DR-SAC: DISTRIBUTIONALLY ROBUST SOFT ACTOR-CRITIC FOR REINFORCEMENT LEARNING UNDER UNCERTAINTY

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep reinforcement learning (RL) has achieved remarkable success, yet its deployment in real-world scenarios is often limited by vulnerability to environmental uncertainties. Distributionally robust RL (DR-RL) algorithms have been proposed to resolve this challenge, but existing approaches are largely restricted to value-based methods in tabular settings. In this work, we introduce Distributionally Robust Soft Actor-Critic (DR-SAC), the first actor–critic based DR-RL algorithm for offline learning in continuous action spaces. DR-SAC maximizes the entropy-regularized rewards against the worst possible transition models within an KL-divergence constrained uncertainty set. We derive the distributionally robust version of the soft policy iteration with a convergence guarantee and incorporate a generative modeling approach to estimate the unknown nominal transition models. Experiment results on five continuous RL tasks demonstrate our algorithm achieves up to $9.8\times$ higher average reward than the SAC baseline under common perturbations. Additionally, DR-SAC significantly improves computing efficiency and applicability to large-scale problems compared with existing DR-RL algorithms.

1 INTRODUCTION

The field of deep reinforcement learning has witnessed remarkable progress, enabling agents to learn complex behaviors in various domains, from game playing to robotic control (Arulkumaran et al., 2017; Francois-Lavet et al., 2018; Chen et al., 2024b). Many deep RL algorithms have demonstrated notable performance without training on real-world systems, by using a simulator or pre-collected data, making them attractive for practical applications. Among them, Soft Actor-Critic (SAC, Haarnoja et al. (2018a;b)) is a principled approach that adopts an entropy regularized learning objective, commonly known as the soft value function. This maximum entropy approach is founded on theoretical principles (Ziebart, 2010) and has been applied to various contexts, including stochastic control (Todorov, 2008; Rawlik et al., 2012) and inverse reinforcement learning (Ziebart et al., 2008; Zhou et al., 2018).

However, a persistent challenge limiting the deployment of deep RL in real-world systems is the inherent sensitivity of learned policies to uncertainties in the environment (Whittle, 1981; Enders et al., 2024). Agents trained in one environment often exhibit significant performance degradation when deployed in a slightly different environment. This model mismatches often stem from uncertain transition and reward functions, observation and actuator errors, model parameter variations, or even adversarial perturbations.

Distributionally robust RL addresses this challenge by optimizing decision-making in the worst-case scenario. Specifically, instead of working on a single Markov Decision Process (MDP), DR-RL considers a Robust Markov Decision Process (RMDP) framework, which includes a set of MDPs defined by an uncertainty set of distributions around the nominal one. Although both value-based (Liu et al., 2022; Lu et al., 2024) and policy-gradient (Wang & Zou, 2022; Kumar et al., 2023) DR-RL algorithms have been proposed, most work focus on the performance guarantees and sample complexity in the tabular setting and cannot be deployed in continuous environments, with the only exception being Robust Fitted Q-Iteration (RFQI, Panaganti et al. (2022)). However, fundamental research gaps remain: 1) RFQI only considers uncertainty sets defined by the Total Variation (TV) distance, which is analytically convenient due to the piece-wise linear dual formulation but cannot be extended to other divergences; and 2) its non-robust baseline, Fitted Q-Iteration (FQI, Ernst et al. (2005)), is value-based and suffers from critical limitations, including deterministic learned policies, low applicability to high-dimensional action spaces and high sensitivity to the learned state-action function (Degris et al., 2012). In contrast, actor-critic based algorithms combine low-variance return estimation with scalable policy optimization, making them preferred in benchmark tasks and practical applications (Konda & Tsitsiklis, 1999; Grondman et al., 2012). However, no distributionally robust counterpart has been developed. This gap motivates our development of Distributionally Robust Soft Actor-Critic (DR-SAC), *the first actor–critic based DR-RL algorithm for offline learning in continuous action spaces*.

In this work, we assume access only to a dataset collected in the training environment and the transition distributions of the deployment environment lie within an uncertainty set, which is defined as a Kullback-Leibler (KL) divergence ball centered around the nominal one. The goal is to learn a policy that maximizes the soft value function under the worst possible distributions. The main contributions of this work are:

- We formulate the maximum entropy learning framework with uncertain transition distributions lying in KL-divergence constrained balls. Within this framework, we derive the distributionally robust soft policy iteration with convergence guarantees and develop the distributionally robust counterpart of SAC, one of the most widely used offline RL benchmark algorithms.
- We exploit the interchange property to reformulate the optimization problems over scalars into functional optimization, resulting in policy iteration that is independent of state–action space dimensionality. This reformulation enables application to continuous action space and saves over 80.0% training time compared to the existing DR-RL algorithm RFQI.
- We incorporate generative models to estimate unknown nominal distributions and construct empirical measures with minor computation and memory increase. This addresses the double-sampling issue caused by the non-linear KL-divergence dual formulation and enables distributionally robust soft policy learning in offline and continuous-space tasks. Our proposed algorithm, DR-SAC, is validated on five offline RL environments with extensive perturbations and achieves up to $9.8\times$ higher average reward than the SAC baseline.

1.1 Related Works

Robust RL. The RMDP and Robust Dynamic Programming method were first introduced in Iyengar (2005); Nilim & El Ghaoui (2005) and have been widely studied in Xu & Mannor (2010); Wiesemann et al. (2013); Yu & Xu (2015) under planning settings. Many works consider robust RL algorithms from different aspects, such as soft-robustness (Derman et al., 2018; Lobo et al., 2020), risk sensitivity (Tamar et al., 2015; Pan et al., 2019; Singh et al., 2020; Queeney & Benosman, 2023), and adversarial training (Pinto et al., 2017; Zhang et al., 2020; Cheng et al., 2022). In recent years, many distributionally robust RL algorithms have been proposed with provable guarantees in the tabular setting, including algorithms based on Q-learning (Wang et al., 2023; 2024; Liang et al., 2024) and value iteration (Zhou et al., 2021; Panaganti & Kalathil, 2022; Xu et al., 2023; Ma et al., 2023; Liu & Xu, 2024). However, these algorithms are not applicable to continuous action space environments.

Model-Free Algorithms for Distributionally Robust RL. In the DR-RL problem, the nominal distributions usually appear in the optimization problem but are unknown in reality. To overcome this difficulty, some model-free algorithms (Liu et al., 2022; Zhou et al., 2023; Ramesh et al., 2024) assume access to a simulator that generates *i.i.d* samples from the nominal environment, which does not satisfy the offline requirement. Some algorithms (Derman & Mannor, 2020; Clavier et al., 2023; Shi & Chi, 2024) compute empirical frequencies of state transitions in the offline dataset, which is not applicable to the continuous space task. Lastly, Empirical Risk Minimization (ERM) method has also been used to estimate the loss function in a special structure (Mankowitz et al., 2019; Wang & Zou, 2021; Kordabad et al., 2022) but is not widely applicable.

VAE in Offline RL. Variational Autoencoders (VAEs) have wide applications in non-robust offline learning algorithms. A major use of VAE is to estimate the behavior policy from the offline dataset, and add policy constraints or apply pessimistic value (Fujimoto et al., 2019; Wei et al., 2021; Xu et al., 2022; Lyu et al., 2022). See Chen et al. (2024a) for a more detailed discussion. Using VAE to reconstruct states has also been found in Van Hoof et al. (2016). *To the best of our knowledge, we are the first to incorporate VAE models in a DR-RL algorithm, to estimate nominal transition distributions and generate samples without a simulator.*

Note that although Smirnova et al. (2019) proposed a close name algorithm, their settings are completely different from ours and most DR-RL literature. The authors assume estimation error in the evaluation step and use KL divergence to limit the behavior policy, with all analysis on a single MDP rather than an RMDP.

2 FORMULATION

2.1 NOTATION AND BASICS OF SOFT ACTOR-CRITIC

A standard framework for reinforcement learning is the discounted Markov Decision Process (MDP), formally defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, P, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively, both continuous in this work. The random reward function is denoted by $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{P}([0, R_{\max}])$, where $\mathbb{P}([0, R_{\max}])$ is the set of random variables supported on $[0, R_{\max}]$. The transition distribution is denoted by $P : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$, where $\Delta(S)$ is the set of probability function on set $\mathcal{S}$ and $\gamma \in [0, 1)$ is the discount factor. We denote $r = R(s, a)$ as the random reward and s' as the next state reached following the transition distribution $p_{s,a} = P(\cdot \mid s, a)$. A policy $\pi : \mathcal{S} \mapsto \Delta(\mathcal{A})$ represents the conditional probability of actions taken. We consider a stochastic stationary policy class, denoted by Π. The entropy of a stochastic policy π at state s is defined as $\mathcal{H}(\pi(s)) = \mathbb{E}\left[-\log \pi(a|s)\right]$, measuring the randomness of action. The set of integers from 1 to n is denoted as $[n]$.

