# Optimal transport final project
# Wasserstein Robust Reinforcement Learning

Clémence GRISLAIN

January 9, 2023

**Abstract**

Reinforcement learning trained agents are often strongly dependent on the simulation environment they have been trained on. As a consequence, they lack of robustness when it comes to unseen dynamics occurring in the real environment. The studied paper [ARB+19] presents a robust reinforcement learning training based on min max optimization problem on a Wasserstein ball centered in a reference dynamics. To solve this problem, the authors introduce an approximation method of the Hessian of the Wasserstein distance. We will focus on this latter and try to evaluate the error introduce by the different approximation methods. We will also apply the end-to-end method on a simple reinforcement learning problem.

## 1 Introduction

### 1.1 The problem

In reinforcement learning, an agent learns to interact with an unknown and stochastic environment in order to maximize a reward signal.This problem is formalized as infinite horizon Markov decision problem defined as $(\mathcal{P}, R, S, A, \gamma)$ where $\mathcal{P}$ is the transition function characterizing the dynamics of the environment, $R$ immediate the reward function, $S$ and $A$ the state and action spaces, and $\gamma$ the discount term which specifies the degree to which the reward is discounted over time. This formulation can refer to numerous domain such as games, autonomous driving, finance or even robotics. In Reinforcement learning setting, the dynamics, the reward function as well as the discount term are unknown. When training, the agent only interacts with the environment: at time $t$, being in a state $s_t$, the agent makes the action $a_t$ and observes the reward $r_t$ and ends up in the state $s_{t+1}$. This transition is ruled by the underlying dynamics $\mathcal{P}_t$. The model is trained on a huge number of such interactions.

However, on high dimension environment, the space of the dynamics $\mathcal{P}_t$ is very large and some dynamics remain unseen by the agent during the training. The problem also occurs when the agent aims to be deployed in a real-world environment - like autonomous driving. The agent cannot be safely trained in the real world and has to be trained on a simulator. Yet, constructing a simulator able to recreate every possible dynamics is impossible. On the autonomous driving example, the dynamics are different for every car, driving system or even climate condition, road characteristics etc. . It is thus capital to make the agent's learning robust to unseen dynamics, in the limits that the dynamics is "not too far" from seen ones. Hence the need for reinforcement learning to be robust.

### 1.2 Related work

Reinforcement learning research has recently been focused on developing algorithms that can make robust decisions in real-world applications. Many of these algorithms borrow from multiple domains of applied mathematics (game theory, function analysis, optimal control etc.) and are often developed for discrete state-action spaces. To treat with action uncertainty and/or dynamics variations, the authors cite Sargent and Hansen (2001) [HS01]; Nilim and El Ghaoui (2005) [NG05]; Iyengar (2005) [Iye05]; Namkoong and Duchi (2016) [ND16], which explore the idea of "worse-case scenario both in terms of agents' policies and of environments' dynamics. These methods have also been extended to continuous state-action space using linear function approximators, Chow et al. (2015) [CTMP15] and deep neural networks Peng et al. (2017) [PAZA17].

Despite their success in practice, current techniques for robust decision-making are task-specific and have two major drawbacks. The first drawback is that they do not provide a general framework for robustness due to their specialized heuristics. Moreover, the algorithms often cannot be applied to both continuous and discrete state-action space. The second drawback pointed out by the paper is that the process of empirical validation for these techniques rarely involves comparison with other robust algorithms in the literature, which completely bias the evaluation of the performances of the methods.

Work on robust MDPs (Markov Decision Problems $(\mathcal{P}, R, S, A)$ where $\mathcal{P}$ and $R$ are known) are also cited in the article. In particular, Yang (2017) [Yan17] defines uncertainty sets of dynamics as the Wasserstein $\epsilon$-ball around a particular dynamics. The paper also shows that finding the optimal policy using the Bellman equation amount to solve a *max-min* optimization problem. Yet, in the field of reinforcement learning, these methods are not sufficient because they are not able to provide efficient solutions for large state and action spaces, and because the dynamics of the system are not explicitly known. One could solve this issue by training model-based reinforcement learning algorithm, that is to say, to learn beforehand the explicit form of the dynamics and of the reward function. However, this could easily lead excessive computational costs and memory.

Rajeswaran et al. (2017) [RGLR16] approaches the robustness problem by training an agent in sets of dynamics. The algorithm called Ensemble Policy Optimisation is decomposed in two phases : (i) find a good policy over a given set of dynamics (ii) sample trajectory from the current policy on the set of dynamics and uses the worse $\epsilon$ trajectories to update the policy. This is a soft way to find the worse-case dynamics among the given set. However, this method requires a considerable number of interactions with the environment to sample the trajectories for the entire set of dynamics at each step. Instead on working on uncertainty over dynamics, Tessler et al. (2019) [TEM19] studies robustness with respect to action perturbations. Two main approaches have been explored: *Probabilistic Action Robust* MDP - given an action, the agent takes an adversarial action or *Noisy Action Robust* MSP - given an action, the agent takes this action on which a certain noise has been added. These methods are also suitable for applying deep neural network as parameters of the policy and/or of the dynamics.

Finally, Lecarpentier and Rachelson (2019) [LR19] studied non-stationary Markov Decision Process, that is to say, MDPs in which the dynamics change over time $\mathcal{P}_{\phi_t}$. In this setup, the dynamics between two time steps $t$ and $t'$ are constrained by the Wasserstein distance between the two distributions $\mathcal{P}_{\phi_t}$ and $\mathcal{P}_{\phi_{t'}}$ which has to be $L$-Lipschitz: $\mathcal{W}_2(\mathcal{P}_{\phi_t}, \mathcal{P}_{\phi'_t}) \leq L|t - t'|$. The algorithm tackled the problem by using a *min max* optimization problem. In fact, nature is considered as an opponent. The final approach, called *Risk Averse Tree Search*, is a tree search algorithm that considers the potential future dynamics of a slowly changing system (as constrained by the Lipschitz condition on the Wasserstein distance) and plans for the worst-case outcome. However, this algorithm is made for discrete state-action space and have been tested on tabular settings. It may not be easily adaptable to environments with continuous states and actions.

### 1.3 Paper contributions

In the studied paper, the authors present a method for robust reinforcement learning that can handle both discrete and continuous state and action spaces. The proposed algorithm is called *Wasserstein Robust Reinforcement Learning WRL*[2] and it aims to find the best policy by evaluating the worst-case dynamics among a set of candidate "close" to a reference dynamics $\mathcal{P}_{\phi_0}$. To evaluate how close two dynamics are, the authors use the Wasserstein distance $\mathcal{W}_2$. Thus, the constraint space of the optimization problem is the average Wasserstein ball around a reference dynamics. In fact, the optimization problem can be seen in term of game theory: a player evolving in an environment acts according to the worse plausible scenario (in terms of cumulative rewards) - the notion of "plausible scenario" being encapsulated by the dynamics belonging to the Wasserstein ball $\mathcal{B}_2(\mathcal{P}_{\phi_0}, \epsilon)$. In the course of Reinforcement Learning, we have seen that certain algorithms are based on behavioral notion such as *optimism* (e.g UCB algorithm) or *pessimism*. In this case, this strategy can refers to conservative behavior.

Finding the best policy in the worse-case scenario is formalized as a *max min* optimization problem on the reward function $(\theta, \phi)$: the *max* over the policy parameters $\theta$, and the *min* over the dynamics parameters $\{\phi \text{ s.t } \mathcal{P}_\phi \in \mathcal{B}_2(\mathcal{P}_{\phi_0}, \epsilon)\}$. The algorithm is actor-critic reinforcement learning: both the

policy and the dynamics are parameterized. The policy parameters $\theta$ are the weights of a neural network $\mathcal{F}_\theta$ and the dynamics parameters $\phi$ are the parameters of a simulator or differential equation solver $\mathcal{P}_\phi$ (the dynamics are not parameterized by a neural network in order to ensure every generated dynamics are physically plausible). The model supposes that the reference dynamics $\mathcal{P}_{\phi_0}$ is known and is an input of the algorithm.

The algorithm performs gradient descent in the dynamics parameters space, in order to find the worse-case scenario, and updates the policy parameters after convergence, best policy according to the worse dynamics. The authors introduce a zero-order sampling method to approximate the gradients of the reward function given a policy and a dynamics. The constraints on the dynamics parameters search space is approximated using the second order Taylor expansion which simplifies the problem of finding the worse-case dynamics into a quadratic constrained problem. The quadratic constraint is defined using the Hessian of the Wasserstein distance evaluated in the reference dynamics: $H_0 = \nabla^2 \mathcal{W}_2^2(\mathcal{P}_\phi, \mathcal{P}_{\phi_0})|_{\phi=\phi_0}$. The zero-order approximation method is extended to estimate the Hessian. As the reference dynamics $\mathcal{P}_{\phi_0}$ is an input of the algorithm, in practice, the estimate of $H_0$ is computed once and also considered as an input.

