Neural Computing and Applications (2024) 36:585–598
https://doi.org/10.1007/s00521-023-09029-3

ORIGINAL ARTICLE

Offline reinforcement learning in high-dimensional stochastic environments

Félicien Hêche[1,2] · Oussama Barakat[2] · Thibaut Desmettre[4] · Tania Marx[3] · Stephan Robert-Nicoud[1]

Received: 18 April 2023 / Accepted: 6 September 2023 / Published online: 11 October 2023

Abstract

Offline reinforcement learning (RL) has emerged as a promising paradigm for real-world applications since it aims to train policies directly from datasets of past interactions with the environment. The past few years, algorithms have been introduced to learn from high-dimensional observational states in offline settings. The general idea of these methods is to encode the environment into a latent space and train policies on top of this smaller representation. In this paper, we extend this general method to stochastic environments (i.e., where the reward function is stochastic) and consider a risk measure instead of the classical expected return. First, we show that, under some assumptions, it is equivalent to minimizing a risk measure in the latent space and in the natural space. Based on this result, we present Latent Offline Distributional Actor-Critic (LODAC), an algorithm which is able to train policies in high-dimensional stochastic and offline settings to minimize a given risk measure. Empirically, we show that using LODAC to minimize Conditional Value-at-Risk (CVaR) outperforms previous methods in terms of CVaR and return on stochastic environments.

Keywords Offline RL · Risk-averse RL · High-dimensional RL · Distributional RL

1 Introduction

Often, human decisions are stored and this builds interesting datasets. With the success of modern machine learning tools, comes the hope of exploiting them to construct useful decision helpers. To achieve this, we can use an imitation learning approach [1]. However, in this case, we will at best be as good as humans. Moreover, the performance of this approach depends heavily on the quality of the training dataset. In this work, we would like to avoid these limitations and thus, we consider another framework: reinforcement learning (RL).

RL has achieved impressive results in a number of challenging areas, including games [2, 3], robotic control [4, 5] and even health care [6, 7]. In particular, offline RL seems to be really interesting for real-world applications as it aims to train agents from a dataset of past interactions with the environment.

With the digitization of society, more and more features could be used to represent the environment. Unfortunately, classical RL algorithms are not able to work with high-dimensional states. Although we could manually choose a feature subset, this choice is not straightforward and it

✉ Félicien Hêche
felicien.heche@hes-so.ch

Oussama Barakat
oussama.barakat@univ-fcomte.fr

Thibaut Desmettre
Thibaut.Desmettre@hcuge.ch

Tania Marx
tmarx@chu-besancon.fr

Stephan Robert-Nicoud
stephan.robert@hes-so.ch

1 School of Engineering and Management, University of Applied Sciences and Arts Western Switzerland (HES-SO), Yverdon-les-Bains, Switzerland

2 Nanomedecine Lab, Imagery and Therapeutics, University of Franche-Comté, Besançon, France

3 Emergency Department, Centre Hospitalier Universitaire (CHU), Besançon, France

4 Emergency Department, Hôpitaux Universitaires de Genève (HUG), Genève, Switzerland

could have a huge impact on the performances. Therefore, it may be more practical to use RL algorithms capable of learning from high-dimensional states.

It is common to evaluate RL algorithms on deterministic (in the sense that the reward function is deterministic) environments such as the DeepMind Control suite [8]. However, in many real-world applications, environments are not deterministic but stochastic. In stochastic environments, the same action may lead to different outcomes due to randomness. This can make it difficult for the agent to distinguish between the quality of different actions and states, thereby complicating the learning process. Due to this inherent randomness, stochastic environments demand a larger number of samples to accurately estimate value functions and policies. On the other hand, offline learning has limited data available for training. The combination of these factors makes offline RL in stochastic environments particularly challenging. Furthermore, the performance of classical RL methods in such contexts remains unclear. Moreover, and as presented in Sect. 5.2, our experiments suggest that adding stochasticity to the environment decreases significantly the performance of these algorithms.

Based on these observations, we aim to develop a method which is able to train policies in offline settings with high-dimensional states while being robust to stochasticity. In this paper, we present Latent Offline Distributional Actor-Critic (LODAC), an algorithm designed to train policies in high-dimensional, stochastic environments within offline settings. The main idea is to encode the natural environment space into a smaller representation and train the agent directly in this latent space. But instead of considering the expected return, we use a risk measure. First, assuming some hypotheses, we show that minimizing this risk measure in the latent space is equivalent to minimizing the risk measure directly in the natural state. This theoretical result provides a natural framework: employ a latent variable model to encode the natural state into a latent space, and subsequently train a policy atop this latent space using a risk-sensitive RL algorithm. In the experimental part, we evaluate our algorithm on high-dimensional stochastic and deterministic datasets. In the best of our knowledge, we are the first authors to propose an algorithm to train policies in high-dimensional, stochastic and offline settings.

2 Related work

Before going further into our work, we present some related research.

As discussed and motivated in the previous section, this paper focuses on offline RL [9]. Offline RL is a specific approach of RL that aims to learn from past interactions with the environment. This framework holds promise for real-world applications as it enables the deployment of trained policies. Therefore, it is not surprising that offline RL has received a lot of attention in recent years [10–15]. One of the main problems in offline RL is that the Q-function tends to be overly optimistic in the case of out-of-distribution (OOD) states-actions pairs, where observational data is limited [16]. Several approaches have been introduced to address this issue. While some studies extend importance sampling methods [17, 18] other propose model-based techniques [19, 20] or even incorporate dataset re-weight sampling [21]. In our work, and following previous researches [22–24], we build a conservative estimation of the Q-function for OOD state-action pairs.

Another challenge encountered in our work is training policies from high-dimensional states. Over the past years, several algorithms have been proposed to address policy training within such contexts. For instance, several methods have been developed to directly handle image inputs [25–28]. Acquiring a meaningful representation of the observation is crucial to tackle this type of problem. Therefore, this issue has been studied by previous works [29, 30]. Some researchers employ data augmentation techniques to facilitate the learning of optimal representations [31]. However, these methods are limited to scenarios involving image states. Hence, in our work, we adhere to other common practices [32, 33] and we encode high-dimensional states into a latent space without relying on data augmentation. Furthermore, we integrate planning within this latent space, building upon studies that highlight its potential for performance enhancement [34–36]. Subsequently, a common approach involves training policies using classical RL algorithms, such as Soft Actor-Critic (SAC) [4], on top of this compact representation [37, 38]. Unfortunately, these classical RL algorithms struggle to train agents in stochastic environment. Therefore, in our study, we employ a risk-sensitive RL method atop this latent space.

Risk-sensitive RL is a specific approach to safe RL [39]. In safe RL, policies are trained to maximize performance while satisfying some safety constraints during training and/or in the deployment. In risk-sensitive RL, the focus is on minimizing a measure of the risk induced by the cumulative rewards, rather the maximizing the expected return. Risk-sensitive RL has gained attention in recent years [40–43]. Various risk measures may be considered, such that Exponential Utility [44] Cumulative Prospect Theory [45] or Conditional Value-at-Risk (CVaR) [46]. In this work, given the robust theoretical underpinnings and intuitive nature of Conditional Value-at-Risk [47, 48], we opt to focus on this operator. Furthermore, prior research suggests that taking into account Conditional Value-at-

Risk instead of the classical expectation, can prevent the performance gap between simulations and real-world applications [49]. CVaR is really popular and has been strongly studied in the context of RL for many years [50–52]. For instance, [53] extend SAC in a distributional settings, enabling the training of risk-averse policies. However, this algorithm is not suited for offline RL. Other authors presented O-RAAC [54], an algorithm which is able to train risk-averse policies in an offline settings. In this paper, OOD errors are managed through an imitation learning component [55]. Finally, Ma et al. [56] presented a method for training risk-averse policies in an offline setting. In our work, we aim to extend this method to enable the training of policies in the context of high-dimensional states.

3 Preliminaries

In this section, we introduce notations and recall concepts we will use later.

Coherent risk measure Let $(\Omega, \mathcal{F}, \mathbb{P})$ a probability space and $\mathcal{L}^2 := \mathcal{L}^2(\Omega, \mathcal{F}, \mathbb{P})$. A function $\mathcal{R}$: $\mathcal{L}^2 \to (-\infty, +\infty]$ is called a coherent risk measure [57] if

1. $\mathcal{R}(C) = C$ for all constants C.
2. $\mathcal{R}((1-\lambda)X + \lambda X)$ $\leq (1-\lambda)\mathcal{R}(X) + \lambda\mathcal{R}(X)$ for $\lambda \in (0,1)$.
3. $\mathcal{R}(X) \leq \mathcal{R}(X')$ when $X \leq X'$.
4. $\mathcal{R}(X) \leq 0$ when $\|X_k - X\|_2 \to 0$ with $\mathcal{R}(X_k) \leq 0$.
5. $\mathcal{R}(\lambda X) = \lambda\mathcal{R}(X)$ for $\lambda > 0$.

Coherent risk measures have some interesting properties. In particular, a risk measure is coherent if and only if, there exists a risk envelope $\mathcal{U}$ such that

$$\mathcal{R}(X) = \sup_{\delta \in \mathcal{U}} \mathbb{E}_q[\delta X] \tag{1}$$

[48, 58, 59]. A risk envelope is a nonempty convex subset of $\mathcal{P}$ that is closed and where $\mathcal{P} := \{\delta \in \mathcal{L}^2 \mid \delta \geq 0, \mathbb{E}_p[\delta] = 1\}$.

