprint arXiv:1906.10652*, 2019.

Nilim, A. and El Ghaoui, L. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

Osband, I., Blundell, C., Pritzel, A., and Roy, B. V. Deep exploration via bootstrapped DQN. *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 4033–4041, 2016.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE international conference on robotics and automation (ICRA)*, pp. 1–8. IEEE, 2018.

Perolat, J., Scherrer, B., Piot, B., and Pietquin, O. Approximate dynamic programming for two-player zero-sum Markov games. In *International Conference on Machine Learning (ICML)*, pp. 1321–1329, 2015.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 2817–2826, 2017.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. Epopt: Learning robust neural network policies using model ensembles. In *International Conference on Learning Representations (ICLR)*, 2017.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, pp. 387–395, 2014.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. Gaussian process optimization in the bandit setting: no regret and experimental design. In *International Conference on Machine Learning (ICML)*, pp. 1015–1022, 2010.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tamar, A., Mannor, S., and Xu, H. Scaling up robust MDPs using function approximation. In *International Conference on Machine Learning (ICML)*, pp. 181–189, 2014.

Tessler, C., Efroni, Y., and Mannor, S. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning (ICML)*, pp. 6215–6224, 2019.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30. IEEE, 2017.

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

Valko, M., Korda, N., Munos, R., Flaounas, I., and Cristianini, N. Finite-time analysis of kernelised contextual bandits. In *Uncertainty in Artificial Intelligence (UAI)*, pp. 654, 2013.

van Hasselt, H., Guez, A., Hessel, M., Mnih, V., and Silver, D. Learning values across many orders of magnitude. *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 4294–4302, 2016.

Vinitsky, E., Du, Y., Parvate, K., Jang, K., Abbeel, P., and Bayen, A. Robust reinforcement learning using adversarial populations. *arXiv preprint arXiv:2008.01825*, 2020.

Wiesemann, W., Kuhn, D., and Rustem, B. Robust Markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

Zhang, K., Kakade, S., Basar, T., and Yang, L. Model-based multi-agent rl in zero-sum markov games with near-optimal sample complexity. *Conference on Neural Information Processing Systems (NeurIPS)*, 33, 2020.