The presented approach is interesting as it does not require explicit knowledge of the reference dynamics and does not build a full model of the dynamics, making it computationally efficient. It is also not demanding in terms of domain knowledge, requiring only access to a simulator or differentiable equation solver that can parameterize the dynamics, that is to say, given $\phi$ returns samples drawn from the dynamics $\mathcal{P}_\phi$. The method is generic and can be applied both on continuous and discrete setups, suitable for applying deep neural network parametrization and efficient for high dimensional problems. The paper also paid particular attention to testing and comparing their work with other reinforcement learning algorithms trained especially for robustness. In fact, the authors specified that most of the time, robust reinforcement learning training methods are only compared to state-of-the-art, but not trained for robustness, algorithms. This bias is pointed out in the paper.

In this project, we will focus on finding the worst-case scenario within the set of plausible dynamics as an optimization problem over the Wasserstein ball centered on the reference dynamics. For the purposes of simplicity, in a first part, we will consider a problem outside of the context of reinforcement learning. However, we will use the same solution technique involving the second-order Taylor expansion of the Wasserstein distance, which transforms the problem into quadratic constraint form. We will examine the approximation of the Hessian of the Wasserstein distance at the reference dynamics and attempt to evaluate the error introduced by this approximation using an example in which the Hessian can be analytically computed. In the second part of the project, we will apply the robust reinforcement learning algorithm in a simple, non-stationary setting in which an agent aims to reach a goal as quickly as possible in a tabular environment.

# 2 Main body

## 2.1 Presentation of the method

### 2.1.1 Reinforcement learning (RL)

In reinforcement learning setup, an agent interacts with an unknown and stochastic environment with the goal of maximizing a cumulative reward function. These problem are formalized as Markov Decision Processes (MDPs) from stochastic processes theory. The problem is defined as $\mathcal{M} = (\mathcal{P}, R, S, A, \gamma)$ where:

- $\mathcal{P}$ is the dynamics that rules the probability of a transition knowing an action in the environment $S \times A \times S \longrightarrow [0, 1]$
- $R$ the reward function $S \times A \longrightarrow \mathcal{R}$
- $S \subseteq \mathbb{R}^d$, $A \subseteq \mathbb{R}^n$ the state and action spaces
- $\gamma \in [0, 1)$ the discount term (the closer to 1, the more important the future rewards are)

In practice, $\mathcal{P}$ and $R$ are unknown. The agent only accesses to interaction with the environment. At each time $t$ the agent is in a state $s_t$ and take the action $a_t$. It ends up in a state $s_{t+1}$ with probability $P(s_{t+1}|s_a, a_t)$ and receive the reward $r_t = (s_t, a_t)$. A policy $\pi$ defines the probability for the agent of taking an action $a_t$ being in a current state $s_t$, written $\pi(a_t|s_t)$. Yet, the agent is not trained directly on the transition $(s_t, a_t, r_t, s_{t+1})$ as it does not aim to maximize the immediate reward but the cumulative reward. Thus, the agent is trained on trajectories which are succession of transition: a trajectory $\tau$ of length $T$ is defined as $\tau = (s_0, a_0, r_0, s_1, ..., s_{-1}, a_{T-1}, r_{T-1}, s_T)$ and its cumulative reward is:

$$R_{total}(\tau) = \sum_{t=0}^{T-1} \gamma^t \times r_t$$

The optimal policy is the policy that maximize the cumulative reward over the trajectories and is found by solving the optimization problem:

$$\pi^* = \arg\max_\pi \mathbb{E}_{\tau \sim p\pi(\tau)}[R_{total}(\tau)]$$

where $p_\pi(\tau)$ is the probability of occurrence of the trajectory $\tau$ if the agent follows the policy $\pi$. It is defined as:

$$p_\pi(\tau) = \mu_0(s_0)\pi(a_0|s_0) \prod_{t=0}^{T-1} \underbrace{\mathcal{P}(s_{t+1}|s_t, a_t)\pi(a_t|s_t)}_{\text{probability of the transition } (s_t, a_t, s_{t+1})}$$

and $\mu_0(s_0)$ is the probability of starting the trajectory in the state $s_0$.

For non-stationary environment, the dynamics $\mathcal{P}$ are not constant in time, $\mathcal{P}_t$. All the above notions are can be extended to this case. One can also note that the policy $\pi$ might also depend over time, $\pi_t$ to adapt to the evolution of the dynamics.

### 2.1.2 Wasserstein distance on RL dynamics

In this problem, a distance measure is used to bound the allowed variations from a reference transition density $\mathcal{P}_{\phi_0}$. As we have seen in class, there are several common ways to measure the closeness between two probability distributions, such as total variation distance and Kullback-Leibler divergence. However, this paper uses the Wasserstein distance to measure the distance between two different dynamics. The Wasserstein distance has several advantages: it is a true distance that is symmetrical, it can be used to compare a wide range of distributions (including discrete, continuous, and a mixture of both), and it takes into account the underlying geometry of the space in which the distributions are defined, which may be useful for learning optimal control.

In fact, the authors assume that the variation in dynamics is controlled by a threshold on the Wasserstein distance between the reference dynamics $\mathcal{P}_{\phi_0}$ and some other distribution. This assumption can be intuitively understood as saying that if an error is recurrent, it should be small in magnitude (strong probability but small distance), but if the error is significant, it should be a rare occurrence (strong distance but small probability). Essentially, this means that small variations around $\mathcal{P}_{\phi_0}$ are expected to be common, but large variations should be infrequent. For instance, in the context of autonomous driving, if we consider only the effect of climate conditions on the vehicle's dynamics, the reference dynamics might be the movement equations for dry and sunny conditions. Small variations from this reference might include slightly windy or lightly rainy conditions, while extreme variations might include snow or frost on the road. In fact, small variations are likely to occur more frequently, while extreme variations are less common.

The Wasserstein distance between two distributions $\mu$ and $\nu$ on the metric spaces $\mathcal{X}$ and $\mathcal{Y}$, is defined as

$$\mathcal{W}_p^p = \left( \min_{\kappa \in K(\mu,\nu)} \int_{X \times \mathcal{Y}} d(x,y)^p d_\kappa(x,y) \right)$$

where

$$K(\mu,\nu) = \{\kappa \in \mathcal{X} \times \mathcal{Y}; \forall (A,B) \in \mathcal{X} \times \mathcal{Y}, \kappa(A \times \mathcal{Y}) = \mu(A), \kappa(\mathcal{X} \times B) = \nu(B)\}$$

is the set of couplings between $\mu$ and $\nu$.

**Remark:** the authors only consider the case $p \geq 1$ and use the $L_2$ norm as inner distance ($p = 2$) in their algorithm.

At the beginning of this work, we used the Wasserstein distance to compare two dynamics. While this gives an idea of the method employed in the paper, it is inappropriate because the dynamics $\mathcal{P}$ are not a distribution but rather a set of distributions. In fact, for every state-action pair $(s, a)$, $\mathcal{P}_\phi(.|s, a)$ represents the probability distribution of the next state $s'$. The paper introduces the concept of the "Expected Wasserstein distance" in order to generalize the idea of the "Wasserstein distance between two dynamics".

$$\hat{\mathcal{W}}_2^2(\mathcal{P}_\phi(.), \mathcal{P}_{\phi_0}(.)) = \int_{(s,a)} \mathcal{P}(s, a)\mathcal{W}_2^2 \left(\mathcal{P}\phi(.|s, a), \mathcal{P}_{\phi_0}(.|s, a)\right) d(s, a)$$

$$= \mathbb{E}_{(s,a)\sim\mathcal{P}} \left[\mathcal{W}_2^2(\mathcal{P}_\phi(.|s, a), \mathcal{P}_{\phi_0}(.|s, a)\right]$$

The dynamics $\mathcal{P}$ is defined by the equation $\mathcal{P}(s, a) = \mathcal{P}(a|s)\mathcal{P}(s) = \pi(a|s)\rho_\pi^{\phi_0}(s)$. The purpose of this dynamics is to explore the possible state-action pairs. In the context of reinforcement learning, the policy $\pi$ is referred to as a behavioral policy and must satisfy certain conditions with $\rho_\pi^{\phi_0}(.)$, which represents the probability distribution of starting in state $s$, in order to effectively explore the state-action space. Commonly, $\rho_\pi^{\phi_0}(.)$ and $\pi(.)$ are uniform distributions over, respectively, the state and action spaces. In practice, the expected Wassertsein distance between two dynamics is computed in three steps:

- step 1: from a behavioral policy $\pi$ trajectories are sampled
- step 2: from the set of trajectories, a bucket of state-action pairs $(s, a)$ is created
- step 3: an estimate of the expected Wasserstein distance is computed using the Monte Carlo method on the set of pairs

This method is called "off-policy" because, at optimization step $t$, it does not use the current estimate of the policy $\pi_{\theta_t}$ that the algorithm is looking for, i.e., the best policy in the worst-case dynamics.