The definition of a risk measure $\mathcal{R}$ might depend on a probability distribution p. In some cases, it may be useful to specify which distribution we are working with. Thus, we sometimes use the notation $\mathcal{R}_p$. There exists a lot of different coherent risk measures, for example, the Wang risk measure [60], the entropic Value-at-Risk [61] or Conditional Value-at-Risk [46, 62].

Conditional Value-at-Risk (CVaR_α) with probability level $\alpha \in (0,1)$ is defined as

$$\text{CVaR}_\alpha(X) := \min_{t \in \mathbb{R}} \left\{ t + \frac{1}{1-\alpha}\mathbb{E}_p\left[\max\{0, X-t\}\right] \right\} \tag{2}$$

Moreover the risk envelope associated to CVaR_α, can be written as $\mathcal{U} = \{\delta \in \mathcal{P} \mid \mathbb{E}_p[\delta] = 1, 0 \leq \delta \leq \frac{1}{\alpha}\}$ [59, 63]. This rigorous definition may not be intuitive, but roughly speaking, CVaR_α is the expected value of X given the upper α-tail of its conditional distribution, representing the $(1-\alpha)$ worst-case scenarios. In this paper, we use the classical definition of Conditional Value-at-Risk presented in risk measure literature. In particular, X should be interpreted as a loss function.

Offline RL We consider a Markov Decision Process (MDP), $(S, \mathcal{A}, p, r, \mu_0, \gamma)$ where S is the environment space, $\mathcal{A}$ the action space, r the reward distribution $(r_t \sim r(\cdot|s_t, a_t))$, p the transition probability distribution $(s_{t+1} \sim p(\cdot|s_t, a_t))$. μ_0 is the initial state distribution and $\gamma \in (0,1)$ denotes the discount factor. For the purpose of notation, we define $p(s_0) := \mu_0(s_0)$ and we write the MDP $(S, \mathcal{A}, p, r, \gamma)$.

Actions are taking following a policy π which depends on the environment state, (i.e., $a_t \sim \pi(\cdot|s_t)$). A sequence on the MDP $(S, \mathcal{A}, p, r, \gamma)$, $\tau = s_0, a_0, r_0, \ldots s_H$, with $s_i \in S$ and $a_i \in \mathcal{A}$ is called a trajectory. A trajectory of fixed length $H \in \mathbb{N}$ is called an episode. Given a policy π, a rollout from state-action $(s,a) \in S \times \mathcal{A}$ is a random sequence $\{(s_0, a_0, r_0), (s_1, a_1, r_1), \ldots\}$ where $a_0 = a$, $s_0 = s$, $s_{t+1} \sim p(\cdot|s_t, a_t)$, $r_t \sim r(\cdot|s_t, a_t)$ and $a_t \sim \pi(\cdot|s_t)$. Given a policy π and a fixed length H, we have a trajectory distribution given by

$$p_\pi(\tau) = p(s_0)\prod_{t=0}^{H-1} \pi(a_t|s_t)p(s_{t+1}|s_t, a_t)r(r_t|s_t, a_t)$$

The goal of classical risk-neutral RL algorithms is to find a policy that maximizes the expected discounted return $\mathbb{E}_\pi[\sum_{t=0}^{H} \gamma^t r_t]$ where H could be infinite. Equivalently, we can look for the policy that maximizes the Q-function, which is defined as Q_π: $S \times \mathcal{A} \to \mathbb{R}$, $Q_\pi(s_t, a_t) := \mathbb{E}_\pi[\sum_{t'=t}^{H} \gamma^{t'-t} r_{t'}]$.

Instead of this classical objective function, other choices are possible. For example, the maximum entropy RL objective function $\mathbb{E}_\pi[\sum_{t=0}^{H} r_t + \mathcal{H}(\pi(\cdot|s_t))]$, where $\mathcal{H}$ denotes the entropy. This function has an interesting connection with variational inference [64] and it has shown impressive results in recent years [4, 19, 65].

In offline RL, we have access to a fixed dataset $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}$, where $s_t \in S$, $a_t \in \mathcal{A}$, $r_t \sim r(s_t, a_t)$ and $s_{t+1} \sim p(\cdot|s_t, a_t)$, and we aim to train policies without interaction with the environment. A such dataset comes with the empirical behavior policy

$$\pi_\beta(a|s) := \frac{\sum_{(s_t,a_t)\in\mathcal{D}} \mathbb{1}_{\{s_t=s, a_t=a\}}}{\sum_{s_t\in\mathcal{D}} \mathbb{1}_{\{s_t=s\}}}.$$

Latent variable model There are different methods for learning directly from high-dimensional states [29, 65, 66]. However, in this work, we build upon the framework presented in Stochastic Latent Actor-Critic (SLAC) [38]. The main idea of this work is to train a latent variable model to encode the natural MDP $(S, \mathcal{A}, p, r, \gamma)$ into a latent space $(Z, \mathcal{A}, q, r, \gamma)$ and to train policies directly in this space. To achieve this, the variational distribution $q(z_{1:H}, a_{t+1:H}|s_{1:t}, a_{1:t})$ is factorized into a product of inference term $q(z_{i+1}|z_i, s_{i+1}, a_i)$, latent dynamic term $q(z_{i+1}|z_i, a_i)$ and policy term $\pi(a_i|s_{1:t}, a_{1:t-1})$ as follows

$$\begin{aligned} &q(z_{1:H}, a_{t+1:H}|s_{1:t}, a_{1:t}) \\ &= \prod_{i=0}^{t} q(z_{i+1}|z_i, s_{i+1}, a_i) \prod_{i=t+1}^{H-1} q(z_{i+1}|z_i, a_i) \\ &\prod_{i=t+1}^{H-1} \pi(a_i|s_{1:i}, a_{1:i-1}) \end{aligned} \tag{3}$$

Using this factorization, the evidence lower bound (ELBO) [67] and a really interesting theoretical approach [64], the following objective function for the latent variable model is derived

$$\begin{aligned} &\mathbb{E}_{z_{1:t}, a_{t+1:H} \sim q} \Bigg[\sum_{i=0}^{t} \log D(s_{i+1}|z_{i+1}) \\ &- D_{KL}\Big(q(z_{i+1}|z_i, s_{i+1}, a_i)||p(s_{i+1}|s_i, a_i)\Big)\Bigg] \end{aligned} \tag{4}$$

where D_{KL} is the Kullback–Leibler divergence and D is a decoder.

Distributional RL The aim of distributional RL is to learn the distribution of the discounted cumulative rewards $Z_\pi := -\sum_t^H \gamma^t r_t$. Z_π is a random variable. A classical approach to distributional RL is to learn Z_π implicitly using its quantile function $F^{-1}_{Z(s,a)}$: $[0, 1] \to \mathbb{R}$, which is defined as $F^{-1}_{Z(s,a)}(y) := \inf\{x \in \mathbb{R} \mid y \leq F_{Z(s,a)}(x)\}$ and where $F_{Z(s,a)}$ is the cumulative density function of the random variable $Z(s, a)$. A model $Q_\theta(\eta, s, a)$ is used to approximate $F^{-1}_{Z(s,a)}(\eta)$. For $(s, a, r, s') \sim \mathcal{D}$, $a' \sim \pi(\cdot|s')$, $\eta, \eta' \sim \text{Uniform}[0, 1]$, we define $\delta := r + \gamma Q_{\theta'}(\eta', s', a') - Q_\theta(\eta, s, a)$. Q_θ is trained using the τ-Huber quantile regression loss at threshold k [68]

$$\begin{aligned} &\mathcal{L}_k(\delta, \eta) \\ &:= \begin{cases} |\eta - \mathbb{1}_{\{\delta<0\}}|(\delta^2/2k) & \text{if } |\delta| < k \\ |\eta - \mathbb{1}_{\{\delta<0\}}|(|\delta| - k/2) & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

With this function $F_{Z(s,a)}$, different risk measures can be computed, such that Cumulative Probability Weight (CPW) [45], Wang measure [60] or Conditional Value-at-Risk. For example, the following equation [69] is used to compute CVaR_α

$$\text{CVaR}_\alpha(X) = \frac{1}{1-\alpha} \int_\alpha^1 F_{Z^{-1}(s,a)}(\tau) \mathrm{d}\tau \tag{6}$$

It is also possible to extend distributional RL to offline settings. For example, O-RAAC [54] decomposes the actor into two different components an imitation actor and a perturbation model. CODAC [56] extends DSAC [70] in offline settings. More precisely, the $Q_\theta(\eta, s, a)$ are trained using the following loss function

$$\begin{aligned} \alpha \mathbb{E}_{\eta \sim U} \Bigg[&\mathbb{E}_{s \sim \mathcal{D}} \Bigg[\log \sum_a \exp(Q_\theta(\eta, s, a)) \Bigg] \\ &- \mathbb{E}_{(s,a) \sim \mathcal{D}}[Q_\theta(\eta, s, a)] \Bigg] + \mathcal{L}_k(\delta, \eta) \end{aligned} \tag{7}$$

where $U = \text{Uniform}[0, 1]$. The first term of the equation, is introduced to avoid overly optimistic estimations for OOD state-action pairs [22]. The second term (i.e. $\mathcal{L}_k(\delta, \tau)$) is the classical objective function used to train the Q-function in a distributional settings.

