WASSERSTEIN ROBUST REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning algorithms, though successful, tend to over-fit to training environments, thereby hampering their application to the real-world. This paper proposes WR^{2}L – a robust reinforcement learning algorithm with significant robust performance on low and high-dimensional control tasks. Our method formalises robust reinforcement learning as a max-min game with a constraint derived from Wasserstein distance, and we propose an efficient and scalable algorithm to solve it. Our algorithm applies zero-order optimisation method that we believe can be useful to numerical optimisation in general. Apart from the formulation, we also propose an efficient and scalable solver following a novel zero-order optimisation method that we believe can be useful to numerical optimisation in general. We empirically demonstrate significant gains compared to standard and robust state-of-the-art algorithms on high-dimensional MuJuCo environments.

1 INTRODUCTION

Reinforcement learning (RL) has become a standard tool for solving decision-making problems with feedback, and though significant progress has been made, algorithms often over-fit to training environments and fail to generalise across even slight variations of transition dynamics (Packer et al., 2018; Zhao et al., 2019). Robustness to changes in transition dynamics is a crucial component for adaptive and safe RL in real-world environments. Motivated by real-world applications, recent literature has focused on the above problems, proposing a plethora of algorithms for robust decision-making (Morimoto & Doya, 2005; Pinto et al., 2017; Tessler et al., 2019). Most of these techniques borrow from game theory to analyse, typically in a discrete state and actions spaces, worst-case deviations of agents' policies and/or environments, see Sargent & Hansen (2001); Nilim & El Ghaoui (2005); Iyengar (2005); Namkoong & Duchi (2016) and references therein. These methods have also been extended to linear function approximators (Chow et al., 2015), and deep neural networks (Peng et al., 2017) showing (modest) improvements in performance gain across a variety of disturbances, e.g., action uncertainties, or dynamical model variations. In this paper, we propose a generic framework for robust reinforcement learning that can cope with both discrete and continuous state and actions spaces. Our algorithm, termed *Wasserstein Robust Reinforcement Learning* (WR^{2}L), aims to find the best policy, where any given policy is judged by the worst-case dynamics amongst all candidate dynamics in a certain set. This set is essentially the average Wasserstein ball around a reference dynamics $\mathcal{P}_0$. The constraints makes the problem well-defined, as searching over arbitrary dynamics can only result in non-performing system. The measure of performance is the standard RL objective, the expected return. Both the policy and the dynamics are parameterised; the policy parameters $\boldsymbol{\theta}_k$ may be the weights of a deep neural network, and the dynamics parameters $\boldsymbol{\phi}_j$ the parameters of a simulator or differential equation solver. The algorithm performs estimated descent steps in $\boldsymbol{\phi}$ space and - after (almost) convergence - performs an update of policy parameters, i.e., in $\boldsymbol{\theta}$ space. Since $\boldsymbol{\phi}_j$ may be high-dimensional, we adapt a zero'th order sampling method based extending Salimans et al. (2017) to make estimations of gradients, and in order to define the constraint set which $\boldsymbol{\phi}_j$ is bounded by, we generalise the technique to estimate Hessians (Proposition 2). We emphasise that although access to a simulator with parameterisable dynamics is required, the actual reference dynamics $\mathcal{P}_0$ need not be known explicitly nor learnt by our algorithm. Put another way, we are in the "RL setting", not the "MDP setting" where the transition probability matrix is known *a priori*. The difference is made obvious, for example, in the fact that we cannot perform dynamic programming, and the determination of a particular probability transition can only be estimated from sampling, not retrieved explicitly. Hence, our algorithm is not model-based in the traditional sense of learning a model to perform planning.

We believe our contribution is useful and novel for two main reasons. Firstly, our framing of the robust learning problem is in terms of dynamics uncertainty sets defined by Wasserstein distance. Whilst we are not the first to introduce the Wasserstein distance into the context of MDPs (see, e.g., Yang (2017) or Lecarpentier & Rachelson (2019)), we believe our formulation is amongst the first suitable for application to the demanding application-space we desire, that being, high-dimensional, continuous state and action spaces. Secondly, we believe our solution approach is both novel and effective (as evidenced by experiments below, see Section 5), and does not place a great demand on model or domain knowledge, merely access to a simulator or differentiable equation solver that allows for the parameterisation of dynamics. Furthermore, it is not computationally demanding, in particular, because it does not attempt to build a model of the dynamics, and operations involving matrices are efficiently executable using the Jacobian-vector product facility of automatic differentiation engines.

2 Background

A Markov decision process (MDP)[1] is denoted by $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S} \subseteq \mathbb{R}^d$ denotes the state space, $\mathcal{A} \subseteq \mathbb{R}^n$ the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a state transition probability describing the system's dynamics, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function measuring the agent's performance, and $\gamma \in [0, 1)$ specifies the degree to which rewards are discounted over time.

At each time step t, the agent is in state $\boldsymbol{s}_t \in \mathcal{S}$ and must choose an action $\boldsymbol{a}_t \in \mathcal{A}$, transitioning it to a new state $\boldsymbol{s}_{t+1} \sim \mathcal{P}\left(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t\right)$, and yielding a reward $\mathcal{R}(\boldsymbol{s}_t, \boldsymbol{a}_t)$. A policy $\pi : \mathcal{S} \times A \to [0, 1]$ is defined as a probability distribution over state-action pairs, where $\pi(\boldsymbol{a}_t | \boldsymbol{s}_t)$ represents the density of selecting action $\boldsymbol{a}_t$ in state $\boldsymbol{s}_t$. Upon subsequent interactions with the environment, the agent collects a trajectory $\boldsymbol{\tau}$ of state-action pairs. The goal is to determine an optimal policy $\pi^\star$ by solving:

$$\pi^\star = \arg\max_\pi \mathbb{E}_{\boldsymbol{\tau} \sim p_\pi(\boldsymbol{\tau})} \left[\mathcal{R}_{\text{Total}}(\boldsymbol{\tau})\right], \tag{1}$$

where $p_\pi(\boldsymbol{\tau})$ denotes the trajectory density function, and $\mathcal{R}_{\text{Total}}(\boldsymbol{\tau})$ the return, that is, the total accumulated reward:

$$p_\pi(\boldsymbol{\tau}) = \mu_0(\boldsymbol{s}_0)\pi(\boldsymbol{a}_0|\boldsymbol{s}_0) \prod_{t=1}^{T-1} \mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)\pi(\boldsymbol{a}_t|\boldsymbol{s}_t) \text{ and } \mathcal{R}_{\text{Total}}(\boldsymbol{\tau}) = \sum_{t=0}^{T-1} \gamma^t \mathcal{R}(\boldsymbol{s}_t, \boldsymbol{a}_t), \tag{2}$$

with $\mu_0(\cdot)$ denoting the initial state distribution.

We make use of the ***Wasserstein distance*** to quantify variations from a reference transition density $\mathcal{P}_0(\cdot)$. The latter being a probability distribution, one may consider other divergences, such as Kullback-Leibler (KL) or total variation (TV). Our main intuition for choosing Wasserstein distance is explained below, but we note that it has a number of desirable properties: Firstly, it is symmetric ($W_p(\mu, \nu) = W_p(\nu, \mu)$, a property K-L lacks). Secondly, it is well-defined for measures with different supports (which K-L also lacks). Indeed, the Wasserstein distance is flexible in the forms of the measures that can be compared - discrete, continuous or a mixture. Finally, it takes into account the underlying geometry of the space the distributions are defined on, which can encode valuable information. It is defined as follows: Let $\mathcal{X}$ be a metric space with metric $d(\cdot, \cdot)$. Let $\mathcal{C}(\mathcal{X})$ be the space of continuous functions on $\mathcal{X}$ and let $\mathcal{M}(\mathcal{X})$ be the set of probability measures on $\mathcal{X}$. Let $\mu, \nu \in \mathcal{M}(\mathcal{X})$. Let $\mathbf{K}(\mu, \nu)$ be the set of couplings between μ, ν:

$$\mathbf{K}(\mu, \nu) := \{\kappa \in \mathcal{M}(\mathcal{X} \times \mathcal{X})\,;\, \forall (A, B) \subset \mathcal{X} \times \mathcal{X}, \kappa(A \times \mathcal{X}) = \mu(A), \kappa(\mathcal{X} \times B) = \nu(B)\} \tag{3}$$

That is, the set of joint distributions $\kappa \in \mathcal{M}(\mathcal{X} \times \mathcal{X})$ whose marginals agree with μ and ν respectively. Given a metric (serving as a cost function) $d(\cdot, \cdot)$ for $\mathcal{X}$, the p'th Wasserstein distance $W_p(\mu, \nu)$ for $p \geq 1$ between μ and ν is defined as:

$$W_p(\mu, \nu) := \left(\min_{\kappa \in \mathbf{K}(\mu, \nu)} \int_{\mathcal{X} \times \mathcal{Y}} d(x, y)^p d\kappa(x, y)\right)^{1/p} \tag{4}$$

(in this paper, and mostly for computational convenience, we use $p = 2$, though other values of p are applicable).

[1]Please note that we present reinforcement learning with continuous states and actions. This allows us to easily draw similarities to optimal control as detailed later. Extending these notions to discrete settings is relatively straight-forward.