However, the explicit form of $\mathcal{P}_\phi(.|s, a)$ is not known. The agent only accesses samples drawn from these distributions given by a simulator or a black box solver. Thus, the 2-Wasserstein distance between two dynamics need to be approximated by finding the 2-Wasserstein distance between their empirical discrete distributions. Let's, $\mu$ and $\nu$, be two measures. We define their empirical distributions evaluated at samples of size $n$ as $\mu_n = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ and $\nu_n = \frac{1}{n}\sum_{j=1}^n \delta_{y_j}$, where $x_i$ and $y_j$ are iid, respectively, according to $\mu$ and $\nu$. Thus, to computed the expected Wasserstein distance, as described above, the authors use the approximation: $\mathcal{W}_2(\mu, \nu) \approx \mathcal{W}_2(\mu_n, \nu_n)$.

### 2.1.3 *Min Max* problem

The robust reinforcement learning algorithm *WRL²* aims to train a agent able to identify effective policies in worst-case scenarios -the considered dynamics for decision-making being constrained within certain bounds. The motivation for this approach comes from the field of robust optimal control, which focuses on finding optimal actions to take in the face of uncertainty caused by modeling assumptions. In these situations, an agent controlling a system may encounter an adversary trying to minimize the rewards received by the agent through the use of a disturbance controller. These types of problems can be understood as two-player games. However, solving these problems often involve making assumptions about the way disturbances are applied or limiting the range of disturbances to those with maximally bounded norms in order to optimize well-behaved objectives. In this paper, the dynamics are constrained to belong to the expected Wasserstein ball around the reference dynamics.

**Unconstrained formulation**
As previously mentioned, the goal of reinforcement learning is to identify the best policy, or the one that maximizes the cumulative reward. However, in the worst case scenario, we need to consider not just the policy that maximizes the reward, but also the dynamics that minimize the reward. This is because we want to ensure that our policy is robust and can still perform well even in the presence of unfavorable dynamics. This can be mathematically defined as follows:

$$\underbrace{\max_{\theta}}_{\text{best policy}} \left[ \underbrace{\min_{\phi}}_{\text{worse dynamics}} \mathbb{E}_{\tau \sim p_{\theta}^{\phi}(\tau)}[R_{total}(\tau)] \right]$$

where $p_{\pi}^{\phi}(\tau)$ is the probability of occurrence of the trajectory $\tau$ if the agent follows the policy $\pi$ in an environment ruled by the dynamics $\mathcal{P}_{\phi}$. It is defined by:

$$p_{\pi}^{\phi}(\tau) = \mu_0(s_0)\pi(a_0|s_0)\prod_{t=0}^{T-1}\mathcal{P}_{\phi}(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$$

$\mu_0$ is the distribution of the first state $s_0$.

**Constraints**

To improve the behavior of the optimization objectives, we will add constraints to limit the search space on the dynamics parameters and ensure that the optimization converges to a feasible transition model. Our method requires access to samples from a reference dynamics $\mathcal{P}_{\phi_0}$, which will be used to bound the learned transitions within a certain distance of the reference model. This distance is defined using the $\epsilon$-Wasserstein ball around the reference model i.e the set defined as:

$$\mathcal{B}_2(\mathcal{P}_{\phi_0}(.), \epsilon) = \left\{\mathcal{P}\phi(.) : \mathcal{W}_2^2\left(\mathcal{P}_{\phi}(.|s,a), \mathcal{P}_{\phi_0}(.|s,a)\right) \leq \epsilon, \quad \forall (s,a) \in S \times A\right\}$$

where $\epsilon \in \mathbb{R}^+$ is a hyperparameter used to specify the "degree of robustness".
However, in the case of continuous state and/or action spaces, this definition leads to an infinite number of constraints. To address this issue, the authors propose a relaxed version of the constraint that utilizes the expected Wasserstein distance between two dynamics $\hat{\mathcal{W}}_2^2(.)$, which was defined earlier in this work. This relaxed constraint is referred to as the average Wasserstein constraint:

$$\hat{\mathcal{B}}_2(\mathcal{P}_{\phi_0}(.|s,a), \epsilon) = \left\{\mathcal{P}_{\phi}(.) : \hat{\mathcal{W}}_2^2(\mathcal{P}_{\phi}(.), \mathcal{P}_{\phi_0}(.)) \leq \epsilon\right\}$$

$$= \left\{\mathcal{P}_{\phi}(.) : \mathbb{E}_{(s,a)\sim\mathcal{P}}\left[\mathcal{W}_2^2(\mathcal{P}_{\phi}(.|s,a), \mathcal{P}_{\phi_0}(.|s,a)\right] \leq \epsilon\right\}$$

**Complete problem definition**

Using this relaxed constraint, the author define the *WR²L*'s optimization problem as the search for the best policies under the assumption of worst-case bounded inside the expected $\epsilon$-Wasserstein ball:

Wasserstein Robust Reinforcement Learning Objective:

$$\max_{\theta}\left[\min_{\phi} \mathbb{E}_{\tau \sim p_{\theta}^{\phi}(\tau)}[R_{total}(\tau)]\right]$$

$$\text{s.t. } \mathbb{E}_{(s,a)\sim\mathcal{P}}\left[\mathcal{W}_2^2(\mathcal{P}_{\phi}(.|s,a), \mathcal{P}_{\phi_0}(.|s,a)\right] \leq \epsilon$$

### 2.1.4 Resolution strategy

To solve this *max min* optimization problem for $\phi$ and $\theta$, the authors use an alternating process in which, iteratively, one variable is updated while the other is fixed. In fact, one can note that the policy parameters $\theta$ do not intervene in the constraint, as we have seen that the expected Wasserstein distance and its constraint can be computed from any behavioral policy $\pi$.

**Updating policy parameters $\theta$**

For fixed dynamics parameters $\phi$, the problem simplifies to a standard reinforcement learning problem of finding the best policy in an environment governed by the dynamics $\mathcal{P}_{\phi}(.)$. Thus, the update can be made by solving the sub-problem:

$$\max_{\theta} \mathbb{E}_{\tau \sim p_{\theta}^{\phi}(\tau)}[R_{total}(\tau)]$$

As a consequence, any policy search methods can be used to update the policy parameters $\theta$ under fixed dynamics $\mathcal{P}_{\phi}$. In the article, they use the the proximal policy optimization (PPO) (Sculman et al. (2017 [SWD+17]).

**Updating dynamics parameters $\phi$**

Given fixed policy parameters $\theta$, updating the dynamics parameters involves finding the dynamics that are the most detrimental to the policy $\pi_\theta$. In contrast to the previous update of $\theta$, the update of $\phi$ is directly dependent on the average Wasserstein constraint. The sub-problem to update $\phi$ is:

$$\min_{\phi} \mathbb{E}_{\tau \sim p_\theta^\phi(\tau)}[R_{total}(\tau)]$$
$$\text{s.t. } \mathbb{E}_{(s,a)\sim\mathcal{P}}\left[\mathcal{W}_2^2(\mathcal{P}_\phi(.|s,a), \mathcal{P}_{\phi_0}(.|s,a)\right] \leq \epsilon$$

However, in a reinforcement learning setting, the agent does not have access to the explicit form of $\mathcal{P}\phi(.|s,a)$ or $\mathcal{P}\phi(.|s,a)$ (for a given pair $(s,a)$) and can only obtain samples from these distributions. Even thought the relaxation of the constraint simplifies the problem, it remains difficult to evaluate the expected Wasserstein distance $\hat{\mathcal{W}}_2^2(\mathcal{P}\phi(.), \mathcal{P}_{\phi_0}(.))$ and thus to solve the sub-problem to update $\phi$. To address this issue, the authors approximate the expected Wasserstein constrain by its second Taylor expansion up to second order.
They define the function:

$$W(\phi) := \hat{\mathcal{W}}_2^2(\mathcal{P}_\phi(.), \mathcal{P}_{\phi_0}(.)) = \mathbb{E}_{(s,a)\sim\mathcal{P}}\left[\mathcal{W}_2^2(\mathcal{P}_\phi(.|s,a), \mathcal{P}_{\phi_0}(.|s,a))\right]$$

and approximate it around $\phi_0$ by they second-order Taylor expansion:

$$W(\phi) \approx W(\phi_0) + \nabla_\phi W(\phi_0)^T(\phi - \phi_0) + \frac{1}{2}(\phi - \phi_0)^T \nabla_\phi^2 W(\phi_0)(\phi - \phi_0)$$

Moreover, the expected Wasserstein distance is null for $\phi = \phi_0$ as $\mathcal{W}_2^2$ is a distance. Moreover, it is positive (expected value of positive variable) and reachs its minimum in $\phi = \phi_0$. Thus we have $\nabla_\phi W(\phi_0)$=0. The Taylor expansion simplifies into:

$$W(\phi) \approx \frac{1}{2}(\phi - \phi_0)^T \nabla_\phi^2 W(\phi_0)(\phi - \phi_0)$$

Finally, the sub-problem used to update the dynamics parameters $\phi$ for fixed parameters $\theta$ is:

$$\min_{\phi} \mathbb{E}_{\tau \sim p_\theta^\phi(\tau)}[R_{total}(\tau)] \quad \text{s.t. } \frac{1}{2}(\phi - \phi_0)^T H_0(\phi - \phi_0) \leq \epsilon$$

where

$$H_0 = \nabla_\phi^2 W(\phi_0) = \nabla_\phi^2 \hat{\mathcal{W}}(\mathcal{P}_\phi(.), \mathcal{P}_{\phi_0}(.)) = \nabla_\phi^2 \mathbb{E}_{(s,a)\sim\mathcal{P}}\left[\mathcal{W}_2^2(\mathcal{P}_\phi(.|s,a), \mathcal{P}_{\phi_0}(.|s,a))\right]\Big|_{\phi=\phi_0}$$

is the Hessian of the expected squared 2-Wasserstein distance evaluated at $\phi_0$.