4 Theoretical considerations

In this paper, we aim to train policies in high-dimensional, stochastic environment within offline settings. For a practical point of view, our idea is straightforward. Since training with high-dimensional states directly fails and following previous works [19, 38], we encode our high-dimensional states into a more compact representation using ϕ: $S \to Z$, where $Z = \phi(S)$. Then, using ϕ, we build an MDP in the latent space and train policies directly on top of this space.

However, from a theoretical point of view, this general idea is not entirely clear. Is this latent MDP always well-defined? If we find a policy that minimizes a risk measure in the latent space, will it also minimize the risk measure in the natural state?

The first goal of this section is to rigorously construct an MDP in the latent space. Then, we show that, for certain coherent risk measures and under some assumptions, minimizing the risk measure in the latent space is equivalent to minimizing it in the natural space. In particular, we demonstrate that it is the case for Conditional Value-at-Risk.

4.1 Theoretical results

First, we make the following assumptions

1. $\forall a \in \mathcal{A}$ we have $r(\cdot|s, a) = r(\cdot|s', a)$ if $\phi(s) = \phi(s')$.

2. We note $\mathbb{P}(\cdot|s_t, a_t)$ the probability measure with probability density function $p(\cdot|s_t, a_t)$. We suppose that if $s_t, s'_t \in S$ satisfy $\phi(s_t) = \phi(s'_t)$, then $\mathbb{P}(\cdot|s_t, a_t) = \mathbb{P}(\cdot|s'_t, a_t)$.
3. We note $\mathbb{Q}(\cdot|z_t, a_t)$ the probability image of $\mathbb{P}(\cdot|s_t, a_t)$ by ϕ and where s_t is any element of $\phi^{-1}(z_t)$. We suppose $\mathbb{Q}(\cdot|z_t, a_t)$ admits a probability density functions $q(\cdot|z_t, a_t)$.

Under these assumptions, ϕ induces an MDP $(Z, \mathcal{A}, q, r', \gamma)$ where the reward distribution r' is defined as $r'(\cdot|z, a) := r(\cdot|s, a)$ for any $s \in \phi^{-1}(z)$. r' is well defined by surjectivity of ϕ and by assumption (1). $\mathbb{Q}$ is well defined by hypothesis (2).

Obviously, for a given policy π' on the MDP $(Z, \mathcal{A}, q, r', \gamma)$ and fixed length H, we have a trajectory distribution

$$q_{\pi'}(\tau') = q(z_0) \prod_{t=0}^{H-1} \pi'(a_t|z_t) q(z_{t+1}|z_t, a_t) r'(r_t|z_t, a_t)$$

Given a policy π and a fixed length H, we denote Ω the set of all trajectories on the MDP $(S, \mathcal{A}, p, r, \gamma)$, $\mathcal{F}$ the σ-algebra generated by these trajectories and $\mathbb{P}_\pi$ the probability with probability distribution p_π. Following the same idea and given a policy π', we denote Ω' the set of trajectories of length H on the MDP $(Z, \mathcal{A}, q, r')$, $\mathcal{F}'$ the σ-algebra generated by these trajectories and $\mathbb{Q}_{\pi'}$ the probability with probability distribution $q_{\pi'}$. $(\Omega, \mathcal{F}, \mathbb{P}_\pi)$ and $(\Omega', \mathcal{F}', \mathbb{Q}_{\pi'})$ are probability spaces.

ϕ: $S \to Z$ induces a map

$$\Omega : \to \Omega'$$
$$(s_0, a_0, r_0, \ldots, s_H) \mapsto (\phi(s_0), a_0, r_0, \ldots, \phi(s_H))$$

With a slight abuse of notation, we write it ϕ.

We denote Π the set of all policies π on the MDP $(S, \mathcal{A}, p, r)$ which satisfies $\pi(a|s) = \pi(a|s')$ if $\phi(s) = \phi(s')$. If $\pi \in \Pi$, then π induces a policy π' on the MDP $(Z, \mathcal{A}, q, r')$, taking $\pi'(a|z) := \pi(a|s)$ where s is any element of $\phi^{-1}(z)$.

For any $\pi \in \Pi$, we denote π' the associated policy on the MDP $(Z, \mathcal{A}, q, r')$ as defined above. Furthermore, we note $\Pi' := \{\pi' \mid \pi \in \Pi\}$. If $\pi' \in \Pi'$, a policy $\pi \in \Pi$ can be defined as $\pi(a|s) := \pi'(a|\phi(s))$. X is a random variable on Ω and X' a random variable on Ω'.

With all these notations, we can introduce our first result.

Lemma 1 *Let $\pi \in \Pi$. Then, $\mathbb{Q}_{\pi'}$ is the probability image of $\mathbb{P}_\pi$ by ϕ.*

Proof Let $B' := Z_0 \times A_0 \times R_0 \times \ldots Z_H \in \mathcal{F}'$. We have

$$\begin{aligned}
&\mathbb{P}_\pi(\phi^{-1}(B')) \\
&\quad = \int_{\phi^{-1}(B')} p_\pi(\tau) \mathrm{d}\tau \\
&\quad = \int_{\phi^{-1}(Z_0) \times A_0 \times R_0 \times \ldots \phi^{-1}(Z_H)} p(s_0)\pi(a_0|s_0) \\
&r(r_0|a_0, s_0) p(s_1|a_0, s_0) \ldots p(s_H|a_{H-1}, s_{H-1}) \\
&\mathrm{d}s_0 \mathrm{d}a_0 \mathrm{d}r_0 \mathrm{d}s_1 \ldots \mathrm{d}s_H \\
&\quad = \int_{\phi^{-1}(Z_0) \times A_0 \times R_0 \times \ldots \phi^{-1}(Z_H)} p(s_0)\pi'(a_0|\phi(s_0)) \\
&r'(r_0|a_0, \phi(s_0)) p(s_1|a_0, s_0) \ldots p(s_H|a_{H-1}, s_{H-1}) \\
&\mathrm{d}s_0 \mathrm{d}a_0 \mathrm{d}r_0 \mathrm{d}s_1 \ldots \mathrm{d}s_H
\end{aligned}$$

Then, denoting $\bar{s_0}$ one arbitrary element of $\phi^{-1}(z_0)$, we obtain

$$\begin{aligned}
&\mathbb{P}_\pi(\phi^{-1}(B')) \\
&\quad = \int_{Z_0 \times A_0 \times R_0 \times \ldots \phi^{-1}(Z_H)} q(z_0)\pi'(a_0|z_0) r'(r_0|a_0, z_0) \\
&p(s_1|a_0, \bar{s_0}) \ldots p(s_H|a_{H-1}, s_{H-1}) \mathrm{d}z_0 \ldots \mathrm{d}s_1 \ldots \mathrm{d}s_H
\end{aligned}$$

Iterating this process, we find

$$\begin{aligned}
&\mathbb{P}_\pi(\phi^{-1}(B')) \\
&\quad = \int_{Z_0 \times A_0 \times R_0 \times \ldots Z_H} q(z_0)\pi'(a_0|z_0) r'(r_0|a_0, z_0) \\
&q(z_1|a_0, z_0) \ldots q(z_H|a_{H-1}, z_{H-1}) \mathrm{d}z_0 \mathrm{d}a_0 \mathrm{d}r_0 \ldots \mathrm{d}z_H \\
&\quad = \int_{B'} q_{\pi'}(\tau') \mathrm{d}\tau' = \mathbb{Q}_{\pi'}(B')
\end{aligned}$$

□

This result is not a surprise but it has an interesting implication. Indeed, it implies that if $X' \circ \phi = X$, then $\mathbb{E}_{q_{\pi'}}[X'] = \mathbb{E}_{p_\pi}[X]$. And thus, we get the following result.

Corollary 1 *Suppose $X' \circ \phi = X$. Then, if a policy $\pi'_\star$ satisfies*

$$\pi'_\star = \mathrm{argmin}_{\pi' \in \Pi'} \mathbb{E}_{q_{\pi'}}[X']$$

its associated policy $\pi_\star$ satisfies

$$\pi_\star = \mathrm{argmin}_{\pi \in \Pi} \mathbb{E}_{p_\pi}[X]$$

For a coherent risk measure $\mathcal{R}$, we write $\mathcal{U}$ the risk envelope associated to $\mathcal{R}$ in Ω and $\mathcal{U}'$ the corresponding risk envelope in Ω'. If $\forall \delta \in \mathcal{U}$, $\exists \delta' \in \mathcal{U}'$ such that $\delta = \delta' \circ \phi$ and $\forall \delta' \in \mathcal{U}'$ $\exists \delta \in \mathcal{U}$ with $\delta = \delta' \circ \phi$ almost everywhere, we write $\mathcal{U} = \mathcal{U}' \circ \phi$.