3 Wasserstein Robust Reinforcement Learning

The desirable properties of Wasserstein distance aside, our main intuition for choosing it is described thus: Per the definition, constraining the possible dynamics to be within an ϵ-Wasserstein ball of a reference dynamics $\mathcal{P}_0(\cdot)$ means constraining it in a certain way. Wasserstein distance has the form mass $\times$ distance. If this quantity is constrained to be less than a constant ϵ, then if the mass is large, the distance is small, and if the distance is large, the mass is small. Intuitively, when modelling the dynamics of a system, it may be reasonable to concede that there could be a systemic error - or bias - in the model, but that bias should not be too large. It is also reasonable to suppose that occasionally, the behaviour of the system may be wildly different to the model, but that this should be a low-probability event. If the model is frequently wrong by a large amount, then there is no use in it. In a sense, the Wasserstein ball formalises these assumptions.

3.1 Problem Definition: Robust Objectives and Constraints

Due to the continuous nature of the state and action spaces considered in this work, we resort to deep neural networks to parameterise policies, which we write as $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)$, where $\boldsymbol{\theta} \in \mathbb{R}^{d_1}$ is a set of tunable hyper-parameters to optimise. For instance, these policies can correspond to multi-layer perceptrons for MuJoCo environments, or to convolutional neural networks in case of high-dimensional states depicted as images. Exact policy details are ultimately application dependent and, consequently, provided in the relevant experiment sections.

In principle, one can similarly parameterise dynamics models using deep networks (e.g., LSTM-type models) to provide one or more action-conditioned future state predictions. Though appealing, going down this path led us to agents that discover worst-case transition models which minimise training data but lack any valid physical meaning. For instance, original experiments we conducted on CartPole ended up involving transitions that alter angles without any change in angular velocities. More importantly, these effects became more apparent in high-dimensional settings where the number of potential minimisers increases significantly. It is worth noting that we are not the first to realise such an artifact when attempting to model physics-based dynamics using deep networks. Authors in (Lutter et al., 2019) remedy these problems by introducing Lagrangian mechanics to deep networks, while others (Koller et al., 2018; Chen et al., 2018) argue the need to model dynamics given by differential equation structures directly.

Though incorporating physics-based priors to deep networks is an important and challenging task that holds the promise of scaling model-based reinforcement learning for efficient solvers, in this paper we rather study an alternative direction focusing on perturbing differential equation solvers and/or simulators with respect to the dynamic specification parameters $\boldsymbol{\phi} \in \mathbb{R}^{d_2}$. Not only would such a consideration reduce the dimension of parameter spaces representing transition models, but would also guarantee valid dynamics due to the nature of the simulator. Though tackling some of the above problems, such a direction arrives with a new set of challenges related to computing gradients and Hessians of black-box solvers. In Section 4, we develop an efficient and scalable zero-order method for valid and accurate model updates.

Unconstrained Loss Function: Having equipped agents with the capability of representing policies and perturbing transition models, we are now ready to present an unconstrained version of WR^{2}L's loss function. Borrowing from robust optimal control, we define robust reinforcement learning as an algorithm that learns best-case policies under worst-case transitions:

$$\max_{\boldsymbol{\theta}} \left[\min_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{\tau} \sim p_{\boldsymbol{\theta}}^{\boldsymbol{\phi}}(\boldsymbol{\tau})} \left[\mathcal{R}_{\text{total}}(\boldsymbol{\tau}) \right] \right], \tag{5}$$

where $p_{\boldsymbol{\theta}}^{\boldsymbol{\phi}}(\boldsymbol{\tau})$ is a trajectory density function parameterised by both policies and transition models, i.e., $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, respectively:

$$p_{\boldsymbol{\theta}}^{\boldsymbol{\phi}}(\boldsymbol{\tau}) = \mu_0(\boldsymbol{s}_0)\pi(\boldsymbol{a}_0|\boldsymbol{s}_0) \prod_{t=1}^{T-1} \underbrace{\mathcal{P}_{\boldsymbol{\phi}}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)}_{\text{specs vector and diff. solver}} \underbrace{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}_{\text{deep network}}.$$

At this stage, it should be clear that our formulation, though inspired from robust optimal control, is, truthfully, more generic as it allows for parameterised classes of transition models without incorporating additional restrictions on the structure or the scope by which variations are executed[2].

Constraints & Complete Problem Definition: Clearly, the problem in Equation 5 is ill-defined due to the arbitrary class of parameterised transitions. To ensure well-behaved optimisation objectives, we next introduce constraints to bound search spaces and ensure convergence to feasible transition models. For a valid constraint set, our method assumes access to samples from a *reference dynamics model* $\mathcal{P}_0(\cdot|\boldsymbol{s},\boldsymbol{a})$, and bounds learnt transitions in an ϵ-Wasserstein ball around $\mathcal{P}_0(\cdot|\boldsymbol{s},\boldsymbol{a})$, i.e., the set defined as:

$$\mathcal{W}_\epsilon\left(\mathcal{P}_{\boldsymbol{\phi}}(\cdot),\mathcal{P}_0(\cdot)\right)=\left\{\mathcal{P}_{\boldsymbol{\phi}}(\cdot):\mathcal{W}_2^2\left(\mathcal{P}_{\boldsymbol{\phi}}(\cdot|\boldsymbol{s},\boldsymbol{a}),\mathcal{P}_0(\cdot|\boldsymbol{s},\boldsymbol{a})\right)\leq\epsilon,\ \forall(\boldsymbol{s},\boldsymbol{a})\in\mathcal{S}\times\mathcal{A}\right\},\tag{6}$$

where $\epsilon\in\mathbb{R}_+$ is a hyperparameter used to specify the "degree of robustness" in a similar spirit to maximum norm bounds in robust optimal control. It is worth noting, that though we have access to samples from a reference simulator, our setting is by no means restricted to model-based reinforcement learning in an MDP setting. <u>That is, our algorithm operates successfully given only traces from $\mathcal{P}_0$ accompanied with its specification parameters, e.g., pole lengths, torso masses, etc. – a more flexible framework that does not require full model learners.</u>

Though defining a better behaved optimisation objective, the set in Equation 6 introduces infinite number of constraints when considering continuous state and/or actions spaces. To remedy this problem, we target a relaxed version that considers a constraint of *average* Wasserstein distance bounded by a hyperparameter ϵ:

$$\begin{aligned}\hat{\mathcal{W}}_\epsilon^{(\text{average})}\left(\mathcal{P}_{\boldsymbol{\phi}}(\cdot),\mathcal{P}_0(\cdot)\right)&=\left\{\mathcal{P}_{\boldsymbol{\phi}}(\cdot):\int_{(\boldsymbol{s},\boldsymbol{a})}\mathcal{P}(\boldsymbol{s},\boldsymbol{a})\mathcal{W}_2^2\left(\mathcal{P}_{\boldsymbol{\phi}}(\cdot|\boldsymbol{s},\boldsymbol{a}),\mathcal{P}_0(\cdot|\boldsymbol{s},\boldsymbol{a})\right)d(\boldsymbol{s},\boldsymbol{a})\leq\epsilon\right\}\\&=\left\{\mathcal{P}_{\boldsymbol{\phi}}(\cdot):\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})}\left[\mathcal{W}_2^2\left(\mathcal{P}_{\boldsymbol{\phi}}(\cdot|\boldsymbol{s},\boldsymbol{a}),\mathcal{P}_0(\cdot|\boldsymbol{s},\boldsymbol{a})\right)\right]\leq\epsilon\right\}\end{aligned}\tag{7}$$

The sampling $(\boldsymbol{s},\boldsymbol{a})$ in the expectation is done as follows: We sample trajectories using reference dynamics $\mathcal{P}_0$ and a policy π that chooses actions uniformly at random (**uar**). Then $(\boldsymbol{s},\boldsymbol{a})$ pairs are sampled **uar** from those collected trajectories. For a given pair $(\boldsymbol{s},\boldsymbol{a})$, $\mathcal{W}_2^2\left(\mathcal{P}_{\boldsymbol{\phi}}(\cdot|\boldsymbol{s},\boldsymbol{a}),\mathcal{P}_0(\cdot|\boldsymbol{s},\boldsymbol{a})\right)$ is approximated through the empirical distribution: we use the state that followed $(\boldsymbol{s},\boldsymbol{a})$ in the collected trajectories as a data point. Estimating Wasserstein distance using empirical data is standard, see, e.g., Peyré et al. (2019). One approach which worked well in our experiments, was to assume that the dynamics are given by deterministic functions plus Gaussian noise with diagonal convariance matrices. This makes estimation easier in high dimensions since sampling in each dimension is independent of others, and the total samples needed is a constant factor of the number of dimensions. Gaussian distributions also have closed-form expressions for Wasserstein distance, given in terms of mean and covariance.

We thus arrive at WR^{2}L's optimisation problem allowing for best policies under worst-case yet bounded transition models:

<u>**Wasserstein Robust Reinforcement Learning Objective:**</u>

$$\max_{\boldsymbol{\theta}}\left[\min_{\boldsymbol{\phi}}\mathbb{E}_{\boldsymbol{\tau}\sim p_{\boldsymbol{\theta}}^{\phi}(\boldsymbol{\tau})}\left[\mathcal{R}_{\text{total}}(\boldsymbol{\tau})\right]\right]\ \text{ s.t. }\ \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})}\left[\mathcal{W}_2^2\left(\mathcal{P}_{\boldsymbol{\phi}}(\cdot|\boldsymbol{s},\boldsymbol{a}),\mathcal{P}_0(\cdot|\boldsymbol{s},\boldsymbol{a})\right)\right]\leq\epsilon\tag{8}$$

3.2 Solution Methodology

Our solution alternates between updates of $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, keeping one fixed when updating the other. Fixing dynamics parameters $\boldsymbol{\phi}$, policy parameters $\boldsymbol{\theta}$ can be updated by solving $\max_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{\tau}\sim p_{\boldsymbol{\theta}}^{\phi}(\boldsymbol{\tau})}\left[\mathcal{R}_{\text{total}}(\boldsymbol{\tau})\right]$, which is the formulation of a standard RL problem. Consequently, one can easily adapt any policy search method for updating policies under fixed dynamical models. As

[2]Of course, allowed perturbations are ultimately constrained by the hypothesis space. Even then, our model is more general compared to robust optimal control that assumes additive, multiplicative, or other forms of disturbances.

described later in Section 4, we make use of proximal policy optimisation (Schulman et al., 2017). When updating $\boldsymbol{\phi}$ given a set of fixed policy parameters $\boldsymbol{\theta}$, the Wasserstein constraints must be respected. Unfortunately, even with the simplification introduced in Section 3.1 the constraint is still difficult to compute.