Such quadratic constraint optimization problem can be solved using interior point method. A closed form solution can also be found when considering a sub-problem on the pair of parameters $\theta^{[k]}$ and $\phi^{[j]}$ (which correspond respectively to k'th update of the policy parameters the j'th update of the dynamics parameters) to find the update dynamics parameters $\phi^{[j+1]}$:

$$\min_{\phi} \mathbb{E}_{\tau \sim p_\theta^\phi(\tau)}[R_{total}(\tau)]\Big|_{\theta^{[k]}, \phi^{[j]}} \quad \text{s.t. } \frac{1}{2}(\phi - \phi_0)^T H_0(\phi - \phi_0) \leq \epsilon$$

A minimiser to this equation can derived in a closed-form as:

$$\phi^{[j+1]} = \phi_0 - \sqrt{\frac{2\epsilon}{g^{[k,j]T} H_0^{-1} g^{[k,j]}}} H_0^{-1} g^{[k,j]}$$

where $g^{[k,j]}$ is the gradient of the expected cumulative reward $R_{total}$ evaluated at $\phi = \phi^{[j]}$ and $\theta = \theta^{[k]}$ i.e., $g^{[k,j]} = \nabla_\phi \mathbb{E}_{\tau \sim p_\theta^\phi(\tau)}[R_{total}(\tau)]\Big|_{\theta^{[k]}, \phi_{[j]}}$.

**Generic Algorithm**

These two main steps are repeated iteratively inside the final algorithm. The authors do not explicitly define the stopping criterion, but suggest that a threshold on the norm of the gradient $g^{[k,j]}$ could be used as a termination condition. We have seen that the Hessian of the expected Wasserstein distance evaluated at $\phi_0$ can be computed using any behavioral policy. Thus, it is passed, along with $\phi_0$ and $\epsilon$ as inputs of the algorithm.

### 2.1.5 Zero'th order approximation

The method described above assumes that it is easy to obtain $g^{[k,j]} = \nabla_\phi \mathbb{E}_{\tau \sim p^\phi \theta(\tau)}[R_{total}(\tau)]\Big|_{\theta^{[k]},\phi[j]}$ and $H_0$. However, in reinforcement learning, the explicit form of the reward function $R_{total}$ as well as the dynamics $\mathcal{P}_\phi$ as a function of $\phi$ are not known. A black box optimization solver (or a simulator) is use to return samples drawn from $\mathcal{P}_\phi$. The reference dynamics is also only partially known: the parameter $\phi_0$ is known, but the explicit form of the dynamics $\mathcal{P}_{\phi_0}$ is not. To solve this issue, the authors introduce an approximation method used both to estimate the gradient $g^{[k,j]}$ and the Hessian $H_0$.

Let $\mathbb{S}_\phi$ be a simulator (or black box differential equation solver) parameterized by $\phi \in \mathbb{R}^d$ and from which one can extract trajectories i.e. given a state $s$ and an action $a$ the simulator return the next state $s'$ drawn from the distribution $\mathcal{P}_\phi(.|s,a)$ and the immediate reward $r = R(s,a)$. It is assumed that the simulator returns transitions that are grounded in physics, allowing the dynamics to be varied by directly modifying the parameters $\phi$. The authors present a method to approximate the gradient of the expected cumulative reward and of the Hessian of the expected Wasserstein distance based only on function evaluations of $\mathbb{S}_\phi$.

**Gradient estimation**
For the second updating phase of the algorithm, the gradient of the expected cumulative reward $g^{[k,j]} = \nabla_\phi \mathbb{E}_{\tau \sim p^\phi_\theta(\tau)}[R_{total}(\tau)]\Big|\theta^{[k]}, \phi^{[j]}$ is required for each inner loop. The proposed estimate uses sampling from Gaussian distribution with 0 mean and $\sigma^2 I$ covariance matrix in the space of the dynamics parameters $\phi \in \mathbb{R}^d$. For fixed parameters $\theta$ and $\phi$, the gradient can be estimated by:

$$\begin{aligned}\nabla_\phi \mathbb{E}_{\tau \sim p^\phi_\theta(\tau)}[R_{total}((\tau)] &= \frac{1}{\sigma^2}\mathbb{E}_{\xi \sim \mathcal{N}(0,\sigma^2 I)}\left[\xi \int_\tau p_\theta^{\phi+\xi} R_{total}(\tau)\right] \\ &= \frac{1}{\sigma^2}\mathbb{E}_{\xi \sim \mathcal{N}(0,\sigma^2 I)}\left[\xi \mathcal{E}_{\tau \sim p_\theta^{\phi+\xi}(\tau)}[R_{total}(\tau)]\right]\end{aligned}$$

The authors provide a proof of this proposition in the paper. However, they do not specify how to compute the expected value of the reward under the dynamics $\phi + \xi$ in practice. In the numerical experiments, we use the Monte Carlo estimate based on samples of trajectories generated by the policy $\theta$ under the dynamics parameters $\phi + \xi$.

**Hessian estimation**
The concept of a zero-order gradient estimator can be extended to estimate the Hessian. Specifically, the Hessian of the expected square-Wasserstein distance can be estimated at $\phi_0$ by:

$$H_0 = \frac{1}{\sigma^2}\mathbb{E}_{\xi \sim \mathcal{N}(0,\sigma^2 I)}\left[\frac{1}{\sigma^2}\xi W(\phi+\xi)\xi^T - W(\phi+\xi)I\right]$$

**NB:** this approximation has to be computed only once as $H_0$ is considered to be an input of the algorithm.

## 2.2 Theoretical guarantees

While the zero-order approximation is the main theoretical result of the paper, we will only provide a proof for the approximation of the Hessian, as that is the focus of our discussion.

*Proof of the zero'th order estimate of the Hessian:* the authors use the second-order Taylor expansion centered at $\phi_0$ (under the assumption that the variation $\xi$ is small) on the two terms inside the expected value. Using the previous result: $W(\phi_0+\xi) \approx W(\phi_0) + \nabla_\phi W(\phi_0)^T \xi + \frac{1}{2}\xi^T \nabla^2_\phi W(\phi_0)\xi \approx \frac{1}{2}\xi^T \nabla^2_\phi W(\phi_0)\xi$, we obtain:

$$\frac{1}{\sigma^2}\mathbb{E}_{\xi \sim \mathcal{N}(0,\sigma^2 I)}\left[\frac{1}{\sigma^2}\xi W(\phi+\xi)\xi^T - W(\phi+\xi)I\right]$$

$$\approx \frac{1}{\sigma^2}\mathbb{E}_{\xi \sim \mathcal{N}(0,\sigma^2 I)}\left[\frac{1}{\sigma^2}\frac{1}{2}\underbrace{\xi\xi^T \nabla^2_\phi W(\phi_0)\xi\xi^T}_{B} - \frac{1}{2}\underbrace{\xi^T \nabla^2_\phi W(\phi_0)\xi}_{C} I\right]$$

Expected value of $B$: $B$ can be decomposed as $B_{i,j} = \sum_{n=1}^{d_2}\sum_{m=1}^{d_2} \xi_i\xi_j\xi_n\xi_m H_0^{[m,n]}$. Three sub-cases can be identify both for diagonal and off-diagonal elements:

- **Diagonal elements** $i = j$
  - **case** $i = j = n = m$**:** $\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}[\xi_i\xi_j\xi_n\xi_m H_0^{[m,n]}] = H_0^{[i,i]}\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}(\xi_i^4) = 3\sigma^4 H_0^{[i,i]}$ (moment of order four of a zero mean Gaussian of co-variance $\sigma^2 I$)
  - **case** $i = j \neq m = n$**:** $\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}[\xi_i\xi_j\xi_n\xi_m H_0^{[m,n]}] = H_0^{[m,m]}\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}(\xi_i^2\xi_m^2) = \sigma^4 H_0^{[m,m]}$
  - **case** $i = j \neq n \neq m$**:** $\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}[\xi_i\xi_j\xi_n\xi_m H_0^{[m,n]}] = 0$

  which leads to the result that $\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}[B_{i,i}] = 2\sigma^2 H_0^{[i,i]} + \sigma^4 trace(H_0)$
- Off-Diagonal elements $i \neq j$
  - **case** $i = m \neq j = n$**:** $\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}[\xi_i\xi_j\xi_n\xi_m H_0^{[m,n]}] = H_0^{[i,j]}\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}(\xi_i^2\xi_j^2) = \sigma^4 H_0^{[i,j]}$
  - **case** $i = n \neq j = m$**:** $\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}[\xi_i\xi_j\xi_n\xi_m H_0^{[m,n]}] = H_0^{[j,i]}\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}(\xi_i^2\xi_j^2) = \sigma^4 H_0^{[j,i]}$
  - **case** $i \neq j \neq n \neq m$**:** $\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}[\xi_i\xi_j\xi_n\xi_m H_0^{[m,n]}] = 0$

