A Simple Mixture Policy Parameterization for Improving Sample Efficiency of CVaR Optimization

Yudong Luo[1,4], **Yangchen Pan**[2], **Han Wang**[3], **Philip Torr**[2], **Pascal Poupart**[1,4]
[1]University of Waterloo, [2]University of Oxford, [3]University of Alberta, [4]Vector Institute

Abstract

Reinforcement learning algorithms utilizing policy gradients (PG) to optimize Conditional Value at Risk (CVaR) face significant challenges with sample inefficiency, hindering their practical applications. This inefficiency stems from two main facts: a focus on tail-end performance that overlooks many sampled trajectories, and the potential of gradient vanishing when the lower tail of the return distribution is overly flat. To address these challenges, we propose a simple mixture policy parameterization. This method integrates a risk-neutral policy with an adjustable policy to form a risk-averse policy. By employing this strategy, all collected trajectories can be utilized for policy updating, and the issue of vanishing gradients is counteracted by stimulating higher returns through the risk-neutral component, thus lifting the tail and preventing flatness. Our empirical study reveals that this mixture parameterization is uniquely effective across a variety of benchmark domains. Specifically, it excels in identifying risk-averse CVaR policies in some Mujoco environments where the traditional CVaR-PG fails to learn a reasonable policy.

1 Introduction

Avoiding risks is a practical consideration in real world applications, inspiring risk-averse reinforcement learning (RL). Risk-averse RL involves optimizing some risk measures of the return random variable. Many risk measures have been studied, for instance, variance (Tamar et al., 2012; La & Ghavamzadeh, 2013), exponential utility functions (Borkar, 2002; Fei et al., 2021), Value at Risk (VaR) (Chow et al., 2018; Jung et al., 2022), and Conditional VaR (CVaR) (Tamar et al., 2015; Lim & Malik, 2022). We focus on CVaR in this work, which emphasizes the worst case outcome of a policy's return. Intuitively, CVaR measures the expected return below a specific quantile level α, termed the risk level. This kind of risk is also known as the tail risk measure (Liu & Wang, 2021), since only the tail of a distribution is considered, and CVaR is more often preferred than VaR because it is coherent (Delbaen & Biagini, 2000).

Among the existing CVaR algorithms in RL (Tamar et al., 2015; Chow et al., 2018; Tang et al., 2019; Yang et al., 2021; Ying et al., 2022), policy gradient (PG) is a common choice. CVaR-PG samples a batch of N trajectories and maximizes the mean return of the αN trajectories with worst returns (Tamar et al., 2015). This approach suffers from sample inefficiency due to two major facts (Greenberg et al., 2022): 1) $1-\alpha$ portion of sampled trajectories are discarded; 2) gradients vanish when the tail of the return quantile function is overly flat, which is discussed later in Sec. 3.1. Another line of research on optimizing CVaR is based on distributional RL (Bellemare et al., 2017), e.g., Dabney et al. (2018a); Tang et al. (2019); Keramati et al. (2020). However, due to the time-inconsistency of the risk, the objectives of some approaches differ from maximizing the α-CVaR of the total return, while the behavior of some others are not well-understood yet (Lim & Malik, 2022).

In this paper, we focus on the policy gradient approach and propose a simple mixture policy parameterization to improve sample efficiency. Our key insight is that in many real-world risk-sensitive domains, the agent may only need to perform risk-averse actions in a subset of states, e.g., related to risky regions, and behave akin to a risk-neutral agent in other states. We give an example in

Sec. 3.3. This motivates representing a risk-averse policy via integrating a risk-neutral policy and an adjustable component. With this parameterization, all collected trajectories can be used to update the policy under the mixture framework, and gradient vanishing is counteracted by stimulating higher returns with the help of its risk-neutral component, thus lifting the tail and preventing flatness of the quantile function. To demonstrate the effectiveness of our method in learning risk-averse policies, we modify several domains (Maze (Greenberg et al., 2022), Lunar Lander(Brockman et al., 2016), Mujoco (Todorov et al., 2012)) where risk-aversion can be clearly verified. We empirically show that our method can learn a risk-averse policy when others fail to learn a reasonable policy.

Contributions. To the best of our knowledge, a generally applicable approach to improve the sample efficiency of CVaR-PG algorithms remains unclear. In summary, our work provides 1) insights into a novel perspective on scenarios where risk-averse behavior is required only in a subset of states; 2) a simple mixture policy parameterization to address sample inefficiency. Notably, our algorithm, in certain Mujoco domains, marks a pioneering advancement in CVaR optimization.

2 Background: CVaR Optimization in RL

In standard RL settings, agent-environment interactions are modeled as a Markov Decision Process (MDP), represented as a tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \mu_0, \gamma \rangle$ (Puterman, 2014). $\mathcal{S}$ and $\mathcal{A}$ denote state and action spaces. $P(\cdot|s,a)$ defines the transition. R is the state and action dependent reward. μ_0 is the initial state distribution, and $\gamma \in (0,1]$ is a discount factor. An agent takes actions according to its policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, +\infty)$. The return at time step t is defined as $G_t^\pi = \sum_{i=0}^{\infty} \gamma^i R(s_{t+i}, a_{t+i})$. Thus, G_0^π is the random variable indicating the total return starting from the initial state following π. CVaR-based risk-averse RL avoids catastrophic outcomes by optimizing the tail risk measure of G_0^π, e.g., CVaR, instead of $\max_\pi \mathbb{E}[G_0^\pi]$ as done in risk-neutral RL.

2.1 Problem Formulation

Let Z be a bounded random variable with cumulative distribution function $F_Z(z) = \mathcal{P}(Z \leq z)$. Denote the α-quantile as $q_\alpha(Z) = \min\{z | F_Z(z) \geq \alpha\}, \alpha \in (0,1]$. The CVaR at confidence level α is given by (Rockafellar et al., 2000)

$$\text{CVaR}_\alpha(Z) = \frac{1}{\alpha} \int_0^\alpha q_\beta(Z) d\beta \tag{1}$$

When $\alpha \to 1$, $\text{CVaR}_\alpha(Z)$ becomes $\mathbb{E}[Z]$. If Z has a continuous distribution, $\text{CVaR}_\alpha(Z)$ is more intuitively expressed as $\text{CVaR}_\alpha(Z) = \mathbb{E}[Z | Z \leq q_\alpha(Z)]$. Thus, $\text{CVaR}_\alpha(Z)$ can be interpreted as the expected value of the α-portion of the left tail of the distribution of Z. Another way to define $\text{CVaR}_\alpha(Z)$ is (Rockafellar et al., 2000)

$$\text{CVaR}_\alpha(Z) = \max_{k \in \mathbb{R}} k - \frac{1}{\alpha} \mathbb{E}[(k - Z)^+] \tag{2}$$

where $(x)^+ = \max\{x, 0\}$, and the maximum is always attained at $k = q_\alpha(Z)$ as a by product.

In this paper, we consider the problem of maximizing the CVaR of total return G_0^π given a confidence level α (we consider small α in practice) (Tamar et al., 2015), i.e.,

$$\max_\pi \text{CVaR}_\alpha(G_0^\pi) \tag{3}$$

Remark. Some works optimize the CVaR term plus the mean term or treat CVaR as a constraint (Chow et al., 2018; Yang et al., 2021; Ying et al., 2022), which differ from the problem in Eq. 3. In addition, the risk defined on the total return (Eq. 3) is known as the static risk. Another line of research on CVaR works on dynamic risk (Ruszczyński, 2010; Huang et al., 2021; Du et al., 2023), where risk is recursively computed at each time step. The comparison between static and dynamic CVaR is discussed, e.g., in Lim & Malik (2022).

2.2 CVaR Policy Gradient (CVaR-PG)

Consider π is parameterized by θ. Under some mild assumptions, the gradient of Eq. 3 w.r.t. θ can be estimated by sampling trajectories $\{\tau_i\}_{i=1}^N$ from the environment using π_θ (Tamar et al., 2015).

$$\nabla_\theta \text{CVaR}_\alpha(G_0^{\pi_\theta}) \simeq \frac{1}{\alpha N} \sum_{i=1}^{N} \mathbb{I}_{\{R(\tau_i) \leq \hat{q}_\alpha\}} (R(\tau_i) - \hat{q}_\alpha) \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \tag{4}$$

where $R(\tau)$ represents the total return of trajectory τ, $\hat{q}_\alpha$ is the empirical α-quantile estimated from $\{R(\tau_i)\}_{i=1}^N$, and T is the maximum trajectory length. This gradient is derived from Eq. 1. For further details, readers can refer to Tamar et al. (2015). Note that computing policy gradient from Eq. 2 is also feasible and results in a similar update as Eq. 4, e.g., see Algo. 1 in Chow et al. (2018).