To alleviate this problem, we propose to approximate the constraint in (8) by its Taylor expansion up to second order. That is, defining $W(\boldsymbol{\phi}) := \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})}\left[\mathcal{W}_2^2\left(\mathcal{P}_{\boldsymbol{\phi}}(\cdot|\boldsymbol{s},\boldsymbol{a}),\mathcal{P}_0(\cdot|\boldsymbol{s},\boldsymbol{a})\right)\right]$, the above can be approximated around $\boldsymbol{\phi}_0$ by a second-order Taylor as:

$$W(\boldsymbol{\phi}) \approx W(\boldsymbol{\phi}_0) + \nabla_{\boldsymbol{\phi}} W(\boldsymbol{\phi}_0)^{\mathsf{T}}\left(\boldsymbol{\phi}-\boldsymbol{\phi}_0\right) + \frac{1}{2}\left(\boldsymbol{\phi}-\boldsymbol{\phi}_0\right)^{\mathsf{T}}\nabla^2_{\boldsymbol{\phi}} W(\boldsymbol{\phi}_0)\left(\boldsymbol{\phi}-\boldsymbol{\phi}_0\right).$$

Recognising that $W(\boldsymbol{\phi}_0) = 0$ (the distance between the same probability densities), and $\nabla_{\boldsymbol{\phi}} W(\boldsymbol{\phi}_0) = 0$ since $\boldsymbol{\phi}_0$ minimises $W(\boldsymbol{\phi})$, we can simplify the Hessian approximation by writing: $W(\boldsymbol{\phi}) \approx \frac{1}{2}(\boldsymbol{\phi}-\boldsymbol{\phi}_0)^{\mathsf{T}}\nabla^2_{\boldsymbol{\phi}} W(\boldsymbol{\phi}_0)(\boldsymbol{\phi}-\boldsymbol{\phi}_0)$. Substituting our approximation back in the original problem in Equation 8, we reach the following optimisation problem for determining model parameter given fixed policies:

$$\min_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{\tau}\sim p_{\boldsymbol{\theta}}^{\boldsymbol{\phi}}(\boldsymbol{\tau})}\left[\mathcal{R}_{\text{total}}(\boldsymbol{\tau})\right] \text{ s.t. } \frac{1}{2}(\boldsymbol{\phi}-\boldsymbol{\phi}_0)^{\mathsf{T}}\boldsymbol{H}_0(\boldsymbol{\phi}-\boldsymbol{\phi}_0) \le \epsilon, \tag{9}$$

where $\boldsymbol{H}_0 = \nabla^2_{\boldsymbol{\phi}}\,\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})}\left[\mathcal{W}_2^2\left(\mathcal{P}_{\boldsymbol{\phi}}(\cdot|\boldsymbol{s},\boldsymbol{a}),\mathcal{P}_0(\cdot|\boldsymbol{s},\boldsymbol{a})\right)\right]\Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}_0}$ is the Hessian of the expected squared 2-Wasserstein distance evaluated at $\boldsymbol{\phi}_0$. Optimisation problems with quadratic constraints can be efficiently solved using interior-point methods. To do so, one typically approximates the loss with a first-order expansion and determines a closed-form solution. Consider a pair of parameters $\boldsymbol{\theta}^{[k]}$ and $\boldsymbol{\phi}^{[j]}$ (which will correspond to parameters of the j'th inner loop of the k'th outer loop in the algorithm we present). To find $\boldsymbol{\phi}^{[j+1]}$, we solve:

$$\min_{\boldsymbol{\phi}} \nabla_{\boldsymbol{\phi}}\mathbb{E}_{\boldsymbol{\tau}\sim p_{\boldsymbol{\theta}}^{\boldsymbol{\phi}}(\boldsymbol{\tau})}\left[\mathcal{R}_{\text{total}}(\boldsymbol{\tau})\right]\Big|_{\boldsymbol{\theta}^{[k]},\boldsymbol{\phi}^{[j]}}^{\mathsf{T}}\left(\boldsymbol{\phi}-\boldsymbol{\phi}^{[j]}\right) \text{ s.t. } \frac{1}{2}(\boldsymbol{\phi}-\boldsymbol{\phi}_0)^{\mathsf{T}}\boldsymbol{H}_0(\boldsymbol{\phi}-\boldsymbol{\phi}_0) \le \epsilon.$$

It is easy to show that a minimiser to the above equation can derived in a closed-form as:

$$\boldsymbol{\phi}^{[j+1]} = \boldsymbol{\phi}_0 - \sqrt{\frac{2\epsilon}{\boldsymbol{g}^{[k,j]\mathsf{T}}\boldsymbol{H}_0^{-1}\boldsymbol{g}^{[k,j]}}}\,\boldsymbol{H}_0^{-1}\boldsymbol{g}^{[k,j]}, \tag{10}$$

with $\boldsymbol{g}^{[k,j]}$ denoting the gradient[3] evaluated at $\boldsymbol{\theta}^{[k]}$ and $\boldsymbol{\phi}^{[j]}$, i.e., $\boldsymbol{g}^{[k,j]} = \nabla_{\boldsymbol{\phi}}\mathbb{E}_{\boldsymbol{\tau}\sim p_{\boldsymbol{\theta}}^{\boldsymbol{\phi}}(\boldsymbol{\tau})}\mathbb{E}\left[\mathcal{R}_{\text{total}}(\boldsymbol{\tau})\right]|_{\boldsymbol{\theta}^{[k]},\boldsymbol{\phi}^{[j]}}$.

Generic Algorithm: Having described the two main steps needed for updating policies and models, we now summarise these findings in the pseudo-code in Algorithm 1. As the Hessian[4] of the Wasserstein distance is evaluated based on reference dynamics and any policy π, we pass it, along with ϵ and $\boldsymbol{\phi}_0$ as inputs. Then Algorithms 1 operates in a descent-ascent fashion in two main phases. In the first, lines 5 to 10 in Algorithm 1, dynamics parameters are updated using the closed-form solution in Equation 10, while ensuring that learning rates abide by a step size condition (we used the Wolfe conditions (Wolfe, 1969), though it can be some other method). With this, the second phase (line 11) utilises any state-of-the-art reinforcement learning method to adapt policy parameters generating $\boldsymbol{\theta}^{[k+1]}$. Regarding the termination condition for the inner loop, we leave this as a decision for the user. It could be, for example, a large finite time-out, or the norm of the gradient $\boldsymbol{g}^{[k,j]}$ being below a threshold, or whichever happens first.

[3]**Remark:** Although this looks superficially similar to an approximation made in TRPO (Schulman et al., 2015a), the latter aims to optimise the *policy parameter* rather than dynamics. Furthermore, the constraint is based on the Kullback-Leibler divergence rather than the Wasserstein distance

[4]Please note our algorithm does not need to store the Hessian matrix. In line 7 of the algorithm, it is clear that we require Hessian-vector products. These can be easily computed using computational graphs without having access to the full Hessian matrix.

Algorithm 1: Wasserstein Robust Reinforcement Learning

```
1: Inputs: Wasserstein distance Hessian, $\boldsymbol{H}_0$ evaluated at $\boldsymbol{\phi}_0$ under any policy $\pi$, radius of the
   Wasserstein ball $\epsilon$, and the reference simulator specification parameters $\boldsymbol{\phi}_0$
2: Initialise $\boldsymbol{\phi}^{[0]}$ with $\boldsymbol{\phi}_0$ and policy parameters $\boldsymbol{\theta}^{[0]}$ arbitrarily
3: for $k = 0, 1, \dots$ do
4:     $\boldsymbol{x}^{[0]} \leftarrow \boldsymbol{\phi}^{[0]}$, and $j \leftarrow 0$
5:     Phase I: Update model parameter while fixing the policy:
6:     while termination condition not met do
7:         Compute descent direction for the model parameters as given by Equation 10:
           $$\boldsymbol{p}^{[j]} \leftarrow \boldsymbol{\phi}_0 - \sqrt{\frac{2\epsilon}{\boldsymbol{g}^{[k,j]\mathsf{T}}\boldsymbol{H}_0^{-1}\boldsymbol{g}^{[k,j]}}}\boldsymbol{H}_0^{-1}\boldsymbol{g}^{[k,j]} - \boldsymbol{x}^{[j]}$$
8:         Update candidate solution, while satisfying step size conditions (see discussion below) on
           the learning rate $\alpha$: $\boldsymbol{x}^{[j+1]} \leftarrow \boldsymbol{x}^{[j]} + \alpha\boldsymbol{p}^{[j]}$ and $j \leftarrow j + 1$
9:     end while
10:    Perform model update setting $\boldsymbol{\phi}^{[k+1]} \leftarrow \boldsymbol{x}^{[j]}$
11:    Phase II: Update policy given new model parameters:
12:    Use any standard reinforcement learning algorithm for ascending in the gradient direction,
       e.g., $\boldsymbol{\theta}^{[k+1]} \leftarrow \boldsymbol{\theta}^{[k]} + \beta^{[k]}\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{\tau}\sim p_{\boldsymbol{\theta}}^{\boldsymbol{\phi}}(\boldsymbol{\tau})}\left[\mathcal{R}_{\text{total}}(\boldsymbol{\tau})\right]|_{\boldsymbol{\theta}^{[k]},\boldsymbol{\phi}^{[k+1]}}$, with $\beta^{[k]}$ a learning rate.
13: end for
```

4 Zero'th Order Wasserstein Robust Reinforcement Learning

Consider a simulator (or differential equation solver) $\mathbb{S}_{\boldsymbol{\phi}}$ for which the dynamics are parameterised by a real vector $\boldsymbol{\phi}$, and for which we can execute steps of a trajectory (i.e., the simulator takes as input an action $\boldsymbol{a}$ and gives back a successor state and reward). For generating novel physics-grounded transitions, one can simply alter $\boldsymbol{\phi}$ and execute the instruction in $\mathbb{S}_{\boldsymbol{\phi}}$ from some a state $\boldsymbol{s} \in \mathcal{S}$, while applying an action $\boldsymbol{a} \in \mathcal{A}$. Not only does this ensure valid (under mechanics) transitions, but also promises scalability as specification parameters typically reside in lower dimensional spaces compared to the number of tuneable weights when using deep networks as transition models.