In maximum entropy RL tasks, to encourage exploration, the value function includes the cumulative discounted sum of reward and entropy of the stochastic policy π. More precisely, given an MDP $\mathcal{M}$, the value function with entropy (soft value function) under policy π is

$$V^{\pi}_{\mathcal{M}}(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1}\Big(r_t + \alpha \cdot \mathcal{H}\big(\pi(s_t)\big)\Big) \,\middle|\, \pi, s_1 = s\right]. \tag{1}$$

The temperature $\alpha \geq 0$ determines the relative importance of policy stochasticity compared to reward. The optimal value and optimal policy are defined as $V^{\star}_{\mathcal{M}} = \max_{\pi \in \Pi} V^{\pi}_{\mathcal{M}}$ and $\pi^{\star}_{\mathcal{M}} = \operatorname{argmax}_{\pi \in \Pi} V^{\pi}_{\mathcal{M}}$. Similarly, the soft state-action value function (soft Q-function) under policy π can be defined as

$$Q^{\pi}_{\mathcal{M}}(s, a) = \quad \mathbb{E}\left[r_1 + \sum_{t=2}^{\infty} \gamma^{t-1}\Big(r_t + \alpha \cdot \mathcal{H}\big(\pi(s_t)\big)\Big) \,\middle|\, \pi, s_1 = s, a_1 = a\right]. \tag{2}$$

For any mapping $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, Haarnoja et al. (2018a) defined soft Bellman operator as

$$\mathcal{T}^{\pi} Q(s, a) = \quad \mathbb{E}[r] + \gamma \cdot \mathbb{E}_{p_{s,a},\pi}\left[Q\left(s', a'\right) - \alpha \log \pi\left(a' \mid s'\right)\right]. \tag{3}$$

Soft Actor-Critic (SAC) algorithm updates the policy through soft policy iteration with guaranteed convergence in the tabular case. In each iteration, $\mathcal{T}^{\pi}$ will be applied to the estimation of soft Q-function under the current policy π, and the policy is updated by minimizing the KL divergence between the improved policy distribution and the exponential of the soft Q-function:

$$\pi_{k+1} = \operatorname*{argmin}_{\pi \in \Pi} D_{\mathrm{KL}}\left(\pi(\cdot \mid s) \,\middle\|\, \exp\left(\frac{1}{\alpha} Q^{\pi_k}_{\mathcal{M}}(s, \cdot)\right) \Big/ Z(s)\right), \, k = 0, 1, \cdots \tag{4}$$

where $D_{\mathrm{KL}}(P \parallel Q) = \mathbb{E}_P\left[\log\left(\frac{P(x)}{Q(x)}\right)\right]$ denotes the KL divergence and the function $Z(\cdot)$ normalizes the distribution of $\exp\left(\frac{1}{\alpha} Q^{\pi_k}_{\mathcal{M}}(s, \cdot)\right)$.

2.2 ROBUST MARKOV DECISION PROCESS

In real-world RL tasks, the transition distribution P and reward function R in the deployment environment may be different from the environment in which the model is trained or the offline dataset is collected. The potential environmental shift motivates us to study the Robust Markov Decision Process (RMDP) and learn a policy more robust to such perturbation. Unlike standard MDPs, the RMDP formulation considers models in an uncertainty set. Since the analysis and algorithm design will be similar to reward function perturbation, we assume the reward function R is unchanged and consider uncertain transition distributions only.

The RMDP framework is denoted as $\mathcal{M}_\delta = (\mathcal{S}, \mathcal{A}, R, \mathcal{P}(\delta), \gamma)$. We consider the transition distribution perturbed within a KL-divergence ball. Specifically, let $\mathcal{P}^0 = \{p^0_{s,a}\}_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ be the nominal transition distributions. For each state-action pair $(s,a) \in \mathcal{S}\times\mathcal{A}$, given $\delta > 0$, we define the KL ball centered at $p^0_{s,a}$ as

$$\mathcal{P}_{s,a}(\delta) := \left\{p_{s,a} \in \Delta(\mathcal{S}) : D_{\mathrm{KL}}(p_{s,a} \| p^0_{s,a}) \le \delta\right\}. \tag{5}$$

The ambiguity set $\mathcal{P}(\delta)$ is the Cartesian product of $\mathcal{P}_{s,a}(\delta)$ for all pairs $(s,a) \in \mathcal{S}\times\mathcal{A}$, which belongs to the (s,a)-rectangular set in Wiesemann et al. (2013).

In the RMDP framework, the goal is to optimize the worst-case objective value under any model in the ambiguity set. Given $\mathcal{M}_\delta$, similar to (1), the distributionally robust (DR) soft value function is defined as

$$V^\pi_{\mathcal{M}_\delta}(s) = \inf_{\mathbf{p}\in\mathcal{P}(\delta)} \mathbb{E}_{\mathbf{p}}\left[\sum_{t=1}^{\infty}\gamma^{t-1}\Big(r_t + \alpha\cdot\mathcal{H}\big(\pi(s_t)\big)\Big)\,\middle|\,\pi, s_1 = s\right], \forall s \in \mathcal{S} \tag{6}$$

Similarly, the distributionally robust soft Q-function is given by

$$Q^\pi_{\mathcal{M}_\delta}(s,a) = \inf_{\mathbf{p}\in\mathcal{P}(\delta)} \mathbb{E}_{\mathbf{p}}\left[r_1 + \sum_{t=2}^{\infty}\gamma^{t-1}\Big(r_t + \alpha\cdot\mathcal{H}\big(\pi(s_t)\big)\Big)\,\middle|\,\pi, s_1 = s, a_1 = a\right], \forall (s,a) \in (\mathcal{S},\mathcal{A}) \tag{7}$$

The DR optimal value and DR optimal policy are defined accordingly as:

$$V^\star_{\mathcal{M}_\delta}(s) = \max_{\pi\in\Pi} V^\pi_{\mathcal{M}_\delta}(s) \text{ and } \pi^\star_{\mathcal{M}_\delta}(\cdot \mid s) = \operatorname*{argmax}_{\pi\in\Pi} V^\pi_{\mathcal{M}_\delta}(s), \forall s \in \mathcal{S}. \tag{8}$$

3 Algorithm: Distributionally Robust Soft Actor-Critic

In this section, we present the development of the Distributionally Robust Soft Actor-Critic algorithm. We first derive the distributionally robust soft policy iteration and establish its convergence to the optimal policy. To improve computing efficiency, we develop a scalable implementation based on functional optimization. Lastly, to handle the challenge of unknown nominal distributions, we incorporate a VAE model to construct the empirical transition measures.

Assumption 3.1. To ensure that the policy entropy $\mathcal{H}(\pi(s)) = \mathbb{E}_{a\sim\pi(\cdot|s)}[-\log\pi(a|s)]$ is bounded, we assume $|\mathcal{A}| < \infty$.

Remark 3.2. Assumption 3.1 is inherited from the non-robust baseline SAC (Haarnoja et al., 2018a), which establishes theoretical guarantees in the tabular setting while being empirically used as a benchmark in continuous tasks. Our work extends the performance properties of SAC to the DR-RL framework. In Section 3.3, we design a practical algorithm in continuous action spaces.

3.1 Distributionally Robust Soft Policy Iteration

We begin with providing the DR soft policy iteration, which iterates between DR soft policy evaluation and DR soft policy improvement. We also show that the DR soft policy iteration is guaranteed to converge to the DR optimal policy.

In the DR soft policy evaluation step, the DR soft Q-function is estimated by iteratively applying the distributionally robust version of the Bellman operator, considering the worst possible transition distribution in the uncertainty set. For a fixed policy π and any bounded mapping $Q : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$, the distributionally robust soft Bellman operator is defined as:

$$\mathcal{T}^\pi_\delta Q(s,a) := \mathbb{E}[r] + \gamma \cdot \inf_{p_{s,a}\in\mathcal{P}_{s,a}(\delta)} \left\{\mathbb{E}_{p_{s,a},\pi}\left[Q(s',a') - \alpha\cdot\log\pi(a' \mid s')\right]\right\}. \tag{9}$$

Following the results in Iyengar (2005); Xu & Mannor (2010), the DR soft Q-function can be computed via distributionally robust dynamic programming, and $Q^\pi_{\mathcal{M}_\delta}$ is a fixed point of $\mathcal{T}^\pi_\delta$. However, Equation (9) is generally intractable because it requires solving an infinite-dimensional optimization problem. To address this issue, we use the strong duality result on worst-case expectations over a KL-divergence ball to derive the dual form of Equation (9).

Proposition 3.3 (Dual Formulation of the Distributionally Robust Soft Bellman Operator)**.** *Suppose $Q(s, a)$ is bounded, the distributionally robust soft Bellman operator in* (9) *can be reformulated into:*

$$\mathcal{T}^\pi_\delta Q(s,a) = \mathbb{E}[r] + \gamma \cdot \sup_{\beta \geq 0} \left\{ -\beta \log \left(\mathbb{E}_{p^0_{s,a}} \left[\exp \left(-\frac{V(s')}{\beta} \right) \right] \right) - \beta\delta \right\}, \tag{10}$$

where

$$V(s) = \mathbb{E}_{a\sim\pi} \left[Q(s,a) - \alpha \cdot \log \pi(a \mid s) \right]. \tag{11}$$

Derivation is provided in Appendix B.1. The RHS of equation (10) only depends on the nominal transition distribution $\mathcal{P}^0_{s,a}$, instead of an infinite number of distributions in the uncertainty set $\mathcal{P}(\delta)$. Also, the optimization problem on the RHS is over the scalar β, instead of an infinite-dimensional distribution. With the tractable dual formation in Proposition 3.3, DR soft Q-value under any policy π can be computed by iteratively applying the DR soft Bellman operator $\mathcal{T}^\pi_\delta$.

Proposition 3.4 (Distributionally Robust Soft Policy Evaluation)**.** *For any policy $\pi \in \Pi$ fixed, starting from any bounded mapping $Q^0 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, define a sequence $\{Q^k\}$ by iteratively applying distributionally robust soft Bellman operator: $Q^{k+1} = \mathcal{T}^\pi_\delta Q^k$. This sequence will converge to the DR soft Q-value of policy π as $k \to \infty$.*

The main part of the proof shows that the operator $\mathcal{T}^\pi_\delta$ is a γ-contraction mapping, with details in Appendix B.2. Next, the distributionally robust soft policy improvement step is similar to Equation (4), but replacing $Q_\mathcal{M}$ with DR soft Q-value $Q_{\mathcal{M}_\delta}$. The new policy in each update is defined as

$$\pi_{k+1} = \operatorname*{argmin}_{\pi\in\Pi} D_{\mathrm{KL}} \left(\pi(\cdot \mid s) \middle\| \exp\left(\frac{1}{\alpha} Q^{\pi_k}_{\mathcal{M}_\delta}(s,\cdot) \right) \Big/ Z^{\pi_k}(s) \right), \; k = 0, 1, \cdots \tag{12}$$

With policy updating rule (12), we show that the policy sequence $\{\pi_k\}$ has a non-decreasing value with respect to the DR soft Q-function in Proposition 3.5. This extends the non-robust soft policy improvement to cases with uncertain transition probabilities.