Proposition 1 *Let $\mathcal{R}$ be a coherent risk measure. Suppose that $X' \circ \phi = X$ and $\mathcal{U} = \mathcal{U}' \circ \phi$. Then, if a policy $\pi'_\star$ satisfies*

$$\pi'_\star = \text{argmin}_{\pi' \in \Pi'} \mathcal{R}(X')$$

its associated policy $\pi_\star$ verifies

$$\pi_\star = \text{argmin}_{\pi \in \Pi} \mathcal{R}(X)$$

Proof Since $\mathcal{U} = \mathcal{U}' \circ \phi$, we have

$$\sup_{\delta' \in \mathcal{U}'} \mathbb{E}_q[\delta' X'] = \sup_{\delta \in \mathcal{U}} \mathbb{E}_p[\delta X]$$

Thus

$$\mathcal{R}(X') := \sup_{\delta' \in \mathcal{U}'} \mathbb{E}_q[\delta' X'] = \sup_{\delta \in \mathcal{U}} \mathbb{E}_p[\delta X] = \mathcal{R}(X)$$

Now, let $\pi'_\star := \text{argmin}_{\pi' \in \Pi} \mathcal{R}(X')$ and $\pi_\star$ the associated policy. By contradiction, suppose there exists π_1 with $\mathcal{R}_{\pi_1}(X) < \mathcal{R}_{\pi_\star}(X)$. But in this case and by the above observation, we would have $\mathcal{R}_{\pi'_1}(X') < \mathcal{R}_{\pi'_\star}(X')$. □

This result provides a useful criteria to prove the optimality of a method. For example, consider $\mathcal{R}(X) = \mathbb{E}_{p_\pi}[X]$ which is obviously a coherent risk-measure. Its risk envelope is $\mathcal{U} \equiv 1$. And thus, Corollary 1 follows directly from the last proposition too.

We also have the same result in the case where $\mathcal{R}(X) = \text{CVaR}_\alpha(X)$.

Proposition 2 *Suppose that $X' \circ \phi = X$. Then, if a policy $\pi'_\star$ satisfies*

$$\pi'_\star = \text{argmin}_{\pi' \in \Pi'} \text{CVaR}_\alpha(X')$$

its associated policy $\pi_\star$ verifies

$$\pi_\star = \text{argmin}_{\pi \in \Pi} \text{CVaR}_\alpha(X)$$

Proof Recall that the risk envelop of the CVaR_α, takes the form $\mathcal{U} = \{\delta \mid 0 \le \delta \le \frac{1}{\alpha}\ \mathbb{E}_p[\delta] = 1\}$. First, we will show that for each $\delta' \in \mathcal{U}'$ there exists $\delta \in \mathcal{U}$ such that $\mathbb{E}_q[\delta' X'] = \mathbb{E}_p[\delta X]$. Then, we will show that for each $\delta \in \mathcal{U}$ there exists $\delta' \in \mathcal{U}'$ such that $\mathbb{E}_p[\delta X] = \mathbb{E}_q[\delta' X']$.

- Let $\delta' \in \mathcal{U}'$. We define $\delta := \delta' \circ \phi$. By construction, we have $0 \le \delta \le \frac{1}{\alpha}$ and $\mathbb{E}_p[\delta] = \mathbb{E}_p[\delta' \circ \phi] = \mathbb{E}_q[\delta'] = 1$. Thus, $\delta \in \mathcal{U}$. With the same idea, remark that $\mathbb{E}_q[\delta' X'] = \mathbb{E}_p[\delta X]$.
- Let $\delta \in \mathcal{U}$. By construction, δp_π is a density function. Let $\tilde{\mathbb{P}}_\pi$ its associated probability measure. We consider $\tilde{\mathbb{Q}}_{\pi'}$ the probability image of $\tilde{\mathbb{P}}_\pi$ by ϕ. Then, remark that $\tilde{\mathbb{Q}}_{\pi'}$ is absolutely continuous with respect to $\mathbb{Q}_{\pi'}$. Indeed, if A satisfies $\mathbb{Q}_{\pi'}(A) = 0$, we have

$$\begin{aligned}\tilde{\mathbb{Q}}_{\pi'}(A) = \tilde{\mathbb{P}}_\pi(\phi^{-1}(A)) &= \int_{\phi^{-1}(A)} \delta(\tau) p(\tau) \mathrm{d}\tau \\ &\le \frac{1}{\alpha} \mathbb{P}_\pi(\phi^{-1}(A)) = \frac{1}{\alpha} \mathbb{Q}_\pi(A) = 0\end{aligned}$$

Thus by Radon-Nikodym, there exists $\delta' \ge 0$, such that for all $B \in \mathcal{F}'$

$$\tilde{\mathbb{Q}}_{\pi'}(B) = \int_B \delta' \mathrm{d}\mathbb{Q}_{\pi'} = \int_B \delta'(\tau') q_{\pi'}(\tau') \mathrm{d}\tau'$$

We will show that $\delta' \in \mathcal{U}'$. We define $C := \{\tau' \in \Omega' \mid \delta'(\tau) > \frac{1}{\alpha}\}$. By contradiction, suppose that C is not empty. On one hand have

$$\begin{aligned}\tilde{\mathbb{Q}}_{\pi'}(C) &= \int_C \delta'(\tau') q_\pi(\tau') \mathrm{d}\tau' \\ &> \frac{1}{\alpha} \int_C q_\pi(\tau') \mathrm{d}\tau' = \frac{1}{\alpha} \mathbb{Q}_{\pi'}(C)\end{aligned}$$

And on the other hand, we have

$$\begin{aligned}\tilde{\mathbb{Q}}_\pi(C) &= \tilde{\mathbb{P}}_\pi(\phi^{-1}(C)) \\ &= \int_{\phi^{-1}(C)} \delta(\tau) p_\pi(\tau) \mathrm{d}\tau \\ &\le \frac{1}{\alpha} \mathbb{P}_\pi(\phi^{-1}(C)) = \frac{1}{\alpha} \mathbb{Q}_{\pi'}(C)\end{aligned}$$

Therefore C is empty and thus, $0 \le \delta' \le \frac{1}{\alpha}$. Then, since $\delta' q_{\pi'}$ is a probability density function $\delta' \in \mathcal{U}'$.

Finally, since, $\delta' q_{\pi'}$ is the probability density function of the probability image of $\tilde{\mathbb{P}}_\pi$, we obtain $\mathbb{E}_{\delta p_\pi}[X] = \mathbb{E}_{\delta' q_{\pi'}}[X']$

□

In particular, for

$$X(\tau) = \sum_{t=0}^{H} -r_t - \beta \mathcal{H}(\pi(\cdot|s_t))$$

and

$$X'(\tau') := \sum_{t=0}^{H} -r_t - \beta \mathcal{H}(\pi'(\cdot|z_t)),$$

(with β potentially null) we have

$$\begin{aligned}(X' \circ \phi)(\tau) &= \sum_{t=0}^{H} -r_t - \beta \mathcal{H}(\pi'(\cdot|\phi(s_t))) \\ &= \sum_{t=0}^{H} -r_t - \beta \mathcal{H}(\pi(\cdot|s_t)) = X(\tau)\end{aligned}$$

Therefore, we get the following result.

Corollary 2 *Let* $X\colon \Omega \to \mathbb{R}$ *and* $X'\colon \Omega' \to \mathbb{R}$ *defined as* $X(\tau) := \sum_{t=0}^{H} -r_t - \beta\mathcal{H}(\pi(\cdot|s_t))$ *and* $X'(\tau') := \sum_{t=0}^{H} -r_t - \beta\mathcal{H}(\pi'(\cdot|z_t))$. *Then, if a policy* $\pi'_\star$ *satisfies*

$$\pi'_\star = \text{argmin}_{\pi' \in \Pi'} \text{CVaR}_\alpha(X')$$

its associated policy $\pi_\star$ verifies

$$\pi_\star = \text{argmin}_{\pi \in \Pi} \text{CVaR}_\alpha(X)$$

4.2 Discussion

The results presented in the last section establish a theoretical equivalence between minimizing the risk measure in the latent space and in the natural space. However, to obtain this guarantee we need to make some assumptions.

Assumptions (1), (2) ensure that the reward distribution $r(\cdot|s_t, a_t)$ and the probability measure $\mathbb{P}(\cdot|s_t, a_t)$ are insensitive to any change in $\phi^{-1}(z_t)$. Moreover $\pi \in \Pi$ ensures that the policy distribution is stable to any change in $\phi^{-1}(z_t)$. Thus, and roughly speaking, these assumptions guarantee we do not lose information by encoding s_t into $\phi(s_t)$, in terms of reward distribution, transition probability measure and the optimal policy.

These theoretical considerations highlight that in order to learn a meaningful latent representation of the natural MDP, we should consider all components of the MDP and not only focus on the environment space.

5 Latent Offline Distributional Actor-Critic

The theoretical results presented above justify our really natural idea: encode the natural environment space S into a compact representation Z and then, use a risk-sensitive offline RL algorithm to learn on top of this space. This is the general idea of LODAC.

5.1 Practical implementations of LODAC

In this section, we present the practical implementation of LODAC used in our experiments.