Gradient Estimation: Recalling the update rule in Phase I of Algorithm 1, we realise the need for, estimating the gradient of the loss function with respect to the vector specifying the dynamics of the environment, i.e., $\boldsymbol{g}^{[k,j]} = \nabla_{\boldsymbol{\phi}}\mathbb{E}_{\boldsymbol{\tau}\sim p_{\boldsymbol{\theta}}^{\boldsymbol{\phi}}(\boldsymbol{\tau})}\left[\mathcal{R}_{\text{total}}(\boldsymbol{\tau})\right]\Big|_{\boldsymbol{\theta}^{[k]},\boldsymbol{\phi}^{[j]}}$ at each iteration of the inner-loop j. Handling simulators as black-box models, we estimate the gradients by sampling from a Gaussian distribution with mean $\mathbf{0}$ and $\sigma^2\boldsymbol{I}$ co-variance matrix. Our choice for such estimates is not arbitrary but rather theoretically grounded as one can easily prove the following proposition:

Proposition 1 (Zero-Order Gradient Estimate)**.** *For a fixed $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, the gradient can be computed as:*

$$\nabla_{\boldsymbol{\phi}}\mathbb{E}_{\boldsymbol{\tau}\sim p_{\boldsymbol{\theta}}^{\boldsymbol{\phi}}(\boldsymbol{\tau})}\left[\mathcal{R}_{total}(\boldsymbol{\tau})\right] = \frac{1}{\sigma^2}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}\int_{\boldsymbol{\tau}} p_{\boldsymbol{\theta}}^{\boldsymbol{\phi}+\boldsymbol{\xi}}(\boldsymbol{\tau})\mathcal{R}_{total}(\boldsymbol{\tau})d\boldsymbol{\tau}\right].$$

Hessian Estimation: Having derived a zero-order gradient estimator, we now generalise these notions to a form allowing us to estimate the Hessian. It is also worth reminding the reader that such a Hessian estimator needs to be performed one time only before executing the instructions in Algorithm 1 (i.e., $\boldsymbol{H}_0$ is passed as an input). Precisely, we prove the following proposition:

Proposition 2 (Zero-Order Hessian Estimate)**.** *The hessian of the Wasserstein distance around $\boldsymbol{\phi}_0$ can be estimated based on function evaluations. Recalling that $\boldsymbol{H}_0 = \nabla^2_{\boldsymbol{\phi}}\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_{\pi}^{\boldsymbol{\phi}_0}(\cdot)}\left[\mathcal{W}_2^2\left(\mathcal{P}_{\boldsymbol{\phi}}(\cdot|\boldsymbol{s},\boldsymbol{a}),\mathcal{P}_0(\cdot|\boldsymbol{s},\boldsymbol{a})\right)\right]\Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}_0}$, and defining $\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}(\boldsymbol{\phi}) :=$*

$\mathcal{W}_2^2\left(\mathcal{P}_{\boldsymbol{\phi}}(\cdot|\boldsymbol{s},\boldsymbol{a}), \mathcal{P}_0(\cdot|\boldsymbol{s},\boldsymbol{a})\right)$, *we prove:*

$$\boldsymbol{H}_0 = \frac{1}{\sigma^2}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\boldsymbol{0},\sigma^2\boldsymbol{I})}\left[\frac{1}{\sigma^2}\boldsymbol{\xi}\left(\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_\pi^{\phi_0}(\cdot)}\left[\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}\left(\boldsymbol{\phi}_0+\boldsymbol{\xi}\right)\right]\right)\boldsymbol{\xi}^{\mathsf{T}} - \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_\pi^{\phi_0}(\cdot)}\left[\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}(\boldsymbol{\phi}_0+\boldsymbol{\xi})\right]\boldsymbol{I}\right].$$

Proofs of these propositions are given in Appendix A. They allow for a procedure where gradient and Hessian estimates can be simply based on simulator value evaluations while perturbing $\boldsymbol{\phi}$ and $\boldsymbol{\phi}_0$. It is important to note that in order to apply the above, we are required to be able to evaluate $\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_\pi^{\phi_0}(\cdot)}\left[\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}(\boldsymbol{\phi}_0)\right]$ under random $\boldsymbol{\xi}$ perturbations sampled from $\mathcal{N}(\boldsymbol{0},\sigma^2\boldsymbol{I})$. An empirical estimate of the p-Wasserstein distance between two measures μ and ν can be performed by computing the p-Wasserstein distance between the empirical distributions evaluated at sampled data. That is, one can approximation μ by $\mu_n = \frac{1}{n}\sum_{i=1}^n \delta_{\boldsymbol{x}_i}$ where $\boldsymbol{x}_i$ are identically and independently distributed according to μ. Approximating ν_n similarly, we then realise that[5] $\mathcal{W}_2(\mu,\nu) \approx \mathcal{W}_2(\mu_n,\nu_n)$.

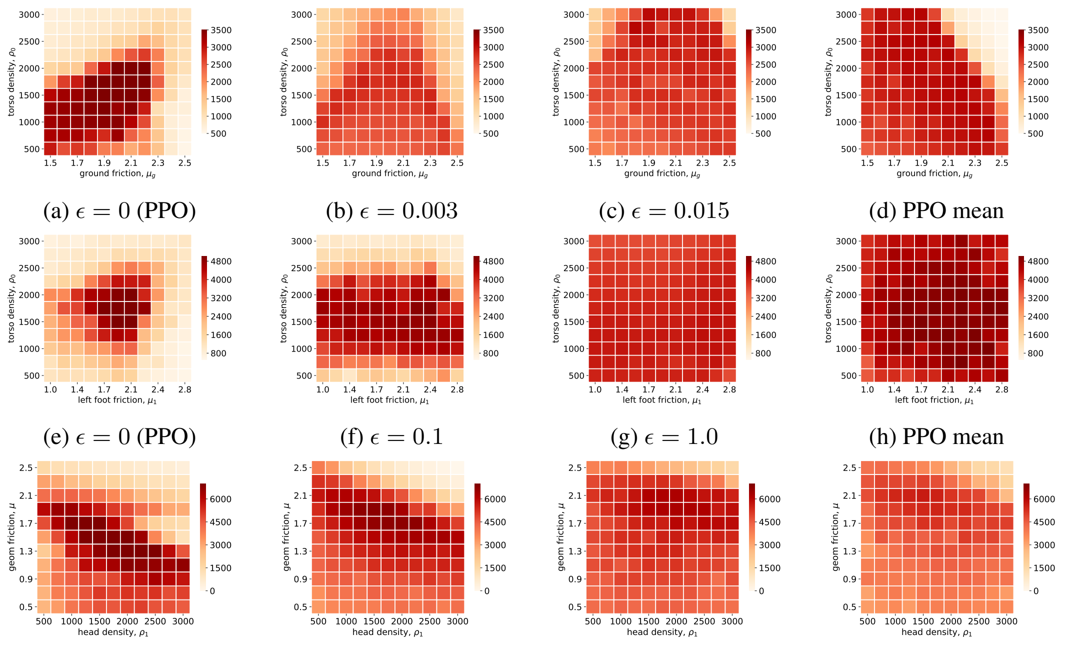
5 EXPERIMENTS & RESULTS

We evaluate WR^{2}L on a variety of continuous control benchmarks from the MuJoCo environment. Dynamics in our benchmarks were parameterised by variables defining physical behaviour, e.g., density of the robot's torso, friction of the ground, and so on. We consider both low and high dimensional dynamics and demonstrate that our algorithm outperforms state-of-the-art from both standard and robust reinforcement learning. We are chiefly interested in policy generalisation across environments with varying dynamics, which we measure using average test returns on novel systems. The comparison against standard reinforcement learning algorithms allows us to understand whether lack of robustness is a critical challenge for sequential decision making, while comparisons against robust algorithms test if we outperform state-of-the-art that considered a similar setting to ours. From standard algorithms, we compare against proximal policy optimisation (PPO) (Schulman et al., 2017), and trust region policy optimisation (TRPO) (Schulman et al., 2015b); an algorithm based on natural actor-crtic (Peters & Schaal, 2008; Pajarinen et al., 2019). From robust algorithms, we demonstrate how WR^{2}L favours against robust adversarial reinforcement learning (RARL) (Pinto et al., 2017), and action-perturbed Markov decision processes (PR-MDP) proposed in (Tessler et al., 2019). Due to space constraints, the results are presented in Appendix B.2. It is worth noting that we attempted to include deep deterministic policy gradients (DDPG) (Silver et al., 2014) in our comparisons. Results including DDPG were, however, omitted as it failed to show any significant robustness performance even on relatively simple systems, such as the inverted pendulum; see results reported in Appendix B.3. During initial trials, we also performed experiments parameterising models using deep neural networks. Results demonstrated that these models, though minimising training data error, fail to provide valid physics-grounded dynamics. For instance, we arrived at inverted pendula models that vary pole angles without exerting any angular speeds. This problem became even more apparent in high-dimensional systems, e.g., Hopper, Walker, etc due to the increased number of possible minima. As such, results presented in this section make use of our zero-order method that can be regarded as a scalable alternative for robust solutions.