Proposition 3.5 (Distributionally Robust Soft Policy Improvement)**.** *Suppose $|\mathcal{A}| < \infty$, let $\pi_k \in \Pi$ and π_{k+1} be the solution of the optimization problem defined in Equation* (12)*. Then $Q^{\pi_{k+1}}_{\mathcal{M}_\delta}(s,a) \geq Q^{\pi_k}_{\mathcal{M}_\delta}(s,a)$ for any $(s,a) \in \mathcal{S} \times \mathcal{A}$.*

Proof is provided in Appendix B.3. The DR soft policy iteration algorithm proceeds by alternatively applying DR soft policy evaluation and DR soft policy improvement. In the following theorem, we show that the policy sequence converges to the optimum under the DR soft policy iteration, with proof in Appendix B.4.

Theorem 3.6 (Distributionally Robust Soft Policy Iteration)**.** *Suppose $|\mathcal{A}| < \infty$, starting from any policy $\pi^0 \in \Pi$, the policy sequence $\{\pi^k\}$ converges to the optimal policy $\pi^\star$ under DR soft policy iteration as $k \to \infty$.*

Key Challenges. Although DR soft policy iteration is guaranteed to find the optimal policy, there are still challenges in extending it to continuous action space and offline setting: 1) the DR soft policy evaluation step in (10) is not efficient enough in large scale problems, 2) the nominal distribution $p^0_{s,a}$ is usually unknown in offline RL tasks, and 3) the DR soft policy iteration can only be implemented exactly in tabular setting. We will resolve these issues step by step in the rest of this section.

3.2 Solving Dual Optimization using Generative Model

In offline RL tasks, the goal is to learn the optimal policy with access to a pre-collected dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^N$, where $(s_i, a_i) \sim \mu$, with μ denoting the data generation distribution determined by the behavior policy, $r_i = R(s_i, a_i)$ and $s_i' \sim P^0(\cdot \mid s_i, a_i)$. In this section, we derive a practical functional optimization method to compute the dual formulation of DR soft Bellman operator in (10) with higher efficiency to address challenge 1, and propose a generative modeling scheme to address challenge 2.

Dual Functional Optimization. In the DR soft policy evaluation step, the Bellman operator $\mathcal{T}_\delta^\pi$ will be applied to Q-function iteratively. By writing out the dual form of the DR soft Bellman operator in (10), the optimization problem is over a scalar $\beta > 0$ and can be routinely solved. However, this optimization problem needs to be solved for every (s, a) pair at each time of update, making the training process slow for a large-scale problem. To improve training efficiency, our idea is to convert a group of scalar optimization problems into a single optimization problem over a function space. This can be achieved by applying the property of interchanging minimization and integration in decomposable space (Rockafellar & Wets, 2009).

Consider the probability space $(\mathcal{S} \times \mathcal{A}, \Sigma(\mathcal{S} \times \mathcal{A}), \mu)$ and let $L^1(\mathcal{S} \times \mathcal{A}, \Sigma(\mathcal{S} \times \mathcal{A}), \mu)$ be the set of absolutely integrable functions on that space, abbreviated as L^1. We can reformulate the expectation of optimal value for each (s, a) pair into a single functional optimization problem.

Proposition 3.7. *For any $\delta > 0$ and function $V : \mathcal{S} \to [0, (R_{\max} + \alpha \log|\mathcal{A}|)/(1-\gamma)]$, let*

$$f\big((s,a),\beta\big) := -\beta \log\left(\mathbb{E}_{p^0_{s,a}}\left[\exp\left(-\frac{V(s')}{\beta}\right)\right]\right) - \beta\delta. \tag{13}$$

Suppose that Assumption 3.1 holds, i.e. $|\mathcal{A}| < \infty$. Define a function set

$$\mathcal{G} := \left\{ g \in L_1 : g(s,a) \in \left[0, \frac{R_{\max} + \alpha \log|\mathcal{A}|}{(1-\gamma)\delta}\right], \forall (s,a) \in \mathcal{S} \times \mathcal{A} \right\}. \tag{14}$$

Then we have

$$\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\sup_{\beta \geq 0} f\big((s,a),\beta\big)\right] = \sup_{g \in \mathcal{G}} \mathbb{E}_{(s,a)\sim\mathcal{D}}\Big[f\big((s,a), g(s,a)\big)\Big]. \tag{15}$$

Proof is provided in Appendix B.5. The RHS of (15) only requires solving one optimization problem instead of $|\mathcal{D}|$ problems on the LHS. This functional optimization method substantially increases training efficiency with negligible robustness loss. We present the training time and performance comparison in Section 4.3 and Appendix C.3.1. Given Proposition 3.7, we introduce a new Bellman operator by replacing the scalar β with a function and removing optimization. For any function $g \in \mathcal{G}$ and mapping $Q : \mathcal{S} \times \mathcal{A} \to [0, (R_{\max} + \alpha \log|\mathcal{A}|)/(1-\gamma)]$, let

$$\begin{aligned}\mathcal{T}^\pi_{\delta,g} Q(s,a) :=& \mathbb{E}[r] + \gamma \cdot f\big((s,a), g(s,a)\big) \\ =& \mathbb{E}[r] + \gamma \cdot \left\{ -g(s,a) \log\left(\mathbb{E}_{p^0_{s,a}}\left[\exp\left(-\frac{V(s')}{g(s,a)}\right)\right]\right) - g(s,a)\delta \right\},\end{aligned} \tag{16}$$

where $V(s) = \mathbb{E}_{a\sim\pi}\left[Q(s,a) - \alpha \cdot \log \pi(a \mid s)\right]$. From Proposition 3.7, we have a direct conclusion that $\|\mathcal{T}^\pi_\delta Q - \mathcal{T}^\pi_{\delta,g^\star} Q\|_{1,\mu} = 0$, where $g^\star = \operatorname{argsup}_{g \in \mathcal{G}} \mathbb{E}_{(s,a)\sim\mathcal{D}}\Big[f\big((s,a), g(s,a)\big)\Big]$.

Generative Modeling for Nominal Distributions. In offline RL tasks, we assume the nominal distributions $\mathcal{P}^0$ are unknown, and no simulator is available to generate additional samples. Under the KL-constrained uncertainty set, the dual optimization problem is non-linear and the empirical risk computed from the offline dataset $\mathcal{D}$ suffers from the *double-sampling issue*, making it inapplicable in our case. More detailed discussion is provided in Appendix A.1. To empirically apply operator $\mathcal{T}^\pi_{\delta,g}$ in continuous space, we incorporate a VAE model to estimate the nominal distributions and generate samples to construct empirical measures. To be specific, the VAE model learns from collected data $(s, a, s') \in \mathcal{D}$ and generate next state samples $\{\tilde{s}'_i\}_{i=1}^m$. We denote $\tilde{p}^0_{s,a}$ as the empirical measures of $p^0_{s,a}$. For any function $h : \mathcal{S} \mapsto \mathbb{R}$, we have $\mathbb{E}_{s'\sim\tilde{p}^0_{s,a}}[h(s')] = \frac{1}{m}\sum_{i=1}^m h(\tilde{s}'_i)$. The empirical Bellman operator with functional optimization is defined as

$$\widetilde{\mathcal{T}}^\pi_{\delta,g} Q(s,a) := \mathbb{E}[r] + \gamma \cdot \widetilde{f}\big((s,a), g(s,a)\big), \tag{17}$$

where

$$\widetilde{f}\big((s,a),\beta\big) = -\beta \log\left(\mathbb{E}_{\tilde{p}^0_{s,a}}\left[\exp\left(-\frac{V(s')}{\beta}\right)\right]\right) - \beta\delta. \tag{18}$$

3.3 Distributionally Robust Soft Actor-Critic

Now we extend the action space to continuous and use neural networks to approximate the DR soft value function and policy. We consider the problem in RMDP $\mathcal{M}_\delta$, with subscripts in V and Q functions omitted. To be specific, our algorithm includes the value network $V_\psi(s)$, the Q-network $Q_\theta(s,a)$ and the stochastic policy $\pi_\phi(a \mid s)$, with ψ, θ, ϕ as the parameters. $\bar{\psi}$ and $\bar{\theta}$ are the target network parameters to help stabilizing training (Mnih et al., 2015). Let φ be the parameters of VAE model. We also use a parametrized neural network $\mathcal{G}_\eta$ to approximate the function set $\mathcal{G}$.

The idea behind our DR-SAC algorithm is to alternate between empirical DR soft policy evaluation with functional optimization and DR soft policy improvement. The loss of Q-network parameters in our algorithm is

$$J_Q^{\text{DR}}(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(s,a) - \mathcal{T}^\pi_{\delta,\widetilde{g}^\star} Q_\theta(s,a)\right)^2\right], \tag{19}$$

where

$$\widetilde{g}^\star = \operatorname*{argsup}_{g\in\mathcal{G}_\eta} \mathbb{E}_{(s,a)\in\mathcal{D}}\left[\widetilde{f}((s,a), g(s,a))\right]. \tag{20}$$

The loss functions of ψ, ϕ and α are the same as SAC in Haarnoja et al. (2018a) and the loss function of φ is the standard VAE loss, with details in Appendix A.2. To reduce the sensitivity on behavior policy in dataset generation, we include V-function as SAC-v1 algorithm (Haarnoja et al., 2018a), with detialed discussion in Section 4.3 and Appendix C.3.3. We also build multiple Q-functions $Q_{\theta_i}, (i \in [n])$, train them independently, and use the minimum of them in updating the value critic and actor function. This has been tested to outperform clipped Q-learning ($n = 2$) in offline RL tasks (An et al., 2021). We formally present the Distributionally Robust Soft Actor-Critic in Algorithm 1.

