RISK-AVERSE OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Training Reinforcement Learning (RL) agents online in high-stakes applications is often prohibitive due to the risk associated with exploration. Thus, the agent can only use data previously collected by *safe* policies. While previous work considers optimizing the *average* performance using offline data, we focus on optimizing a *risk-averse* criterion. In particular, we present the *Offline Risk-Averse Actor-Critic* (O-RAAC), a model-free RL algorithm that is able to learn risk-averse policies in a fully offline setting. We show that O-RAAC learns policies with higher risk-averse performance than risk-neutral approaches in different robot control tasks. Furthermore, considering risk-averse criteria guarantees distributional robustness of the average performance with respect to particular distribution shifts. We demonstrate empirically that in the presence of natural distribution-shifts, O-RAAC learns policies with good average performance.

1 INTRODUCTION

In high-stakes applications, the deployment of highly-performing Reinforcement Learning (RL) agents is limited by prohibitively large costs at early exploration stages (Dulac-Arnold et al., 2019). To address this issue, the offline (or batch) RL setting considers learning a policy from a limited batch of pre-collected data. However, high-stakes decision-making is typically also *risk-averse*: we assign more weight to adverse events than to positive ones (Pratt, 1978). Although several algorithms for risk-sensitive RL exist (Howard & Matheson, 1972; Mihatsch & Neuneier, 2002), none of them addresses the offline setting. On the other hand, existing offline RL algorithms consider the *average* performance criterion and are risk-neutral (Ernst et al., 2005; Lange et al., 2012).

Main contributions We present the first approach towards *learning a risk-averse RL policy for high-stakes applications using only offline data*: the **O**ffline **R**isk-**A**verse **A**ctor-**C**ritic (O-RAAC). The algorithm has three components: a distributional critic that learns the full value distribution (Section 3.1), a risk-averse actor that optimizes a risk averse criteria (Section 3.2), and an imitation learner implemented with a variational auto-encoder (VAE) that reduces the bootstrapping error due to the offline nature of the algorithm (Section 3.3). In Figure 1, we show how these components interact with each other. Finally, in Section 4 we demonstrate the empirical performance of O-RAAC.

1.1 RELATED WORK

Risk-Averse RL The most common risk-averse measure in the literature is the Conditional Value-at-Risk (CVaR) (Rockafellar & Uryasev, 2002), which corresponds to the family of Coherent Risk-Measures (Artzner et al., 1999), and we focus mainly on these risk-measures. Nevertheless, other risk criteria such as Cumulative Prospect Theory (Tversky & Kahneman, 1992) or Exponential Utility (Rabin, 2013) can also be used with the algorithm we propose. In the context of RL, Petrik & Subramanian (2012); Chow & Ghavamzadeh (2014); Chow et al. (2015) propose dynamic programming algorithms for solving the CVaR of the return distribution with *known* tabular Markov Decision Processes (MDPs). For unknown models, Morimura, Tetsuro and Sugiyama, Masashi and Kashima, Hisashi and Hachiya, Hirotaka and Tanaka (2010) propose a SARSA algorithm for (CVaR) optimization but it is limited to the on-policy setting and small action spaces. To scale to larger systems, Tamar et al. (2012; 2015) propose on-policy Actor-Critic algorithms for Coherent Risk-Measures. However, they are extremely sample inefficient though due to the high-variance of the gradient estimate and sample discarding to compute the risk-criteria. While Prashanth et al. (2016) address sample efficiency by considering Cumulative Prospect Theory instead of Coherent Risk-Measures,

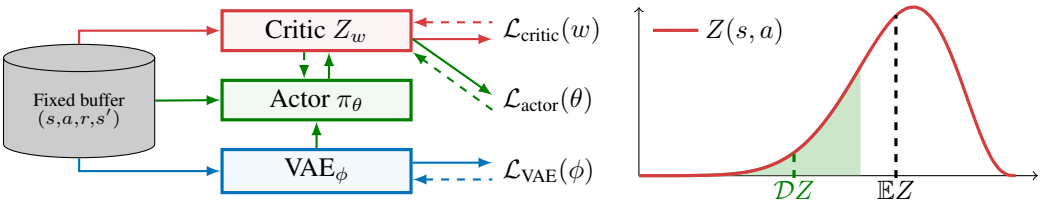

Figure 1: Visualization of the algorithm components. Solid lines indicate the forward flow of data whereas dashed lines indicate the backward flow of gradients. Data is stored in the fixed buffer. The VAE, in blue, learns a generative model of the behavior policy. The actor, in green, perturbs the VAE and outputs a policy. The critic, learns the Z-value distribution of the policy. The actor optimizes a risk-averse distortion of the Z-value distribution, which we denote by $\mathcal{D}Z$. On the right, we show a typical probability density function of Z learned by the critic in red. In dashed black we indicate the expected value of Z, which a risk-neutral actor intends to maximize. Instead, a risk-averse actor intends to maximize a distortion $\mathcal{D}Z$, shown in dashed green. In this particular visualization, we show the ubiquitous Conditional Value at Risk (CVaR).

their algorithm is limited to tabular MDPs and is also on-policy. Instead, Tang et al. (2020) propose an off-policy algorithm that approximates the return distribution with a Gaussian distribution and learns its moments using the Bellman equation for the mean and the variance of the distribution. Instead, we learn the full return distribution without making the Gaussianity assumption (Bellemare et al., 2017). Perhaps most closely related is the work of Singh et al. (2020), who consider also a distributional critic but their algorithm is limited to the CVaR and they do not address the offline RL setting. Furthermore, they use a sample-based distributional critic, which makes the computation of the CVaR inefficient. Instead, we modify Implicit Quantile Networks (Dabney et al., 2018) in order to compute different risk criteria efficiently. Although (Dabney et al., 2018) already investigated risk-related criteria, their scope is limited to discrete action spaces (e.g., the Atari domain) in an off-policy setting whereas we consider continuous actions in an offline setting.

Offline RL The biggest challenge in offline RL is the Bootstrapping Error: a Q-function is evaluated at state-action pairs where there is little or no data and these get propagated through the Bellman equation (Kumar et al., 2019). In turn, a policy optimized with offline data induces a state-action distribution that is shifted from the original data (Ross et al., 2011). To address this, Fujimoto et al. (2019) propose to express the actor as the sum between an imitation learning component and a perturbation model to control the deviation of the behavior policy. Other approaches to control the difference between the data-collection policy and the optimized policy include regularizing the policies with the behavior policy using the MMD distance (Kumar et al., 2019) or f-divergences Wu et al. (2020); Jaques et al. (2019), or using the behavior policy as a prior (Siegel et al., 2020). An alternative strategy in offline RL is to be *pessimistic* with respect to the epistemic uncertainty that arises due to data scarcity. Yu et al. (2020) take a model-based approach and penalize the per-step rewards with the epistemic uncertainty of their dynamical model. Using a model-free approach Kumar et al. (2020); Buckman et al. (2020) propose to learn a lower bound of the Q-function using an estimate of the uncertainty as penalty in the target of the equation. Our work uses ideas from both strategies to address the offline risk-averse problem. First, we use an imitiation learner to control the bootstrapping error. However, by considering a risk-averse criterion, we are also optimizing over a *pessimistic* distribution compatible with the empirical distribution in the data set. The connections between risk-aversion and distributional robustness are well studied in supervised learning (Shapiro et al., 2014; Namkoong & Duchi, 2017; Curi et al., 2019) and in reinforcement learning (Chow et al., 2015; Pan et al., 2019).