2.3 Distributional RL with CVaR

Distributional RL (Bellemare et al., 2017) is recently used for CVaR optimization. Since it directly learns a value distribution, the risk metric is easy to compute. Denote the return random variable at the state-action pair (s, a) as $Z^\pi(s, a) = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$, where $s_0 = s$, $a_0 = a$, $s_{t+1} \sim P(\cdot|s_t, a_t)$, and $a_t \sim \pi(\cdot|s_t)$. Then the distributional Bellman equation is given by $Z^\pi(s, a) \overset{D}{=} R + \gamma Z^\pi(S', A')$, with $S' \sim P(\cdot|s, a)$, $A' \sim \pi(\cdot|S')$, and $X \overset{D}{=} Y$ indicates that random variables X and Y follow the same distribution. The well known Q-value can be extracted by $Q^\pi(s, a) = \mathbb{E}[Z^\pi(s, a)]$.

Dabney et al. (2018a); Keramati et al. (2020) propose to select actions according to

$$Z^\pi(s, a) \overset{D}{=} R + \gamma Z^\pi(S', A'), \qquad A' = \arg\max_{a'} \text{CVaR}_\alpha(Z^\pi(S', a')) \tag{5}$$

This strategy always selects actions leading to the largest α-CVaR at the current step and is termed as "Markov action selection strategy" by Lim & Malik (2022). Within the framework of actor critic, a similar way is applied by updating the actor towards the α-CVaR of the critic, e.g., see Tang et al. (2019). However, Lim & Malik (2022) showed this strategy converges to neither static nor dynamic optimal CVaR policies by counterexamples. Thus, it is not consistent with the problem in Eq. 3.

Bäuerle & Ott (2011) simplified Eq. 2 by avoiding optimizing k and fixing it to some constant k_0, resulting in the problem $\max_\pi -\mathbb{E}[(k_0 - G_0^\pi)^+]$. This problem can be modeled by an augmented MDP with new state $\tilde{s} = (s, k) \in \mathcal{S} \times \mathbb{R}$, where k is a moving variable keeping track of the accumulated rewards so far. Lim & Malik (2022) incorporated this perspective with distributional RL by introducing the tracking variable, and proposed a new action selection strategy as

$$Z^\pi(s, a) \overset{D}{=} R + \gamma Z^\pi(S', A'), \qquad A' = \arg\max_{a'} \mathbb{E}[-(\frac{k - R}{\gamma} - Z^\pi(S', a'))^+] \tag{6}$$

where k is the tracking variable at (s, a), and is set to α-CVaR for the initial state. Lim & Malik (2022) showed that the optimal CVaR policy is a fixed point of Eq. 6 if it exists and it is unique. However, when π is not CVaR optimal, its behavior is generally unknown.

2.4 Other CVaR RL Algorithms

There are several other CVaR algorithms in the context of MDPs, where full knowledge of the MDP is required. Thus, they are not applicable to RL problems where transition dynamics are unknown. These works are less relevant to ours and hence we only provide a brief review here. Based on the theory of CVaR decomposition (Pflug & Pichler, 2016), a dynamic programming approach is developed by decomposing the CVaR via its risk envelope (Chow et al., 2015). This approach returns the optimal α-CVaR value for any $\alpha \in (0, 1]$. Recently, Hau et al. (2023) pointed out this method has some flaws in the control setting, and provided counter examples.

3 Mixture Parameterization Policies

In this section, we examine the difficulties inherent in classical CVaR-PG methods. This examination sets the stage for our proposed solution: a mixture parameterization approach.

3.1 Challenges of CVar-PG: low-efficiency gradient estimation

The classical CVaR-PG (Eq. 4) faces two significant challenges that undermine its sample efficiency and practical applicability. Firstly, to emphasize the tail outcomes, a small value of α is chosen.

Consequently, only an α-fraction of the trajectories contribute to the gradient estimation in Eq. 4, leading to the discarding of the majority of trajectories and resulting in low sample efficiency.

Secondly, as identified by Greenberg et al. (2022), a small α also introduces a gradient vanishing issue. This occurs because the term $\mathbb{I}_{\{R(\tau_i)\leq\hat{q}_\alpha\}}(R(\tau_i)-\hat{q}_\alpha)$ can equal zero for any τ_i satisfying $R(\tau_i)\leq\hat{q}_\alpha$, i.e., $R(\tau_i)=\hat{q}_\alpha$ for those trajectories τ_i selected by the indicator function. This issue arises when the left tail of the quantile function is notably flat, meaning that all quantile values below the α-quantile are identical. Such a scenario is particularly likely in environments with a discrete rewards distribution, a fact that is often overlooked when assuming continuous rewards. For illustration, we present the empirical quantile function of G_0^π obtained through Monte Carlo sampling in Fig. 1(c), during the initial training phase with a random policy in a maze environment (detailed in Sec. 3.3 and shown in Fig. 1(a)). In this scenario, if the agent neither reaches the goal nor enters the red state, the resulting trajectories will yield identical low returns, leading to a markedly flat left tail of the quantile function for G_0^π.

To tackle gradient vanishing, Greenberg et al. (2022) proposed curriculum learning by starting from an α close to 1 (risk-neutral) and gradually decreasing α to its target value. To further improve sample efficiency, Greenberg et al. (2022) proposed a sampling method based on cross-entropy to sample high-risk scenarios from the environment. The algorithm is then focused on learning high-risk parts of the environment and thus improving sample efficiency. However, this sampling strategy requires knowledge of the environment dynamics and the ability to control the parameters of the dynamics in ways that are domain specific, which is not realistic for many RL domains.

3.2 Mixture with Risk-neutral Policy

To address the aforementioned challenges, our key observation is that many real-world risk-sensitive applications exhibit a pattern wherein only a subset of states requires risk-averse behavior. In the remaining portion of the state space, the agent can behave akin to a risk-neutral agent. For example, in scenarios with minimal or no other cars on a highway, a driver may simply need to follow the road without slowing down or braking, as long as the vehicle remains under the speed limit. This observation leads us to propose representing the policy as a mixture of a risk-neutral policy and an adjustable component, i.e.,

$$\pi(a|s)=w(s)\pi'(a|s)+(1-w(s))\pi^n(a|s) \tag{7}$$

where $w(s)\in[0,1]$ is the mixture component weight. π^n is the risk neutral policy, and π' is the adjustable policy. At different phases of a task, the agent self-selects the most suitable policies to execute to ensure the overall policy π is risk averse.

It is evident that the proposed parameterization effectively addresses the challenges outlined earlier. Firstly, it allows for the use of all trajectories collected so far to update the risk-neutral policy within the mixture framework. Secondly, the risk-neutral component encourages the agent to venture into areas of high reward, potentially avoiding the flat tail of the return distribution, and hence mitigates the issue of vanishing gradients. We illustrate the advantages in the following example.

3.3 A Motivating Maze Example

Consider a maze domain in Fig. 1(a), which is originally from Greenberg et al. (2022) and slightly modified by Luo et al. (2023). Starting from the bottom left corner, the goal of the agent is to reach the green goal state. The gray color marks the walls. The per-step reward is deterministic (i.e., -1) except for the red state, whose reward distribution is $-1+\mathcal{N}(0,1)\times 30$. The reward for visiting the goal is a positive constant value (i.e., 10). Thus, the shortest path going through the red state towards the goal is the optimal risk-neutral path, while the longer path (shown in white color) is α-CVaR optimal if α is small, though its expected return is slightly lower.