Algorithm 1 Distributionally Robust Soft Actor-Critic (DR-SAC)

```
Require: Offline dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_i')_{i=1}^N\}$, $V$-function network weights $\psi$, $Q$-function
    network weights $\theta_i, i \in [n]$, policy network weights $\phi$, transition VAE network weights $\varphi$,
    weight $\tau$ for moving average, function class $\mathcal{G}_\eta$
 1: $\bar{\psi} \leftarrow \psi, \bar{\theta}_i \leftarrow \theta_i$ for $i \in [n]$          ▷ Initialize target network weights for soft update
 2: for each gradient step do
 3:     $\varphi \leftarrow \varphi - \lambda_\varphi \hat{\nabla}_\varphi J_{\text{VAE}}(\varphi)$          ▷ Update transition VAE weights
 4:     Generate samples $\{\tilde{s}_i'\}_{i=1}^m$ from VAE, form empirical measures $\widetilde{p}^0_{s,a}$
 5:     Compute optimal function $\widetilde{g}^\star$ according to (20)
 6:     $\psi \leftarrow \psi - \lambda_\psi \hat{\nabla}_\psi J_V(\psi)$          ▷ Update $V$-function weights
 7:     $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q^{\text{DR}}(\theta_i)$ for $i \in [n]$          ▷ Update $Q$-function weights
 8:     $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$          ▷ Update policy weights
 9:     $\alpha \leftarrow \alpha - \lambda_\alpha \hat{\nabla}_\alpha J(\alpha)$          ▷ Adjust temperature
10:     $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}, \bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in [n]$          ▷ Update target network weights
11: end for
Ensure: $\phi$
```

4 Experiments

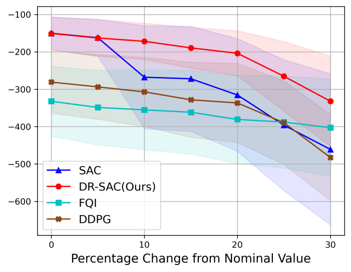
The goal of our experiments is to demonstrate the robustness of DR-SAC in handling environmental uncertainties in offline RL tasks. We evaluate the average episode rewards under different perturbations, comparing with non-robust baselines and RFQI, the only offline DR-RL algorithm applicable to continuous action spaces. To further highlight the practicality of our algorithm, we report the training time to show that DR-SAC significantly improves the training efficiency of DR-RL algorithms.

4.1 Settings

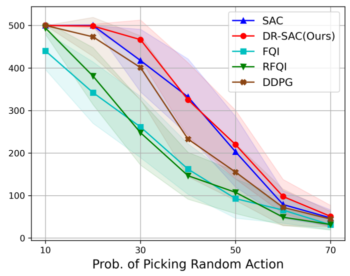
We implement SAC and DR-SAC based on the SAC-N (An et al., 2021). To the best of our knowledge, RFQI is the only distributionally robust offline RL algorithm applicable to continuous space. Besides RFQI, we also compare DR-SAC with Fitted Q-Iteration (FQI), Deep Deterministic Policy Gradient (DDPG, Lillicrap et al. (2015)), and Conservative Q-Learning (CQL, Kumar et al. (2020)).

We consider *Pendulum*, *Cartpole*, *LunarLander*, *Reacher* and *HalfCheetah* environments in Gymnasium (Towers et al., 2024). For *Cartpole*, we consider the continuous action space version in Mehta et al. (2021). For *LunarLander*, we also set the action space to be continuous. All algorithms are trained on the nominal environment and evaluated under different perturbations. In our experiments, we consider perturbations including: environment parameters change, random noise on observed state and random action taken by the actuator. More detailed settings are in Appendix C.1

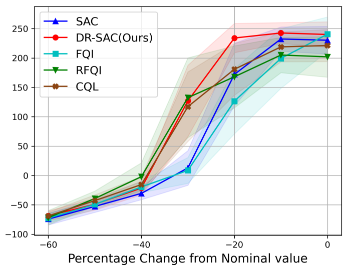
4.2 Performance Analysis



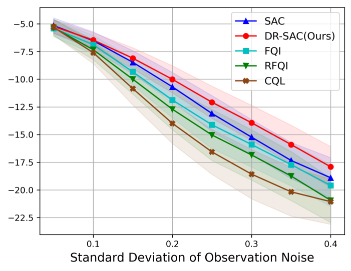
(a) Length Perturbation
Pendulum



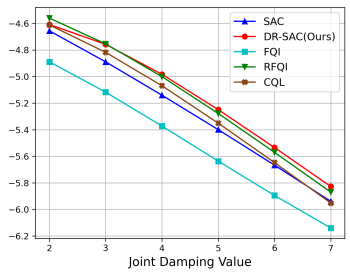
(b) Action Perturbation
Cartpole



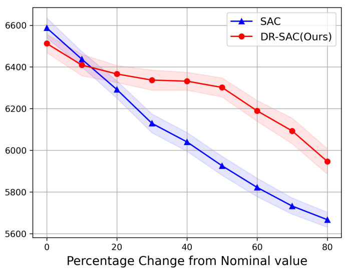
(c) Engine Perturbation
LunarLander



(d) Observation Perturbation
Reacher



(e) Damping Perturbation
Reacher



(f) Back Damping Perturbation
HalfCheetah

Figure 1: Robustness performance in different environments under perturbations. The curves show the average reward over 50 episodes, shaded by ± 0.5 standard deviation (Figure (e) see Table 4). In *Pendulum*, the environment parameter *length* changes. In *Cartpole*, random actions are taken by the actuator. In *LunarLander*, the environment parameters *main_engine_power* and *side_engine_power* change together. In *Reacher*, a Gaussian noise is added to nominal states; and the environment parameter *joint_damping* changes. In *HalfCheetah*, environment parameter *back_damping* changes.

This section reports selected experiment results. Additional experiments are provided in Appendix C.2. In the *Pendulum* environment, we change the parameter *length* to assess algorithm robustness against pendulum length changes. RFQI is omitted due to poor performance in the unperturbed environment. In Figure 1(a), DR-SAC performance outperforms SAC by 35% when the length changes by 20%. In the *Cartpole* environment, the actuator is perturbed by taking random actions with different probabilities. DR-SAC shows superior performance over the robust algorithm RFQI, especially when the probability of random action is less than 50%. In the *LunarLander* environment, we change the environment parameters *main_engine_power* and *side_engine_power* together to model engine power disturbance. DR-SAC shows consistently robust performance compared to other algorithms. In Figure 1(c), when the perturbation percentage is -20%, DR-SAC has an average reward of around 240 while rewards of all other algorithms drop under 180. Moreover, DR-SAC achieves 9.8 times higher reward than the SAC baseline when parameters change by -30%.

To demonstrate the robustness of DR-SAC in more complex environments, we also conduct experiments in *HalfCheetah* and *Reacher* from MuJoCo (Todorov et al., 2012). In the *Reacher* environment, we introduce two types of perturbations: adding Gaussian noise to nominal states and modifying the environment parameter *joint_damping*. In the observation perturbation test on Figure 1(d), DR-SAC shows the best performance in all test cases. In Figure 1(e), DR-SAC outperforms SAC and has similar robustness as RFQI. In the *HalfCheetah* environment, we only present the experiments of SAC and DR-SAC due to the poor performance of FQI and RFQI. When the environment parameter *back_damping* changes less than 50%, DR-SAC achieves a stable average reward of over 6300, while the average reward of SAC keeps decreasing to less than 5950.

Discussion on FQI Failure. It is worth noting that FQI and RFQI do not work well in unperturbed *Pendulum* and *HalfCheetah* environments. One possible reason is that offline RL algorithm performance depends on the dataset differently. SAC works well when the dataset has a broad coverage over the action space (Kumar et al., 2019). Conversely, the FQI algorithm is implemented on Batch-Constrained Deep Q-learning (BCQ, Fujimoto et al. (2019)), which restricts the agent to selecting actions close to the behavior policy. This conflicts with the epsilon-greedy method in data generation, as discussed in Appendix C.1. One major goal of our experiments is to demonstrate that DR-SAC exhibits better robustness over SAC under common environmental perturbations. Addressing the sensitivity of RL algorithms to offline dataset distribution is out of the scope of this study.

4.3 Ablation Studies

Training Efficiency of DR-SAC. Our DR-SAC algorithm is designed to balance efficiency and accuracy. In Section 3.2, we approximate the Bellman operator $\mathcal{T}_{\delta}^{\pi}$ with $\mathcal{T}_{\delta,g}^{\pi}$ to improve the training efficiency. To validate this approximation, we also train a robust algorithm using the accurate operator $\mathcal{T}_{\delta}^{\pi}$. Experimental results show that DR-SAC with functional optimization attains negligible loss in robustness while requiring less than 2% training time. More details are provided in Appendix C.3.1.

In Section 4.2, RFQI shows comparable robustness to DR-SAC in some environments. However, DR-SAC demonstrates notable improvement in the training efficiency. Table 1 shows that the training time of RFQI is at most 23.2 times that of DR-SAC. Compared with each non-robust baseline, RFQI requires no less than 11.3 times the training time of FQI, while DR-SAC training is at most 2.6 times that of SAC. Additional experiments show that this efficiency improvement arises from optimization efficiency. While the RFQI algorithm with functional approximation involves a similar step as (20), it requires 1000 gradient descent (GD) steps in each update to find the optimal function, while DR-SAC requires only 5 GD steps to achieve comparative performance. Experimental results in Appendix C.3.1 reveal that reducing the number of GD steps in RFQI leads to a severe performance drop even in unperturbed environments, suggesting that the loss function structure in RFQI inherently leads to slower convergence and demands more optimization steps.

Table 1: Training time in different environments (minute)

Env	SAC	DR-SAC	FQI	RFQI
Cartpole	2	4	7	93
LunarLander	16	36	17	238
Reacher	13	32	14	159

Robustness of VAE Model While the VAE models inevitably introduce estimation error when constructing empirical measures of the transition distributions, we empirically demonstrate that DR-SAC is largely insensitive to such modeling choices. Specifically, when the latent dimension of the VAE model is varied within the tested range of 5 to 20 in *Pendulum*, DR-SAC maintains superior robustness over the SAC baseline. Detailed experiment results are provided in Appendix C.3.2.