2 Problem Statement

We consider a Markov Decision Process (MDP) with possibly continuous $s \in \mathcal{S}$ and possibly continuous actions $a \in \mathcal{A}$, transition kernel $P(\cdot|s,a)$, reward kernel $R(\cdot|s,a)$ and discount factor γ. We denote by π a stationary policy, i.e., a mapping from states to distribution over actions. We have access to a fixed batch data set collected with an unknown policy π_β. We call d^β the joint *state,*

action, reward, next-state distribution and as ρ^β as the marginal *state* distribution. Similarly, for any policy π, we call d^π the joint *state, action, reward, next-state* distribution induced by π on the MDP.

In risk-neutral RL, the goal is to find a policy that maximizes the *expected* discounted sum of returns $\mathbb{E}_{d^\pi}\left[\sum_{t=1}^{\infty}\gamma^{t-1}R(\cdot|S_t,A_t)\right]$, where the expectation is taken with respect to the stochasticity introduced by the reward kernel, the transition kernel, and the policy. We define as $Z^\pi(s,a) =_D \sum_{t=1}^{\infty}\gamma^{t-1}R(\cdot|S_t,A_t)$ as the return distribution conditioned on $(S_1=s, A_1=a)$ and following π thereafter. Here $=_D$ denotes equality in distribution. The risk-neutral RL objective is the expectation of the distribution of Z^π.

In risk-averse settings, we replace the expectation with a distortion operator $\mathcal{D}$ that is a mapping from the distribution over the returns to the reals. Thus, the goal is to find a policy π that maximizes

$$\max_\pi \mathcal{D}\left[\sum_{t=1}^{\infty}\gamma^{t-1}R(\cdot|S_t,A_t)\right]. \tag{1}$$

With this framework, we address many different risk-averse distortions. For example, this includes probability weighting functions (Gonzalez & Wu, 1999; Tversky & Kahneman, 1992); the CVaR (Rockafellar & Uryasev, 2002); the mean-variance criteria (Namkoong & Duchi, 2017); or the Wang criteria (Wang, 1996; Rabin, 2013).

3 Offline Risk-Averse Actor-Critic (O-RAAC)

We now present our main algorithm – O-RAAC– for offline-risk averse actor-critic. One of the main technical challenges in going beyond expected rewards is to find an analogue of the Q-function for the particular distortion operator we want to optimize. Unfortunately, the Bellman target of most risk-analogues does not have a closed-form expression. Therefore, we instead learn the full distribution of returns as proposed by Bellemare et al. (2017). In Section 3.1, we describe the training procedure for the distributional critic. Next, in Section 3.2 we define the actor loss as the risk-distortion operator on the learned return distribution and optimize it using a gradient-based approach. Up to this point, the actor-critic template is enough to optimize a risk-averse criteria. However, as we focus on the offline setting, we need to control the bootstrapping error. To this end, we use a variational auto-encoder VAE to learn a generative model of the behavior policy in Section 3.3. Finally, in Section 3.4 we bring all the pieces together and instantiate our algorithm for different risk distortions $\mathcal{D}$.

3.1 Distributional Critic Learning

To learn the distributional critic, we exploit the distributional Bellman equation of returns $Z^\pi(s,a) =_D R(s,a) + \gamma Z^\pi(S',A')$ for policy evaluation. The random variables S', A' are distributed according to $s' \sim P(\cdot|s,a)$ and $A' \sim \pi(\cdot|s')$. In particular, we represent the return distribution implicitly through its quantile function as proposed by Dabney et al. (2018). We use this representation because many risk distortion operators can be efficiently computed using the quantile function of the underlying random variable. We parameterize the quantile function through a neural network with learnable parameters w. We express such implicit quantile function as $Z^\pi_w(s,a;\tau)$, where $\tau \in [0,1]$ is the quantile level. Whereas the neural network architecture proposed by Dabney et al. (2018) is for discrete actions only, we extend it to continuous actions by considering all s, a, and τ as the inputs and only the quantile value as the output.

To learn the parameters w, we use the distributional variant of fitted value-iteration (Bellemare et al., 2017; Munos & Szepesvári, 2008) using a quantile Huber-loss (Huber, 1964) as a surrogate of the Wasserstein-distance used by Dabney et al. (2018). To this end, we use a target network with parameters w' and compute the td-error at a sample (s,a,r,s') as

$$\delta_{\tau,\tau'} = r + \gamma Z^\pi_{w'}(s',a';\tau') - Z^\pi_w(s,a;\tau), \tag{2}$$

for two independent samples $\tau, \tau' \sim U([0,1])$ and $a' \sim \pi(\cdot|s')$. The τ-quantile Huber-loss is

$$\mathcal{L}_\kappa(\delta;\tau) = \underbrace{\left|\tau - \mathbb{1}_{\{\delta<0\}}\right|}_{\text{Quantile loss}} \cdot \underbrace{\begin{cases} \frac{1}{2\kappa}\delta^2 & \text{if } |\delta| \le \kappa, \\ |\delta| - \frac{1}{2}\kappa & \text{otherwise.} \end{cases}}_{\text{Huber loss}} \tag{3}$$

We prefer the Huber loss over the $\mathcal{L}_2$ or $\mathcal{L}_1$ loss as it is better behaved due to smooth gradient-clipping (Mnih et al., 2015). Finally, we approximate the quantile loss for all levels τ by sampling N independent quantiles τ and N' independent target quantiles τ'. The critic loss is

$$\mathcal{L}_{\text{critic}}(w) = \mathbb{E}_{\substack{s,a,r,s' \sim \beta(\cdot) \\ a' \sim \pi(\cdot|s')}} \Big[\frac{1}{N \cdot N'} \sum_{i=1}^{N} \sum_{j=1}^{N'} \mathcal{L}_\kappa(\delta_{\tau_i, \tau'_j}; \tau_i) \Big]. \tag{4}$$

3.2 Learning a Risk-Averse Actor

In risk-averse applications, we generally prefer deterministic policies over stochastic ones because introducing extra randomness is against a risk-averse behavior (Pratt, 1978), the optimal policy is deterministic (Puterman, 2014), and there is no benefit of exploration often associated with stochastic policies in the offline setting. Hence, we consider parameterized deterministic policies $\pi_\theta(s) : \mathcal{X} \to \mathcal{A}$. Given a learned distributional critic, we define the actor loss as

$$\mathcal{L}_{\text{actor}}(\theta) = \mathbb{E}_{s \sim \rho^\beta(\cdot)} \left[\mathcal{D} \left(Z_w^{\pi_\theta}(s, \pi_\theta(s)) \right) \right], \tag{5}$$

where $\mathcal{D}$ is the operator that models risk aversion. As a particular example of $\mathcal{D}$ we consider once again the CVaR. We leverage the fact that we parameterize $Z_w^{\pi_\theta}(s, a; \tau)$ using an implicit quantile network and directly use Acerbi's integral formula for the CVaR (Acerbi & Tasche, 2002)

$$\text{CVaR}_\alpha(Z_w^{\pi_\theta}(s, a; \tau)) = \frac{1}{\alpha} \int_0^\alpha Z_w^{\pi_\theta}(s, a; \tau) d\tau, \tag{6}$$

which can be approximated by sampling τ from the uniform distribution in $[0, \alpha]$. In contrast, the sampled-based critic used by Singh et al. (2020) does not lend itself to the use of this formula and instead uses the common Rockafellar truncated optimization (Rockafellar & Uryasev, 2002) to calculate the CVaR, which is sample inefficient. Furthermore, recent work by Curi et al. (2019) suggest that this formula introduces high-variance to the gradient in the context of deep-learning.