First, we need to implement and train a latent variational latent model. Following previous works [19, 23], our latent variable model contains the following components

$$\begin{aligned} \text{Image encoder:} \quad & h_t = E_\theta(s_t) \\ \text{Latent transition model:} \quad & z_t \sim q_\theta(\cdot|z_{t-1}, a_{t-1}) \\ \text{Inference model:} \quad & z_t \sim \phi_\theta(\cdot|h_t, z_{t-1}, a_{t-1}) \\ \text{Image decoder:} \quad & \hat{s}_t \sim D_\theta(\cdot|z_t) \end{aligned}$$

The image encoder, denoted as E_θ, is a classical Convolutional Neural Network consisting of 4 2D convolutional layers. Each of them uses a kernel of size 4, a stride of 2 and Rectified Linear Unit (ReLU) as activation function. These layers use 32, 64, 128 and 256 filters, respectively.

The inference and the latent transition model are implemented as a Recurrent State Space Model (RSSM) [66]. Specifically, a latent state contains two components $z_t = [d_t, x_t]$ where d_t is the deterministic part and x_t the stochastic. d_t is computed using a dense layer which takes x_{t-1}, a_{t-1} as input, uses Exponential Linear Unit as activation function and contains 256 hidden units. This layer is followed by a gated recurrent unit (GRU) cell which uses the hyperbolic tangent as activation function. The dimension of the output is 256. d_t corresponds to the hidden state of this GRU cell.

Then, the latent transition model employs two dense layers. The first one uses Exponential Linear Unit as activation function and yields an output of dimension 256. The second one does not have any activation function and provide an output of dimension 128. This output is subsequently split into two parts. The first part is interpreted as the mean of a normal distribution, while the second part (after applying a softplus) represents the standard deviation. The stochastic component of the latent state x_t is sampled from this distribution.

The inference model computes the deterministic part of the latent state d_t using the first layers of the latent transition model. Then, d_t and h_t are fed into two consecutive dense layers. The first one uses an Exponential Linear Unit

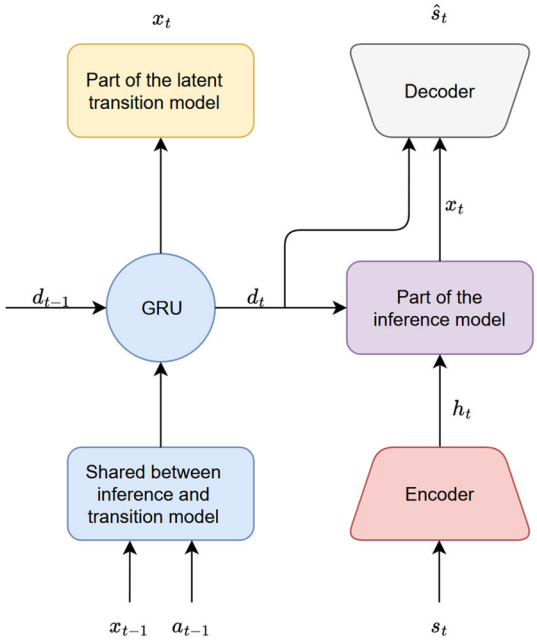

Fig. 1 Architecture of our latent variational model

activation function and 256 hidden units. The final layer does not have any activation function and the dimension of the output is 128. Similar to the latent transition model, the output is divided into two parts, which are interpreted as the mean and the variance of a normal distribution. x_t is sampled from this distribution.

Finally, the decoder D_θ consists of a dense layer with 1024 units followed by four deconvolutional layers. Each of them have a stride of 2 and except the last layer which do not have any activation function. All layers have ReLU as activation function. The first two deconvolutional layers have a kernel size of 5, while the last two have a kernel size of 6. Finally, these layers contain 128, 64, 32, 32 and 3 filters, respectively. The reconstructed image is sampled from a normal distribution where the mean is determined by D_θ and the standard deviation is fixed to one. The main components of our latent variable model are presented in Fig. 1.

These models have been trained using Adam optimizer, a batch size of 64, a learning rate of $6 * 10^{-4}$ and the following objective function [19]

$$\mathbb{E}_{q_\theta}\Bigg[\sum_{t=0}^{H-1}\log D_\theta(s_{t+1}|z_{t+1}) - D_{KL}(\phi_\theta(z_{t+1}|s_{t+1},z_t,a_t)||q_\theta(z_{t+1}|z_t,a_t))\Bigg] \tag{8}$$

Rewards estimation come from a sampling from a normal distribution where the mean is determined by the output of a neural network r_θ and the standard deviation is fixed to one. r_θ contains one hidden dense layer with 128 units and uses Exponential Linear Unit as activation function. r_θ is trained using maximum log-likelihood.

After the training of these models, the dataset $\mathcal{D}$ is encoded into the latent space and stored in a replay buffer $\mathcal{B}_{\text{latent}}$. More precisely, $\mathcal{B}_{\text{latent}}$ contains transitions of the form $(z_{1:H}, r_{1:H}, a_{1:H})$ where $z_{1:H} \sim \phi_\theta(\cdot|s_{1:H}, a_{H-1})$ and $s_{1:H}, r_{1:H-1}, a_{1:H-1} \sim \mathcal{D}$. Next, we introduce a latent buffer $\mathcal{B}_{\text{synthetic}}$ which contains rollouts transitions performed using the policy π_θ, the latent model q_θ and the reward estimator r_θ.

The policy π_θ comprises three dense layers. The first two use ReLU as activation function and 256 hidden units. The final layer does not use any activation function and yields two distinct numbers. The first number is interpreted as the mean of a normal distribution and the second one as a standard deviation. Action is sampled from this distribution.

The critic Q_θ is implemented as a quantile distributional critic network [71]. Specifically, we have $Q_\theta(\eta, z, a) = f_\theta(\psi_\theta(z, a) \odot \xi_\theta(\eta))$ where $\odot$ denotes the element-wise (Hadamard) product. ψ_θ is a dense layer with ReLU as activation function, provides an output of dimension 256 and uses layer normalization [72]. ξ_θ consists of an embedding of η into a space of dimension 64 followed by a dense layer. Specifically, η is embedded into a vector where the components take the form $\cos(i\eta\pi)$ with i ranging from 1 to 64. After that, this embedded vector is fed to a dense layer which uses sigmoid as activation function and produces an output of dimension 256. This layer also employs layer normalization. Finally, the model f_θ consists of two dense layers. The first one uses layer normalization, ReLU as activation function and 256 hidden units. The last layer does not use any activation function and produces an output of dimension one.

Actor π_θ and critic Q_θ are trained on $\mathcal{B} := \mathcal{B}_{\text{synthetic}} \cup \mathcal{B}_{\text{latent}}$. To achieve this, and based on our empirical results, we follow Ma et al. [56]. Thus, the critic Q_θ is iteratively chosen to minimize

$$\alpha\mathbb{E}_{\eta\sim U}\Bigg[\mathbb{E}_{z\sim\mathcal{B}}\Bigg[\log\sum_a \exp(Q_\theta(\eta,z,a))\Bigg] - \mathbb{E}_{(z,a)\sim\mathcal{B}}[Q_\theta(\eta,z,a)]\Bigg] + \mathcal{L}_k(\delta,\eta') \tag{9}$$

where $U = \text{Uniform}[0,1]$,

$$\delta = r_t + \gamma Q_\theta(\eta', z_{t+1}, a_{t+1}) - Q_\theta(\eta, z_t, a_t)$$

with $(z_t, a_t, r_t) \sim \mathcal{B}$, $a_{t+1} \sim \pi_\theta(\cdot|z_t)$ and $\mathcal{L}_k$ is the τ-Huber quantile regression loss as defined in (5). Following Kumar et al. [22], Ma et al. [56], we add two parameters $\zeta, \omega \in \mathbb{R}_{>0}$ to the last equation

$$\max_{\alpha\geq 0}\ \alpha\ \mathbb{E}_{\eta\sim U}\Bigg[\omega\Bigg(\mathbb{E}_{z\sim\mathcal{B}}\Bigg[\log\sum_a \exp(Q_\theta(\eta,z,a))\Bigg] - \mathbb{E}_{(z,a)\sim\mathcal{B}}\Bigg[Q_\theta(\eta,z,a)\Bigg]\Bigg) - \zeta\Bigg] + \mathcal{L}_k(\delta,\eta)$$

ζ is used to threshold the difference between $\mathbb{E}_{(z,a)\sim\mathcal{B}}[Q_\theta(\eta,z,a)]$ and the regularizer $\mathbb{E}_{z\sim\mathcal{B}}\big[\log\sum_a \exp(Q_\theta(\eta,z,a))\big]$. The parameter ω scales this difference. Remark that if this difference (scaled to ω) is smaller than the parameter ζ, then α will be set to 0 and only the term $\mathcal{L}_k(\delta,\eta)$ will be considered.

Furthermore, since the action space $\mathcal{A}$ is continuous, the computation $\log\sum_a \exp(Q_\theta(\eta,z,a)$ is intractable. To overcome this problem, and as introduced in Kumar et al. [22], we use the following approximation

$$\log\sum_a \exp(Q_\theta(\eta,z,a)) \approx \log\Bigg(\frac{1}{2M}\sum_{a_i\sim U(\mathcal{A})}^{M}\Bigg[\frac{\exp(Q_\theta(\eta,z,a))}{U(\mathcal{A})}\Bigg] + \frac{1}{2M}\sum_{a_i\sim\pi(\cdot|z)}^{M}\Bigg[\frac{\exp(Q_\theta(\eta,z,a))}{\pi(a_i|z)}\Bigg]\Bigg) \tag{10}$$

where $U(\mathcal{A}) = \text{Unif}(\mathcal{A})$ and we choose $M = 10$.