Usage of V-Network. In the DR-SAC algorithm, we include a V-network following the SAC-v1 design (Haarnoja et al., 2018a) to improve the applicability across a wider range of offline datasets. Although the V-network is removed in SAC-v2 (Haarnoja et al., 2018b), this version is indeed on-policy, while our setting is off-policy. We observe empirically that SAC with a V-network is less sensitive to the behavior policy used in dataset generation. Details are discussed in Appendix C.3.3.

5 Conclusions

We propose DR-SAC, the first actor-critic based DR-RL algorithm for offline settings and continuous action spaces. Our framework establishes distributionally robust soft policy iteration with convergence guarantees, saves over 80.0% of training time compared to RFQI through functional optimization, and resolves the double-sampling issue in estimating the nominal distributions via generative modeling. Experiments across five environments show that DR-SAC attains up to 9.8× higher reward than SAC under perturbations, demonstrating both robustness and efficiency in practical offline RL tasks.

Ethics Statement. All authors of this submission have read and adhered to the ICLR Code of Ethics.

Reproducibility Statement. We provide our code with detailed comments in the supplementary materials. The detailed experiment settings, dataset processing steps and the devices used in our experiments are provided in Appendix C to ensure reproducibility.

REFERENCES

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

Leemon Baird et al. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the twelfth international conference on machine learning*, pp. 30–37, 1995.

Jiayu Chen, Bhargav Ganguly, Yang Xu, Yongsheng Mei, Tian Lan, and Vaneet Aggarwal. Deep generative models for offline policy learning: Tutorial, survey, and perspectives on future directions. *arXiv preprint arXiv:2402.13777*, 2024a.

Yanjun Chen, Xinming Zhang, Xianghui Wang, Zhiqiang Xu, Xiaoyu Shen, and Wei Zhang. Corrected soft actor critic for continuous control. *arXiv preprint arXiv:2410.16739*, 2024b.

Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. Adversarially trained actor critic for offline reinforcement learning. In *International Conference on Machine Learning*, pp. 3852–3878. PMLR, 2022.

Pierre Clavier, Erwan Le Pennec, and Matthieu Geist. Towards minimax optimality of model-based robust reinforcement learning. *arXiv preprint arXiv:2302.05372*, 2023.

Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

Esther Derman and Shie Mannor. Distributional robustness and regularization in reinforcement learning. *arXiv preprint arXiv:2003.02894*, 2020.

Esther Derman, Daniel J Mankowitz, Timothy A Mann, and Shie Mannor. Soft-robust actor-critic policy-gradient. *arXiv preprint arXiv:1803.04848*, 2018.

Tobias Enders, James Harrison, and Maximilian Schiffer. Risk-sensitive soft actor-critic for robust deep reinforcement learning under distribution shifts. *arXiv preprint arXiv:2402.09992*, 2024.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 2005.

Vincent Francois-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11 (3-4):219–354, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1582–1591, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)*, 42(6):1291–1307, 2012.

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, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Zhaolin Hu and L Jeff Hong. Kullback-leibler divergence constrained distributionally robust optimization. *Available at Optimization Online*, 1(2):9, 2013.

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

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Arash Bahari Kordabad, Rafael Wisniewski, and Sebastien Gros. Safe reinforcement learning using wasserstein distributionally robust mpc and chance constraint. *IEEE Access*, 10:130058–130067, 2022.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Navdeep Kumar, Esther Derman, Matthieu Geist, Kfir Y Levy, and Shie Mannor. Policy gradient for rectangular robust markov decision processes. *Advances in Neural Information Processing Systems*, 36:59477–59501, 2023.

Zhipeng Liang, Xiaoteng Ma, Jose Blanchet, Jiheng Zhang, and Zhengyuan Zhou. Single-trajectory distributionally robust reinforcement learning, 2024. URL `https://arxiv.org/abs/2301.11721`.

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

Zhishuai Liu and Pan Xu. Minimax optimal and computationally efficient algorithms for distributionally robust offline reinforcement learning. *arXiv preprint arXiv:2403.09621*, 2024.

Zijian Liu, Qinxun Bai, Jose Blanchet, Perry Dong, Wei Xu, Zhengqing Zhou, and Zhengyuan Zhou. Distributionally robust q-learning. In *International Conference on Machine Learning*, pp. 13623–13643. PMLR, 2022.

Elita A Lobo, Mohammad Ghavamzadeh, and Marek Petrik. Soft-robust algorithms for batch reinforcement learning. *arXiv preprint arXiv:2011.14495*, 2020.

Miao Lu, Han Zhong, Tong Zhang, and Jose Blanchet. Distributionally robust reinforcement learning with interactive data collection: Fundamental hardness and near-optimal algorithm. *arXiv preprint arXiv:2404.03578*, 2024.

Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1711–1724, 2022.

Xiaoteng Ma, Zhipeng Liang, Jose Blanchet, Mingwen Liu, Li Xia, Jiheng Zhang, Qianchuan Zhao, and Zhengyuan Zhou. Distributionally robust offline reinforcement learning with linear function approximation, 2023. URL `https://arxiv.org/abs/2209.06620`.

Daniel J Mankowitz, Nir Levine, Rae Jeong, Yuanyuan Shi, Jackie Kay, Abbas Abdolmaleki, Jost Tobias Springenberg, Timothy Mann, Todd Hester, and Martin Riedmiller. Robust reinforcement learning for continuous control with model misspecification. *arXiv preprint arXiv:1906.07516*, 2019.

Viraj Mehta, Biswajit Paria, Jeff Schneider, Stefano Ermon, and Willie Neiswanger. An experimental design perspective on model-based reinforcement learning. *arXiv preprint arXiv:2112.05244*, 2021.

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

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

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

Kishan Panaganti and Dileep Kalathil. Sample complexity of robust reinforcement learning with a generative model. In *International Conference on Artificial Intelligence and Statistics*, pp. 9582–9602. PMLR, 2022.

Kishan Panaganti, Zaiyan Xu, Dileep Kalathil, and Mohammad Ghavamzadeh. Robust reinforcement learning using offline data. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32211–32224. Curran Associates, Inc., 2022.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International conference on machine learning*, pp. 2817–2826. PMLR, 2017.

James Queeney and Mouhacine Benosman. Risk-averse model uncertainty for distributionally robust safe reinforcement learning. *Advances in Neural Information Processing Systems*, 36:1659–1680, 2023.

Shyam Sundhar Ramesh, Pier Giuseppe Sessa, Yifan Hu, Andreas Krause, and Ilija Bogunovic. Distributionally robust model-based reinforcement learning with large state spaces. In *International Conference on Artificial Intelligence and Statistics*, pp. 100–108. PMLR, 2024.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*, 2012.

R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

Takuma Seno and Michita Imai. d3rlpy: An offline deep reinforcement learning library. *Journal of Machine Learning Research*, 23(315):1–20, 2022. URL `http://jmlr.org/papers/v23/22-0017.html`.

Alexander Shapiro. Distributionally robust stochastic programming. *SIAM Journal on Optimization*, 27(4):2258–2275, 2017. doi: 10.1137/16M1058297.

Laixi Shi and Yuejie Chi. Distributionally robust model-based offline reinforcement learning with near-optimal sample complexity. *Journal of Machine Learning Research*, 25(200):1–91, 2024.

Rahul Singh, Qinsheng Zhang, and Yongxin Chen. Improving robustness via risk averse distributional reinforcement learning. In *Learning for Dynamics and Control*, pp. 958–968. PMLR, 2020.

Elena Smirnova, Elvis Dohmatob, and Jérémie Mary. Distributionally robust reinforcement learning. *arXiv preprint arXiv:1902.08708*, 2019.

Aviv Tamar, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. Policy gradient for coherent risk measures. *Advances in neural information processing systems*, 28, 2015.

Emanuel Todorov. General duality between optimal control and estimation. In *2008 47th IEEE conference on decision and control*, pp. 4286–4292. IEEE, 2008.

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. doi: 10.1109/IROS.2012.6386109.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Herke Van Hoof, Nutan Chen, Maximilian Karl, Patrick Van Der Smagt, and Jan Peters. Stable reinforcement learning with autoencoders for tactile and visual data. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 3928–3934. IEEE, 2016.

Shengbo Wang, Nian Si, Jose Blanchet, and Zhengyuan Zhou. A finite sample complexity bound for distributionally robust q-learning. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 3370–3398. PMLR, 2023.

Shengbo Wang, Nian Si, Jose Blanchet, and Zhengyuan Zhou. Sample complexity of variance-reduced distributionally robust q-learning. *Journal of Machine Learning Research*, 25(341):1–77, 2024.

Yue Wang and Shaofeng Zou. Online robust reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 34:7193–7206, 2021.

Yue Wang and Shaofeng Zou. Policy gradient method for robust reinforcement learning. In *International conference on machine learning*, pp. 23484–23526. PMLR, 2022.

Hua Wei, Deheng Ye, Zhao Liu, Hao Wu, Bo Yuan, Qiang Fu, Wei Yang, and Zhenhui Li. Boosting offline reinforcement learning with residual generative modeling. *arXiv preprint arXiv:2106.10411*, 2021.

Peter Whittle. Risk-sensitive linear/quadratic/gaussian control. *Advances in Applied Probability*, 13 (4):764–777, 1981.

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

Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8753–8760, 2022.

Huan Xu and Shie Mannor. Distributionally robust markov decision processes. *Advances in Neural Information Processing Systems*, 23, 2010.

Zaiyan Xu, Kishan Panaganti, and Dileep Kalathil. Improved sample complexity bounds for distributionally robust reinforcement learning. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 9728–9754. PMLR, 2023.

Pengqian Yu and Huan Xu. Distributionally robust counterpart in markov decision processes. *IEEE Transactions on Automatic Control*, 61(9):2538–2543, 2015.

Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems*, 33:21024–21037, 2020.

Ruida Zhou, Tao Liu, Min Cheng, Dileep Kalathil, PR Kumar, and Chao Tian. Natural actor-critic for robust reinforcement learning with function approximation. *Advances in neural information processing systems*, 36:97–133, 2023.