To maximize the actor loss (5), we use pathwise derivatives of the objective (Lillicrap et al., 2015; Mohamed et al., 2020), computed by backpropagating the actor through the learned critic at states sampled from the offline data set.

3.3 Off-policy to Offline: Controlling the Bootstraping Error.

Up to this point, the actor-critic procedure we describe in Sections 3.1 and 3.2 is theoretically sufficient to learn a risk-averse RL agent. However, in the offline setting the bootstrapping error appears: when evaluating the td-error (2), the Z-value target will be evaluated at actions where there is no data (Kumar et al., 2019), and propagated through the Bellman equation. To address this phenomenon, we decompose the actor in two components: an imitation learning component π^{IL} and a perturbation model ξ_θ such that the policy is expressed as:

$$\pi_\theta(s) = b + \lambda \xi_\theta(\cdot|s, b), \qquad \text{s.t., } b \sim \pi^{\text{IL}}(\cdot|s). \tag{7}$$

That is, b is an action that the imitation learning component outputs, ξ_θ is a perturbation model that is optimized maximizing the actor loss (5), and λ is a hyper-parameter that modulates the perturbation level. Kumar et al. (2019); Wu et al. (2020); Siegel et al. (2020) address the bootstrapping error by regularizing the actor loss (5) with the behavior policy, but this requires access to the log-probabilities of the actions sampled by π_β, which we do not assume to have. Nevertheless, the high-level idea of these methods is similar and they are interchangeable.

To learn a generative model of the π^{IL} from state-action pairs from the behaviour distribution d^β we use a conditional variational autoencoder (VAE) (Kingma & Welling, 2014). This is also done in Fujimoto et al. (2019) and the main advantage compared to behavioral cloning (Bain & Sammut, 1995) is that it does not suffer from mode-collapse which hinders the actor optimization, and compared to inverse imitation learning (Ziebart et al., 2008; Abbeel & Ng, 2004) it does not assume that the policy is optimal. We chose the VAE over Generative Adversarial Networks (Ho & Ermon, 2016) due to easiness of training (Arjovsky & Bottou, 2017).

The conditional variational autoencoder is a probabilistic model that samples an action $b \sim \text{VAE}_\phi(s, a)$ according to the generative model

$$\mu, \Sigma = E_{\phi_1}(s, a); \qquad z \sim \mathcal{N}(\mu, \Sigma); \qquad b = D_{\phi_2}(s, z), \tag{8}$$

where E_{ϕ_1} is the encoder neural network, D_{ϕ_2} is the decoder neural network, and we sample the code z using the re-parameterization trick. To learn the $\phi = \{\phi_1, \phi_2\}$ parameters we place a prior $\mathcal{N}(0, I)$ on the code z and minimize the variational lower-bound

$$\mathcal{L}_{\text{VAE}}(\phi) = \mathbb{E}_{s,a\sim\beta(\cdot)}\left[\underbrace{(a - D_{\phi_2}(s,z))^2}_{\text{reconstruction loss}} + \frac{1}{2}\underbrace{\text{KL}(\mathcal{N}(\mu,\Sigma),\mathcal{N}(0,I))}_{\text{regularization}}\right]. \tag{9}$$

Upon a new state at test time, the generative model of the $\text{VAE}_\phi(s)$ is $z \sim \mathcal{N}(0, I)$, $b = D_{\phi_2}(s, z)$.

3.4 Final Algorithm

We now combine the critic in Section 3.1, the actor in Section 3.2, and the VAE in Section 3.3 and show the pseudo-code of O-RAAC in Algorithm 1. We replace the expectations in the critic loss (4), actor loss (5), and VAE loss (9) with empirical averages of samples from the data set. As an ablation, we also propose the RAAC algorithm, in which the actor is parameterized with a neural network and there is no VAE component.

The critic's goal is to learn the reward distribution, the VAE goal is to learn a baseline action for the actor, and the goal of the perturbation model is to be risk-averse. Although Santara et al. (2018) and Lacotte et al. (2019) propose risk-averse imitation learning algorithms, these interact with the environment in an on-policy way. Furthermore, the goal of the imitation learning component in O-RAAC is not to be risk-averse, but to provide a baseline to the risk-averse perturbation.

O-RAAC requires a distortion metric $\mathcal{D}$ as an input. For different $\mathcal{D}$, it generalizes existing distributional RL algorithms and extends them to the offline setting. For example, hen $\mathcal{D}$ is the expectation operator, then the agent is risk neutral and O-RAAC is an offline version of D4PG (Barth-Maron et al., 2018). Likewise, when $\mathcal{D}$ is the Rockafellar-truncation operator (Rockafellar & Uryasev, 2002) we recover the algorithm by Singh et al. (2020) for optimizing the CVaR.

Algorithm 1: Offline Risk-Averse Actor Critic (O-RAAC).

input Data set, Critic Z_w and critic-target $Z_{w'}$, $\text{VAE}_\phi = \{E_{\phi_1}, D_{\phi_2}\}$, Perturbation model ξ_θ and perturbation target $\xi_{\theta'}$, modulation parameter λ, Distortion operator $\mathcal{D}$, critic-loss parameters N, N', κ, mini-batch size B, learning rate η, soft update parameter μ.
for $t = 1, \ldots$ **do**
  Sample B transitions (s, a, r, s') from data set.
  Sample N quantiles $\tau \sim [0, 1]$ and N' target quantiles $\tau' \sim [0, 1]$ and compute $\delta_{\tau,\tau'}$ in (2).
  Compute policy $\pi_\theta = b + \lambda\xi_\theta(s, b)$, s.t. $b \sim \text{VAE}_\phi(s, a)$ as in (8).
  Compute critic loss $\mathcal{L}_{\text{critic}}(w)$ in (4); actor loss $\mathcal{L}_{\text{actor}}(\theta)$ in (5); VAE loss $\mathcal{L}_{\text{VAE}}(\phi)$ in (9).
  Gradient step $w \leftarrow w - \eta\nabla\mathcal{L}_{\text{critic}}(w)$; $\theta \leftarrow \theta + \eta\nabla\mathcal{L}_{\text{actor}}(\theta)$; $\phi \leftarrow \phi - \eta\nabla\mathcal{L}_{\text{VAE}}(w)$.
  Perform soft-update on $w' \leftarrow \mu w + (1 - \mu)w'$; $\theta' \leftarrow \mu\theta + (1 - \mu)\theta'$.
end for