In this domain, suppose we are given the optimal risk neutral policy for each state (which is actually easy to get, e.g., via Q-learning or value iteration (Sutton & Barto, 1998), or even by observing the shortest path), it is easy to see most actions along the white (i.e., risk-averse) path are the same as

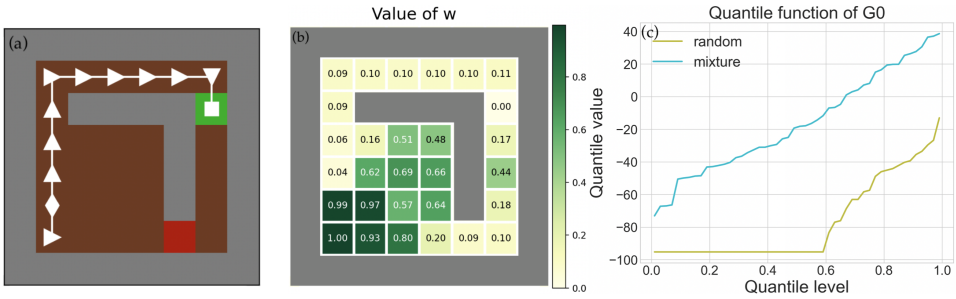

Figure 1: (a) A maze domain with green goal state. The red state returns an uncertain reward (details in Sec. 3.3). Triangle pointers indicate the risk-neutral actions (not unique for the second state). (b) Value of w of Eq. 7 for each state after the mixture policy is updated by CVaR-PG. (c) The empirical quantile function of the total return in maze at an early training stage, if the initial policy is a random and mixture policy.

the risk-neutral policy except the initial state. This means the risk-averse agent only needs to adjust the actions at that state and then follow the optimal risk-neutral policy afterwards. We validate this idea by visualizing the value of $w(s)$ of Eq. 7 in Fig. 1(b), after the mixture policy is trained by CVaR-PG. The value of $w(s)$ represents the probability of choosing π' at each state. Here the risk neutral policy π^n is pre-computed and provided as the softmax of the optimal Q-values (we use temperatures to make the entropy of π^n small). Thus, π' and w are the components that need to be learned by CVaR-PG. As shown in the figure, the probability of choosing π' is only high in the surroundings of the starting state, and the probability of choosing π^n significantly increases after the agent moves far away from the beginning. Also, the empirical quantile function of G_0^π obtained by this mixture policy at the initial training phase is shown in Fig. 1(c). Compared with the randomly initialized policy, the flat tail is eliminated, thereby preventing gradient vanishing.

Remark. This concept, where risk-averse behavior is required only in a subset of states, extends to various fields. For example, in portfolio management, such behavior is crucial only in particular market trends (Ji et al., 2019; Yu et al., 2023), and in healthcare, it is essential only with specific health indicator warnings (Mulligan et al., 2023).

Algorithm 1: Mixture policy for CVaR-PG

Input: risk level α, trajectories sampled per batch N, training steps M, IQL update frequency C

Initialize: policy $\pi_\theta = w_{\theta_2}\pi'_{\theta_1} + (1 - w_{\theta_2})\pi^n_\vartheta$ where $\theta = (\theta_1, \theta_2)$, buffer B, Q-function Q_ϕ (target $Q_{\hat{\phi}}$), value function V_ψ;

for m *in* $1 : M$ **do**

- `// Sample trajectories`
- $\{\tau_i\}_{i=1}^N \leftarrow$ run_episodes(π_θ, N) ;
- Store $\{\tau_i\}_{i=1}^N$ to B;
- `// CVaR PG, i.e., Eq. 4`
- Update θ via CVaR-PG$(\pi_\theta, \{\tau_i\}_{i=1}^N, \alpha)$;
- `// Risk-neutral, e.g., IQL updates`
- **if** m % $C == 0$ **then**
  - Sample $\mathcal{D} = \{(s, a, r, s')\} \sim B$;
  - Update Q_ϕ via Eq. 8;
  - Update V_ψ via Eq. 9;
  - Update π^n_ϑ via Eq. 10;

3.4 Offline RL Risk Neutral Learning

This section explores the process of acquiring a risk-neutral policy under the function approximation setting. We discovered that incorporating a pre-trained deep risk-neutral policy into the mixture policy frequently leads to a suboptimal risk-averse policy, a finding that is further elaborated in Appendix A.3.3.

Observing that the update of CVaR-PG typically involves collecting substantial trajectories, these trajectories naturally constitute an empirical MDP to which an offline RL algorithm can be applied to extract a risk-neutral policy. The field of offline RL has seen rapid advancements in recent years, offering promising solutions for solving the empirical MDP formed from the collected trajectories.

Offline RL attempts to learn an optimal policy from a pre-gathered offline dataset $\mathcal{D} = \{(s, a, s', r)\}_{i=1}^n$, where the learning algorithm is restricted to learning from the samples contained

within $\mathcal{D}$ without any additional interaction with the real environment. One key challenge in offline RL is to not overestimate the action values outside of the dataset (Fujimoto et al., 2019). To address this challenge, there are generally two strategies. The first approach aims to keep the learned policy closely aligned with the dataset's policy by applying some KL constraint, ensuring the learned policy remains within the dataset's support (Peng et al., 2020; Brandfonbrener et al., 2021; Fujimoto & Gu, 2021). The second strategy involves directly optimizing the policy using the samples available in the dataset (Fujimoto et al., 2019; Kostrikov et al., 2022; Xiao et al., 2023).

In our research, we utilize Implicit Q-Learning (IQL) (Kostrikov et al., 2022) for learning risk-neutral policies, chosen for its proven reliability and empirical validation. IQL possesses a Q estimator $Q_\phi(s,a)$, a value estimator $V_\psi(s)$, and a policy $\pi^n_\vartheta(a|s)$. Q-function is updated via minimizing

$$L_Q(\phi) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[(r(s,a) + \gamma V_\psi(s') - Q_\phi(s,a))^2] \tag{8}$$

Value function is updated via expectile regression to avoid overestimation ($Q_{\hat{\phi}}$ is the target function)

$$L_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[L_2^\eta(Q_{\hat{\phi}}(s,a) - V_\psi(s))], \quad L_2^\eta(u) = |\eta - \mathbb{I}_{\{u<0\}}|u^2 \tag{9}$$

Policy is updated by advantage-weighted regression (Peters & Schaal, 2007) with temperature β

$$L_{\pi^n}(\vartheta) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[\exp(\beta(Q_\phi(s,a) - V_\psi(s)))\log\pi^n_\vartheta(a|s)] \tag{10}$$

All the trajectories are stored in a replay buffer for IQL update to learn π^n. In practice, we can perform this update after enough transition data are collected. The overall process of training the mixture policy is described in Algo. 1.

4 Experiments

We modify several domains such that the risk-averse behavior is clear to identify to evaluate the algorithms. We include REINFORCE with baseline method, as a risk-neutral baseline. In more complex domains, we use SAC (Haarnoja et al., 2018) instead.

Baselines. We compare our method with CVaR-PG in Eq. 4 (Tamar et al., 2015), distributional RL with Markov action selection strategy in Eq. 5 (denoted as DRL-mkv), and Lim's action selection strategy in Eq. 6 (Lim & Malik, 2022) (denoted as DRL-lim). In continuous action domains, we adapt DRL-mkv and DRL-lim by DPG (Silver et al., 2014) as done in Tang et al. (2019), see Sec. A.1 for an overview. We use MIX to represent our method. Precomputed π^n in maze is provided to MIX as described in Sec 3.3 since it is easy to get. In other domains, π^n is learned by IQL during training. Please refer to Appendix A for any missing implementation details.

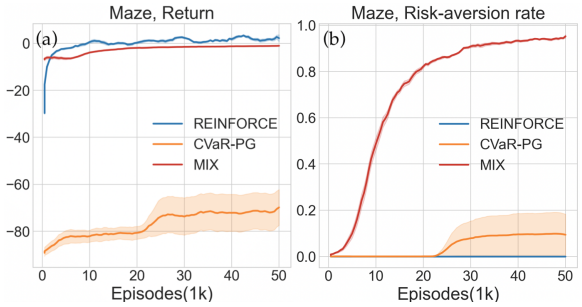

Figure 2: (a) Policy return (y-axsis) and (b) Risk-aversion (long path) rate (y-axsis) v.s. training episodes in Maze. Curves are averaged over 10 seeds with shaded regions indicating standard errors.

Remark. The method in Greenberg et al. (2022) is CVaR-PG with curriculum learning and a special trajectory sampling strategy, which is orthogonal to our approach. It requires to control the environment dynamics, and may not be straightforwardly applicable to most domains discussed here. We compare with it in one domain from Greenberg et al. (2022) in Sec. A.7.

4.1 Tabular case: Maze Problem

This domain is modified from Greenberg et al. (2022) that was previously described in Sec. 3.3. The maximum episode length is 100. CVaR $\alpha = 0.1$. REINFORCE, CVaR-PG, and MIX collect $N = 50$ episodes before updating the policy. Here we report the rate of choosing the long path during

training in Fig. 2(b). Since the policy is non-deterministic, the length of the sampled risk-averse path may not be exactly 11 (the length of the white path in Fig. 1(a)). Here we treat a path as risk-averse if it goes towards the top, reaches the goal, and the path length does not exceed 14.