Zhengqing Zhou, Zhengyuan Zhou, Qinxun Bai, Linhai Qiu, Jose Blanchet, and Peter Glynn. Finite-sample regret bound for distributionally robust offline tabular reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3331–3339. PMLR, 2021.

Zhengyuan Zhou, Michael Bloem, and Nicholas Bambos. Infinite time horizon maximum causal entropy inverse reinforcement learning. *IEEE Transactions on Automatic Control*, 63(9):2787–2802, 2018. doi: 10.1109/TAC.2017.2775960.

Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

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

Appendix

Contents

The Use of Large Language Models. The authors use Large Language Models (LLMs) to assist with grammar checking and language polishing in this submission. LLMs do not play a significant role in research ideation or writing to the extent that they could be regarded as a contributor.

A Discussion

A.1 Necessity of Generative Model

In this section, we discuss why other model-free methods are not applicable in *KL divergence constrained uncertainty set* and why a generative model (VAE) is necessary. Empirical risk minimization (ERM) is a method that minimizes the empirical loss estimation using sampled data, which has been extensively used in machine learning literature. However, it is not applicable in our case due to the non-linearity of the dual formulation and the Bellman operator. In our algorithm, we want to apply operator $\mathcal{T}^{\pi}_{\delta,g^\star}$ where $g^\star = \operatorname{argsup}_{g\in\mathcal{G}} \mathbb{E}_{(s,a)\sim\mathcal{D}}\Big[f\big((s,a),g(s,a)\big)\Big]$. Denote the objective function as

$$\begin{aligned} J(g) :=& \mathbb{E}_{(s,a)\sim\mathcal{D}}\Big[f\big((s,a),g(s,a)\big)\Big] \\ =& \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[-g(s,a)\log\left(\mathbb{E}_{p^0_{s,a}}\left[\exp\left(\frac{-V(s')}{g(s,a)}\right)\right]\right) - g(s,a)\delta\right]. \end{aligned} \tag{21}$$

To obtain a consistent estimator of (21), we encounter the well-known *double-sampling issue* (Baird et al., 1995) caused by the nonlinearity between inner and outer expectations. Specifically, to approximate the inner expectation term $\mathbb{E}_{p^0_{s,a}}[\exp(-V(s')/g(s,a))]$, the dataset $\mathcal{D}$ need to be split into two disjoint parts, $\mathcal{D}_{\text{outer}}$ and $\mathcal{D}_{\text{inner}}$. For each $(s,a)\in\mathcal{D}_{\text{inner}}$, we aggregate the corresponding samples starting from (s,a) contained in $\mathcal{D}_{\text{outer}}$, denoted by $\mathcal{D}_{(s,a)}$, and the empirical risk of (21) becomes

$$\widehat{J}(g) := \frac{1}{|\mathcal{D}_{\text{out}}|}\sum_{(s,a,s')\in\mathcal{D}_{\text{out}}}\left[-g(s,a)\log\left(\frac{1}{|\mathcal{D}_{(s,a)}|}\sum_{(\bar{s},\bar{a},\bar{s}')\in\mathcal{D}_{(s,a)}}\exp\left(\frac{-V(\bar{s}')}{g(s,a)}\right)\right) - g(s,a)\delta\right]. \tag{22}$$

However, in continuous state–action spaces, it is nearly impossible to revisit the exact same state–action pair, leading to $\mathcal{D}_{(s,a)} = \emptyset$.

Note that this issue does not come from the functional optimization technique we use, but from the structure of the dual formulation of the Bellman equation under the KL-based uncertainty set. In contrast, this problem does not occur in the TV-based dual formulation due to its linear structure (Panaganti et al., 2022). Specifically, if we remove the functional approximation and use the exact dual formulation of the DR soft Bellman operator to design an algorithm, the same double-sampling issue occurs in finding the empirical risk of the following Bellman residual:

$$\begin{aligned} \mathcal{L}_Q :=& \mathbb{E}_{(s,a)\in\mathcal{D}}\left[Q(s,a) - \mathcal{T}^{\pi}_{\delta}Q(s,a)\right] \\ =& \mathbb{E}_{(s,a)\in\mathcal{D}}\left[Q(s,a) - \mathbb{E}[r] - \gamma\cdot\sup_{\beta\geq 0}\left\{-\beta\log\left(\mathbb{E}_{p^0_{s,a}}\left[\exp\left(\frac{-V(s')}{\beta}\right)\right]\right) - \beta\delta\right\}\right]. \end{aligned} \tag{23}$$

In other literature introducing distributionally robust algorithms under KL uncertainty set, this difficulty is overcome by using a Monte-Carlo related method (Liu et al., 2022; Wang et al., 2023), estimating nominal distributions from transition frequencies (Wang et al., 2024), or directly estimating the expected value under nominal distributions (Liang et al., 2024). None of these methods is applicable to continuous space offline RL tasks.

A.2 Algorithm Details

In this section, we present a detailed description of the DR-SAC algorithm. In our algorithm, we use neural networks $V_\psi(s)$, $Q_\theta(s,a)$ and $\pi_\phi(a\mid s)$ to approximate the value function, the Q-function and the stochastic policy, respectively, with ψ,θ,ϕ as the network parameters. We also utilize target network $V_{\bar{\psi}}(s)$ and $Q_{\bar{\theta}}(s,a)$, where parameters $\bar{\psi}$ and $\bar{\theta}$ are the exponential moving average of respective network weights. Similar to SAC-v1 algorithm (Haarnoja et al., 2018a), the loss function of V-network is

$$J_V(\psi) = \mathbb{E}_{s\sim\mathcal{D}}\left[\frac{1}{2}\left(V_\psi(s) - \mathbb{E}_{a\sim\pi_\phi}\left[Q_{\bar{\theta}}(s,a) - \alpha\log\pi_\phi(a\mid s)\right]\right)^2\right]. \tag{24}$$

As introduced in Section 3.3, in our algorithm, we modify the loss function of Q-network to

$$J_Q^{\text{DR}}(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(s,a) - \mathcal{T}^{\pi}_{\delta,\widetilde{g}^\star} Q_\theta(s,a)\right)^2\right],$$

where

$$\begin{aligned}\widetilde{g}^\star &= \operatorname*{argsup}_{g\in\mathcal{G}_\eta} \mathbb{E}_{(s,a)\in\mathcal{D}}\left[\widetilde{f}((s,a),g(s,a))\right] \\ &= \operatorname*{argsup}_{g\in\mathcal{G}_\eta} \mathbb{E}_{(s,a)\in\mathcal{D}}\left[-\beta\log\left(\mathbb{E}_{\widetilde{P}^0_{s,a}}\left[\exp\left(\frac{-V_{\bar{\psi}}(s')}{\beta}\right)\right]\right) - \beta\delta\right].\end{aligned} \tag{25}$$

Optimal dual function $\widetilde{g}^\star$ can be found with backpropagation through η. We also keep the assumption of policy network in the standard SAC algorithm by reparameterizing the policy using a neural network transformation $a = f_\phi(\epsilon; s)$, where ϵ is an input noise vector sampled from a spherical Gaussian. The loss of policy is

$$J_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D},\epsilon\sim\mathcal{N}}\Big[\alpha\log\pi_\phi(f_\phi(\epsilon;s)\mid s) - Q_{\bar{\theta}}(s, f_\phi(\epsilon;s))\Big]. \tag{26}$$

In the SAC-v2 algorithm (Haarnoja et al., 2018b), the authors propose an automated entropy temperature adjustment method by using an approximate solution to a constrained optimization problem. The loss of temperature is

$$J(\alpha) = \mathbb{E}_{a\sim\pi_\phi}\left[-\alpha\log\pi_\phi(a\mid s) - \alpha\bar{\mathcal{H}}\right], \tag{27}$$

where $\bar{\mathcal{H}}$ is the desired minimum expected entropy and is usually implemented as the dimensionality of the action space.

In addition, we incorporate the VAE model into our algorithm. VAE is one of the most popular methods to learn complex distributions and has shown superior performance in generating different types of data. In the DR-SAC algorithm, we use VAE to learn the transition function $P^0(s'\mid s,a)$ by modeling the conditional distribution of next states. It assumes a standard normal prior over the latent variable, $p(z) = \mathcal{N}(0,I)$. The encoder maps (s,a,s') to an approximate posterior $q(z\mid s,a,s')$, and the decoder reconstructs s' from the latent sample z and input (s,a). The training loss is the evidence lower bound (ELBO):

$$J_{\text{VAE}}(\varphi) = \mathbb{E}_{q(z\mid s,a,s')}\left[\|s' - \hat{s}'\|^2\right] + D_{\text{KL}}\left(q(z\mid s,a,s')\,\|\,\mathcal{N}(0,I)\right), \tag{28}$$

where $\hat{s}'$ are the reconstructed states from the decoder.

B Proofs

B.1 Proof of Proposition 3.3

We first provide an established result in DRO to compute the worst-case expectation under perturbation in a KL-divergence constrained uncertainty set.