  Using the symmetry of the Hessian matrix, we obtain $\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}[B_{i,j}] = 2\sigma^2 H_0^{[i,j]}$

Expected value of $C$: $\mathbb{E}_{\xi\sim\mathbb{N}(0,\sigma^2 I)}[C] = \mathbb{E}_{\xi\sim\mathbb{N}(0,\sigma^2 I)}[\sum_{i=1}^{d_1}\sum_{j=1}^{d_2}\xi_i\xi_j H_0^{[i,j]}] = \sigma^2 trace(H_0)$

Using the linearity of the expected value and by substituting these results in the approximation, we obtain:

$$\frac{1}{\sigma^2}\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}\left[\frac{1}{\sigma^2}\xi W(\phi+\xi)\xi^T - W(\phi+\xi)I\right] \approx H_0$$

□

## 2.3 Numerics

For the numerical study, we will focus on analyzing the second step of the algorithm, which involves finding the worst-case dynamics for a given policy. First, we will examine the method of approximating the Hessian of the Wasserstein distance, outside of the context of reinforcement learning. In the second part, we will apply the entire problem-solving process to a simple tabular MDP environment in which the dynamics vary over time.

### 2.3.1 Minimization on a Wasserstein ball

In this part, we consider a minimization problem apart from the application to reinforcement learning. The RL dynamics are simplified by removing the states and actions and considering directly $\mathcal{P}_\phi$ as a distribution parameterized par a parameter $\phi \in \mathbb{R}$. The constraint is no more based on the expected Wasserstein distance but rather on the Wasserstein distance between $\mathcal{P}_\phi$ and $\mathcal{P}_{\phi_0}$ . In this setting, the problem can be simplified to:

$$\min_\phi R(\phi) \quad \text{s.t } \underbrace{\mathcal{W}_2^2\left(\mathcal{P}_\phi(.), \mathcal{P}_{\phi_0}(.)\right)}_{W(\phi)} \leq \epsilon$$

Using the second-order Taylor expansion of $W(\phi)$ results in the following approximate problem:

$$\min_\phi R(\phi) \quad \text{s.t } \frac{1}{2}(\phi-\phi_0)^T \underbrace{\nabla_\phi^2 W(\phi_0)}_{H_0}(\phi-\phi_0) \leq \epsilon$$

In order to evaluate the accuracy of the approximations made by the algorithm, we will consider a set of Gaussian distributions parameterized by $\phi = (\mu, \sigma)$, where $\mu$ is the mean and $\text{diag}(\sigma^2)$ is the covariance matrix. Both $\mu$ and $\sigma$ are in $\mathbb{R}^d$ (where $d \geq 1$) and the dimension of the parameter space is $\phi \in \mathbb{R}^{2d}$. The reference dynamics is a Gaussian distribution with zero mean and identity covariance. In summary, we are working with the following:

- $\mathcal{P}_{\phi=(\mu,\sigma)}(.) \sim \mathcal{N}_d(\mu, diag(\sigma^2))$
- reference dynamics $\phi_0 = (\mu_0, \sigma_0) = (0_d, 1_d)$

We choose this set of distributions as the Wasserstein distribution between two Gaussian distributions in $\mathbb{R}^d$, $\alpha \sim \mathcal{N}_d(\mu_\alpha, \Sigma_\alpha)$ and $\beta \sim \mathcal{N}_d(\mu_\beta, \Sigma_\beta)$, can be computed thanks to the formula:

$$\mathcal{W}_2^2(\alpha, \beta) = ||\mu_\alpha - \mu_\beta||^2 + \mathcal{B}(\Sigma_\alpha, \Sigma_\beta)^2$$

Where $\mathcal{B}$ is the Bures's mectric defined as

$$\mathcal{B}(\Sigma_\alpha, \Sigma_\beta)^2 = tr(\Sigma_\alpha + \Sigma_\beta - 2 \times (\Sigma_\alpha^{\frac{1}{2}} \Sigma_\beta \Sigma_\alpha^{\frac{1}{2}})^{\frac{1}{2}})$$

In our case, $\Sigma_\alpha = \text{diag}(\sigma_\alpha^2)$ and $\Sigma_\beta = \text{diag}(\sigma_\beta^2)$

$$\mathcal{B}(\Sigma_\alpha, \Sigma_\beta) = ||\sigma_\alpha - \sigma_\beta||_2$$

And finally,

$$\mathcal{W}_2^2(\alpha, \beta) = ||\mu_\alpha - \mu_\beta||_2 + ||\sigma_\alpha - \sigma_\beta||_2 = ||\phi_\alpha - \phi_\beta||_2$$

**Quantization approximation error**

Since in a RL setting, the true form of the dynamics $\mathcal{P}_\phi$ is unknown and the simulator only provides samples drawn from its distribution, in the paper, the Wasserstein distance (or its expected value) between $\mathcal{P}_\phi$ and the reference dynamics $\mathcal{P}_{\phi_0}$ is approximated using the Wasserstein distance between their empirical (and discrete) distributions. That is to say, we consider:

- $\alpha = \frac{1}{n} \sum_{i=1}^n \delta_{x_i}$ where $x_i$ are identically and independently distributed according to $\mathcal{P}_\phi(.|s, a)$
- $\beta = \frac{1}{n} \sum_{i=j}^n \delta_{y_j}$ where $y_i$ are identically and independently distributed according to $\mathcal{P}_{\phi_0}(.|s, a)$

Where $n$ is a parameter of the problem, to make the approximation:

$$\mathcal{W}_2^2\left(\mathcal{P}_\phi(.), \mathcal{P}_{\phi_0}(.)\right) \approx \mathcal{W}_2^2\left(\alpha, \beta\right)$$

The Wasserstein distance between two discrete distributions, each with uniform mass and same size of supports is a Monge problem, and can be calculated using linear programming. In this case, like in the numerical tours, we utilize the library CVXPY to perform this computation.

To evaluate the error of this approximation, we generate 40 random distributions $\mathcal{P}_\phi(.) = \mathcal{N}(\mu, \text{diag}(\sigma^2))$, where $\mu$ is drawn from the uniform distribution $\mathcal{U}d$ on the interval $[-10, 10]$ and $\sigma$ is drawn from the uniform distribution $\mathcal{U}d$ on the interval $[0, 10]$. For each of these distributions, we draw 10 samples to compute a 95% confidence interval for the error $\mathcal{W}_2^2\left(\alpha, \beta\right) - W_2^2(\mathcal{P}\phi(.), \mathcal{P}\phi_0(.))$ at sample sizes $n = 100$ and $n = 200$. The error is then plotted as a function of the true Wasserstein distance in figure 1.

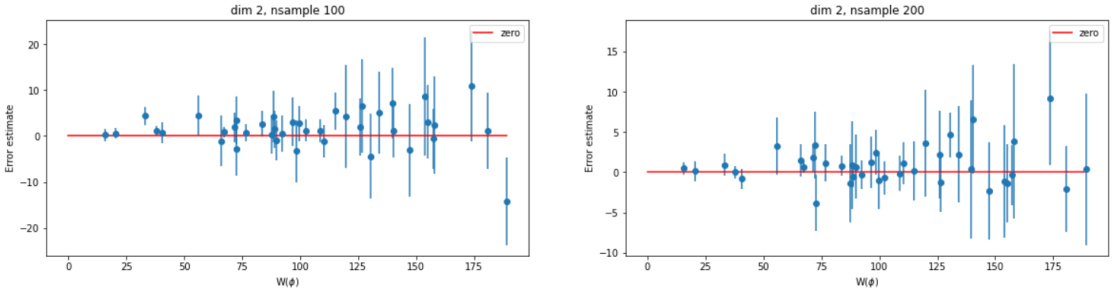



Figure 1: Monte-Carlo estimate of the quantization error on $W(\phi)$ (c.i 95%)

As expected, Figure 1 shows that using $n = 200$ samples for the empirical distributions leads to a smaller error in the Wasserstein distance compared to using $n = 100$ samples (errors range between $-25$ and 20 for $n = 100$ and between $-10$ and 15 for $n = 200$ among the 40 samples). However this reducing of the error has a cost in term of computation time, a single step going from 0.2s to 1.1s of computation time.

Furthermore, in the reinforcement learning setting, this computation must be performed multiple times in order to estimate the expected Wasserstein distance for a given $\phi$ as part of a single step in the Monte-Carlo approximation of the zero'th order approximation of the Hessian. Thus this process may need to be repeated numerous times, which will result in a high computational cost.

Additionally, the mean and variance of the error increase as the Wasserstein distance between the random dynamics and the reference dynamics increases. However, for small distances ($\leq 50$), the errors appear to be negligible for both $n = 100$ and $n = 200$ samples. In the paper, the approximation is made on $W(\phi_0 + \xi)$, where $\xi \sim \mathcal{N}(0, \sigma^2 I)$, in order to approximate the Hessian $H_0$. As a result, it could be argued that for small values of $\sigma$, the Wasserstein distance between $\mathcal{P}\phi_0$ and $\mathcal{P}\phi_0 + \xi$ will be small, leading to a negligible error introduced by the quantization of the distributions.