CVaR-PG fails to learn a reasonable policy even in this simple domain due to gradient vanishing as discussed in Sec. 3.1. We show the gradient norm of CVaR-PG in one run in Fig. 5 in appendix to further illustrate this phenomenon. By initializing MIX with a risk neutral policy, it achieves a relatively high return at the early learning phase, thus potentially avoids gradient vanishing.

4.2 Discrete control: LunarLander

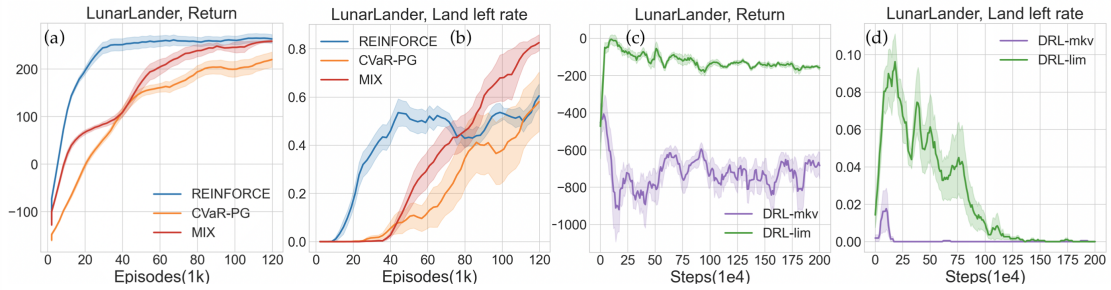

Figure 3: (a,c) Policy return (y-axis), and (b,d) Left-landing rate (i.e., risk-averse landing rate) (y-axis) v.s. training episodes or steps in LunarLander. Curves are averaged over 10 seeds with shaded regions indicating standard errors. For the landing left rate, higher is better.

This domain is taken from OpenAI Gym (Brockman et al., 2016). We refer readers to its official documents for a full description. The goal of the agent is to land on the ground without crashing. We split the ground into left and right parts by the middle line of the landing pad, as shown in Fig. 6 in appendix. If landing on the right, an additional noisy reward sampled from $\mathcal{N}(0, 1)$ times 100 is given. A risk-averse agent should learn to land on the left as much as possible. We set CVaR $\alpha = 0.1$. REINFORCE, CVaR-PG, and MIX collect $N = 30$ episodes before updating the policy. DRL-mkv and DRL-lim are off-policy methods and update policies at each environment step. We train them for 2e6 steps instead of as many episodes as other methods.

We report the left-landing rate of different methods in Fig. 3(b) and (d). Comparing DRL-mkv and DRL-lim against episode-based algorithms is not straightforward within the same figure due to the difference in parameter update frequency. Thus we show them separately. MIX achieves a comparable return with REINFORCE at the end, but shows a clear risk-aversion by landing more on the left. DRL-mkv and DRL-lim can not learn a reasonable policy given the small CVaR α. As mentioned in Section 2.3, they optimize a different objective than CVaR that is not well understood.

4.3 Continuous control: Mujoco

Mujoco (Todorov et al., 2012) is a collection of robotics environments with continuous states and actions in OpenAI Gym (Brockman et al., 2016). Here, we select three domains, namely InvertedPendulum, HalfCheetah, and Ant (disparities in the inherent difficulty of moving backward versus forward are observed in other domains, which likely stems from the intrinsic design of the physics engine's dynamics). Inspired by Malik et al. (2021); Liu et al. (2022), we define the risky region based on the X-position. Specifically, if X-position > 0.04 in InvertedPendulum, X-position < -3 in HalfCheetah and Ant, a zero-mean Gaussian noise is added to the reward ($\mathcal{N}(0, 1) \times 10$ in InvertedPendulum, $\mathcal{N}(0, 1) \times 50$ in HalfCheetah and Ant). To further ensure that agents move both forward and backward with equal preference in terms of expected reward in the two environments, we define the distance-based reward as the difference in distance between the current and previous states from the origin, regardless of the sign of the X-position. Consequently, in InvertedPendulum, a risk-averse agent aims to keep the pendulum balanced while staying out of the noisy region. In HalfCheetah and Ant, a risk-averse agent should learn to move toward the opposite direction of the

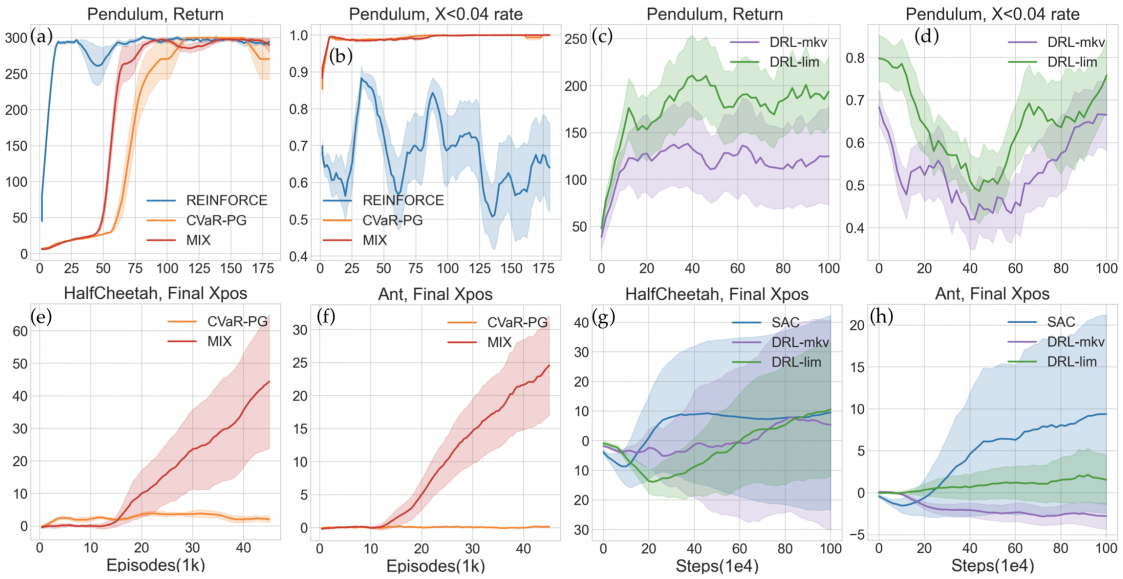

Figure 4: (a, c) Policy return (y-axsis) in InvertedPendulum, (b, d) visiting non-noisy region rate (y-axis) in InvertedPendulum, (e, g) Final X-position (y-axsis) in HalfCheetah, (f, h) Final X-position in Ant (y-axsis) v.s. training episodes or steps in Mujoco. Curves are averaged over 10 seeds with shaded regions indicating standard errors. For the location visiting rate, higher is better.

noisy region. We optimize CVaR $\alpha = 0.2$. REINFORCE still serves as the risk neutral baseline in InvertedPendulum. In HalfCheetah and Ant, we use SAC (Haarnoja et al., 2018) instead, since the vanilla policy gradient is not good at more complex domains. REINFORCE, CVaR-PG, and MIX collect $N = 30$ episodes before updating the policy in InvertedPendulum, and $N = 15$ in HalfCheetah and Ant. DRL-mkv, DRL-lim, and SAC are trained for 1e6 steps.

We report the total return and X<0.04 rate in InvertedPendulum, which are sufficient to reflect the risk-averse behavior of the agent, since the reward is 1 as long as the pendulum is balanced. In HalfCheetah and Ant, we report the final X-position in Fig. 4, as the return can not reflect which direction the agent is moving in the two domains. The policy returns in these two domains are shown in Fig. 9 and 10 in appendix. CVaR-PG achieves a risk-averse policy in InvertedPendulum, i.e., high return and high rate of staying in the non-noisy region. But it fails to learn a reasonable policy in HalfCheetah and Ant, i.e., the final X-position is always close to the origin. MIX learns risk-averse policy by moving away from the noisy region. DRL-mkv and DRL-lim generally do not work well in all three domains since they optimize a different objective than CVaR that is not well understood (see Sec 2.3).

5 Conclusions and Future Work

This paper proposes a mixture policy framework for CVaR-PG. It is motivated to overcome the sample inefficiency of the original CVaR-PG, caused by the waste of most sampled trajectories and gradient vanishing in some domains. We empirically show that our method can succeed when others fail to learn a risk-averse or a reasonable policy by mitigating the sample efficiency issue.