4 Experimental Evaluation

In this section, we test the performance of O-RAAC using $\mathcal{D} = \text{CVaR}_{\alpha=0.1}$ as risk distortion. For further details and extended results please see Appendix A. In particular, we ask:

1. How does RAAC perform as a *risk-averse* agent in the **off-policy** setting? (Section 4.1)
2. How does O-RAAC perform as a *risk-averse* agent in the **offline** setting? (Section 4.2)
3. How does O-RAAC perform as a *risk-neutral* agent in the **offline** setting? (Section 4.3)

Benchmarks We compare O-RAAC with a risk-neutral algorithm (D4PG algorithm by Barth-Maron et al. (2018) with IQN and 1-step returns) and with a competing risk-averse algorithm (WCPG algorithm by Tang et al. (2020)). In the offline setting, we augment D4PG and WCPG with the same VAE imitation learning procedure and we denote them O-D4PG and O-WCPG. As an ablation for the offline setting, we include RAAC (i.e., without the VAE).

Table 1: Results of RAAC, WCPG, and D4PG in the car example. RAAC learns a policy that saturates the velocity before the risky region. WCPG and D4PG learn to accelerate as fast as possible, reaching the goal first with highest average returns but suffer from events with large penalty.

Algorithm	**RAAC**	WCPG	D4PG
$\text{CVaR}_{0.1}$ of Returns	$\mathbf{48.0 \pm 8.3}$	15.8 ± 3.3	15.6 ± 4.4
Risky Steps	$\mathbf{0 \pm 0}$	13 ± 0	13 ± 0
Average of Returns	48.0 ± 8.3	$\mathbf{79.8 \pm 1.3}$	$\mathbf{79.8 \pm 2.0}$
Steps to goal	33 ± 1	$\mathbf{24 \pm 0}$	$\mathbf{24 \pm 0}$

4.1 Off-policy Setting: Risk-Averse performance

In this experiment, we intend to demonstrate the effectiveness of our algorithm as a risk-averse learner without introducing the extra layer of complexity of the offline setting.

4.1.1 Experimental Setup

As a toy example, we chose a 1-D car with state $s = (x, v)$, for position and velocity. The agent controls the car with an acceleration $a \in [-1, 1]$. The car dynamics with a time step $\Delta t = 0.1$ is

$$x_{t+1} = x_t + v_t \Delta t + 0.5 a_t (\Delta t)^2, \qquad v_{t+1} = v_t + a_t \Delta t.$$

The control objective is to move the car to $x_{\text{g}} = 2.5$ as fast as possible, starting from rest. However, when the velocity $v > 1$ the car speeds and it might crash or get a fine with some probability. To model this, we use a random reward function given by

$$R_t(s, a) = -10 + 370 \mathbb{I}_{x_t = x_{\text{g}}} - 25 \mathbb{I}_{v_t > 1} \cdot \mathcal{B}_{0.2},$$

where $\mathbb{I}$ is an indicator function and $\mathcal{B}_{0.2}$ is a Bernoulli Random Variable with probability $p = 0.2$. The episode terminates after 400 steps or when the agent reaches the goal.

4.1.2 Result Discussion

In Table 1, we show the results of the experiment. As expected, D4PG ignores the low probability penalties and learns to accelerate the car with maximum power. Thus it has the largest expected return but the lowest CVaR. WCPG also fails to maximize the CVaR as it assumes that the return distribution is Gaussian. In turn, it under-estimates the variance of the return distribution of the maximum acceleration policy. Consequently, it over-estimates the CVaR of the returns and prefers the latter policy over the maximum-CVaR policy. In contrast, RAAC learns the full value distribution Z and computes the CVaR reliably. Thus, it learns to saturate the velocity below the $v = 1$ threshold and finds the highest CVaR policy. See Appendix A.1 for more experiments in this setting.

4.2 Offline Setting: Risk-Averse performance

In this experiment, we intend to demonstrate the effectiveness of our algorithm as a risk-averse learner in the offline setting and in high-dimensional environments.

4.2.1 Experimental Setup

We test the algorithm on a variety of continuous control benchmark tasks on the data provided in the D4RL dataset (Fu et al., 2020). We use three MuJoCo tasks: HalfCheetah, Walker and Hopper (Todorov et al., 2012). In particular, we chose the medium (M) and expert (E) variants of these datasets. Since the tasks are deterministic, we incorporate stochasticity into the original rewards to have a meaningful assessment of risk and to showcase a practical example of when the risk-averse optimization makes sense. We use the following reward functions:

Half-Cheetah: $R_t(s, a) = \bar{r}_t(s, a) - 70 \mathbb{I}_{v > \bar{v}} \cdot \mathcal{B}_{0.05}$, where $\bar{r}_t(s, a)$ is the original environment reward, v the forward velocity, and $\bar{v}$ is a threshold velocity ($\bar{v} = 4$ for the (M) variant and $\bar{v} = 10$ for the (E) variant). As with the car example, this high-velocity penalization models a penalty to the

Table 2: Performance O-RAAC, O-WCPG, O-D4PG, and RAAC on offline MuJoCo data sets. The top sub-table shows the medium (M) environments whereas the bottom sub-table the expert (E) environments. In all environments, O-RAAC has a higher CVaR than benchmarks. In environments that terminate, O-RAAC has a longer duration too. Finally, O-RAAC also has better or equivalent risk-neutral performance than benchmarks. As an ablation, RAAC (without the VAE component) performs poorly in the offline setting.

Env	Criterion	**O-RAAC**	O-WCPG	O-D4PG	RAAC
half-M	$\text{CVaR}_{0.1}$ Returns	**214.1 ± 35.9**	75.9 ± 14.3	66.0 ± 34.4	-54.9 ± 0.9
	Average Returns	**330.6 ± 30.4**	**316.1 ± 23.3**	**340.6 ± 20.0**	-51.6 ± 0.3
walker-M	$\text{CVaR}_{0.1}$ Returns	**750.5 ± 153.5**	-15.0 ± 40.7	30.6 ± 28.8	54.7 ± 2.0
	Episode Duration	**397.4 ± 18.5**	184.9 ± 12.1	248.8 ± 8.9	199.8 ± 6.5
	Average Returns	**1282.4 ± 19.3**	283.3 ± 36.7	307.7 ± 19.9	91.7 ± 8.9
hopper-M	$\text{CVaR}_{0.1}$ Returns	**1416.1 ± 28.1**	-87.1 ± 24.5	1007.5 ± 28.2	71.3 ± 23.2
	Episode Duration	**499.3 ± 1.2**	99.3 ± 0.1	358.3 ± 3.1	146.1 ± 4.0
	Average Returns	**1482.2 ± 4.3**	68.5 ± 7.6	1098.0 ± 10.6	112.8 ± 4.6
half-E	$\text{CVaR}_{0.1}$ Returns	**595.3 ± 191.0**	248.4 ± 232.3	**555.8 ± 263.2**	2.74 ± 12.8
	Average Returns	**1180.4 ± 77.8**	**905.0 ± 107.2**	**1099.8 ± 152.7**	29.8 ± 3.0
walker-E	$\text{CVaR}_{0.1}$ Returns	**1171.8 ± 71.3**	361.8 ± 32.5	772.5 ± 54.7	53.9 ± 2.0
	Episode Duration	**431.8 ± 11.0**	301.4 ± 31.1	**405.4 ± 12.2**	195.5 ± 6.1
	Average Returns	**2005.8 ± 56.4**	1372.4 ± 160.1	1869.9 ± 62.8	82.9 ± 6.4
hopper-E	$\text{CVaR}_{0.1}$ Returns	**979.5 ± 27.7**	720.0 ± 33.8	606.4 ± 31.4	473.84 ± 0.1
	Episode Duration	**463.5 ± 6.5**	301.2 ± 0.6	267.7 ± 2.5	500 ± 0
	Average Returns	**1384.5 ± 33.11**	897.6 ± 12.1	783.1 ± 18.2	474.75 ± 0.1