5.1 MUJOCO BENCHMARKS

We evaluate our method both in low and *high-dimensional* MuJuCo tasks (Brockman et al., 2016). We consider a variety of systems including CartPole, Hopper, and Walker2D; all of which require direct joint-torque control. Keeping with the generality of our method, we utilise these frameworks as-is with no additional alterations, that is, we use the exact setting of these benchmarks as that shipped with OpenAI gym without any reward shaping, state-space augmentation, feature extraction, or any other modifications of-that-sort. Details are given in section B.

[5] In case the dynamics are assumed to be Gaussian, a similar procedure can be followed or a closed form can be used, see Takatsu (2008).



(a) $\epsilon = 0$ (PPO) (b) $\epsilon = 0.003$ (c) $\epsilon = 0.015$ (d) PPO mean

(e) $\epsilon = 0$ (PPO) (f) $\epsilon = 0.1$ (g) $\epsilon = 1.0$ (h) PPO mean

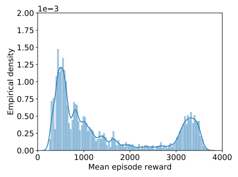
(i) $\epsilon = 0.0$ (PPO) (j) $\epsilon = 0.005$ (k) $\epsilon = 0.03$ (l) PPO mean

Figure 1: Dynamics variations in two dimensions and with different ϵ, for Hopper (top row), Walker (middle row), and HalfCheetah (bottom row). Fourth column is PPO trained with dynamics sampled **uar** from the Wasserstein ball. Darker is better, with the numbers in the graded scale being the range of scores that the tasks can generate.

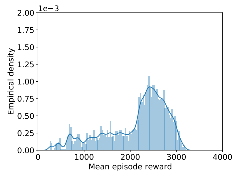
Due to space constraints, results for one-dimensional parameter variations are given in Appendix B.2, where it can be seen that WR^{2}L outperforms both robust and non-robust algorithms when one-dimensional simulator variations are considered. Figure 1 shows results for dynamics variations along two dimensions. Here again, our methods demonstrates considerable robustness. The fourth column, “PPO mean”, refers to experiments where PPO is trained on a dynamics sampled uniformly at random from the Wasserstein constraint set. It displayes more robustness than when trained on just the reference dynamics, however, as can be seen from Fig. 2, our method performs noticably better in high dimensions, which is the main strength of our algorithm.

Results with High-Dimensional Model Variation: Though results above demonstrate robustness, an argument against a min-max objective can be made especially when only considering low-dimensional changes in the simulator. Namely, one can argue the need for such an objective as opposed to simply sampling a set of systems and determining policies performing-well on average similar to the approach proposed in (Rajeswaran et al., 2017).

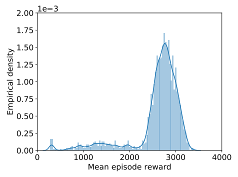
A counter-argument to the above is that a gradient-based optimisation scheme is more efficient than a sampling-based one when high-dimensional changes are considered. In other words, a sampling procedure is hardly applicable when more than a few parameters are altered, while WR^{2}L can remain suitable. To assess these claims, we conducted two additional experiments on the Hopper and HalfCheetah benchmarks. In the first, we trained robustly while changing friction and torso densities, and tested on 1,000 systems generated by varying all 11 dimensions of the Hopper dynamics, and 21 dimensions of the HalfCheetah system. The histogram Figures 2(b) and (f) demonstrate that the empirical densities of the average test returns are mostly centered around 3,000 for the Hopper, and around 4,500 for the Cheetah, which improves that of PPO trained on reference (Figures 2(a) and (e)) with return masses mostly accumulated at around 1,000 in the case of the Hopper and almost equally distributed when considering HalfCheetah. Such improvements, however, can be an artifact of the careful choice of the low-dimensional degrees of freedom allowed to be modified during Phase I of Algorithm 1. To get further insights, Figures 2(c) and (g) demonstrate the effectiveness of our method trained and tested while allowing to tune all 11 dimensional parameters of the Hopper sim-



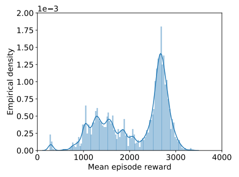
(a) PPO Test High



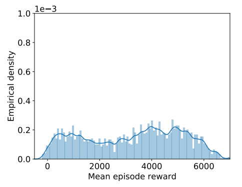
(b) Train Low Test High



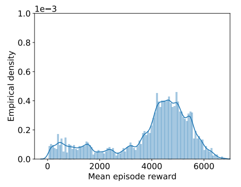
(c) Train High Test High



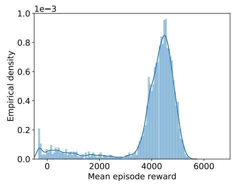
(d) PPO mean



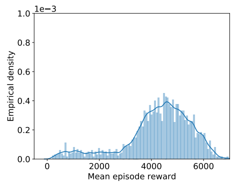
(e) PPO Test High



(f) Train Low Test High



(g) Train High Test High



(h) PPO mean

Figure 2: Results evaluating performance when considering high-dimensional variations on the Hopper (top row) and HalfCheetah (bottom row) environment. All figures show the empirical distribution of returns on 1,000 testing systems. The first column demonstrates the robustness of PPO when trained on the reference dynamics and trained with high dimensional variations. The second column reports empirical test returns of WR^{2}L's policy trained on only two parameter changes (e.g., friction and density) of the environment but tested on systems with all high-dimensional dynamical parameters modified. The third column trains and tests WR^{2}L altering all dimensional parameters of the simulator. The fourth column reports results when PPO is trained on dynamics chosen **uar** and tested with high dimensional variation.

ulator, and the 21 dimensions of the HalfCheetah. Indeed, our results are in accordance with these of the previous experiment depicting that most of the test returns' mass remains around 3,000 for the Hopper, and improves to accumulate around 4,500 for the HalfCheetah. Interestingly, however, our algorithm is now capable of acquiring higher returns on all systems[6] since it is allowed to alter all parameters defining the simulator. As such, we conclude that WR^{2}L outperforms others when high-dimensional simulator variations are considered. In Figures 2(d) and (h), we see the results for PPO trained with dynamics sampled **uar** from the Wasserstein constraint set. We see that although in the two-dimensional variation case this training method worked well (see Figures 1(d), (h), (l)), it does not scale well to high dimensions, and our method does better.

6 Related work

Previous work on robust MDPs, e.g., Iyengar (2005); Nilim & El Ghaoui (2005); Wiesemann et al. (2013); Tirinzoni et al. (2018); Petrik & Russell (2019), whilst valuable in its own right, is not sufficient for the RL setting due to the need in the latter case to give efficient solutions for large state and action spaces, and the fact that the dynamics are not known *a priori*.

Closer to our own work, Rajeswaran et al. (2017) approaches the robustness problem by training on an ensemble of dynamics in order to be deployed on a target environment. The algorithm introduced, Ensemble Policy Optimisation (EPOpt), alternates between two phases: (i) given a distribution over dynamics for which simulators (or models) are available (the source domain), train a policy that performs well for the whole distribution; (ii) gather data from the deployed environment (target domain) to adapt the distribution. The objective is not max-min, but a softer variation defined by conditional value-at-risk (CVaR). The algorithm samples a set of dynamics $\{\phi_k\}$ from a distribution over dynamics $\mathcal{P}_{\psi}$, and for each dynamics ϕ_k, it samples a trajectory using the current policy parameter $\boldsymbol{\theta}_i$. It then selects the worst performing ϵ-fraction of the trajectories to use to update the policy parameter. Clearly this process bears some resemblance to our algorithm, but there is a crucial difference: our algorithm takes *descent steps* in the ϕ space. The difference if important when the dynamics parameters sit in a high-dimensional space, since in that case, optimisation-from-sampling could demand a considerable number of samples. In any case, our experiments demonstrate our algorithm performs well even in these high dimensions. We note that we were

[6]Please note that we attempted to compare against Rajeswaran et al. (2017). Due to the lack of open-source code, we were not able to regenerate their results.