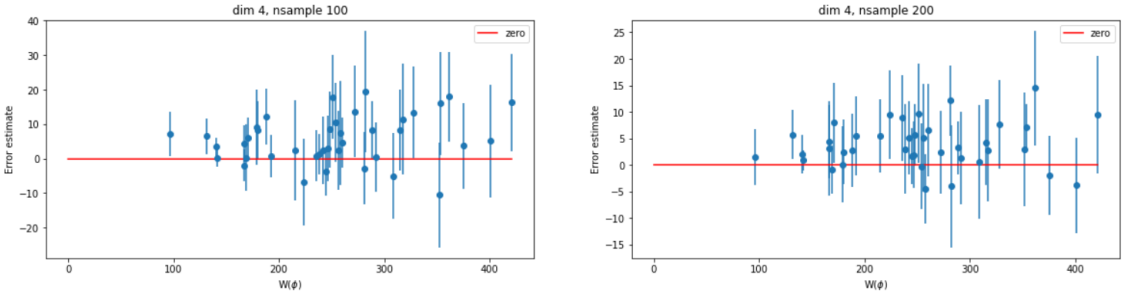



Figure 2: lMonte-Carlo estimate of the quantization error on $W(\phi)$ (c.i 95%)

However, when the dimension of the Gaussian distributions is increased to 4 (as shown in Figure 2), using the same method to generate the random distributions $\mathcal{P}_\phi$, both the mean and variance of the error significantly increase for both $n = 100$ and $n = 200$. This can be attributed to the increased number of degrees of freedom of the disturbances, allowing the random distribution to deviate more significantly from the reference distribution. This might be an issue as the authors of the paper specify that one of the advantages of the algorithm was that it can be used for high dimensional RL environment.

**Zero'th order approximation error**

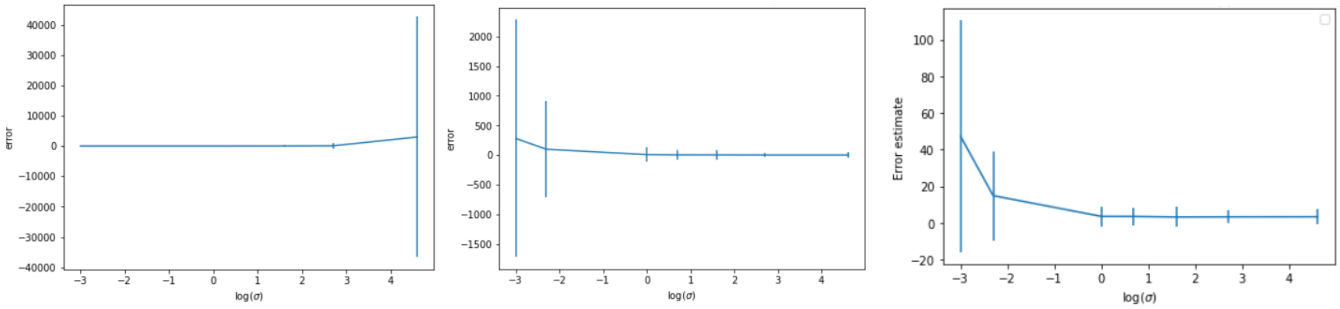
$$H_0 \approx \frac{1}{\sigma^2}\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}\left[\frac{1}{\sigma^2}\xi(W(\phi_0+\xi))\xi^T - W(\phi_0+\xi)I\right]$$

An additional approximation in the Hessian calculation is the zero-order approximation. To determine the error introduced by this approximation, we derived the analytic form of the Hessian using the explicit formula for the Wasserstein distance between two Gaussian distributions:

$$H_0 = \nabla_\phi W(\phi) = 2 \times I_{2d}$$

To evaluate the error between the estimate of the Hessian and its true value, the Frobenius norm is used on the difference between the two terms. According to the previous experiment, we expected to see the error increase as the deviation, characterized by the parameter $\sigma$, increases. However, in Figure 3 a) b), we can see that while the error of the Wasserstein distance increases with $\sigma$, the error of the term inside the expected value of the zero-order estimate $\frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}\xi(W(\phi_0+\xi))\xi^T - W(\phi_0+\xi)I\right)$ decreases. As a result, in Figure 3 c), the error of the zero-order approximation using the Wasserstein distance on the empirical distributions is smaller for strong deviation from the reference dynamics, i.e. for large $\sigma$.

To conduct this experiment, we used empirical distributions with support of size $n = 100$ and ran 40 trials for each value of $\sigma$. The tested values were in the range $[0.05, 0.1, 1, 2, 5, 15, 100]$.




(a) error Wasserstein distance

(b) error on the term inside the zero-order expected value

(c) error on the final estimate of $H_0$

Figure 3: Monte-Carlo estimate of the error on $W(\phi)$, on the term inside the expected value in the zero-order approximation and on the final estimate of $H_0$, as a function of the $log(\sigma)$ (c.i 95%)

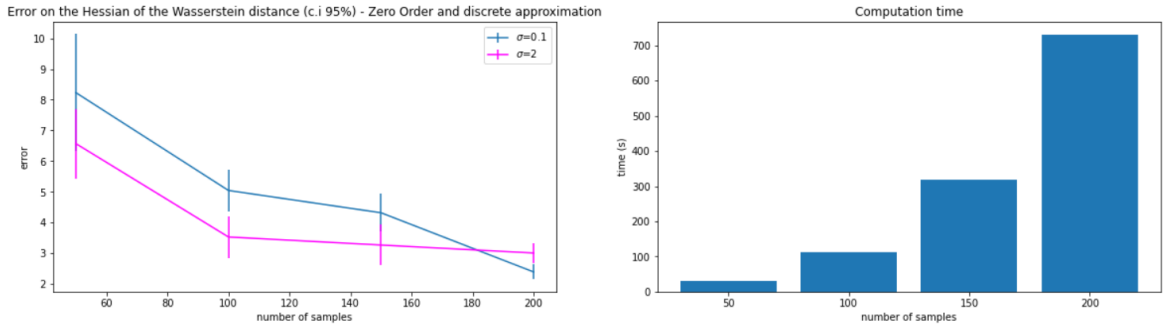



Figure 4: Monte-Carlo estimate of the error on $H_0$ as a function of the number of samples used in the empirical distributions, for standard deviation in the zero-order approximation of 2 and 0.1 (c.i 95%)

As shown in Figure 4, when the number of samples used to generate the empirical distributions is increased, the difference in behavior between the standard deviation values becomes less noticeable. For both standard deviation values of 2 and 0.1, a larger number of samples leads to a significant decrease in the error on the Hessian computation. With a sample size of $n = 200$, a standard deviation of 0.1 actually results in lower error than a standard deviation of 2. However, it's important to note that this improvement in accuracy comes at the cost of increased computational expense.

In summary, the performance of the Hessian approximation for the Wasserstein distance presented in the paper depends on three key factors: the number of samples used to estimate the empirical distributions, the standard deviation of the perturbations applied to the reference parameters, and the number of Monte Carlo runs of the zero-order approximation. It should be noted that achieving a low error rate requires a significant computational effort.

### 2.3.2 Worse-case dynamics in a tabular MDP

In this section, we will use the method described in the paper to find the worst-case dynamics for a given policy $\pi$ (step 2 of the algorithm) in a simple reinforcement learning setting. As previously discussed, this problem can be approximated by the following formulation:

$$\min_{\phi} \mathbb{E}_{\tau \sim p_\theta^\phi(\tau)} \left[ R_{total}(\tau) \right] \quad \text{s.t } \frac{1}{2} (\phi - \phi_0)^T \underbrace{\nabla_\phi^2 W(\phi_0)}_{H_0} (\phi - \phi_0) \leq \epsilon$$

In this study, we explore a 2D table-based reinforcement learning environment in which an agent begins at an initial state ($s_{init}$) and tries to reach a goal state ($s_{final}$) in the shortest amount of time. The time horizon is infinite, but the cumulative reward is influenced by a discount factor $\gamma \in [0, 1]$. The

agent has four possible actions, which correspond to moving in one of the four cardinal directions in 2D space. When the agent is in a state $s$ and takes action $a$, it will move $n$ spaces in the direction of $a$, where $n$ is drawn from a geometric distribution with parameter $(1-\phi)$, and $\phi \in [0,1]$ is a parameter of the dynamics $\mathcal{P}_\phi$(transition function) of the environment. The reference value of the dynamics parameter is $\phi_0 = 0.5$. In summary, the environment is characterized as follows:

- the action space $A = \{top, bottom, right, left\}$ and the state space $S = \{(x,y) \text{ for } x \in [1,N], y \in [1,M]\}$ are finite and discrete
- the state goal is $s_{final} = (N, M)$
- the initial state is $s_{init} = (0,0)$
- the transition matrices $P_\phi \in \mathbb{R}^{(|S|\times|A|\times|S|)}$, $\mathcal{P}_\phi(s'|s,a) = \begin{cases} s + (G,0) & \text{if } a = top, \\ s - (G,0) & \text{if } a = bottom, \\ s + (0,G) & \text{if } a = right, \\ s - (0,G) & \text{if } a = left, \end{cases}$
  with $G \sim \mathcal{G}(1-\phi)$ geometric law of parameter $\phi \in [0,1]$
- the reward function is $R(s) = \begin{cases} 1 & \text{if } s = s_{final}, \\ 0 & \text{otherwise} \end{cases}$
- the cumulative reward of a trajectory $\tau$ is $R_{total}(\tau) = \gamma^{T_\tau}$ with $T_\tau = inf\{T, \text{ s.t } s_T^\tau = s_{final}\}$

Figure 5 shows dynamics for all the directions at a certain state for tree values of $\phi$ in an environment of size $(10, 20)$. The dynamics' parameters are drawn from a Gaussian distribution around $\phi_0 = 0.5$ of standard deviation 0.1 and clipped inside $[0,1]$.