rare but catastrophic event of the robot breaking – we want to be risk-averse to it. We evaluate the Half-Cheetah for 200 time steps.

Walker2D/Hopper: $R_t(s,a) = \bar{r}_t(s,a) - p\mathbb{I}_{|\theta|>\bar{\theta}} \cdot \mathcal{B}_{0.1}$, where $\bar{r}_t(s,a)$ is the original environment reward, θ is the pitch angle, $\bar{\theta}$ is a threshold angle ($\bar{\theta} = 0.5$ for the Walker2d-M/E and $\bar{\theta} = 0.1$ for the Hopper-M/E) and $p = 30$ for the Walker2d-M/E and $p = 50$ for the Hopper-M/E. When $|\theta| > 2\bar{\theta}$ the robot falls, the episode terminates, and we stop collecting such rewards. To avoid such situation, we shape the rewards with the stochastic event at $\theta > \bar{\theta}$. The maximum duration of the Walker2D and the Hopper is 500 time steps.

4.2.2 Result Discussion

In Table 2, we show the results of the experiments. In all environments, O-RAAC has a higher CVaR than the benchmarks. In environments that terminate, it also has longer duration than competitors. As an ablation, RAAC without the VAE performs poorly in all categories. This indicates that the offline version enhancement of RAAC is crucial for these data sets.

We visualize the risk-averse performance by looking at the support of the risk-event in Figure 2. Most of the support of the data set induced by the behavior policy lies in the risky region. O-RAAC learns to shift the distribution towards the the risk-free region (green shaded area). On the other hand, O-D4PG struggles to shift the distribution as it ignores the rare penalties in the risky region. Only in the Half Cheetah-E experiment O-D4PG manages to shift the distribution towards the safe region. This is because most of the behavior policy is on the risky region and the *average* performance of the behavior policy under the new stochastic rewards is low.

4.3 Offline setting: Risk-Neutral performance

It is well-known that coherent risk-related criteria have a dual distributional robust criterion formulation (Shapiro et al., 2014; Iyengar, 2005; Osogami, 2012). In particular, the following holds:

$$\max_{\pi} \mathcal{D}\left[Z^{\pi}(x,a)\right] = \max_{\pi} \min_{d \in \overline{\mathcal{D}}_{\pi}} \mathbb{E}_d\left[Z^{\pi}(x,a)\right], \tag{10}$$

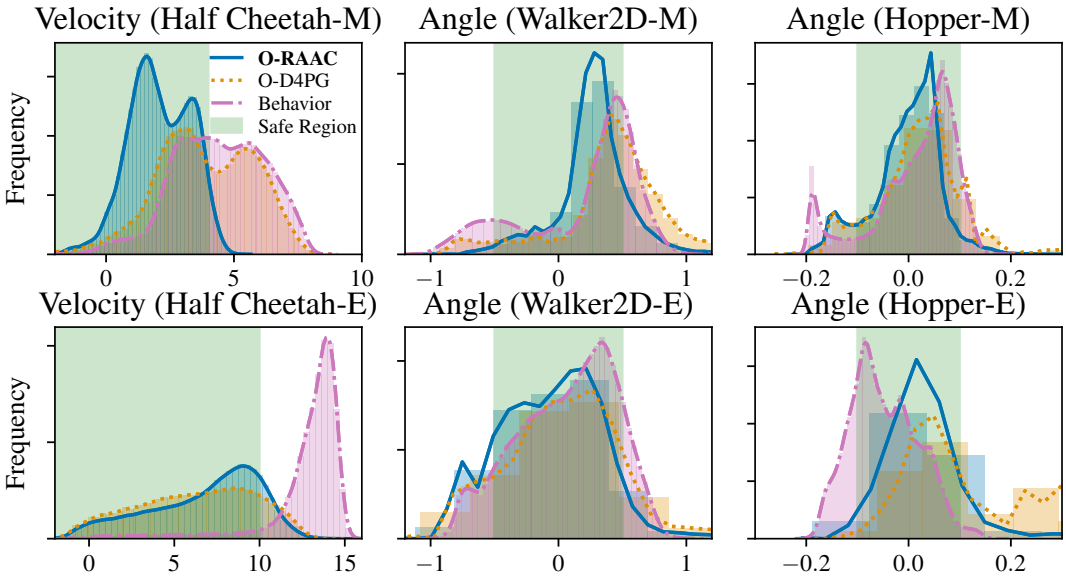

Figure 2: Empirical distribution of the risk-event of O-RAAC, O-D4PG, and the behavior policies. O-RAAC shifts the support towards the risk-free region (green area). On the other hand, O-D4PG ignores the risk-related rewards and imitates the behavior distribution.

where $\overline{\mathcal{D}}_\pi$ is a dual set of distribution that is induced by the distortion measure $\mathcal{D}$ and the distribution d_π. When the distortion set is the expectation, the dual set collapses to a singleton $\overline{\mathcal{D}}_\pi = \{d^\pi\}$. For the CVaR, Chow & Ghavamzadeh (2014) expresses the dual set for MDPs in Proposition 1. Given this dual result, it is straightforward to show that $\mathbb{E}_{d^\pi}[Z^\pi(x,a)] \geq \mathcal{D}[Z^\pi(x,a)]$ by definition of the minimum. In this sense, optimizing $\mathcal{D}[Z^\pi(x,a)]$ is equivalent to optimizing a *pessimistic* estimate of the risk-neutral performance, in a similarly way to Buckman et al. (2020) and Kumar et al. (2020).

Thus, despite the goal being to maximize a risk-neutral objective, we evaluate whether it is beneficial to optimize a coherent risk-sensitive criterion in the offline setting. We test this hypothesis the same setup as in Section 4.2, but we evaluate the risk-neutral performance.

4.3.1 Result Discussion

We show the risk-neutral results in the Average Returns rows in Table 2. In all data sets, O-RAAC performs better or similar to the benchmarks. Despite longer episodes causes larger returns, this is not the only reason. In particular, for the Walker-E environment, both O-D4PG and O-RAAC have similar episode duration. Yet, the average returns of O-RAAC are higher than those of O-D4PG.