were unable to find the code for this paper, and did not attempt to implement it ourselves. The CVaR criterion is also adopted in Pinto et al. (2017), in which, rather than sampling trajectories and finding a quantile in terms of performance, two policies are trained simultaneously: a "protagonist" which aims to optimise performance, and an adversary which aims to disrupt the protagonist. The protagonist and adversary train alternatively, with one being fixed whilst the other adapts. We made comparisons against this algorithm in our experiments. More recently, Tessler et al. (2019) studies robustness with respect to action perturbations. There are two forms of perturbation addressed: (i) Probabilistic Action Robust MDP (PR-MDP), and (ii) Noisy Action Robust MDP (NR-MDP). In PR-MDP, when an action is taken by an agent, with probability α, a different, possibly adversarial action is taken instead. In NR-MDP, when an action is taken, a perturbation is added to the action itself. Like Rajeswaran et al. (2017) and Pinto et al. (2017), the algorithm is suitable for applying deep neural networks, and the paper reports experiments on InvertedPendulum, Hopper, Walker2d and Humanoid. We tested against PR-MDP in some of our experiments, and found it to be lacking in robustness (see Section 5). In Lecarpentier & Rachelson (2019) a non-stationary Markov Decision Process model is considered, where the dynamics can change from one time step to another. The constraint is based on Wasserstein distance, specifically, the Wasserstein distance between dynamics at time t and t' is bounded by $L|t - t'|$, i.e., is L-Lipschitz with respect to time, for some constant L. They approach the problem by treating nature as an adversary and implement a Minimax algorithm. The basis of their algorithm is that due to the fact that the dynamics changes slowly (due to the Lipschitz constraint), a planning algorithm can project into the future the scope of possible future dynamics and plan for the worst. The resulting algorithm, known as *Risk Averse Tree Search*, is - as the name implies - a tree search algorithm. It operates on a sequence "snapshots" of the evolving MDP, which are instances of the MDP at points in time. The algorithm is tested on small grid world, and does not appear to be readily extendible to the continuous state and action scenarios our algorithm addresses. To summarise, our paper uses the Wasserstein distance for quantifying variations in possible dynamics, in common with Lecarpentier & Rachelson (2019), but is suited to applying deep neural networks for continuous state and action spaces. Our algorithm does not require a full dynamics available to it, merely a parameterisable dynamics. It competes well with the above papers, and operates well for high dimensional problems, as evidenced by the experiments.

7 Conclusion & Future Work

In this paper, we proposed a robust reinforcement learning algorithm capable of outperforming others in terms of test returns on unseen dynamics. The algorithm makes use of Wasserstein constraints for policies generalising across varying domains, and considers a zero-order method for scalable solutions. Empirically, we demonstrated superior performance against state-of-the-art from both standard and robust reinforcement learning on low and high-dimensional MuJuCo environments. In future work, we aim to consider robustness in terms of other components of MDPs, e.g., state representations, reward functions, and others. Furthermore, we will implement WR^{2}L on real hardware, considering sim-to-real experiments.

References

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6571–6583. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7892-neural-ordinary-differential-equations.pdf`.

Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 1522–1530. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/6014-risk-sensitive-and-robust-decision-making-a-cvar-optimization-approach.pdf`.

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

Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. Learning-based model predictive control for safe exploration and reinforcement learning. *CoRR*, abs/1803.08287, 2018. URL `http://arxiv.org/abs/1803.08287`.

Erwan Lecarpentier and Emmanuel Rachelson. Non-stationary markov decision processes a worst-case approach using model-based reinforcement learning. *arXiv preprint arXiv:1904.10090*, 2019.

Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning, 07 2019.

Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural Comput.*, 17(2):335–359, February 2005. ISSN 0899-7667. doi: 10.1162/0899766053011528. URL `http://dx.doi.org/10.1162/0899766053011528`.

Hongseok Namkoong and John C. Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 2216–2224, USA, 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9. URL `http://dl.acm.org/citation.cfm?id=3157096.3157344`.

Yurii Nesterov. Random gradient-free minimization of convex functions. CORE Discussion Papers 2011001, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2011. URL `https://EconPapers.repec.org/RePEc:cor:louvco:2011001`.

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

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *CoRR*, abs/1810.12282, 2018. URL `http://arxiv.org/abs/1810.12282`.

Joni Pajarinen, Hong Linh Thai, Riad Akrour, Jan Peters, and Gerhard Neumann. Compatible natural gradient policy search. *CoRR*, abs/1902.02823, 2019. URL `http://arxiv.org/abs/1902.02823`.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *CoRR*, abs/1710.06537, 2017. URL `http://arxiv.org/abs/1710.06537`.

Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomput.*, 71(7-9):1180–1190, March 2008. ISSN 0925-2312. doi: 10.1016/j.neucom.2007.11.026. URL `http://dx.doi.org/10.1016/j.neucom.2007.11.026`.

Marek Petrik and Reazul Hasan Russell. Beyond confidence regions: Tight bayesian ambiguity sets for robust mdps. *arXiv preprint arXiv:1902.07605*, 2019.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2817–2826, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL `http://proceedings.mlr.press/v70/pinto17a.html`.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *International Conference on Learning Representations (ICLR) 2017, arXiv preprint arXiv:1610.01283*, 2017.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Thomas Sargent and Lars Hansen. Robust control and model uncertainty. *American Economic Review*, 91(2):60–66, 2001. URL `https://EconPapers.repec.org/RePEc:aea:aecrev:v:91:y:2001:i:2:p:60-66`.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, Lille, France, 07–09 Jul 2015a. PMLR. URL `http://proceedings.mlr.press/v37/schulman15.html`.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015b. URL `http://arxiv.org/abs/1502.05477`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pp. I–387–I–395. JMLR.org, 2014. URL `http://dl.acm.org/citation.cfm?id=3044805.3044850`.

Asuka Takatsu. Wasserstein geometry of gaussian measures, 2008.

Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6215–6224, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/tessler19a.html`.

Andrea Tirinzoni, Marek Petrik, Xiangli Chen, and Brian Ziebart. Policy-conditioned uncertainty sets for robust markov decision processes. In *Advances in Neural Information Processing Systems*, pp. 8939–8949, 2018.

Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

Philip Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969.

Insoon Yang. A convex optimization approach to distributionally robust markov decision processes with wasserstein distance. *IEEE control systems letters*, 1(1):164–169, 2017.

Chenyang Zhao, Olivier Sigaud, Freek Stulp, and Timothy M. Hospedales. Investigating generalisation in continuous deep reinforcement learning. *CoRR*, abs/1902.07015, 2019. URL `http://arxiv.org/abs/1902.07015`.

A Proofs

Proof of Proposition 1. The proof of the above proposition can easily be derived by combining the lines of reasoning in (Salimans et al., 2017; Nesterov, 2011), while extending to the parameterisation of dynamical models. To commence, begin by defining $\mathcal{J}_{\boldsymbol{\theta}}(\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{\tau}\sim p_{\boldsymbol{\theta}}^{\phi}}\left[\mathcal{R}_{\text{total}}(\boldsymbol{\tau})\right]$ for fixed policy parameters $\boldsymbol{\theta}$. Given any perturbation vector, $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \sigma^2\boldsymbol{I})$, we can derive (through a Taylor expansion) the following:

$$\mathcal{J}_{\boldsymbol{\theta}}(\boldsymbol{\phi}+\boldsymbol{\xi}) = \mathcal{J}_{\boldsymbol{\theta}}(\boldsymbol{\phi}) + \boldsymbol{\xi}^{\mathsf{T}}\nabla_{\boldsymbol{\phi}}\mathcal{J}_{\boldsymbol{\theta}}(\boldsymbol{\phi}) + \frac{1}{2}\boldsymbol{\xi}^{\mathsf{T}}\nabla^2_{\boldsymbol{\phi}}\mathcal{J}_{\boldsymbol{\theta}}(\boldsymbol{\phi})\boldsymbol{\xi} + \mathcal{O}\left(\text{higher-order terms}\right).$$

Multiplying by $\boldsymbol{\xi}$, and taking the expectation on both sides of the above equation, we get:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}\mathcal{J}_{\boldsymbol{\theta}}(\boldsymbol{\phi}+\boldsymbol{\xi})\right] &= \mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}\mathcal{J}_{\boldsymbol{\theta}}(\boldsymbol{\phi}) + \boldsymbol{\xi}\boldsymbol{\xi}^{\mathsf{T}}\nabla_{\boldsymbol{\phi}}\mathcal{J}_{\boldsymbol{\theta}}(\boldsymbol{\phi}) + \frac{1}{2}\boldsymbol{\xi}\boldsymbol{\xi}^{\mathsf{T}}\nabla^2_{\boldsymbol{\phi}}\mathcal{J}_{\boldsymbol{\theta}}(\boldsymbol{\phi})\boldsymbol{\xi}\right] \\ &= \sigma^2\nabla_{\boldsymbol{\phi}}\mathcal{J}_{\boldsymbol{\theta}}(\boldsymbol{\phi}).\end{aligned}$$

Dividing by σ^2, we derive the statement of the proposition as:

$$\nabla_{\boldsymbol{\phi}}\mathcal{J}_{\boldsymbol{\theta}}(\boldsymbol{\phi}) = \frac{1}{\sigma^2}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}\mathcal{J}_{\boldsymbol{\theta}}(\boldsymbol{\phi}+\boldsymbol{\xi})\right]$$