Limitations and future work. We have pinpointed a class of risk-averse RL problems characterized by requiring risk-averse behavior in a subset of states, suitable for our mixture approach. Though this category intuitively covers a broad range of scenarios, situations that do not fit this framework remain unexplored in this paper. Additionally, our method can be potentially integrated with other techniques aimed at enhancing sample efficiency, e.g., Greenberg et al. (2022), given its versatile nature. However, exploring such hybrid methodologies falls outside the scope of our current research. Observing the two limitations, researching algorithms to enhance the sample efficiency for

the broader class of risk-averse problems, as well as possible integration with existing methods to improve sample efficiency, remains valuable for future work.

Broader Impact Statement

This paper presents work whose goal is to advance the risk-averse reinforcement learning. There may be potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Riad Akrour, Davide Tateo, and Jan Peters. Continuous action reinforcement learning from a mixture of interpretable experts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6795–6806, 2021.

Nicole Bäuerle and Jonathan Ott. Markov decision processes with average-value-at-risk criteria. *Mathematical Methods of Operations Research*, 74:361–379, 2011.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 449–458. PMLR, 2017.

Vivek S Borkar. Q-learning for risk-sensitive control. *Mathematics of operations research*, 27(2): 294–311, 2002.

David Brandfonbrener, William F Whitney, Rajesh Ranganath, and Joan Bruna. Offline RL without off-policy evaluation. In *Advances in Neural Information Processing Systems*, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Yunho Choi, Kyungjae Lee, and Songhwai Oh. Distributional deep reinforcement learning with a mixture of gaussians. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9791–9797. IEEE, 2019.

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*, 28, 2015.

Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18 (167):1–51, 2018.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *Proceedings of the International conference on machine learning (ICML)*, pp. 1096–1105. PMLR, 2018a.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 32, 2018b.

Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pp. 273–281. PMLR, 2012.

Freddy Delbaen and Sara Biagini. *Coherent risk measures*. Springer, 2000.

Yihan Du, Siwei Wang, and Longbo Huang. Provably efficient risk-sensitive reinforcement learning: Iterated cvar and worst path. In *Proceedings of the Eleventh International Conference on Learning Representations, ICLR-23*, 2023.

Yingjie Fei, Zhuoran Yang, and Zhaoran Wang. Risk-sensitive reinforcement learning with function approximation: A debiasing approach. In *International Conference on Machine Learning*, pp. 3198–3207. PMLR, 2021.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *International Conference on Machine Learning*, pp. 2052–2062, 2019.

Ido Greenberg, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. Efficient risk-averse reinforcement learning. *Advances in Neural Information Processing Systems*, 35:32639–32652, 2022.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Jia Lin Hau, Erick Delage, Mohammad Ghavamzadeh, and Marek Petrik. On dynamic programming decompositions of static risk measures in markov decision processes. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Audrey Huang, Liu Leqi, Zachary C Lipton, and Kamyar Azizzadenesheli. On the convergence and optimality of policy gradient for markov coherent risk. *arXiv preprint arXiv:2103.02827*, 2021.

Ran Ji, KC Chang, and Zhenlong Jiang. Risk-aversion adjusted portfolio optimization with predictive modeling. In *2019 22th International Conference on Information Fusion (FUSION)*, pp. 1–8. IEEE, 2019.

Whiyoung Jung, Myungsik Cho, Jongeui Park, and Youngchul Sung. Quantile constrained reinforcement learning: A reinforcement learning framework constraining outage probability. *Advances in Neural Information Processing Systems*, 35:6437–6449, 2022.

Ramtin Keramati, Christoph Dann, Alex Tamkin, and Emma Brunskill. Being optimistic to be conservative: Quickly learning a cvar policy. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 4436–4443, 2020.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, pp. 5556–5566. PMLR, 2020.

Prashanth La and Mohammad Ghavamzadeh. Actor-critic algorithms for risk-sensitive mdps. *Advances in neural information processing systems*, 26, 2013.

Shiau Hong Lim and Ilyas Malik. Distributional reinforcement learning for risk-sensitive policies. *Advances in Neural Information Processing Systems*, 35:30977–30989, 2022.

Fangda Liu and Ruodu Wang. A theory for measures of tail risk. *Mathematics of Operations Research*, 46(3):1109–1128, 2021.

Guiliang Liu, Yudong Luo, Ashish Gaurav, Kasra Rezaee, and Pascal Poupart. Benchmarking constraint inference in inverse reinforcement learning. *arXiv preprint arXiv:2206.09670*, 2022.

Yudong Luo, Guiliang Liu, Haonan Duan, Oliver Schulte, and Pascal Poupart. Distributional reinforcement learning with monotonic splines. In *International Conference on Learning Representations*, 2022.

Yudong Luo, Guiliang Liu, Pascal Poupart, and Yangchen Pan. An alternative to variance: Gini deviation for risk-averse policy gradient. *Advances in Neural Information Processing Systems*, 36, 2023.

Shehryar Malik, Usman Anwar, Alireza Aghasi, and Ali Ahmed. Inverse constrained reinforcement learning. In *International conference on machine learning*, pp. 7390–7399. PMLR, 2021.

Karen Mulligan, Drishti Baid, Jason N Doctor, Charles E Phelps, and Darius N Lakdawalla. Risk preferences over health: Empirical estimates and implications for healthcare decision-making. Technical report, National Bureau of Economic Research, 2023.

Takayuki Osa, Akinobu Hayashi, Pranav Deo, Naoki Morihira, and Takahide Yoshiike. Offline reinforcement learning with mixture of deterministic policies. *Transactions on Machine Learning Research*, 2023.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2020.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

Georg Ch Pflug and Alois Pichler. Time-consistent decisions and temporal decomposition of coherent risk functionals. *Mathematics of Operations Research*, 41(2):682–699, 2016.

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

R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

Andrzej Ruszczyński. Risk-averse dynamic programming for markov decision processes. *Mathematical programming*, 125:235–261, 2010.

Tim Seyde, Wilko Schwarting, Igor Gilitschenski, Markus Wulfmeier, and Daniela Rus. Strength through diversity: Robust behavior learning via mixture policies. In *Conference on Robot Learning*, pp. 1144–1155. PMLR, 2022.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. Pmlr, 2014.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

Aviv Tamar, Dotan Di Castro, and Shie Mannor. Policy gradients with variance related risk criteria. In *Proceedings of the twenty-ninth international conference on machine learning*, pp. 387–396, 2012.

Aviv Tamar, Yonatan Glassner, and Shie Mannor. Optimizing the cvar via sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Yichuan Charlie Tang, Jian Zhang, and Ruslan Salakhutdinov. Worst cases policy gradients. *Conference on Robot Learning*, 2019.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Markus Wulfmeier, Abbas Abdolmaleki, Roland Hafner, Jost Tobias Springenberg, Michael Neunert, Tim Hertweck, Thomas Lampe, Noah Siegel, Nicolas Heess, and Martin Riedmiller. Compositional transfer in hierarchical reinforcement learning. *Robotics: Science and Systems*, 2020.

Markus Wulfmeier, Dushyant Rao, Roland Hafner, Thomas Lampe, Abbas Abdolmaleki, Tim Hertweck, Michael Neunert, Dhruva Tirumala, Noah Siegel, Nicolas Heess, et al. Data-efficient hindsight off-policy option learning. In *International Conference on Machine Learning*, pp. 11340–11350. PMLR, 2021.

Chenjun Xiao, Han Wang, Yangchen Pan, Adam White, and Martha White. The in-sample softmax for offline reinforcement learning. *International Conference on Learning Representations*, 2023.

Qisong Yang, Thiago D Simão, Simon H Tindemans, and Matthijs TJ Spaan. Wcsac: Worst-case soft actor critic for safety-constrained reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10639–10646, 2021.

ChengYang Ying, Xinning Zhou, Hang Su, Dong Yan, Ning Chen, and Jun Zhu. Towards safe reinforcement learning via constraining conditional value-at-risk. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 3673–3680, 7 2022.

Shi Yu, Haoran Wang, and Chaosheng Dong. Learning risk preferences from investment portfolios using inverse optimization. *Research in International Business and Finance*, 64:101879, 2023.

Yuguang Yue, Zhendong Wang, and Mingyuan Zhou. Implicit distributional reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7135–7147, 2020.

Shangtong Zhang and Shimon Whiteson. Dac: The double actor-critic architecture for learning options. *Advances in Neural Information Processing Systems*, 32, 2019.

Fan Zhou, Jianing Wang, and Xingdong Feng. Non-crossing quantile regression for distributional reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 15909–15919. Curran Associates, Inc., 2020.

Fan Zhou, Zhoufan Zhu, Qi Kuang, and Liwen Zhang. Non-decreasing quantile function network with efficient exploration for distributional reinforcement learning. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3455–3461. International Joint Conferences on Artificial Intelligence Organization, 2021.