In Figure 2 we see that in most cases there *is* a distribution shift between the behaviour distribution and both O-RAAC and O-D4PG. As the shift increases, we see the benefits of learning using distributionally robust objectives. As a particular example we take the Hopper-E distribution. The behavior policy is safe but much of the data is on the risky region. O-RAAC learns to center the distribution in the safe region and yet hops forward efficiently. On the other hand, O-D4PG also learns to shift away of the risky region. However, it is not risk averse and it overshoots towards the other end of the risky region, where there is not sufficient data to have good critic estimates.

5 Conclusion

In high-stakes applications, decision-making is usually risk-averse and no interactions with the environment are allowed. For this pracical setting, we introduce O-RAAC, the first fully offline risk-averse algorithm. Empirically, O-RAAC optimizes risk-averse criteria and, due to the distributional-robust properties of risk-sensitive criteria, it also optimizes risk-neutral criteria under natural distributions shift that occur in the offline setting.

REFERENCES

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Carlo Acerbi and Dirk Tasche. On the coherence of expected shortfall. *Journal of Banking and Finance*, 2002. ISSN 03784266. doi: 10.1016/S0378-4266(02)00283-2.

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent risk measures. *Mathematical Finance*, 9(3):203–228, 1999.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pp. 103–129, 1995.

Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, T B Dhruva, Alistair Muldal, Nicolas Manfred Otto Heess, and Timothy P Lillicrap. Distributed Distributional Deterministic Policy Gradients. *ArXiv*, abs/1804.0, 2018.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pp. 449–458, 2017.

Jacob Buckman, Carles Gelada, and Marc G. Bellemare. The importance of pessimism in fixed-dataset policy optimization, 2020.

Yinlam Chow and Mohammad Ghavamzadeh. Algorithms for CVaR optimization in MDPs. *Advances in Neural Information Processing Systems*, 4(January):3509–3517, 2014. ISSN 10495258.

Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: A CVaR optimization approach. *Advances in Neural Information Processing Systems*, 2015-Janua:1522–1530, 2015. ISSN 10495258.

Sebastian Curi, Kfir Levy, Stefanie Jegelka, Andreas Krause, et al. Adaptive sampling for stochastic risk-averse learning. *arXiv preprint arXiv:1910.12511*, 2019.

Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. Implicit quantile networks for distributional reinforcement learning. In *35th International Conference on Machine Learning, ICML 2018*, 2018. ISBN 9781510867963.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for Deep Data-Driven Reinforcement Learning, 2020.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *36th International Conference on Machine Learning, ICML 2019*, 2019. ISBN 9781510886988.

Richard Gonzalez and George Wu. On the shape of the probability weighting function. *Cognitive psychology*, 38(1):129–166, 1999.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pp. 4565–4573, 2016.

Ronald A Howard and James E Matheson. Risk-Sensitive Markov Decision Processes. *Management Science*, 18(7):356–369, 1972. ISSN 00251909, 15265501. URL `http://www.jstor.org/stable/2629352`.

Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 1964. ISSN 0003-4851. doi: 10.1214/aoms/1177703732.

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

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *CoRR*, abs/1906.00949, 2019. URL `http://arxiv.org/abs/1906.00949`.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning, 2020.

Jonathan Lacotte, Mohammad Ghavamzadeh, Yinlam Chow, and Marco Pavone. Risk-sensitive generative adversarial imitation learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2154–2163. PMLR, 2019.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pp. 45–73. Springer, 2012.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Oliver Mihatsch and Ralph Neuneier. Risk-sensitive reinforcement learning. *Machine learning*, 49 (2-3):267–290, 2002.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62, 2020.

Toshiyuki Morimura, Tetsuro and Sugiyama, Masashi and Kashima, Hisashi and Hachiya, Hirotaka and Tanaka. Nonparametric return distribution approximation for reinforcement learning. In *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, 2010. ISBN 9781605589077.

Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(May):815–857, 2008.

Hongseok Namkoong and John C Duchi. Variance-based regularization with convex objectives. In *Advances in neural information processing systems*, pp. 2971–2980, 2017.

Takayuki Osogami. Robustness and risk-sensitivity in markov decision processes. In *Advances in Neural Information Processing Systems*, pp. 233–241, 2012.

Xinlei Pan, Daniel Seita, Yang Gao, and John Canny. Risk averse robust adversarial reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8522–8528. IEEE, 2019.

Marek Petrik and Dharmashankar Subramanian. An approximate solution method for large risk-averse markov decision processes. In *Uncertainty in Artificial Intelligence - Proceedings of the 28th Conference, UAI 2012*, 2012. ISBN 9780974903989.

LA Prashanth, Cheng Jie, Michael Fu, Steve Marcus, and Csaba Szepesvári. Cumulative prospect theory meets reinforcement learning: Prediction and control. In *International Conference on Machine Learning*, pp. 1406–1415, 2016.

John W Pratt. Risk aversion in the small and in the large. In *Uncertainty in Economics*, pp. 59–79. Elsevier, 1978.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Matihew Rabin. Risk aversion and expected-utility theory: A calibration theorem. In *Handbook of the fundamentals of financial decision making: Part I*, pp. 241–252. World Scientific, 2013.

R Tyrrell Rockafellar and Stanislav Uryasev. Conditional value-at-risk for general loss distributions. *Journal of banking & finance*, 26(7):1443–1471, 2002.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

Anirban Santara, Abhishek Naik, Balaraman Ravindran, Dipankar Das, Dheevatsa Mudigere, Sasikanth Avancha, and Bharat Kaul. Rail: Risk-averse imitation learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2062–2063, 2018.

Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2014.

Noah Y. Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning, 2020.

Rahul Singh, Qinsheng Zhang, and Yongxin Chen. Improving robustness via risk averse distributional reinforcement learning. *arXiv preprint arXiv:2005.00585*, 2020.

Aviv Tamar, Dotan Di Castro, and Shie Mannor. Policy gradients with variance related risk criteria. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 2012. ISBN 9781450312851.

Aviv Tamar, Yonatan Glassner, and Shie Mannor. Optimizing the CVaR via sampling. *Proceedings of the National Conference on Artificial Intelligence*, 4:2993–2999, 2015.

Yichuan Charlie Tang, Jian Zhang, and Ruslan Salakhutdinov. Worst Cases Policy Gradients. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1078–1093. PMLR, 2020. URL `http://proceedings.mlr.press/v100/tang20a.html`.

Empel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE International Conference on Intelligent Robots and Systems*, 2012. ISBN 9781467317375. doi: 10.1109/IROS.2012.6386109.

Amos Tversky and Daniel Kahneman. Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and uncertainty*, 5(4):297–323, 1992.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Shaun Wang. Premium calculation by transforming the layer premium density. *ASTIN Bulletin: The Journal of the IAA*, 26(1):71–92, 1996.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning, 2020. URL https://openreview.net/forum?id=BJg9hTNKPH.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A Extended Experimental Results

A.1 Car

We ran the Car environment for RAAC, D4PG and WCPG, using 5 independent random seeds. We evaluate final policies for 1000 interactions and report the average results with corresponding standard deviation in Table 1. In Figure 3, we show the trajectories following aforementioned policies.