The actor is trained to minimize Conditional Value-at-Risk of the negative cumulative reward, which can be computed using the formula

$$\text{CVaR}_\alpha(Z_\pi) = \frac{1}{1-\alpha}\int_\alpha^1 Q_\theta(\eta, z, a)\text{d}\eta \tag{11}$$

π_θ and Q_θ have been trained using Adam optimizer with a batch size of 256. Following previous works [19, 23], batches of equal data mixed from $\mathcal{B}_{\text{latent}}$ and $\mathcal{B}_{\text{synthetic}}$ are used. While the critic has been trained using a learning rate of $3 * 10^{-4}$, we used a value of $3 * 10^{-5}$ for the actor. Finally, it is worth mentioning we used the clipped double Q-learning trick [73] for training Q_θ. A summary of LODAC can be found in Algorithm 1.

Remark that for training a policy using the approach presented above, we need to be able to compute an expectation of a uniform distribution (Eq. 9), approximate $\log \sum_a \exp(Q_\theta(\eta, z, a))$ using 10 and estimate Conditional Value-at-Risk using the formula 11 to compute the loss of the policy. Unfortunately, all these computations take time and are unavoidable. Thus, it is not surprising that optimizing a policy using this approach takes more time than classical methods.

Algorithm 1 LODAC

Require: dataset $\mathcal{D}$, models train steps, initial latent steps, number iterations, number actor-critic steps.

for models train steps **do**
  Sample a batch of sequence $(s_{1:H}, a_{1:H-1}, r_{1:H-1})$ from $\mathcal{D}$ and train a variational latent model using equation (8) and a reward estimator r_θ.
end for
for Initial latent step **do**
  Sample a batch of sequence $(s_{1:H}, a_{1:H-1}, r_{1:H-1})$ from $\mathcal{D}$
  Sample $z_{1:H}$ from the latent model and add the transitions $(z_{1:H}, a_{1:H-1} r_{1:H-1})$ to $\mathcal{B}_{\text{latent}}$.
end for
for initial synthetic latent steps **do**
  Sample a batch of sequences $(s_{1:H}, a_{1:H-1}, r_{H-1})$ from $\mathcal{D}$
  Sample a set of latent states $S \sim \phi_\theta(\cdot|s_{1:H}, a_{1:H-1})$ from the trained latent model.
  for $s_0 \in S$ **do**
    for $h \in \{1 : H\}$ **do**
      Sample a random action a_{h-1} and $z_h \sim q_\theta(\cdot|z_{h-1}, a_{h-1})$; estimate the reward $r_t \sim r_\theta$
      Add $(z_{h-1}, a_{h-1}, z_h, r_h)$ to $\mathcal{B}_{\text{synthetic}}$.
    end for
  end for
end for
for number iterations **do**
  for number actor-critic steps **do**
    Sample transitions from $\mathcal{B}_{\text{latent}} \cup \mathcal{B}_{\text{synthetic}}$
    Train the critic to minimize (9)
    Train the actor to minimize $\text{CVaR}_\alpha(Z_\pi)$ which can be computed using $Q(\eta, z, a)$ and (11).
  end for
  for rollout steps **do**
    Sample a batch of sequences $(s_{1:H}, a_{1:H-1}, r_{H-1})$ from $\mathcal{D}$.
    Sample a set of latent states $S \sim \phi_\theta(\cdot|s_{1:H}, a_{1:H-1})$ from the trained latent model.
    for $s_0 \in S$ **do**
      for $h \in \{1 : H\}$ **do**
        Sample action $a_{h-1} \sim \pi_\theta(\cdot|z_{h-1})$ and $z_h \sim q_\theta(\cdot|z_{h-1}, a_{h-1})$; estimate the reward r_h.
        Add $(z_{h-1}, a_{h-1}, z_h, r_h)$ to $\mathcal{B}_{\text{synthetic}}$.
      end for
    end for
  end for
end for

5.2 Experimental setup

In this section, we evaluate the performance of LODAC.

First, our method is compared with LOMPO [19], an offline high-dimensional risk-free algorithm. Then, we build a version of LODAC where the actor and the critic

are trained using O-RAAC [54]. We denote it LODAC-O. Moreover, it is also possible to use a risk-free offline RL in the latent space. Thus, a risk-free policy is also trained in the latent space using COMBO [23].

These algorithms are evaluated on the standard walker walk task from the DeepMind Control suite [8], but here we learn directly from the pixels of dimensions $3 \times 64 \times 64$. As a standard practice, each action is repeated two times on the ground environment and episodes of length 1000 are used. These algorithms are tested on three different datasets: expert, medium and expert-replay. Each dataset consists of 100K transitions steps. More precisely, the following datasets are built.

- **Expert** For the expert dataset, actions are chosen according to an expert policy that has been trained online, in a risk-free environment using SAC for 500K training steps. The states used for this training are the classical states provided by DeepMind Control suite.
- **Medium** In this dataset, actions are chosen according to a policy that has been trained using the same method as above. However, here, the training was stopped when the policy achieves about half the performance of the expert policy.
- **Expert_replay** The expert_replay dataset consists of episode that were sampled from the expert policy during the training.

The setup presented above allows to test our approach on deterministic environments.

However, we would like to evaluate our algorithm on stochastic environments too. To achieve this, we use the same datasets, but we modify the reward using the following formula

$$r_t \sim \left(r(s_t, a_t) - \lambda \mathbb{1}_{\{r(s_t,a_t) > \overline{r}\}} \mathcal{B}_{p_0} \right)$$

where r is the classical reward function. $\overline{r}$, λ are hyperparameters and $\mathcal{B}_{p_0}$ is a Bernoulli distribution of parameter p_0. In our experiments, we choose $\lambda = 8$ and $p_0 = 0.1$. Different values of $\overline{r}$ are used for each dataset such that about the half of the states verify $r(s_t, a_t) > \overline{r}$.

Hyperparameters of each model have been manually tuned based on our experiments and hyperparameters used in previous papers. However, due to the substantial time required to train these models (specifically LODAC and LODAC-O), we were unable to test a wide range of hyperparameter combinations.

Algorithms have been tested using the following procedure. First of all, to avoid excessive computation time and for a more accurate comparison, we use the same latent variable model for all algorithms.

We evaluate each algorithm using 100 episodes, reporting the mean and CVaR_α of the returns. LODAC and LODAC-O are trained to minimize $\text{CVaR}_{0.7}$. LOMPO and COMBO are trained to maximize the return. We run 4 different random seeds. As introduced in Fu et al. [74], we use the normalized score to compare our algorithms. Specifically, a score of 0 corresponds to a fully random policy and a score of 100 corresponds to an expert policy on the deterministic task. However, and as suggested in Agarwal et al. [75], instead of taking the mean of the results, we consider the interquartile means (IQM).

5.3 Results discussion

In this section, we discuss the results of our experiments, which are presented in Table 1 for the stochastic environment and in Table 2 for the deterministic environment. We bold the highest score across all methods. Complete results of all different runs can be found in Appendix A.

Stochastic environment The first general observation is that LODAC and LODAC-O generally outperform risk-free algorithms. The only exception is with the expert_replay dataset where LODAC-O provides worse results. This is not really surprising since actors trained with O-RAAC contains an imitation component, and obviously, the imitation agent provides really poor performance on this dataset. LODAC provides really interesting results as it significantly outperforms risk-free algorithm in terms of $\text{CVaR}_{0.7}$ and return on all datasets. Moreover, it provides better results than LODAC-O on the medium and the expert_replay dataset, while achieving comparable result on the expert dataset. A final observation is that risk-free

Table 1 Performances for the stochastic offline high-dimensional walker_walk task on expert, medium and expert_replay datasets

Dataset type	Algorithm	Mean	$\text{CVaR}_{0.7}$
Expert	LOMPO	8.67	1.92
Expert	COMBO	10.96	3.72
Expert	LODAC-O	18.54	**4.92**
Expert	LODAC	**18.70**	4.32
Medium	LOMPO	19.03	13.31
Medium	COMBO	16.25	11.18
Medium	LODAC-O	21.31	14.99
Medium	LODAC	**27.85**	**21.37**
Expert_replay	LOMPO	23.22	16.03
Expert_replay	COMBO	20.01	14.12
Expert_replay	LODAC-O	5.73	0.28
Expert_replay	LODAC	**39.4**	**30.14**

We compare the mean and the $\text{CVaR}_{0.7}$ of the returns. LODAC outperforms other algorithms on the medium and expert_replay datasets while achieving comparable results on the expert dataset. We bold the highest score across all methods

Table 2 Performances for the deterministic offline high-dimensional walker_walk task on expert, medium and expert_replay datasets

Dataset type	Algorithm	Mean	$CVaR_{0.7}$
Expert	LOMPO	14.56	0.99
Expert	COMBO	**16.74**	1.97
Expert	LODAC-O	13.25	**2.79**
Expert	LODAC	16.03	0.0
Medium	LOMPO	**52.2**	**42.56**
Medium	COMBO	43.71	25.89
Medium	LODAC-O	37.33	31.12
Medium	LODAC	33.08	5.79
Expert_replay	LOMPO	55.86	**35.10**
Expert_replay	COMBO	**62.88**	27.69
Expert_replay	LODAC-O	15.81	2.90
Expert_replay	LODAC	48.01	27.25

We compare the mean and the $CVaR_{0.7}$ of the returns. We bold the highest score across all methods

RL policies provide generally bad performances on this stochastic environment.