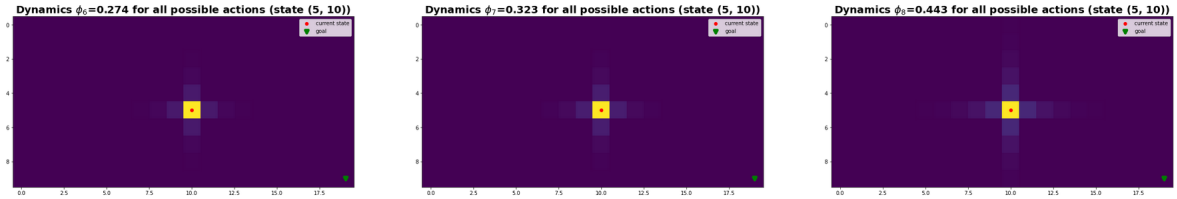



Figure 5: Examples of dynamics in all the direction for the state $s = (5, 10)$ for different values of parameter $\phi$

**Remark:** The probability distribution $\mathcal{P}_\phi(.|s,a)$ must sum to one, which means that all probability mass must be accounted for. When the distribution is evaluated at the border of the state space, some states may not be available. In this case, the probability mass for these unavailable states is uniformly redistributed among the available states i.e. the intersection between the support of the Geometric law and the state space.

**Policy**

The worst-case dynamics is calculated for a given policy. In this work, we use a naive policy that aims to reach the goal, at the lower right corner of the table, starting in the most left top corner, by moving right or bottom with probability 0.5, or choosing one of the two options if the other is not available. In other words:

$$\pi_0(s,a) = \begin{cases} 0 & \text{if } a = left \\ 0 & \text{if } a = top, \\ 0.5 & \text{if } a = right, \text{ and } y_s \neq M \ 0 \text{ otherwise.} \\ 0.5 & \text{if } a = bottom, \text{ and } x_s \neq N \ 0 \text{ otherwise.} \end{cases}$$

**Cumulative reward**

As shown in Figure 6, the cumulative reward increases with increasing values of $\phi$. This is because higher values of $\phi$ result in larger displacements within the RL environment, leading to shorter times to reach the goal and higher rewards. The cumulative reward is calculated by simulating 1000 trajectories of an agent following policy $\pi_0$ in transition model $\mathcal{P}_\phi$, for various values of $\phi$. The value of $\gamma$ used is 0.9.

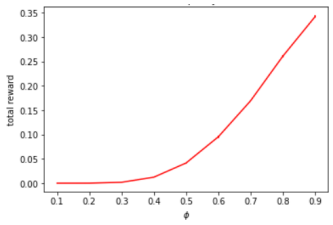



Figure 6: Estimation of the cumulative reward as a function of $\phi$ (c.i 95%)

**Expected Wasserstein distance**

In the algorithm, the Wasserstein distance is computed on the empirical distribution of the dynamics on a state-action pair $(s, a)$. This approximated distance is denoted as $\hat{\mathcal{W}}(s, a)(\phi)$. Then, to determine the set of possible dynamics, we consider the expected Wasserstein distance, denoted by $\mathbb{E}(s, a) \sim \pi(.)\rho_{\pi}^{\phi_0}(.)[\hat{\mathcal{W}}(s, a)(\phi)]$, for all state-action pairs $(s, a)$ that are generated by a behavioral dynamics $\mathcal{P}(.) = \pi(.)\rho_{\pi}^{\phi_0}(.)$ with the goal of exploration, as seen in the previous sections. To compute this expectation we follow the instruction given in the paper:

- We create a batch of trajectories sampled from the uniform over actions policy $\pi_u$ (the trajectory can have different length as the trajectory finishes when the agent reaches the goal)
- From the batch of trajectories, we create a bucket of state-action pairs
- Given such data, we compute the expected Wasserstein distance over the pairs using Monte-Carlo estimation

Figure 7 illustrates the estimate of the expected Wasserstein distance computed on various numbers of state-action pairs $(s, a)$, plotted against the range of dynamics parameter $\phi$ from 0 to 1.

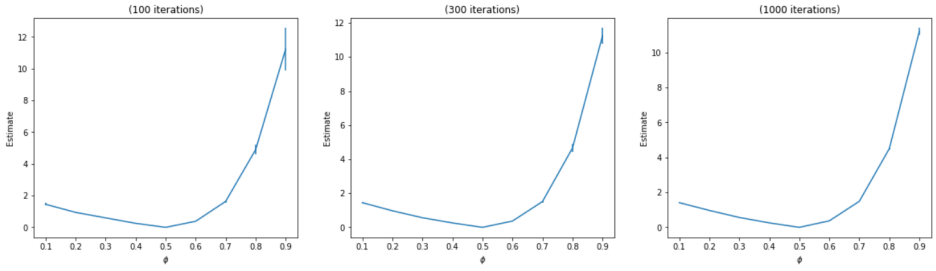



Figure 7: Expected Wasserstein distance computed with various number of state-action pairs 100, 300 and 1000

The expected distance reaches 0 at $\phi_0$. It is also interesting to observe that the distance is higher for values of $\phi > \phi_0$ than for values $\phi < \phi_0$. The Wasserstein ball (an interval in this case) can be obtained directly from this plot, for a given robustness parameter $\epsilon$.

**Resolution**

Intuitively, the solution to the problem for a given $\epsilon$, or the worst-case dynamics in terms of cumulative reward for the naive policy $\pi_0$, will be the smallest value of $\phi$ such that the expected Wasserstein distance to the reference dynamics $\mathcal{P}_{\phi_0}$ is less than $\epsilon$. This represents the minimum bound of the interval known as the $\epsilon$ expected Wasserstein ball.

We want to find this result using the method described in the paper, which involves using the closed-form formula:

$$\phi_{opt} = \phi_0 - \sqrt{\frac{2\epsilon}{g_j^T H_0^{-1} g_j}} H_0^{-1} g_j$$

where $g_j$ is the zero-order approximation of the gradient for a particular dynamics parameter $\phi_j$, given by $g_j = \frac{1}{\sigma^2}\mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I)}\left[\xi \int \tau p_{\theta}^{\phi_j+\xi}(\tau) R_{total}(\tau) d\tau\right]$, and $H_0$ is also computed using a zero-order approximation as described in the paper.

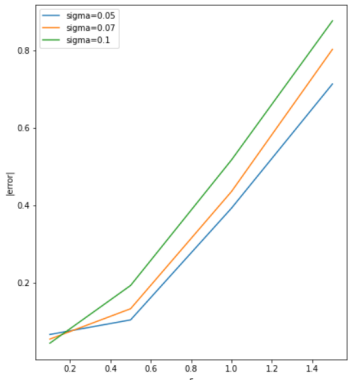



Figure 8: Expected Wasserstein distance computed with various number of state-action pairs 100, 300 and 1000

The plot Figure 8 displays the error in the expected Wasserstein constraint for different values of parameter $\epsilon$ in [0.05,0.07,0.1]. In fact, as the constraint should be saturated, the expected Wasserstein distance between the dynamics of the optimal parameters $\phi_{opt}$ and the reference ones should be equal to the robustness parameter $\epsilon$. The plotted error represents the absolute value of the difference between these two quantities. One can note that the error is the smallest for a robust parameter of $\epsilon = 0.05$, thus for small disturbance of the reference dynamics' parameters $\phi_0$. Yet, the results are highly variable and do not exhibit a clear pattern even when multiple trials are conducted using different samples.

To do this experiment, we used 200 samples of state-action pairs to compute the expected Wasserstein distance in 1000 Monte-Carlo steps for the zero-order approximation of the Hessian. The same number of steps is used to compute the zero-order approximation of the gradient evaluated for the value $\phi = 0.6.\text{t}$

# 3 Conclusion and perspective

In the paper, the authors propose a method for robust reinforcement learning that involves formulating the problem as a constrained min-max optimization. The goal of this approach is to make the training process more robust by training the agent to follow a policy that is resilient to the worst-case dynamics and so to any "bad" dynamics. The set of candidate dynamics is represented by an expected Wasserstein ball centered on a reference dynamics and with a radius of $\epsilon$, which is a robustness parameter. To solve this problem, the authors also introduce an approximation method of the Hessian of the (expected) Wasserstein distance evaluated at the reference dynamics.