Reward function design We use the reward function given by

$$R_t(s,a) = -10 + 370\mathbb{I}_{x_t=x_g} - 25\mathbb{I}_{v_t>1} \cdot \mathcal{B}_{0.2},$$

where $\mathbb{I}$ is an indicator function and $\mathcal{B}_{0.2}$ is a Bernoulli Random Variable with probability $p = 0.2$. That is, $r_f = 370$ is a sparse reward that the agent gets at the goal and $r_d = -10$ is a negative reward that penalizes delays on reaching the goal. Finally, the agent receives a negative reward of $r_v = -25$ with probability 0.2 when it exceeds the $v > 1$ threshold. As the returns is a sum of bernoulli R.V. we know that it will be a Binomial distribution. For this particular case, we expect that if the number of steps is large enough, the Gaussianity assumption that WCPG does is good as Binomial distributions are asymptotically Gaussian (Vershynin, 2018). However, the episode terminates after at most thirteen risky steps and the approximation is not good.

We show in Figure 3 the trajectories for RAAC, D4PG, and WCPG.

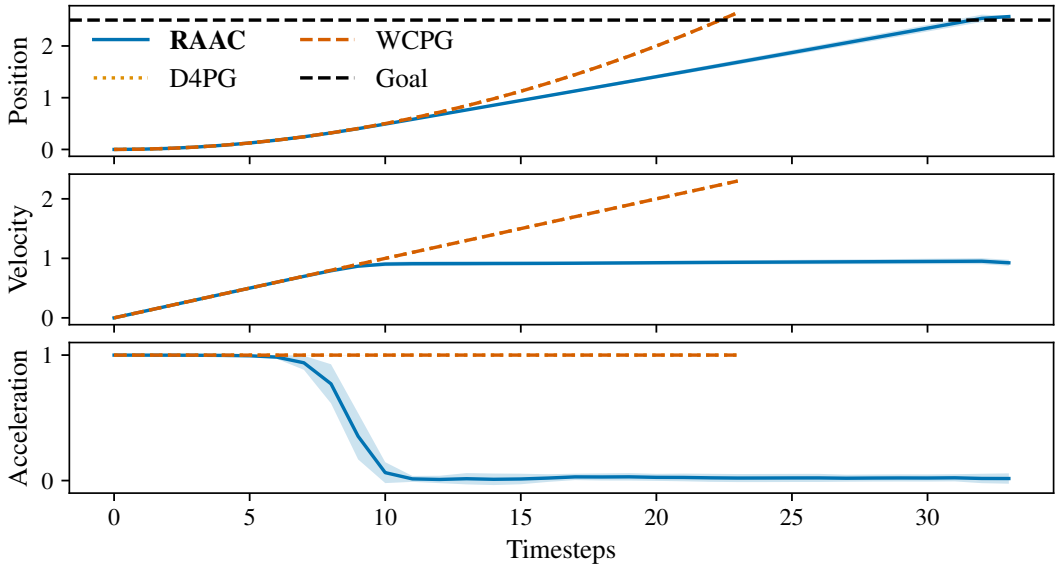

Figure 3: Evolution of car states and input control using learnt policies after training for RAAC and WCPG and D4PG (are the same trajectory).We use policies from 5 independent seeds for each algorithm. RAAC learns to saturate the velocity below the speed limit.

Table 3: Reference of average and CVaR of returns with stochastic reward function

Environment	Average of Returns	$\text{CVaR}_{0.1}$ of Returns
HalfCheetah-M	345.7 ± 0.9	9.6 ± 4.5
Walker2d-M	917.5 ± 6.2	337.0 ± 3.8
Hopper-M	1355.0 ± 1.6	1096.0 ± 1.8
HalfCheetah-E	1781.6 ± 1.6	1157.6 ± 11.1
Walker2d-E	2202.1 ± 3.2	1682.0 ± 8.8
Hopper-E	1774.6 ± 0.9	1618.0 ± 2.7

A.2 MuJoCo Environments

We ran 5 independent random seeds and evaluate for 20 episodes the policy every 100 gradient steps for *HalfCheetah* and 500 gradient steps for *Hopper* and *Walker2d*. We plot the learning curves of the medium variants in Figure 4 and expert variant in Figure 5. To report the tests in Table 2, we early-stop the policy that outputs the best CVaR and evaluate on 50 episodes with 5 different random seeds.

A.2.1 Behavior policies performance

For sake of reference, we evaluate the stochastic reward function on the state-action pairs in the behavior data set. To estimate the returns, we use the state-action distribution in the data set and split it into chunks of 200 time steps for the Half-Cheetah and 500 time steps for the Walker2D and the Hopper. We then sample from the stochastic reward function on these state-actions pairs and compute the returns by adding them. We finally split the list of returns into K folds (K=5) and for each fold we compute the average and $\text{CVaR}_{0.1}$ of the returns. We show the averaged results across folds with corresponding standard deviation in Table 3.

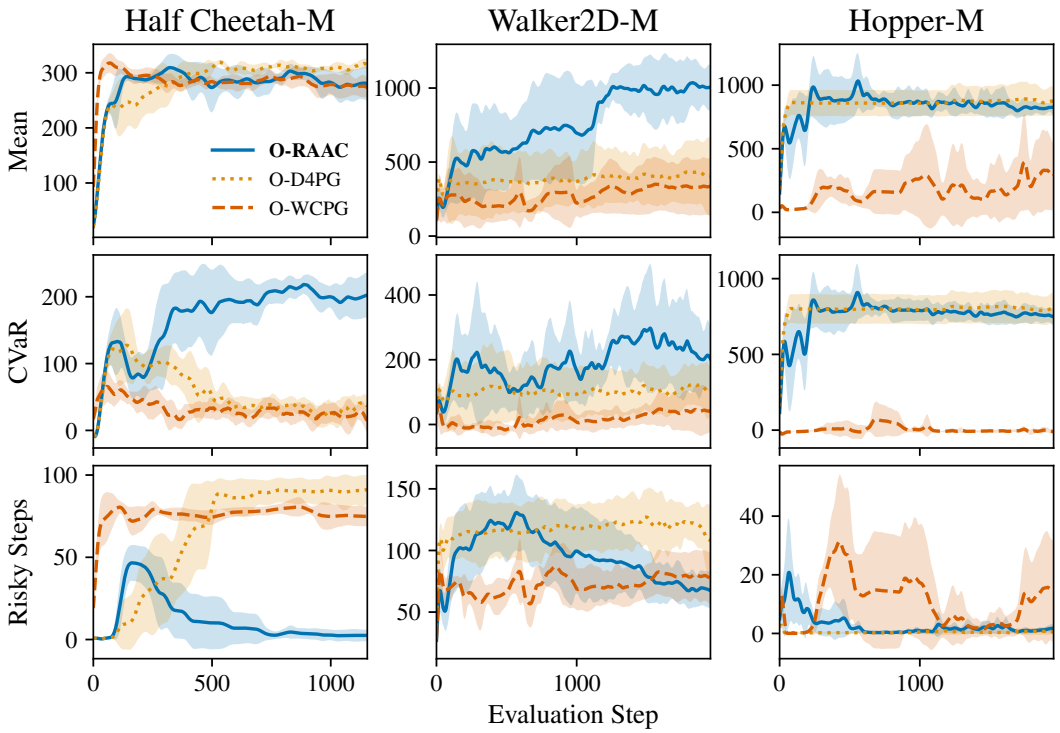

Figure 4: Experimental results across several Mujoco tasks for the Medium variant of each dataset.