A Experiments Details

A.1 General Descriptions of Different Methods

Monte Carlo methods. Among the methods included in this paper, REINFORCE (Sutton & Barto, 1998), CVaR-PG (Tamar et al., 2015), and the CVaR part of our mixture policy are on-policy policy gradient methods using Monte Carlo sampling. In our implementation, we update the policy after gathering N trajectories.

Time difference methods. DRL-mkv (Dabney et al., 2018a; Keramati et al., 2020), DRL-lim (Lim & Malik, 2022), SAC (Haarnoja et al., 2018) are off-policy time difference methods, i.e., updating policy at each environment step. Continuous action domains are not considered in the original paper of Lim & Malik (2022). We follow Tang et al. (2019) to apply DRL-mkv and DRL-lim to continuous action domains, e.g., for DRL-mkv, the actor is updated via

$$\nabla_\theta J_\alpha = \mathbb{E}_{s,a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\text{CVaR}_\alpha(Z^\pi(s,a))] \tag{11}$$

for DRL-lim, the actor is updated via

$$\nabla_\theta J_\alpha = \mathbb{E}_{s,k,a\sim\pi_\theta}\Big[\nabla_\theta - \log \pi(a|s)\mathbb{E}[(k - Z^\pi(s,a))^+]\Big] \tag{12}$$

where k is the tracking variable at state s.

Both DRL-mkv and DRL-lim are built on top of distributional RL (Bellemare et al., 2017). The most commonly used approach to update distributional value function (critic) is quantile regression (Dabney et al., 2018b;a; Zhou et al., 2020; 2021; Luo et al., 2022). We also adopt quantile regression in our implementation.

Solving the quantile crossing issue. In particular, Zhou et al. (2020) pointed out some previous quantile regression based work, e.g., QR-DQN (Dabney et al., 2018b), IQN (Dabney et al., 2018a) suffered from the quantile crossing issue, i.e., the predicted quantile values do not satisfy the monotonicity of the quantile function. This is shown to hinder policy learning and exploration (Zhou et al., 2020). The monotonicity of the quantile is also important in DRL-mkv and DRL-lim to make sure the estimated quantities, e.g. α-CVaR, are correct. We follow the approach in Yue et al. (2020) by sorting the predicted quantile values to make them non decreasing.

A.2 The Maze Problem

The maze consists of a 6×6 grid. The initial state of the agent is fixed at the bottom left corner. The action space is four (up, down, left, right). The maximum episode length is 100.

Policy function. For CVaR-PG, the policy is represented as

$$\pi_\theta(a|s) = \frac{e^{\zeta(s,a)\cdot\theta}}{\sum_b e^{\zeta(s,b)\cdot\theta}} \tag{13}$$

where $\zeta(s,a)$ is the state-action feature vector, basically a one-hot encoding in our implementation. Thus, the dimension of $\zeta(s,a)$ is $6 \times 6 \times 4$. The derivative of the logarithm is

$$\nabla_\theta \log \pi_\theta(a|s) = \zeta(s,a) - \mathbb{E}_{b\sim\pi_\theta(\cdot|s)}\zeta(s,b) \tag{14}$$

For our mixture policy, the policy parameter θ consists of two parts $\theta = (\theta_1, \theta_2)$, where θ_1 is for the adjustive policy π'_{θ_1}, and θ_2 is for the weight w.

$$\pi_\theta(a|s) = \sigma(\zeta(s,a)\cdot\theta_2)\pi'_{\theta_1}(a|s) + \Big(1 - \sigma(\zeta(s,a)\cdot\theta_2)\Big)\pi^n(a|s) \tag{15}$$

$$\text{with } \pi'_{\theta_1} = \frac{e^{\zeta(s,a)\cdot\theta_1}}{\sum_b e^{\zeta(s,b)\cdot\theta_1}} \tag{16}$$

where $\sigma(\cdot)$ is the sigmoid function. The derivative of the logarithm is

$$\nabla_{\theta_1} \log \pi_\theta(a|s) = \frac{1}{\pi_\theta(a|s)} \sigma(\zeta(s,a) \cdot \theta_2) \pi'_{\theta_1}(a|s) \nabla_{\theta_1} \log \pi'_{\theta_1}(a|s) \tag{17}$$

$$\nabla_{\theta_2} \log \pi_\theta(a|s) = \frac{1}{\pi_\theta(a|s)} (\pi'_{\theta_1}(a|s) - \pi^n(a|s)) \sigma(\zeta(s,a) \cdot \theta_2) \Big(1 - \sigma(\zeta(s,a) \cdot \theta_2)\Big) \zeta(s,a) \tag{18}$$

Value function. The value function of REINFORCE baseline is represented as $V_v(s) = \zeta(s) \cdot v$. Similarly, $\zeta(s)$ is a one-hot encoding.

A.2.1 Learning Parameters

Discount factor $\gamma = 0.999$. Optimizer is stochastic gradient descent (SGD).

REINFORCE: Policy learning rate is 1e-2$\in$ {1e-2, 1e-3, 1e-4}. Value learning rate is 10 times policy learning rate.

CVaR-PG: Learning rate is 1e-2$\in$ {1e-1, 1e-2, 1e-3, 1e-4}.

MIX: Learning rate is 1e-2$\in$ {1e-2, 1e-3, 1e-4}.

A.2.2 Policy Gradient Norm of CVaR-PG

We show the policy gradient norm of CVaR-PG in one run to further demonstrate the gradient vanishing issue in Maze. The norm is computed by `numpy.linalg.norm`. As shown in Fig. 5, in most of the time, the policy gradients are zero.

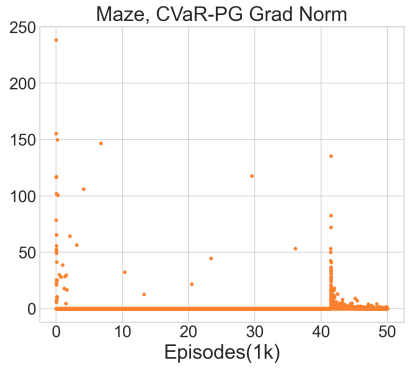

Figure 5: Policy gradient norm (y-axsis) of CVaR-PG in one run in Maze

A.3 LunarLander Discrete

The goal is to land the agent on the ground without crashing. The state dimension is 8. The action dimension is 4. The detailed reward information is available at this webpage [1]. Here, we split the ground into left and right parts by the middle line of the landing pad as shown in Figure 6. If the agent lands on the right part of the ground, an additional noisy reward signal $\mathcal{N}(0,1) \times 100$ is given. The maximum episode length is 500.

Policy function. The policy is a categorical distribution in REINFORCE and CVaR-PG, modeled as a neutral network. Hidden layer: 2. Hidden size: 128. Activation: ReLU. Softmax function is applied to the output to generate categorical probabilities.

The policy of MIX is a weighted summation of π' and π^n with weight w. π' and w are modeled as a neutral network with two output heads. π^n is a separate neutral network. Both of them have: Hidden layer: 2. Hidden size: 128. Activation: ReLU.

[1] https://www.gymlibrary.dev/environments/box2d/lunar_lander/

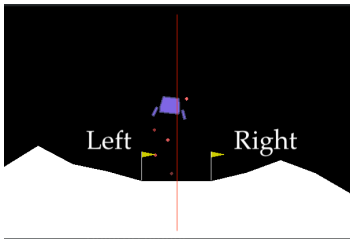

Figure 6: Split the ground of LunarLander into left and right parts by the middle (red) line. If landing on the right, an additional reward smapled from $\mathcal{N}(0, 1)$ times 100 is given.

Value function. For value function in REINFORCE baseline, Q and V function in IQL of MIX. Hidden layer: 2. Hidden size: 128. Activation: ReLU.

For distributional value function in DRL-mkv and DRL-lim. Hidden layer: 2. Hidden size: 128. Activation: ReLU. Quantile size (i.e., final layer output size): 80.

A.3.1 Learning Parameters

Discount factor $\gamma = 0.999$. Optimizer is Adam.

REINFORCE: Policy learning rate is 7e-4$\in$ {1e-3, 7e-4, 3e-4, 1e-4}. Value learning rate is 10 times policy learning rate.

CVaR-PG: Learning rate is 7e-4$\in$ {1e-3, 7e-4, 3e-4, 1e-4}.