In this work, we focus on the sub-problem of constrained optimization with respect to the Wasserstein distance. In the first part of our study, we attempted to estimate the error introduced by successive approximation methods when computing the Hessian of the Wasserstein distance, and found that a large number of computations and appropriate hyperparameter selection are necessary for accurate estimation of the Hessian $H_0$. In the second part of the study, we apply the resolution method to a simple non-stationary 2D tabular RL environment. This numerical experiment demonstrates that the method can estimate the parameters of the worst-case dynamics for a given policy, but it is computationally costly and not very precise.

The results of the paper, on the provided examples, in which dynamics deviations from a reference dynamics are applied in complex Mujoco RL environments, are quite impressive. Despite only having access to samples drawn from the reference dynamics during training, the agent is able to adapt to the new environment and exhibit good behavior. While the results are not very precise, as we have seen in the numerical experiments, the goal of the algorithm is not necessarily to find the optimal policy, but rather to find a policy that does not diverge on unseen transition models. In fact, the zero-order approximation method used to estimate the Hessian of the Wasserstein distance is not highly precise, but it requires very little information about the reference dynamics on which it is being evaluated to return a relatively good solution.

One of the major limitations of this method, which is a common issue for many reinforcement learning algorithms, is the high computational cost. In our numerical experiment, we only considered a small sub-problem in a low-dimensional setting, and the method took over two hour to return a result. Additionally, this sub-problem would need to be solved multiple times in the overall algorithm, even though the approximation of the Hessian $H_0$ is done only once. In addition, a second sub-problem involving the computation of an optimal policy for a given dynamics, would have to be solved at each step, which may also be computationally expensive. While the paper provides comparisons with other robust reinforcement learning algorithms, it does not address computational cost. It would also have been useful to mention the hardware architecture used to train the algorithm, as this information is necessary for reproducibility of the study.

Another limitation of the method is that it assumes that the parameters of the reference dynamics $\phi_0$ are known. However, in high-dimensional and complex RL environments, for which the algorithm is claimed to be scalable, it may not be straightforward to find a reference dynamics that satisfies the assumption that any deviations from it are either small or occur with low probability.

### 3.1 Improvement(s) and extension(s)

It would have been helpful if the authors had provided more information about the training architecture of their RL agent and the computation time needed to obtain the results. Additionally, the paper states that the reference dynamics parameters must be known, but in a real-world setting, it may not be possible to have access to this information. It would be interesting to see how the algorithm performs when the parameter $\phi_0$ is estimated using statistical tests or neural networks, for example.

## 4 Link with the course

### 4.1 Gaussian distribution and Monge problem

The sub-problem we were interested in involve the computation the Wasserstein distance between a reference set of dynamics $\mathcal{P}_{\phi_0}$ and parameterized set of dynamics $\mathcal{P}_{\phi}$. Yet, in the context of reinforcement learning, only samples drawn from the distributions are available. To solve this issue, the authors approximate the true distance by the distance on the empirical distributions extracted from the sampled data. In fact, we saw in class that computing the 2-Wasserstein distance $\mathcal{W}_2(\alpha, \beta)$ on measures discrete measures with the same support's size $\alpha = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}$ and $\beta = \frac{1}{n}\sum_{i=j}^{n} \delta_{y_j}$ is defined as a Monge optimal matching problem. The optimal coupling in this case is a permutation i.e. a binary bi-stochastic matrix. The Figure 9 illustrates this result. In the paper, the authors do not provide the method used to compute the empirical Wasserstein distance but we saw in the course that such problem could be solve using linear programming. In particular, we saw in the numerical tour how to solve the linear constrained problem using the library CVXP.

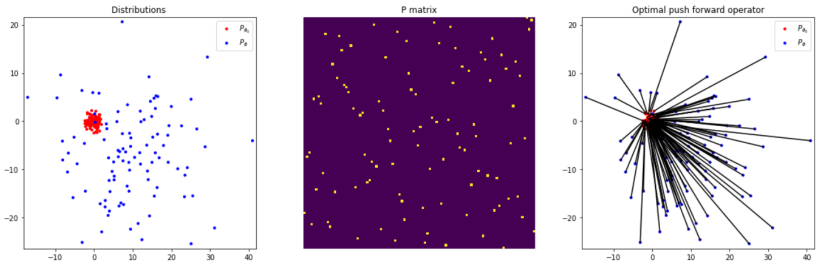



Figure 9: Optimal coupling of Monge problem between two Gaussian distributions dynamics of parameters $(\mu_0, \sigma_0) = (0_2, 1_2)$, $(\mu, \sigma) = ((8.83, -5.96), (8.89, 9.70))$ (the squared 2-Wasserstein distance is 283.71)

Moreover, , to evaluate the method of approximating the Hessian of the Wasserstein distance introduced in the paper, we applied the results from the course on Gaussian distributions. In fact, we saw that the Wasserstein distance between two Gaussian distributions $\alpha$ and $\beta$ in any dimension $\mathcal{R}^d$ of the form $\mathcal{N}(\mu, diag(\sigma^2))$ is given by the formula $\mathcal{W}_2^2(\alpha, \beta) = ||\mu_\alpha - \mu_\beta||^2 + \mathcal{B}(\Sigma_\alpha, \Sigma_\beta)^2 =$

$||\mu_\alpha - \mu_\beta||_2 + ||\sigma_\alpha - \sigma_\beta||_2$. As a result, we were able to compute the true values of the Wasserstein distance and its Hessian and compare them to their estimated values.

## 4.2 1D Kantorovitch problem

In the second part of the numerical work, we applied the resolution method to a simple 2D tabular RL environment. For a given state-action pair $(s, a)$, we used a Geometric distribution with parameter $(1 - \phi)$ as the dynamics $\mathcal{P}\phi(.|s, a)$. However, we wanted the agent to travel in the chosen direction $a$. Therefore, we applied the Geometric law only in this direction. As a result, computing the true Wasserstein distance between $\mathcal{P}\phi(.|s, a)$ and $\mathcal{P}\phi_0(.|s, a)$ is equivalent to computing the distance between the discrete measures on the support of the distributions $S\phi = \{s' \in S : \mathcal{P}\phi(s'|s, a) \geq tol\}$, where $tol$ is a threshold, with mass the probability , i.e., on $\alpha = \sum_{s' \in S_\phi} \delta_{s'}$ and $\alpha_0 = \sum_{s' \in S_{\phi_0}} \delta_{s'}$ with mass the probability of reaching the state $\mathcal{P}_\phi(s'|s, a)$. One can note that, for a given action $a$, the two supports are always on the same column or row of the grid environment. Therefore, computing the Wasserstein distance between $\alpha$ and $\alpha_0$ is equivalent to solving a Kantorovich problem in one dimension. We saw in the course that for 1D discrete measures, the Brenier theorem ensures that the optimal transport map $T$ is increasing, as $T = \nabla\Phi$ with $\Phi$ convex, which means that $\forall(x, y) \in \mathcal{X} \times \mathcal{Y}$ $(T(x) - T(y))(x - y) \geq 0$. As a result, an optimal coupling matrix $P$ can be computed by sorting the points and then sweeping the mass in a single pass from left to right, as shown in Figure 10 (bottom to top in Figure 10). This process has a time complexity of $O(nlog(n) + mlog(m))$ (in our case, the complexity is $O(1)$). In the numerical work, we used the CVXP linear programming algorithm.

Figure 10: Optimal coupling of 1D Kantorovitch problem between two Geometric distributions dynamics of parameters $\phi = 0.221$ and $\phi_0 = 0.5$ (threshold=$1e - 3$)

# References


[ARB+19] Mohammed Amin Abdullah, Hang Ren, Haitham Bou-Ammar, Vladimir Milenkovic, Rui Luo, Mingtian Zhang, and Jun Wang. Wasserstein robust reinforcement learning. *CoRR*, abs/1907.13196, 2019.

[CTMP15] Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. *CoRR*, abs/1506.02188, 2015.

[HS01] LarsPeter Hansen and Thomas J. Sargent. Robust control and model uncertainty. *American Economic Review*, 91(2):60–66, May 2001.

[Iye05] Garud N. Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

[LR19] Erwan Lecarpentier and Emmanuel Rachelson. Non-stationary markov decision processes a worst-case approach using model-based reinforcement learning. *CoRR*, abs/1904.10090, 2019.

[ND16] Hongseok Namkoong and John C Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[NG05] Arnab Nilim and Laurent El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[PAZA17] 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.

[RGLR16] Aravind Rajeswaran, Sarvjeet Ghotra, Sergey Levine, and Balaraman Ravindran. Epopt: Learning robust neural network policies using model ensembles. *CoRR*, abs/1610.01283, 2016.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[TEM19] Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6215–6224. PMLR, 09–15 Jun 2019.

[Yan17] Insoon Yang. A convex optimization approach to distributionally robust markov decision processes with wasserstein distance. *IEEE Control Systems Letters*, 1(1):164–169, 2017.