□

Proof of Proposition 2. Commencing with the right-hand-side of the above equation, we perform second-order Taylor expansions for each of the two terms under under the expectation of $\boldsymbol{\xi}$. Namely, we write:

$$\begin{aligned}\boldsymbol{H}_0 = \frac{1}{\sigma^2}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\Bigg[\frac{1}{\sigma^2}\boldsymbol{\xi}\left(\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_\pi^{\phi_0}(\cdot)}\left[\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}\left(\boldsymbol{\phi}_0+\boldsymbol{\xi}\right)\right]\right)\boldsymbol{\xi}^{\mathsf{T}} \qquad (11)\\ - \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_\pi^{\phi_0}(\cdot)}\left[\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}(\boldsymbol{\phi}_0+\boldsymbol{\xi})\right]\boldsymbol{I}\Bigg] \\ \approx \frac{1}{\sigma^4}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\Bigg[\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_\pi^{\phi_0}}\left[\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}(\boldsymbol{\phi}_0)\right]\boldsymbol{\xi}\boldsymbol{\xi}^{\mathsf{T}} + \boldsymbol{\xi}\boldsymbol{\xi}^{\mathsf{T}}\nabla_{\boldsymbol{\phi}}\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_\pi^{\phi_0}}\left[\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}(\boldsymbol{\phi}_0)\right]\boldsymbol{\xi}^{\mathsf{T}} \\ + \frac{1}{2}\boldsymbol{\xi}^{\mathsf{T}}\nabla^2_{\boldsymbol{\phi}}\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_\pi^{\phi_0}}\left[\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}(\boldsymbol{\phi}_0)\right]\boldsymbol{\xi}\boldsymbol{I}\Bigg] \\ - \frac{1}{\sigma^2}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\Bigg[\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_\pi^{\phi_0}}\left[\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}(\boldsymbol{\phi}_0)\right]\boldsymbol{I} + \boldsymbol{\xi}^{\mathsf{T}}\nabla_{\boldsymbol{\phi}}\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_\pi^{\phi_0}}\left[\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}(\boldsymbol{\phi}_0)\right]\boldsymbol{I} \\ + \frac{1}{2}\boldsymbol{\xi}^{\mathsf{T}}\nabla^2_{\boldsymbol{\phi}}\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_\pi^{\phi_0}}\left[\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}(\boldsymbol{\phi}_0)\right]\boldsymbol{\xi}\boldsymbol{I}\Bigg].\end{aligned}$$

Now, we analyse each of the above terms separately. For ease of notation, we define the following variables:

$$\boldsymbol{g} = \nabla_{\boldsymbol{\phi}}\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_\pi^{\phi_0}}\left[\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}(\boldsymbol{\phi}_0)\right] \qquad \boldsymbol{H} = \nabla^2_{\boldsymbol{\phi}}\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi(\cdot)\rho_\pi^{\phi_0}}\left[\mathcal{W}_{(\boldsymbol{s},\boldsymbol{a})}(\boldsymbol{\phi}_0)\right]$$

$$\boldsymbol{A} = \boldsymbol{\xi}\boldsymbol{\xi}^{\mathsf{T}}\boldsymbol{g}\boldsymbol{\xi}^{\mathsf{T}} \qquad \boldsymbol{B} = \boldsymbol{\xi}\boldsymbol{\xi}^{\mathsf{T}}\boldsymbol{H}\boldsymbol{\xi}\boldsymbol{\xi}^{\mathsf{T}}$$

$$c = \boldsymbol{\xi}^{\mathsf{T}}\boldsymbol{H}\boldsymbol{\xi}.$$

Starting with $\boldsymbol{A}$, we can easily see that any (i,j) component can be written as $\boldsymbol{A} = \sum_{n=1}^{d_2}\boldsymbol{\xi}_i\boldsymbol{\xi}_j\boldsymbol{\xi}_n\boldsymbol{g}_n$. Therefore, the expectation under $\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})$ can be derived as:

$$\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}_i\boldsymbol{\xi}_j\boldsymbol{\xi}_n\boldsymbol{g}_n\right] = \boldsymbol{g}_n\mathbb{E}_{\boldsymbol{\xi}_i\sim\mathcal{N}(0,\sigma^2)}\left[\boldsymbol{\xi}_i^3\right] = 0 \quad \text{if } i=j=n \text{ and 0 otherwise.}$$

Thus, we conclude that $\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}[\boldsymbol{A}] = \mathbf{0}_{d_2\times d_2}$.

Continuing with the second term, i.e., $\boldsymbol{B}$, we realise that any (i,j) component can be written as $\boldsymbol{B}_{i,j} = \sum_{n=1}^{d_2}\sum_{m=1}^{d_2}\boldsymbol{\xi}_i\boldsymbol{\xi}_j\boldsymbol{\xi}_n\boldsymbol{\xi}_m\boldsymbol{H}_{m,n}$. Now, we consider two cases:

- Diagonal Elements (i.e., when $i=j$): The expectation under $\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})$ can be further split in three sub-cases
  - Sub-Case I when $i=j=m=n$: We have $\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}_i\boldsymbol{\xi}_j\boldsymbol{\xi}_n\boldsymbol{\xi}_m\boldsymbol{H}_{m,n}\right] = \boldsymbol{H}_{i,i}\mathbb{E}_{\boldsymbol{\xi}_i\sim\mathcal{N}(0,\sigma^2)} = 3\sigma^4\boldsymbol{H}_{i,i}$.
  - Sub-Case II when $i=j\neq m=n$: We have $\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}_i\boldsymbol{\xi}_j\boldsymbol{\xi}_n\boldsymbol{\xi}_m\boldsymbol{H}_{m,n}\right] = \boldsymbol{H}_{m,m}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}_i^2\boldsymbol{\xi}_m^2\right] = \sigma^4\boldsymbol{H}_{m,m}$.

- Sub-Case III when indices are all distinct: We have $\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}_i\boldsymbol{\xi}_j\boldsymbol{\xi}_n\boldsymbol{\xi}_m\boldsymbol{H}_{m,n}\right] = 0$.

Diagonal Elements Conclusion: Using the above results we conclude that $\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}[\boldsymbol{B}_{i,i}] = 2\sigma^4\boldsymbol{H}_{i,i} + \sigma^4\text{trace}(\boldsymbol{H})$.

- Off-Diagonal Elements (i.e., when $i \neq j$): The above analysis is now repeated for computing the expectation of the off-diagonal elements of matrix $\boldsymbol{B}$. Similarly, this can also be split into three sub-cases depending on indices:
  - Sub-Case I when $i = m \neq j = n$: We have $\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}_i\boldsymbol{\xi}_j\boldsymbol{\xi}_n\boldsymbol{\xi}_m\boldsymbol{H}_{m,n}\right] = \boldsymbol{H}_{i,j}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}_i^2\boldsymbol{\xi}_j^2\right] = \sigma^4\boldsymbol{H}_{i,j}$.
  - Sub-Case II when $i = n \neq j = m$: We have $\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}_i\boldsymbol{\xi}_j\boldsymbol{\xi}_n\boldsymbol{\xi}_m\boldsymbol{H}_{m,n}\right] = \boldsymbol{H}_{j,i}\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}_i^2\boldsymbol{\xi}_j^2\right] = \sigma^4\boldsymbol{H}_{j,i}$.
  - Sub-Case III when indices are all distinct: We have $\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\left[\boldsymbol{\xi}_i\boldsymbol{\xi}_j\boldsymbol{\xi}_n\boldsymbol{\xi}_m\boldsymbol{H}_{m,n}\right] = 0$.

Off-Diagonal Elements Conclusion: Using the above results and due to the symmetric properties of $\boldsymbol{H}$, we conclude that $\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\left[\boldsymbol{B}_{i,j}\right] = 2\sigma^4\boldsymbol{H}_{i,j}$

Finally, analysing c, one can realise that $\mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}[c] = \mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\left[\sum_{i=1}^{d_2}\sum_{j=1}^{d_2}\boldsymbol{\xi}_i\boldsymbol{\xi}_j\boldsymbol{H}_{i,j}\right] = \sigma^2\text{trace}(\boldsymbol{H})$.

Substituting the above conclusions back in the original approximation in Equation 11, and using the linearity of the expectation we can easily achieve the statement of the proposition. □

B Further details on experiments

For clarity, we summarise variables parameterising dynamics in Table 1, and detail specifics next.

CartPole: The goal of this classic control benchmark is to balance a pole by driving a cart along a rail. The state space is composed of the position x and velocity $\dot{x}$ of the cart, as well as the angle θ and angular velocities of the pole $\dot{\theta}$. We consider two termination conditions in our experiments: 1) pole deviates from the upright position beyond a pre-specified threshold, or 2) cart deviates from its zeroth initial position beyond a certain threshold. To conduct robustness experiments, we parameterise the dynamics of the CartPole by the pole length l_p, and test by varying $l_p \in [0.3, 3]$.

Hopper: In this benchmark, the agent is required to control a hopper robot to move forward without falling. The state of the hopper is represented by positions, $\{x, y, z\}$, and linear velocities, $\{\dot{x}, \dot{y}, \dot{z}\}$, of the torso in global coordinate, as well as angles, $\{\theta_i\}_{i=0}^2$, and angular speeds, $\{\dot{\theta}_i\}_{i=0}^2$, of the three joints. During training, we exploit an early-stopping scheme if "unhealthy" states of the robot were visited. Parameters characterising dynamics included densities $\{\rho_i\}_{i=0}^3$ of the four links, armature $\{a_i\}_{i=0}^2$ and damping $\{\zeta_i\}_{i=0}^2$ of three joints, and the friction coefficient μ_g. To test for robustness, we varied both frictions and torso densities leading to significant variations in dynamics. We further conducted additional experiments while varying all 11 dimensional specification parameters.

Walker2D: This benchmark is similar to Hopper except that the controlled system is a biped robot with seven bodies and six joints. Dimensions for its dynamics are extended accordingly as reported in Table 1. Here, we again varied the torso density for performing robustness experiments in the range $\rho_0 \in [500, 3000]$.

Halfcheetah: This benchmark is similar to the above except that the controlled system is a two-dimensional slice of a three-dimensional cheetah robot. Parameters specifying the simulator consist of 21 dimensions, with 7 representing densities. In our two-dimensional experiments we varied the

	1D experiment	2D experiment	High-dimensional experiment
Inverted Pendulum	l_p	None	None
Hopper	ρ_0	$\{\rho_0, \mu_g\}$	$\{\rho_i\}_{i=0}^3 \cup \{a_i\}_{i=0}^2 \cup \{\zeta_i\}_{i=0}^2 \cup \mu_g$
Walker2D	ρ_0	$\{\rho_0, \mu_g\}$	$\{\rho_i\}_{i=0}^6 \cup \{a_i\}_{i=0}^5 \cup \{\zeta_i\}_{i=0}^5 \cup \mu_g$
HalfCheetah	μ	$\{\rho_0, \mu\}$	None

Table 1: Parameterisation of dynamics. See section 5.1 for the physical meaning of these parameters.

torso-density and floor friction, while in high-dimensional ones, we allowed the algorithm to control all 21 variables.