Lemma B.1 (Hu & Hong (2013), Theorem 1)**.** *Suppose $G(X)$ has a finite moment generating function in the neighborhood of zero. Then for any $\delta > 0$,*

$$\sup_{P:D_{KL}(P\|P_0)\le\delta} \mathbb{E}_P[G(X)] = \inf_{\beta\ge 0}\left\{\beta\log\left(\mathbb{E}_{P_0}\left[\exp\left(\frac{G(X)}{\beta}\right)\right]\right)+\beta\delta\right\} \tag{29}$$

Proof of Proposition 3.3.

$$\begin{aligned}
\mathcal{T}^{\delta}_{\pi}Q(s,a) &= \mathbb{E}[r] + \gamma\cdot\inf_{p\in\mathcal{P}_{s,a}(\delta)}\left\{\mathbb{E}_{s'\sim p(\cdot|s,a)}\left[\mathbb{E}_{a'\sim\pi(\cdot|s')}[Q(s',a')-\alpha\log\pi(a'|s')]\right]\right\}\\
&= \mathbb{E}[r] - \gamma\cdot\sup_{p\in\mathcal{P}_{s,a}(\delta)}\left\{\mathbb{E}_{s'\sim p(\cdot|s,a)}[-V(s')]\right\}\\
&= \mathbb{E}[r] - \gamma\cdot\inf_{\beta\ge 0}\left\{\beta\log\left(\mathbb{E}_{s'\sim p^0(\cdot|s,a)}\left[\exp\left(\frac{-V(s')}{\beta}\right)\right]\right)+\beta\delta\right\} \quad \text{(Lemma B.1)}\\
&= \mathbb{E}[r] + \gamma\cdot\sup_{\beta\ge 0}\left\{-\beta\log\left(\mathbb{E}_{s'\sim p^0(\cdot|s,a)}\left[\exp\left(\frac{-V(s')}{\beta}\right)\right]\right)-\beta\delta\right\}
\end{aligned}$$

To apply Lemma B.1, let $P = p(\cdot|s,a)$, $P_0 = p^0(\cdot|s,a)$, and $G(X) = G(s') = -V(s')$. As stated in Section 2.1, the rewards $r = R(s,a)$ are bounded, and the discount factor $\gamma\in[0,1)$. From the assumption that $Q(s,a)$ is bounded, we know $V(s')$ is bounded as well. This implies that $G(s') = -V(s')$ has a finite moment generating function (MGF) under the nominal distribution $p^0(\cdot|s,a)$, i.e., $\mathbb{E}_{s'\sim p^0(\cdot|s,a)}[e^{\lambda G(s')}] < \infty$, for λ in a neighborhood of zero. This ensures that $G(s')$ has a finite MGF under P_0 as required by Lemma B.1. ☐

B.2 Proof of Proposition 3.4

Before providing the proof of Proposition 3.4, we present the optimality conditions of Lemma B.1.

Lemma B.2 (Hu & Hong (2013), Proposition 2)**.** *Let $\beta^\star$ be an optimal solution of the optimization problem in* (29)*. Let $H = \operatorname{esssup}_{X\sim P_0} G(X)$ and $\kappa = \mathbb{P}_{X\sim P_0}(G(X) = H)$. Suppose the assumption in Lemma B.1 still holds, then $\beta^\star = 0$ or $G(X)$ has a finite moment generating function at $1/\beta^\star$. Moreover, $\beta^\star = 0$ if and only if $H < \infty$, $\kappa > 0$ and $\log\kappa + \delta \ge 0$.*

This lemma tells us the optimal solution is unique when $\beta^\star = 0$. This happens if and only if there is a large enough probability mass on the finite essential supremum of X, under the distribution center P_0. We use this lemma to discuss either $\beta^\star = 0$ or $\beta^\star > 0$ in the following proof.

Proof of Proposition 3.4. Similar to the standard convergence proof of policy evaluation, we want to prove that the operator $\mathcal{T}^{\pi}_{\delta}$ is a γ-contraction mapping. Suppose there are two mappings $Q_{1,2} : \mathcal{S}\times\mathcal{A}\to\mathbb{R}$ and define $V_i = \mathbb{E}_{a\sim\pi}[Q_i(s,a)] - \alpha\mathcal{H}(\pi(s))$, $i = 1,2$. For any state $s\in\mathcal{S}$, we have

$$|V_1(s) - V_2(s)| = |\mathbb{E}_{a\sim\pi}[Q_1(s,a) - Q_2(s,a)]| \le \|Q_1 - Q_2\|_\infty.$$

Thus, $\|V_1 - V_2\|_\infty \le \|Q_1 - Q_2\|_\infty$.

Next, for any $\beta > 0$ and (s,a) fixed, define function

$$F_\beta(V) := -\beta\log\mathbb{E}_{p^0_{s,a}}\left[\exp\left(-\frac{V(s')}{\beta}\right)\right] - \beta\delta.$$

Let $\|V_1 - V_2\|_\infty = d$. Then for any $s'\in\mathcal{S}$, $V_2(s') - d \le V_1(s') \le V_2(s') + d$. After exponential, expectation, and logarithm operations, monotonicity is preserved. We have

$$\begin{aligned}
-\beta\log\mathbb{E}_{p^0_{s,a}}\left[\exp\left(-\frac{V_2(s')}{\beta}\right)\right] - d \le &-\beta\log\mathbb{E}_{p^0_{s,a}}\left[\exp\left(-\frac{V_1(s')}{\beta}\right)\right]\\
\le &-\beta\log\mathbb{E}_{p^0_{s,a}}\left[\exp\left(-\frac{V_2(s')}{\beta}\right)\right] + d.
\end{aligned}$$

This gives us $|F_\beta(V_1) - F_\beta(V_2)| \le \|V_1 - V_2\|_\infty$.

Lastly, we reformulate DR soft Bellman operator as $\mathcal{T}_\delta^\pi Q(s,a) = \mathbb{E}[r] + \gamma \cdot \sup_{\beta \ge 0} F_\beta(V)$. Let $\beta_i^\star$ be an optimal solution of $\sup_{\beta\ge 0} F_\beta(V_i),\ i = 1,2$. From Lemma B.2, we know $\beta_i^\star$ is unique when $\beta_i^\star = 0$ is optimal. And the optimal value is the essential infimum H_i when $\beta_i^\star = 0$. We want to show $|F_{\beta_1^\star}(V_1) - F_{\beta_2^\star}(V_2)|$ is bounded in all cases of $\beta_i^\star$.

- Case 1: $\beta_1^\star = \beta_2^\star = 0$.

  In this case, the optimal value is the essential infimum value for both V_i. We have

$$|F_{\beta_1^\star}(V_1) - F_{\beta_2^\star}(V_2)| = \left| \operatorname*{essinf}_{s' \sim P^0_{s,a}} V_1(s') - \operatorname*{essinf}_{s' \sim P^0_{s,a}} V_2(s') \right| \le \|V_1 - V_2\|_\infty.$$

  The last inequality holds because monotonicity is preserved after taking the essential infimum.

- Case 2: $\beta_1^\star = 0,\ \beta_2^\star > 0$, WLOG.

  In this case, we know from optimality that

$$H_1 = \operatorname*{essinf}_{s' \sim P^0_{s,a}} V_1(s') \ge F_{\beta_2^\star}(V_1),\ H_2 = \operatorname*{essinf}_{s' \sim P^0_{s,a}} V_2(s') \le F_{\beta_2^\star}(V_2).$$

  Then we have

$$\begin{aligned} H_1 - F_{\beta_2^\star}(V_2) &\le H_1 - H_2 \le \|V_1 - V_2\|_\infty, \\ F_{\beta_2^\star}(V_2) - H_1 &\le F_{\beta_2^\star}(V_2) - F_{\beta_2^\star}(V_1) \le \|V_1 - V_2\|_\infty. \end{aligned}$$

  Thus, $|F_{\beta_1^\star}(V_1) - F_{\beta_2^\star}(V_2)| = |H_1 - F_{\beta_2^\star}(V_2)| \le \|V_1 - V_2\|_\infty$.

- Case 3: $\beta_1^\star > 0,\ \beta_2^\star > 0$.

  Suppose $F_{\beta_1^\star}(V_1) \le F_{\beta_2^\star}(V_2)$, WLOG. Then

$$|F_{\beta_1^\star}(V_1) - F_{\beta_2^\star}(V_2)| = F_{\beta_2^\star}(V_2) - F_{\beta_1^\star}(V_1) \le F_{\beta_2^\star}(V_2) - F_{\beta_2^\star}(V_1) \le \|V_1 - V_2\|_\infty,$$

  where the first inequality comes from the optimality of $\beta_1^\star$.

Thus for any (s,a) pair, we have we

$$\begin{aligned} |\mathcal{T}_\delta^\pi Q_1(s,a) - \mathcal{T}_\delta^\pi Q_2(s,a)| =& \gamma \cdot \left| \sup_{\beta_1 \ge 0} F_{\beta_1}(V_1) - \sup_{\beta_2 \ge 0} F_{\beta_2}(V_2) \right| \\ \le& \gamma \cdot \|V_1 - V_2\|_\infty \\ \le& \gamma \cdot \|Q_1 - Q_2\|_\infty. \end{aligned}$$

Since $\mathcal{T}_\delta^\pi$ is a γ-contraction, using Banach Fixed-Point Theorem, the sequence $\{Q^k\}$ convergences to the unique fixed-point $\mathcal{T}_\delta^\pi$. From Iyengar (2005); Xu & Mannor (2010), we know this fixed point is the DR soft Q-value. $\square$

B.3 Proof of Proposition 3.5

Proof. Given $\pi_k \in \Pi$, let $Q^{\pi_k}_{\mathcal{M}_\delta}$ and $V^{\pi_k}_{\mathcal{M}_\delta}$ be the corresponding DR soft Q-function and value function. Denote the function for determining the new policy as

$$J_\pi(\pi'(\cdot \mid s)) := D_{\mathrm{KL}} \left(\pi'(\cdot \mid s)) \middle\| \exp\left(\frac{1}{\alpha} Q^\pi_{\mathcal{M}_\delta}(s,\cdot) - \log Z^{\pi_k}(s) \right) \right). \tag{30}$$

According to Equation (12), $\pi_{k+1} = \operatorname{argmin}_{\pi' \in \Pi} J_{\pi_k}(\pi')$ and $J_{\pi_k}(\pi_{k+1}) \le J_{\pi_k}(\pi_k)$. Hence

$$\begin{aligned} &\mathbb{E}_{a\sim\pi_{k+1}}[\alpha \log \pi_{k+1}(a \mid s) - Q^{\pi_k}_{\mathcal{M}_\delta}(s,\cdot) + \alpha \log Z^{\pi_k}(s)] \\ \le& \mathbb{E}_{a\sim\pi_k}[\alpha \log \pi_k + (a \mid s) - Q^{\pi_k}_{\mathcal{M}_\delta}(s,\cdot) + \alpha \log Z^{\pi_k}(s)], \end{aligned}$$

and after deleting $Z^{\pi_k}(s)$ on both sides, the inequality is reformulated to

$$\mathbb{E}_{a\sim\pi_{k+1}}\left[Q^{\pi_k}_{\mathcal{M}_\delta}(s,\cdot) - \alpha \log \pi_{k+1}(a \mid s) \right] \ge V^{\pi_k}_{\mathcal{M}_\delta}(s).$$