Deterministic environment First, a drop of performance between the deterministic and the stochastic environment can be noticed. This is not surprising, as stochastic environments are generally more challenging than deterministic ones. However, this change appears to affect risk-free algorithms more significantly than LODAC-O or LODAC. Indeed, we get a difference of more than 27% with LOMPO and COMBO on the medium and expert_replay dataset in terms of return between the deterministic and the stochastic environment. This difference even reached 42% for COMBO on the expert_replay dataset. For LODAC-O and for the same tasks, we obtain a deterioration of less than 17%. In contrast to risk-free methods, and in a lesser degree LODAC-O, adding stochasticity to the dataset does not seem to have a significant impact on the performance of LODAC. Indeed, with this algorithm we observe a drop in performance of less than 9%. For the medium dataset, a difference of only 5.23% can even be noticed.

6 Conclusion

While offline RL appears to be an interesting paradigm for real-world applications, many of these real-world applications are high-dimensional and stochastic. However, current high-dimensional offline RL algorithms are trained and tested in deterministic environments. Our empirical results suggest that adding stochasticity to the training dataset significantly decreases the performance of high-dimensional risk-free offline RL algorithms.

Based on this observation, we develop LODAC. LODAC can be used to minimize various risk measures like Conditional Value-at-Risk. Our theoretical considerations in Sect. 4 show that our algorithm relies on a strong theoretical foundation. Finally, the use of LODAC to minimize CVaR empirically outperforms previous algorithms in term of CVaR and return on stochastic high-dimensional environments.

Appendix A Experimental results

In this section, we present our complete empirical results. More precisely, you can find all results of the deterministic environment in Table 3, and all results of the stochastic environment can be found in Table 4.

Table 3 Complete results for the deterministic offline high-dimensional walker_walk task on expert, medium and expert_replay datasets

	Expert		Medium		Expert_replay	
	Mean	$CVaR_{0.7}$	Mean	$CVaR_{0.7}$	Mean	$CVaR_{0.7}$
LOMPO	174	65	419	298	410	274
	177	98	583	501	523	341
	71	26	515	437	601	411
	193	37	527	428	586	387
COMBO	217	62	345	263	662	395
	190	58	458	296	742	501
	201	95	554	476	576	197
	38	21	428	246	359	160
LODAC-O	149	68	385	334	107	59
	178	44	428	357	194	104
	149	73	381	309	224	51
	186	67	384	321	180	78
LODAC	265	43	628	458	460	271
	103	31	416	52	505	313
	306	65	275	134	421	193
	113	34	194	56	509	313

We compare the mean and the $CVaR_{0.7}$ of the returns

Table 4 Complete results for the stochastic offline high-dimensional walker_walk task on expert, medium and expert_replay datasets

	Expert		Medium		Expert_replay	
	Mean	$CVaR_{0.7}$	Mean	$CVaR_{0.7}$	Mean	$CVaR_{0.7}$
LOMPO	150	62	219	161	258	187
	129	57	227	167	259	201
	114	62	201	141	252	191
	108	46	214	171	238	173
COMBO	169	123	196	153	238	182
	167	101	202	159	171	101
	73	40	179	132	250	192
	118	51	186	136	213	161
LODAC-O	210	110	238	181	60	44
	173	69	241	178	97	43
	214	105	237	190	92	51
	228	69	208	153	118	45
LODAC	303	93	299	244	406	326
	112	23	287	235	391	279
	227	87	299	241	413	336
	200	76	296	228	401	311

We compare the mean and the $CVaR_{0.7}$ of the returns

Acknowledgements We would like to thank the emergency medical services dispatch center from the Centre Hospitalier Universitaire Vaudois (CHUV) and more specifically Dr. Fabrice Dami for interesting discussions, that hat highlighted the necessity to develop high-dimensional risk-sensitive RL algorithms.

Funding Open access funding provided by University of Applied Sciences and Arts Western Switzerland (HES-SO). This work was funded by Interreg France-Suisse (project SIA-REMU) (https://www.interreg-francesuisse.eu/).

Data availability statement The datasets generated during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interest to disclose.

References

1. Hussein A, Gaber MM, Elyan E, Jayne C (2017) Imitation learning: a survey of learning methods. ACM Comput Surv (CSUR) 50(2):1–35. https://doi.org/10.1145/3054912
2. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M et al (2016) Mastering the game of go with deep neural networks and tree search. Nature 529(7587):484–489. https://doi.org/10.1038/nature16961
3. Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T et al (2018) A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science 362(6419):1140–1144. https://doi.org/10.1126/science.aar6404
4. Haarnoja T, Zhou A, Abbeel P, Levine S (2018) Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning, pp 1861–1870. PMLR
5. Johannink T, Bahl S, Nair A, Luo J, Kumar A, Loskyll M, Ojea JA, Solowjow E, Levine S (2019) Residual reinforcement learning for robot control. In: 2019 international conference on robotics and automation (ICRA), pp 6023–6029. https://doi.org/10.1109/ICRA.2019.8794127. IEEE
6. Wang L, Zhang W, He X, Zha H (2018) Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 2447–2456. https://doi.org/10.1145/3219819.3219961
7. Yu C, Liu J, Nemati S, Yin G (2021) Reinforcement learning in healthcare: a survey. ACM Comput Surv (CSUR) 55(1):1–36. https://doi.org/10.1145/3477600
8. Tassa Y, Doron Y, Muldal A, Erez T, Li Y, Casas DL, Budden D, Abdolmaleki A, Merel J, Lefrancq A et al (2018) Deepmind control suite. arXiv:1801.00690. https://doi.org/10.48550/arXiv.1801.00690
9. Prudencio RF, Maximo MR, Colombini EL (2023) A survey on offline reinforcement learning: taxonomy, review, and open problems. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2023.3250269
10. Liu M, Zhao H, Yang Z, Shen J, Zhang W, Zhao L, Liu T-Y (2021) Curriculum offline imitating learning. Adv Neural Inf Process Syst 3:4
11. Kostrikov I, Nair A, Levine S (2021) Offline reinforcement learning with implicit q-learning. In: Deep RL workshop NeurIPS 2021
12. Xu H, Jiang L, Jianxiong L, Zhan X (2022) A policy-guided imitation approach for offline reinforcement learning. Adv Neural Inf Process Syst 35:4085–4098
13. Xu H, Jiang L, Li J, Yang Z, Wang Z, Chan VWK, Zhan X (2022) Offline rl with no ood actions: In-sample learning via implicit value regularization. In: The eleventh international conference on learning representations
14. Snell CV, Kostrikov I, Su Y, Yang S, Levine S (2022) Offline rl for natural language generation with implicit language q learning. In: The eleventh international conference on learning representations
15. Zheng Q, Henaff M, Amos B, Grover A (2023) Semi-supervised offline reinforcement learning with action-free trajectories. In: International conference on machine learning, pp 42339–42362. PMLR
16. Kumar A, Fu J, Soh M, Tucker G, Levine S (2019) Stabilizing off-policy q-learning via bootstrapping error reduction. In: Advances in neural information processing systems, vol 32