B.1 Experimental protocol

Our experiments included training and a testing phases. During the training phase we applied Algorithm 1 for determining robust policies while updating transition model parameters according to the min-max formulation. Training was performed independently for each of the algorithms on the relevant benchmarks while ensuring best operating conditions using hyper-parameter values reported elsewhere (Schulman et al., 2017; Pinto et al., 2017; Tessler et al., 2019).

For all benchmarks, policies were represented using parametrised Gaussian distributions with their means given by a neural network and standard derivations by a group of free parameters. The neural network consisted of two hidden layers with 64 units and hyperbolic tangent activations in each of the layers. The final layer exploited linear activation so as to output a real number. Following the actor-critic framework, we also trained a standalone critic network having the same structure as that of the policy.

For each policy update, we rolled-out in the current worst-case dynamics to collect a number of transitions. The number associated to these transitions was application-dependent and varied between benchmarks in the range of 5,000 to 10,000. The policy was then optimised (i.e., Phase II of Algorithm 1) using proximal policy optimization with a generalised advantage estimation. To solve the minimisation problem in the inner loop of Algorithm 1, we sampled a number of dynamics from a diagonal Gaussian distribution that is centered at the current worst-case dynamics model. The number of sampled dynamics and the variance of the sampled distributions depended on both the benchmark itself, and well as the dimensions of the dynamics. Gradients needed for model updates were estimated using the results in Propositions 7 and 8. Finally, we terminated training when the policy entropy dropped below an application-dependent threshold.

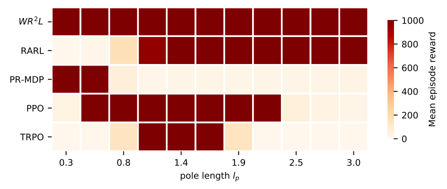
When testing, we evaluated policies on unseen dynamics that exhibited simulator variations as described earlier. We measured performance using returns averaged over 20 episodes with a maximum length of 1,000 time steps on testing environments. We note that we used non-discounted mean episode rewards to compute such averages.

B.2 Results with One-Dimensional Model Variation

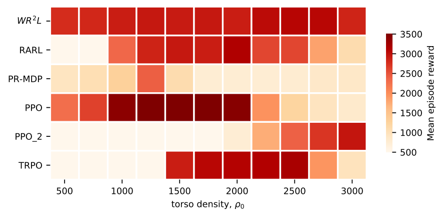
Figure 3 shows the robustness of policies on various taks. For a fair comparison, we trained two standard policy gradient methods (TRPO (Schulman et al., 2015b) and PPO (Schulman et al., 2017)), and two robust RL algorithms (RARL (Pinto et al., 2017), PR-MDP (Tessler et al., 2019)) with the reference dynamics preset by our algorithm. The range of evaluation parameters was intentionally designed to include dynamics ***outside of the ϵ-Wasserstein ball***. Clearly, WR^{2}L outperforms all baselines in this benchmark.

B.3 Deep Deterministic Policy Gradients Results

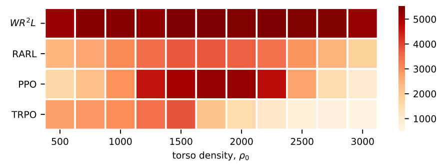
As mentioned in the experiments section of the main paper, we refrained from presenting results involving deep deterministic policy gradients (DDPG) due to its lack in robustness even on simple systems, such as the CartPole.



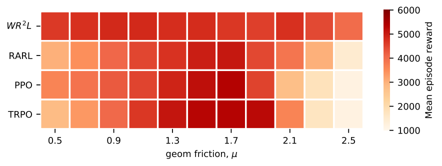
(a) Inverted pendulum, reference is $l_p = 1.65$.



(b) Hopper. Reference is $\rho_0 = 1750$; PPO$_2$ uses the same implementation as PPO but trained with $\rho_0 = 3000$.



(c) Walker. Reference is $\rho_0 = 1750$.



(d) HalfCheetah. Reference is $\mu = 1.5$.

Figure 3: Robustness tests for various tasks; dynamics varied along one dimension.

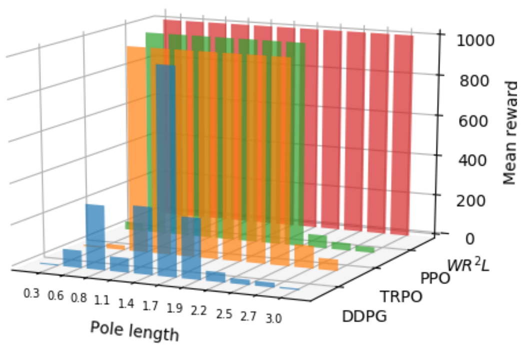


Figure 4: Robustness results on the inverted pendulum demonstrating that our method outperforms state-of-the-art in terms of average test returns and that DDPG lacks in robustness performance.

Figure 4 depicts these results showing that DDPG lacks robustness even when minor variations in the pole length are introduced. TRPO and PPO, on the other hand, demonstrate an acceptable performance retaining a test return of 1,000 across a broad range of pole lengths variations.

C Derivation of the Closed Form Solution

In Section 3.2 we presented a closed form solution to the following optimisation problem:

$$\min_{\boldsymbol{\phi}} \nabla_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{\tau}\sim p_{\boldsymbol{\theta}}^{\boldsymbol{\phi}}(\boldsymbol{\tau})}\left[\mathcal{R}_{\text{total}}(\boldsymbol{\tau})\right]\Big|_{\boldsymbol{\theta}^{[k]},\boldsymbol{\phi}^{[j]}}^{\mathsf{T}} (\boldsymbol{\phi}-\boldsymbol{\phi}^{[j]}) \;\text{ s.t. }\; \frac{1}{2}(\boldsymbol{\phi}-\boldsymbol{\phi}_0)^{\mathsf{T}}\boldsymbol{H}_0(\boldsymbol{\phi}-\boldsymbol{\phi}_0) \leq \epsilon,$$

which took the form of:

$$\boldsymbol{\phi}^{[j+1]} = \boldsymbol{\phi}_0 - \sqrt{\frac{2\epsilon}{\boldsymbol{g}^{[k,j]\mathsf{T}}\boldsymbol{H}_0^{-1}\boldsymbol{g}^{[k,j]}}}\boldsymbol{H}_0^{-1}\boldsymbol{g}^{[k,j]}.$$

In this section of the appendix, we derive such an update rule from first principles. We commence transforming the constraint optimisation problem into an unconstrained one using the Lagrangian:

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\lambda}) = \boldsymbol{g}^{[k,j],\mathsf{T}} \left(\boldsymbol{\phi} - \boldsymbol{\phi}^{[j]}\right) + \boldsymbol{\lambda} \left[\frac{1}{2}(\boldsymbol{\phi} - \boldsymbol{\phi}_0)^\mathsf{T} \boldsymbol{H}_0 (\boldsymbol{\phi} - \boldsymbol{\phi}_0) - \epsilon\right],$$

where $\boldsymbol{\lambda}$ is a Lagrange multiplier, and $\boldsymbol{g}^{[k,j]} = \nabla_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{\tau} \sim p_{\boldsymbol{\theta}}^{\boldsymbol{\phi}}(\boldsymbol{\tau})} \left[\mathcal{R}_{\text{total}}(\boldsymbol{\tau})\right] \Big|_{\boldsymbol{\theta}^{[k]}, \boldsymbol{\phi}^{[j]}}^{\mathsf{T}}.$

Deriving the Lagrangian with respect to the primal parameters $\boldsymbol{\phi}$, we write:

$$\nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\lambda}) = \boldsymbol{g}^{[k,j]\mathsf{T}} + \boldsymbol{\lambda}(\boldsymbol{\phi} - \boldsymbol{\phi}_0)^\mathsf{T} \boldsymbol{H}_0. \tag{12}$$

Setting Equation 12 to zero and solving for primal parameters, we attain:

$$\boldsymbol{\phi} = \boldsymbol{\phi}_0 - \frac{1}{\boldsymbol{\lambda}} \boldsymbol{H}_0^{-1} \boldsymbol{g}^{[k,j]}.$$

Plugging $\boldsymbol{\phi}$ back into the equation representing the constraints, we derive:

$$\left(\boldsymbol{\phi}_0 - \frac{1}{\boldsymbol{\lambda}} \boldsymbol{H}_0^{-1} \boldsymbol{g}^{[k,j]} - \boldsymbol{\phi}_0\right)^\mathsf{T} \boldsymbol{H}_0 \left(\boldsymbol{\phi}_0 - \frac{1}{\boldsymbol{\lambda}} \boldsymbol{H}_0^{-1} \boldsymbol{g}^{[k,j]} - \boldsymbol{\phi}_0\right) = 2\epsilon \implies \boldsymbol{\lambda}^2 = \frac{1}{2\epsilon} \boldsymbol{g}^{[k,j]\mathsf{T}} \boldsymbol{H}_0^{-1} \boldsymbol{g}^{[k,j]}.$$

It is easy to see that with the positive solution for $\boldsymbol{\lambda}$, the Karush–Kuhn–Tucker (KKT) conditions are satisfied. Since the objective and constraint are both convex, the KKT conditions are sufficient and necessary for optimality, thus finalising our derivation.