MIX: Learning rate for π' and w is 7e-4$\in$ {1e-3, 7e-4, 3e-4, 1e-4}. Learning rate for IQL part (including policy and value functions) is 1e-4$\in$ {3e-4, 1e-4}. IQL update frequency $C = 50$, by sampling 2e5 transitions from buffer. $\eta = 0.8$ in Eq. 9. $\beta = 1$ in Eq. 10.

DRL-mkv: Learning rate is 7e-4$\in$ {1e-3, 7e-4, 3e-4, 1e-4, 7e-5}.

DRL-lim: Learning rate is 1e-4$\in$ {1e-3, 7e-4, 3e-4, 1e-4, 7e-5}.

A.3.2 Performance of the risk neutral component of the mixture policy

In this domain, the risk neutral component π^n of the mixture policy is updated via IQL (Kostrikov et al., 2022). We report the total successful landing rate and left landing rate of π^n during training in Fig. 7. π^n does not demonstrates a preference of landing location, which indicates the risk-aversion is achieved by the mixture policy.

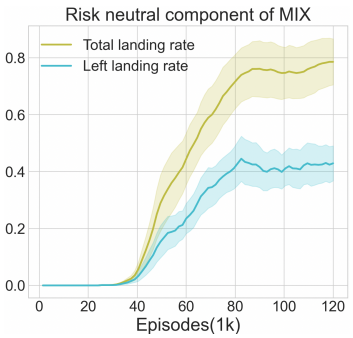

Figure 7: Total landing rate and left landing rate of IQL (y-axsis) in MIX during training. Curves are averaged over 10 seeds with shaded regions indicating standard errors.

A.3.3 Incorporate pre-trained risk-neutral policy

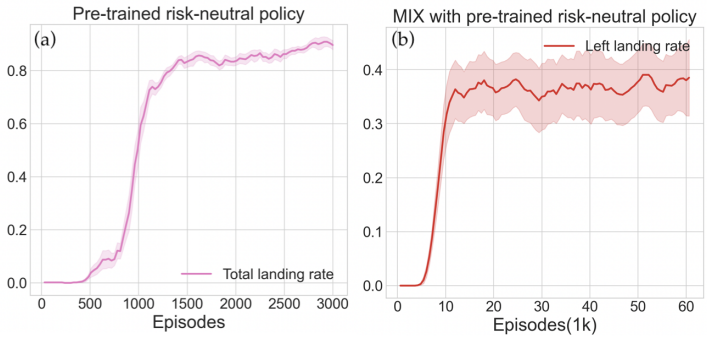

Figure 8: (a) The total successful landing rate (y-axsis) of pre-trained risk-neutral policy. (b) The left landing (i.e., risk-averse) (y-axsis) rate of MIX by incorporating this pre-trained risk-neutral policy. Curves are averaged over 10 seeds with shaded regions indicating standard errors.

A natural question raises that whether we can use a pre-trained risk-neural policy to form the mixture policy, as done in maze (Sec. 3.3), when using deep RL. We conduct the experiments in this LunarLander domain.

Similar as the risk-neutral policy in maze, we represent the risk-neutral policy by the softmax of Q-values with temperature. The Q-values are learned by deep Q-network (DQN). To validate the pre-trained risk-neutral policy performs well, we show its total successful landing rate in Fig. 8(a). The whole training process for MIX is as follows. The first 3k episodes are used to update the risk-neutral policy only (i.e., update DQN), with the remaining part of the MIX policy unchanged. After the first 3k episodes, the risk-neutral policy is fixed, and the remaining part of the MIX policy begins to update. However, as indicated by the left landing rate in Fig. 8(b), MIX leads to a suboptimal risk-averse policy. One possible reason may be when training the deep risk-neutral RL algorithm, the data distribution tends to concentrate on those states in the optimal (or near optimal) trajectories. Thus, the learned value or policy function approximator may not generalize well around the risk-averse path.

A.4 InvertedPendulum

The description of the Mujoco environments can be found at this webpage [2].

The goal is to balance a inverted pendulum on a cart. The state dimension is 4 (X-position is already contained in the observation). The action dimension is 1. Per step reward is 1. If the agent reaches the region X-position > 0.04, and additional noisy reward sampled from $\mathcal{N}(0, 1)$ times 10 is given. The game ends if angle between the pendulum and the cart is greater than 0.2 radian or a maximum episode length 300 is reached.

Policy function. The policy is a normal distribution in REINFORCE and CVaR-PG, modeled as a neutral network. Hidden layer: 2. Hidden size: 128. Activation: ReLU. Tanh is applied to the last layer. The logarithm of standard deviation is an independent trainable parameter.

For MIX, π' and w is a neutral network with two output heads. One for the mean of the normal distribution π', one for w. The logarithm of standard deviation is an independent trainable parameter. π^n is a seperate neutral network as above. Both of them have: Hidden layer: 2. Hidden size: 128. Activation: ReLU. Tanh is applied to the output of the distribution layer.

Value function. For value function in REINFORCE baseline, Q and V function in IQL of MIX. Hidden layer: 2. Hidden size: 128. Activation: ReLU.

[2] https://www.gymlibrary.dev/environments/mujoco/

For distributional value function in DRL-mkv and DRL-lim. Hidden layer: 2. Hidden size: 128. Activation: ReLU. Quantile size (i.e., final layer output size): 80.

A.4.1 Learning Parameters

Discount factor $\gamma = 0.999$. Optimizer is Adam.

REINFORCE: Policy learning rate is 3e-4$\in$ {7e-4, 3e-4, 1e-4}. Value learning rate is 10 times policy learning rate.

CVaR-PG: Learning rate is 3e-4$\in$ {7e-4, 3e-4, 1e-4}.

MIX: Learning rate for π' and w is 3e-4$\in$ {7e-4, 3e-4, 1e-4}. Learning rate for IQL part (including policy and value functions) is 1e-4$\in$ {3e-4, 1e-4}. IQL update frequency $C = 50$, by sampling 1e5 transitions from buffer. $\eta = 0.9$ in Eq. 9. $\beta = 2$ in Eq. 10.

DRL-mkv: Learning rate is 7e-4$\in$ {1e-3, 7e-4, 3e-4, 1e-4, 7e-5}.

DRL-lim: Learning rate is 1e-3$\in$ {1e-3, 7e-4, 3e-4, 1e-4, 7e-5}.

A.5 HalfCheetah

The agent controls a robot with two legs. The state dimension is 18 (add X-position). The action dimension is 6. One part of the reward is determined by the distance covered between the current and the previous time step. Originally, it is positive only when the agent moves toward the forward (right) direction. To encourage the agent to freely move forward (left) and backward (right), we modify this part of the reward to make it positive as long as the agent is moving far from the origin. If the agent reaches the region X-position <-3, an additional noisy reward sampled from $\mathcal{N}(0, 1)$ times 50 is given. The game ends when a maximum episode length 500 is reached.

Policy function. Hidden size: 256. Others are the same as the case in InvertedPendulum.

Value function. Hidden size: 256. Others are the same as the case in InvertedPendulum.

A.5.1 Learning Parameters

Discount factor $\gamma = 0.99$. Optimizer is Adam.

SAC: Learning rate is 3e-4$\in$ {7e-4, 3e-4, 1e-4}.

CVaR-PG: Learning rate is 3e-4$\in$ {7e-4, 3e-4, 1e-4}.

MIX: Learning rate for π' and w is 3e-4$\in$ {7e-4, 3e-4, 1e-4}. Learning rate for IQL part (including policy and value functions) is the same. IQL update frequency $C = 30$, by sampling 2e5 transitions from buffer. $\eta = 0.8$ in Eq. 9. $\beta = 2$ in Eq. 10.

DRL-mkv: Learning rate is 3e-4$\in$ {7e-4, 3e-4, 1e-4, 7e-5}.

DRL-lim: Learning rate is 1e-4$\in$ {7e-4, 3e-4, 1e-4, 7e-5}.

A.5.2 Policy Return in HalfCheetah

The policy return of different methods in HalfCheetah is shown in Fig. 9. Note that the policy return does not indicate the risk-aversion of a policy.