17. Liu Y, Swaminathan A, Agarwal A, Brunskill E (2020) Off-policy policy gradient with stationary distribution correction. In: Uncertainty in artificial intelligence, pp. 1180–1190. PMLR
18. Rashidinejad P, Zhu H, Yang K, Russell S, Jiao J (2022) Optimal conservative offline rl with general function approximation via augmented Lagrangian. In: The eleventh international conference on learning representations
19. Rafailov R, Yu T, Rajeswaran A, Finn C (2021) Offline reinforcement learning from images with latent space models. In: Learning for dynamics and control, pp 1154–1168. PMLR
20. Argenson A, Dulac-Arnold G (2020) Model-based offline planning. arXiv:2008.05556. https://doi.org/10.48550/arXiv.2008.05556
21. Hong Z-W, Agrawal P, Combes RT, Laroche R (2022) Harnessing mixed offline reinforcement learning datasets via trajectory weighting. In: The eleventh international conference on learning representations
22. Kumar A, Zhou A, Tucker G, Levine S (2020) Conservative q-learning for offline reinforcement learning. Adv Neural Inf Process Syst 33:1179–1191
23. Yu T, Kumar A, Rafailov R, Rajeswaran A, Levine S, Finn C (2021) Combo: conservative offline model-based policy optimization. Adv Neural Inf Process Syst 34:28954–28967
24. Shi L, Li G, Wei Y, Chen Y, Chi Y (2022) Pessimistic q-learning for offline reinforcement learning: towards optimal sample complexity. In: International conference on machine learning, pp 19967–20025. PMLR
25. Ha D, Schmidhuber J (2018) World models. arXiv:1803.10122. https://doi.org/10.48550/arXiv.1803.10122
26. Chen AS, Nam H, Nair S, Finn C (2021) Batch exploration with examples for scalable robotic reinforcement learning. IEEE Robot Autom Lett 6(3):4401–4408. https://doi.org/10.1109/LRA.2021.3068655
27. Driess D, Schubert I, Florence P, Li Y, Toussaint M (2022) Reinforcement learning with neural radiance fields. Adv Neural Inf Process Syst 35:16931–16945
28. Yi Q, Zhang R, Peng S, Guo J, Hu X, Du Z, Guo Q, Chen R, Li L, Chen Y (2023) Learning controllable elements oriented representations for reinforcement learning. Neurocomputing. https://doi.org/10.1016/j.neucom.2023.126455
29. Cui B, Chow Y, Ghavamzadeh M (2020) Control-aware representations for model-based reinforcement learning. arXiv:2006.13408. https://doi.org/10.48550/arXiv.2006.13408
30. Laskin M, Srinivas A, Abbeel P (2020) Curl: cunsupervised representations for reinforcement learning. In: International conference on machine learning, pp 5639–5650. PMLR
31. Ma G, Wang Z, Yuan Z, Wang X, Yuan B, Tao D (2022) A comprehensive survey of data augmentation in visual reinforcement learning. arXiv:2210.04561
32. Nair AV, Pong V, Dalal M, Bahl S, Lin S, Levine S (2018) Visual reinforcement learning with imagined goals. In: Advances in neural information processing systems, vol 31
33. Gelada C, Kumar S, Buckman J, Nachum O, Bellemare MG (2019) Deepmdp: learning continuous latent space models for representation learning. In: International conference on machine learning, pp 2170–2179. PMLR
34. Hafner D, Lillicrap T, Ba J, Norouzi M (2019) Dream to control: learning behaviors by latent imagination. In: International conference on learning representations
35. Hafez MB, Weber C, Kerzel M, Wermter S (2019) Efficient intrinsically motivated robotic grasping with learning-adaptive imagination in latent space. In: 2019 Joint IEEE 9th international conference on development and learning and epigenetic robotics (Icdl-Epirob). IEEE, pp 1–7. https://doi.org/10.1109/DEVLRN.2019.8850723
36. Hafez MB, Weber C, Kerzel M, Wermter S (2020) Improving robot dual-system motor learning with intrinsically motivated meta-control and latent-space experience imagination. Robot Auton Syst 133:103630. https://doi.org/10.1016/j.robot.2020.103630
37. Han D, Doya K, Tani J (2019) Variational recurrent models for solving partially observable control tasks. arXiv:1912.10703. https://doi.org/10.48550/arXiv.1912.10703
38. Lee AX, Nagabandi A, Abbeel P, Levine S (2020) Stochastic latent actor-critic: deep reinforcement learning with a latent variable model. Adv Neural Inf Process Syst 33:741–752
39. Garcıa J, Fernández F (2015) A comprehensive survey on safe reinforcement learning. J Mach Learn Res 16(1):1437–1480
40. Fei Y, Yang Z, Chen Y, Wang Z (2021) Exponential bellman equation and improved regret bounds for risk-sensitive reinforcement learning. In: Advances in neural information processing systems, vol 34
41. Zhang K, Zhang X, Hu B, Basar T (2021) Derivative-free policy optimization for linear risk-sensitive and robust control design: implicit regularization and sample complexity. In: Advances in neural information processing systems, vol 34
42. Greenberg I, Chow Y, Ghavamzadeh M, Mannor S (2022) Efficient risk-averse reinforcement learning. Adv Neural Inf Process Syst 35:32639–32652
43. Théate T, Ernst D (2023) Risk-sensitive policy with distributional reinforcement learning. Algorithms 16(7):325. https://doi.org/10.3390/a16070325
44. Rabin M (2000) Risk aversion and expected-utility theory: a calibration theorem. Econometrica 68(5):1281–1292
45. Tversky A, Kahneman D (1992) Advances in prospect theory: cumulative representation of uncertainty. J Risk Uncertain 5(4):297–323
46. Rockafellar RT, Uryasev S (2002) Conditional value-at-risk for general loss distributions. J Bank Finance 26(7):1443–1471
47. Sarykalin S, Serraino G, Uryasev S (2008) Value-at-risk vs. conditional value-at-risk in risk management and optimization. In: State-of-the-art decision-making tools in the information-intensive age. Informs, Maryland, pp 270–294. https://doi.org/10.1287/educ.1080.0052
48. Artzner P, Delbaen F, Eber J-M, Heath D (1999) Coherent measures of risk. Math Finance 9(3):203–228
49. Pinto L, Davidson J, Sukthankar R, Gupta A (2017) Robust adversarial reinforcement learning. In: International conference on machine learning. PMLR, pp 2817–2826
50. Chow Y, Ghavamzadeh M (2014) Algorithms for cvar optimization in mdps. In: Advances in neural information processing systems, vol 27
51. Chow Y, Tamar A, Mannor S, Pavone M (2015) Risk-sensitive and robust decision-making: a cvar optimization approach. In: Advances in neural information processing systems, vol 28
52. Ying C, Zhou X, Su H, Yan D, Chen N, Zhu J (2022) Towards safe reinforcement learning via constraining conditional value-at-risk. arXiv:2206.04436. https://doi.org/10.48550/arXiv.2206.04436
53. Ma X, Xia L, Zhou Z, Yang J, Zhao Q (2020) Dsac: distributional actor critic for risk-sensitive reinforcement learning. arXiv:2004.14547. https://doi.org/10.48550/arXiv.2004.14547
54. Armengol Urpí N, Curi S, Krause A (2021) Risk-averse offline reinforcement learning. In: International conference on learning representations (ICLR 2021). OpenReview
55. Fujimoto S, Meger D, Precup D (2019) Off-policy deep reinforcement learning without exploration. In: International conference on machine learning. PMLR, pp 2052–2062
56. Ma Y, Jayaraman D, Bastani O (2021) Conservative offline distributional reinforcement learning. In: Advances in neural information processing systems, vol 34

57. Rockafellar RT (2007) Coherent approaches to risk in optimization under uncertainty. In: OR tools and applications: glimpses of future technologies. Informs, Maryland, USA, pp 38–61
58. Delbaen F (2002) Coherent risk measures on general probability spaces. In: Advances in finance and stochastics. Springer, Berlin, pp 1–37. https://doi.org/10.1007/978-3-662-04790-3_1
59. Rockafellar RT, Uryasev SP, Zabarankin M (2002) Deviation measures in risk analysis and optimization. University of Florida, Department of Industrial & Systems Engineering Working Paper (2002–7)
60. Wang SS (2000) A class of distortion operators for pricing financial and insurance risks. J Risk Insur. https://doi.org/10.2307/253675
61. Ahmadi-Javid A (2011) An information-theoretic approach to constructing coherent risk measures. In: 2011 IEEE international symposium on information theory proceedings. IEEE, pp 2125–2127. https://doi.org/10.1109/ISIT.2011.6033932
62. Rockafellar RT, Uryasev S et al (2000) Optimization of conditional value-at-risk. J Risk 2:21–42
63. Rockafellar RT, Uryasev S, Zabarankin M (2006) Generalized deviations in risk analysis. Finance Stoch 10(1):51–74. https://doi.org/10.1007/s00780-005-0165-8
64. Levine S (2018) Reinforcement learning and control as probabilistic inference: Tutorial and review. arXiv:1805.00909. https://doi.org/10.48550/arXiv.1805.00909
65. Kim S, Jo H, Song J-B (2022) Object manipulation system based on image-based reinforcement learning. Intell Serv Robot. https://doi.org/10.1007/s11370-021-00402-6
66. Hafner D, Lillicrap T, Fischer I, Villegas R, Ha D, Lee H, Davidson J (2019) Learning latent dynamics for planning from pixels. In: International conference on machine learning. PMLR, pp 2555–2565
67. Odaibo S (2019) Tutorial: deriving the standard variational autoencoder (vae) loss function. arXiv:1907.08956. https://doi.org/10.48550/arXiv.1907.08956
68. Huber PJ (1964) Robust estimation of a location parameter. Ann Math Stat 35:73–101
69. Acerbi C (2002) Spectral measures of risk: a coherent representation of subjective risk aversion. J Bank Finance 26(7):1505–1518. https://doi.org/10.1016/S0378-4266(02)00281-9
70. Duan J, Guan Y, Li SE, Ren Y, Sun Q, Cheng B (2021) Distributional actor-critic: off-policy reinforcement learning for addressing value estimation errors. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2021.3082568
71. Dabney W, Ostrovski G, Silver D, Munos R (2018) Implicit quantile networks for distributional reinforcement learning. In: International conference on machine learning. PMLR, pp 1096–1105
72. Ba JL, Kiros JR, Hinton GE (2016) Layer normalization. arXiv:1607.06450
73. Fujimoto S, Hoof H, Meger D (2018) Addressing function approximation error in actor-critic methods. In: International conference on machine learning. PMLR, pp 1587–1596
74. Fu J, Kumar A, Nachum O, Tucker G, Levine S (2020) D4rl: datasets for deep data-driven reinforcement learning. arXiv:2004.07219. https://doi.org/10.48550/arXiv.2004.07219
75. Agarwal R, Schwarzer M, Castro PS, Courville AC, Bellemare M (2021) Deep reinforcement learning at the edge of the statistical precipice. Adv Neural Inf Process Syst 34:29304–29320