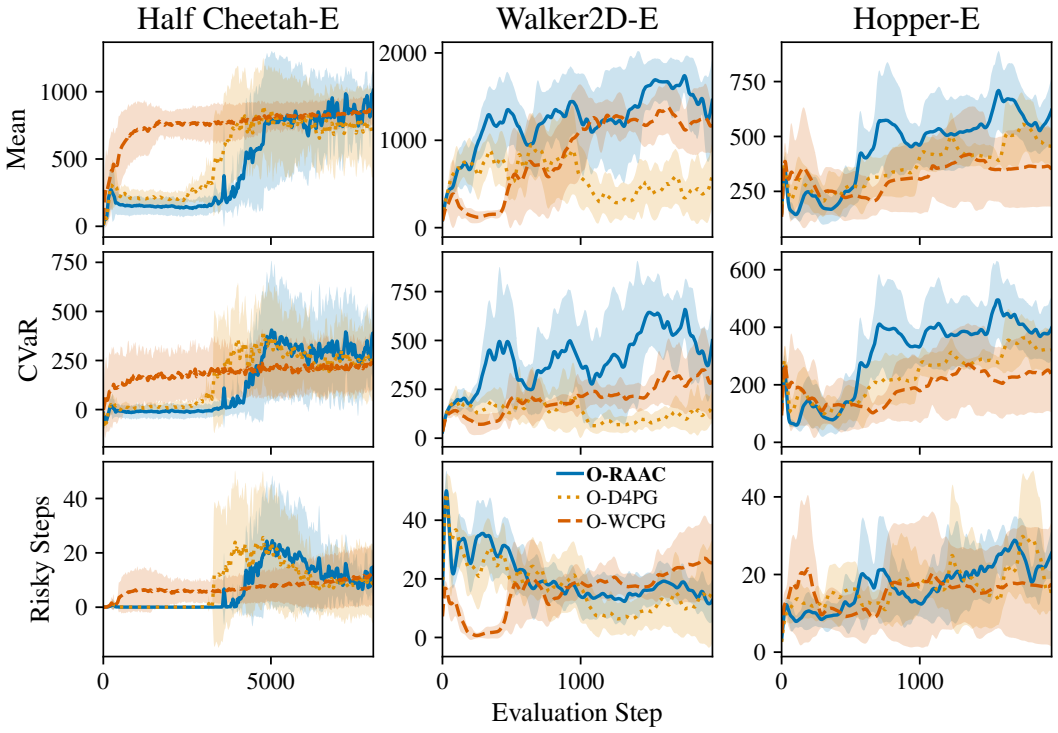

Figure 5: Experimental results across several Mujoco tasks for the Expert variant of each dataset.

A.3 Additional Experimental Details

We proceed to explain detailed information about the architectures and hyperparameters used to deploy the experiments. Across all methods and experiments, for fair comparison and whenever possible, each algorithm generally uses the same hyper-parameters and architecture unless explicitly stated.

A.3.1 Architectures

We use neural networks as function approximators for all the elements in the architecture.

Critic architecture: For the critic architecture, we build on the IQN network Dabney et al. (2018) but we extend it to the continuous action setting by adding an additional action input to the critic network, resulting in the function:

$$Z(s,a;\tau) = f(m_{sa\tau}(\left[m_{sa}([\psi_s(s),\psi_a(a)]),\psi_\tau(\tau)\right]), \tag{11}$$

where $\psi_s : \mathcal{X} \to \mathbb{R}^d$, $\psi_a : \mathcal{A} \to \mathbb{R}^d$, $m_{sa} : \mathbb{R}^{d\times d} \to \mathbb{R}^n$, $m_{sa\tau} : \mathbb{R}^n \to \mathbb{R}^n$, $\psi_\tau : \mathbb{R} \to \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$

For the embedding ψ_τ we use a linear function of n cosine basis functions of the form $\cos(\pi i\tau)$ $i = 1..,n$, with $n = 16$. For ψ_s, ψ_a we use a multi-layer perceptron (MLP) with a single hidden layer with $d = 64$ units for the Car experiment and with $d = 256$ units for all MuJoCo experiments. For the merging function m_{sa}, which takes as an input the concatenation of $\psi_s(s)$ and $\psi_a(a)$, we use a single hidden layer with $n = 16$ units. For the merging function $m_{sa\tau}$, we force interaction between its two inputs via a multiplicative function $m_{sa\tau}(u_1,u_2) = u_1 \odot u_2$, where $u_1 = m_{sa}(\psi_s(s),\psi_a(a))$ and $u_2 = \psi_\tau(\tau)$ and $\odot$ denotes the element-wise product of two vectors. For f we use a MLP with a single hidden layer with 32 units We used ReLU non-linearities for all the layers.

Actor architecture: The architecture of the actor model is

$$\pi(a|s) = b + \lambda\xi_\theta(s,b) \tag{12}$$

where $\xi : \mathcal{A} \rightarrow \mathbb{R}^{\|\mathcal{A}\|}$ and b is the output of the imitation learning component. For the RAAC algorithm we remove b and set $\lambda = 1$.

For the Car experiments, we used a MLP with 2 hidden layers of size 64. For the MuJoCo experiments, based on Fujimoto et al. (2019), we used a MLP embedding with 3 hidden layers of sizes 400, 300 and 300. We used ReLU non-linearities for all the hidden layers and we saturate the output with a Tanh non-linearity.

VAE architecture: The architecture of the conditional VAE_ϕ is also based on Fujimoto et al. (2019). It is defined by two networks, an encoder $E_{\phi_1}(s, a)$ and decoder $D_{\phi_2}(s, z)$. Each network has two hidden layers of size 750 and it uses ReLU non-linearities.

A.3.2 HYPERPARAMETERS

All the network parameters are updated using Adam (Kingma & Ba, 2015) with learning rates $\eta = 0.001$ for the critic and the VAE, and $\eta = 0.0001$ for the actor model. The target networks for the critic and the perturbation models are updated softly with $\mu = 0.005$.

For the critic loss (4) we use $N = N' = 32$ quantile samples, whereas to approximate the CVaR to compute the actor loss (5) (6) we use 8 samples from the uniform distribution between $[0, 0.1]$.

For all MuJoCo experiments, the λ parameter which modulates the action perturbation level was experimentally set to 0.75, except for the ***HalfCheetah*** experiment for which it was set to 0.5. Lower levels of λ were found to affect performance negatively.