A.6 Ant

The agent controls a robot with four legs attached to it with each leg having two links. The state dimension is 113 (add X-position). The action dimension is 8. Similar to HalfCheetah, we modify the reward to make the distance based reward positive as long as the agent is moving far from the

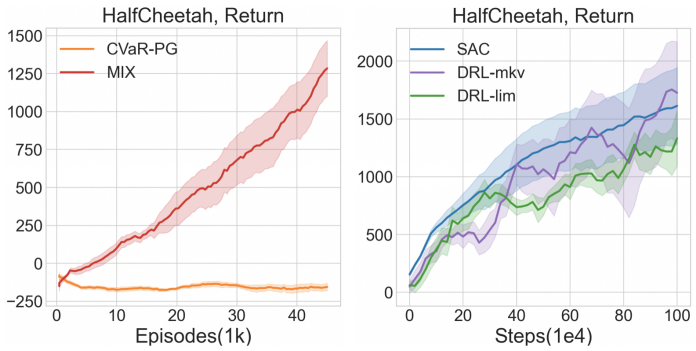

Figure 9: Policy return (y-axsis) v.s. training episodes or steps in HalfCheetah. Curves are averaged over 10 seeds with shaded regions indicating standard errors.

origin. If the agent reaches the region X-position <-3, an additional noisy reward sampled from $\mathcal{N}(0,1)$ times 50 is given. The game ends when a maximum episode length 500 is reached.

Policy function. Hidden size: 256. Others are the same as the case in InvertedPendulum.

Value function. Hidden size: 256. Others are the same as the case in InvertedPendulum.

A.6.1 Learning Parameters

Discount factor $\gamma = 0.99$. Optimizer is Adam.

SAC: Learning rate is 3e-4$\in$ {7e-4, 3e-4, 1e-4}.

CVaR-PG: Learning rate is 3e-4$\in$ {7e-4, 3e-4, 1e-4}.

MIX: Learning rate for π' and w is 3e-4$\in$ {7e-4, 3e-4, 1e-4}. Learning rate for IQL part (including policy and value functions) is the same. IQL update frequency $C = 30$, by sampling 2e5 transitions from buffer. $\eta = 0.8$ in Eq. 9. $\beta = 2$ in Eq. 10.

DRL-mkv: Learning rate is 7e-5$\in$ {7e-4, 3e-4, 1e-4, 7e-5}.

DRL-lim: Learning rate is 7e-5$\in$ {7e-4, 3e-4, 1e-4, 7e-5}.

A.6.2 Policy Return in Ant

The policy return of different methods in Ant is shown in Fig. 10. Note that the policy return does not indicate the risk-aversion of a policy.

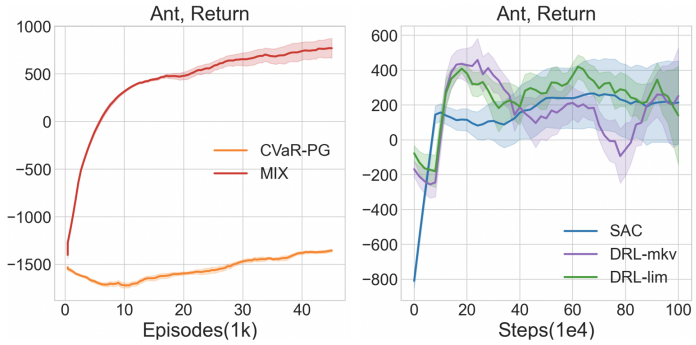

Figure 10: Policy return (y-axsis) v.s. training episodes or steps in Ant. Curves are averaged over 10 seeds with shaded regions indicating standard errors.

A.7 Driving Game

The goal of this game is to control the agent's car to follow the leader car without colliding. The state dimension is 5. The action dimension is 5. We refer reader to Sec. 5.2 of (Greenberg et al., 2022) for more details.

The method proposed in Greenberg et al. (2022) is named CeSoR, which includes a curriculum learning scheduler to adjust CVaR α during learning, i.e., starting from a large value for α and gradually decreasing to its target value; and a trajectory generator which controls the environment dynamic. In this domain, it controls the behavior of the leader car.

We directly use the code provided by Greenberg et al. (2022) to produce the results for CeSoR. CeSoR is orthogonal to MIX, and these two can be combined. We combine MIX with curriculum learning (denoted as MIX+SoR, SoR means soft risk to represent curriculum learning in Greenberg et al. (2022)), and combine MIX with CeSoR (denoted as MIX+CeSoR) in this domain.

Policy function. Policy of CeSoR and CVaR-PG: Hidden size: 32. Hidden layer: 2. Activation: Tanh.

Policy of MIX: Hidden size: 32. Activation: Tanh. Others are the same as the case in InvertedPendulum.

Value function. Q and V of IQL: Hidden size: 32. Others are the same as the case in InvertedPendulum.

A.7.1 Learning Parameters

CVaR $\alpha = 0.05$. Update policy after gathering $N = 80$ trajectories. The starting value for CVaR α is 0.8.

CVaR-PG: Learning rate 1e-2$\in$ {2e-2, 1e-2, 5e-3}.

CeSoR: Learning rate 1e-2$\in$ {2e-2, 1e-2, 5e-3}.

MIX: Learning rate for π' and w is 1e-2$\in$ {2e-2, 1e-2, 5e-3}. Learning rate for IQL part (including policy and value function) is 5e-3. IQL update frequency $C = 50$, by sample 2e4 transitions from buffer. $\eta = 0.8$ in Eq. 9. $\beta = 2$ in Eq. 10.

MIX+SoR: Learning parameters are the same as MIX.

MIX+CeSoR: Learning parameters are the same as MIX.

We report the mean return and the 0.05-CVaR of the return in Fig. 11. For both the mean and tail of the return, CeSoR, MIX, and MIX variants are better than CVaR-PG. CeSoR is slightly better than MIX, since CeSoR possesses the environment dynamic information while MIX does not. MIX with curriculum learning, i.e., MIX+SoR, learns faster than MIX at the early training stage than MIX, though the final mean return is the same as MIX. MIX+CeSoR is better than MIX and MIX+SoR with respect to the mean and tail of the return, and is comparable to CeSoR. CeSoR learns faster and achieves the highest risk averse rewards among all techniques, however it requires access to the environment dynamics and the ability to change the parameters of the dynamics in a way that is domain specific. In contrast, MIX and MIX+SoR do not need dynamics information and therefore can be applied directly to any domain.

Remark. Our mixture policy method differs from the curriculum learning idea in Greenberg et al. (2022). Tough the CVaR α starts from a large value in curriculum learning, where the objective is close to a risk-neutral problem, it is an on-policy policy gradient method, i.e., the trajectories used to update the policy is generated by the current policy (if no importance sampling is assumed). In contrast, the risk-neutral component of our mixture policy is trained by an off-policy (offline) algorithm, in this case, all the encountered trajectories can be stored in the replay buffer for policy update.

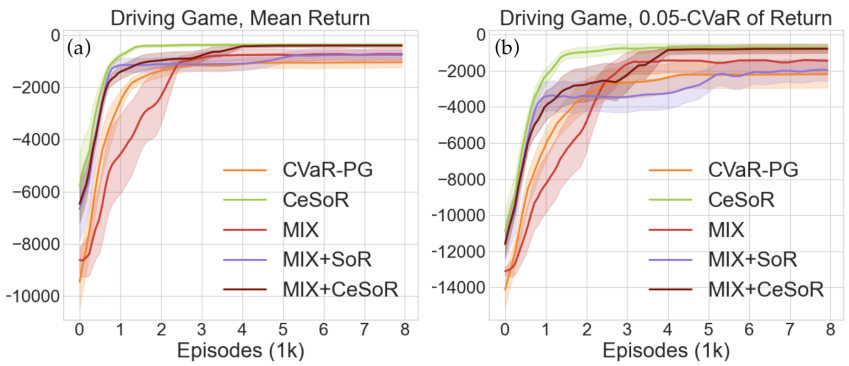

Figure 11: (a) The expected return (y-axsis), and (b) the 0.05-CVaR of the return (y-axsis) achieved by CVaR-PG, CeSoR, MIX, MIX+SoR, and MIX+CeSoR in driving game. Curves are averaged over 10 seeds with shaded regions indicating standard errors.

B Additional Related Work

Due to its two-layered structure, a mixture policy is also called a hierarchical policy (Daniel et al., 2012). Though the idea of mixture policy is not new, it is mainly applied in risk-neutral settings. Osa et al. (2023) constructed a mixture of deterministic policies for offline RL tasks and shown it can mitigate the issue of critic error accumulation in offline RL. Wulfmeier et al. (2020) and Seyde et al. (2022) utilized mixture policy to capture the diverse motivations of the robots such that the skill learned by each sub-policy can be transferred. Akrour et al. (2021) adopted mixture policy to enhance the interpretability of decision making. The mixture policy is also used for option discovery (Zhang & Whiteson, 2019; Wulfmeier et al., 2021). The similar mixture structure also appears in value (critic) function learning, for instance, mixture critic is utilized for distributional RL (Choi et al., 2019; Kuznetsov et al., 2020).