Next, consider the DR soft Bellman equation:

$$\begin{aligned}
Q^{\pi_k}_{\mathcal{M}_\delta}(s,a) =&\mathbb{E}[r] + \gamma \cdot \inf_{p_{s,a}\in\mathcal{P}_{s,a}(\delta)} \left\{\mathbb{E}_{s'\sim p_{s,a}}\left[V^{\pi_k}_{\mathcal{M}_\delta}(s')\right]\right\} \\
\leq&\mathbb{E}[r] + \gamma \cdot \inf_{p_{s,a}\in\mathcal{P}_{s,a}(\delta)} \left\{\mathbb{E}_{s'\sim p_{s,a}}\left[\mathbb{E}_{a'\sim\pi_{k+1}}\left[Q^{\pi_k}_{\mathcal{M}_\delta}(s',a') - \alpha\log\pi_{k+1}(a'\mid s')\right]\right]\right\} \\
=&\mathcal{T}^{\pi_{k+1}}_\delta\left(Q^{\pi_k}_{\mathcal{M}_\delta}\right)(s,a) \\
&\vdots \\
\leq&Q^{\pi_{k+1}}_{\mathcal{M}_\delta}(s,a),\ \forall(s,a)\in\mathcal{S}\times\mathcal{A}
\end{aligned} \tag{31}$$

where operator $\mathcal{T}^{\pi_{k+1}}_\delta$ is repeatedly applied to $Q^{\pi_k}_{\mathcal{M}_\delta}$ and its convergence is guaranteed by Proposition 3.4. □

B.4 Proof of Theorem 3.6

Proof. Let π_k be the policy at iteration k. By Proposition 3.5, $Q^{\pi_k}_{\mathcal{M}_\delta}$ is non-decreasing with k. Since function $Q^{\pi_k}_{\mathcal{M}_\delta}$ is bounded by $(R_{\max} + \alpha\log|\mathcal{A}|)/(1-\gamma)$, sequence $\{Q^{\pi_k}_{\mathcal{M}_\delta}\}$ converges. Thus policy sequence $\{\pi_k\}$ convergences to some $\pi^\star$. It remains to show that $\pi^\star$ is indeed optimal. According to Equation (12), $J_{\pi^\star}(\pi^\star) \leq J_{\pi^\star}(\pi)$, $\forall\pi\in\Pi$. Using the same argument in proof of Proposition 3.5, we can show that $Q^{\pi}_{\mathcal{M}_\delta}(s,a) \leq Q^{\pi^\star}_{\mathcal{M}_\delta}(s,a)$ for any $\pi\in\Pi$ and $(s,a)\in\mathcal{S}\times\mathcal{A}$. Hence $\pi^\star$ is an optimal policy. □

B.5 Proof of Proposition 3.7

Before providing the proof, we first introduce two technical lemmas. Specifically, Lemma B.4 establishes the *interchange of minimization and integration* property in decomposable spaces. This property has wide applications in replacing point-wise optimality conditions by optimization in a functional space (Shapiro, 2017; Panaganti et al., 2022).

Lemma B.3 (Rockafellar & Wets (2009), Exercise 14.29)**.** *Function $f:\Omega\times\mathbb{R}^n\mapsto\mathbb{R}$ (finite-valued) is a normal integrand if $f(\omega,x)$ is measurable in ω for each x and continuous in x for each ω.*

Lemma B.4 (Rockafellar & Wets (2009), Theorem 14.60, Exercise 14.61)**.** *Let $f:\Omega\times\mathbb{R}\mapsto\mathbb{R}$ (finite-valued) be a normal integrand. Let $\mathcal{M}(\Omega,\mathcal{A};\mathbb{R})$ be the space of all measurable functions $x:\Omega\to\mathbb{R}$, $\mathcal{M}_f$ be the collection of all $x\in\mathcal{M}(\Omega,\mathcal{A};\mathbb{R})$ with $\int_{\omega\in\Omega} f(\omega,x(\omega))\mu(d\omega)<\infty$. Then, for any space with $\mathcal{M}_f\subset\mathcal{X}\subset\mathcal{M}(\Omega,\mathcal{A};\mathbb{R})$, we have*

$$\inf_{x\in\mathcal{X}}\int_{\omega\in\Omega} f(\omega,x(\omega))\mu(d\omega) = \int_{\omega\in\Omega}\left(\inf_{x\in\mathbb{R}} f(\omega,x)\right)\mu(d\omega).$$

Proof of Proposition 3.7. First we want to prove $\beta^\star = \operatorname{argsup}_{\beta\geq 0} f((s,a),\beta)$ is bounded in interval $\mathcal{I}_\beta := \left[0, \frac{R_{\max}+\alpha\log|\mathcal{A}|}{(1-\gamma)\delta}\right]$ for any $(s,a)\in\mathcal{S}\times\mathcal{A}$. Rewriting the optimization problem to its primal form, it is clear that

$$f((s,a),\beta^\star) = \inf_{p_{s,a}\in\mathcal{P}_{s,a}(\delta)}\mathbb{E}\left[V(s')\right]\geq 0.$$

When β is greater than $\frac{R_{\max}+\alpha\log|\mathcal{A}|}{(1-\gamma)\delta}$, it can never be optimal since

$$\begin{aligned}
f((s,a),\beta) =& -\beta\log\left(\mathbb{E}_{p^0_{s,a}}\left[\exp\left(-\frac{V(s')}{\beta}\right)\right]\right) - \beta\delta \\
\leq& -\beta\log\left(\exp\left(-\frac{R_{\max}+\alpha\log|\mathcal{A}|}{(1-\gamma)\beta}\right)\right) - \beta\delta \\
=&\frac{R_{\max}+\alpha\log|\mathcal{A}|}{1-\gamma} - \beta\delta < 0.
\end{aligned}$$

Now we know that $f((s,a),\beta)$ is a finite-valued function for each $(s,a)\in\mathcal{S}\times\mathcal{A}$ and $\beta\in\mathcal{I}_\beta$. Also, it is $\Sigma(\mathcal{S}\times\mathcal{A})$-measurable in $(s,a)\in\mathcal{S}\times\mathcal{A}$ for each $\beta\in\mathcal{I}_\beta$ and is continuous in β for each $(s,a)\in\mathcal{S}\times\mathcal{A}$. From Lemma B.3, we know that $f((s,a),\beta)$ is a normal integrand.

Moreover, all functions in $\mathcal{G}$ is upper bounded and measurable so $\mathcal{M}_f \subset \mathcal{G} \subset \mathcal{M}((\mathcal{S} \times \mathcal{A}), \Sigma(\mathcal{S} \times \mathcal{A}); \mathbb{R})$. Proposition 3.7 is a direct conclusion of Lemma B.4. $\square$

C EXPERIMENT DETAILS

C.1 MORE SETTING DETAILS

To allow for comparability of results, all tools were evaluated on equal-cost hardware, a Ubuntu 24.04 LTS system with one Intel(R) Core(TM) i7-6850K CPU, one NVIDIA GTX 1080 Ti GPU with 11 GB memory, and 64 GB RAM. All experiments use 12 CPU cores and 1 GPU.

We implement FQI and RFQI algorithms from `https://github.com/zaiyan-x/RFQI`. DDPG and CQL are implemented from the offline RL library d3rlpy (Seno & Imai, 2022).

Hyperparameter Selection Across all environments, we use $\gamma = 0.99$ for discount rate, $\tau = 0.005$ for both V and Q critic soft-update, $\alpha = 0.12$ as initial temperature, $|B| = 256$ for mini-batch size, $|\mathcal{D}| = 10^6$ for data buffer size. Actor, Q and V critic and VAE networks are multilayer perceptrons (MLPs) with $[256, 256]$ as hidden dimension. In the *HalfCheetah* and *Reacher* environments, we use two hidden layers in the actor and critic networks. All other networks have one hidden layer.

There are multiple learning rates in our algorithm. Learning rate for VAE network λ_φ is 5×10^{-5} in the *Pendulum* environment and 5×10^{-4} in others. In Step 5 of Algorithm 1, optimal function $\widetilde{g}^\star$ is found via backpropagation with learning rate λ_η. All other learning rates λ_ψ, λ_θ, λ_ϕ and λ_α are the same in each environment and represented by λ_ψ.

Value of learning rates λ_ψ and λ_η, number of Q-critics and latent dimensions in VAE are separately tuned in each environment and presented in Table 2.

Table 2: Hyper-parameters selection in SAC and DR-SAC algorithm training.

Environment	λ_ψ	λ_η	Q-Critic Number	latent dimensions
Pendulum	5×10^{-4}	5×10^{-5}	2	5
Cartpole	3×10^{-4}	5×10^{-4}	2	5
LunarLander	5×10^{-4}	5×10^{-4}	2	10
HalfCheetah	3×10^{-4}	5×10^{-5}	5	32
Reacher	3×10^{-4}	5×10^{-5}	5	10

Offline Dataset To ensure fairness in performance comparison, all models in each environment are trained on the same dataset. Each datasets contains 10^6 samples, generated by first training a behavior policy and applying the epsilon-greedy method. For most environments, the behavior policy is trained by the Twin Delayed DDPG (TD3, Fujimoto et al. (2018)) implemented from the d3rlpy offline RL library (Seno & Imai, 2022), while in the *Cartpole* environment we use SAC. To ensure a fair robustness evaluation, all models are trained to achieve the same performance (500, the maximum reward) under unperturbed conditions in *Cartpole*. Datasets generated by behavior policies trained by TD3 (SAC) are denoted as TD3-datasets (SAC-datasets) throughout this work. Additional details including the algorithm to train behavior policy, training steps and the random-action probability ϵ are presented in Table 3.

Table 3: Experiment details in dataset generation

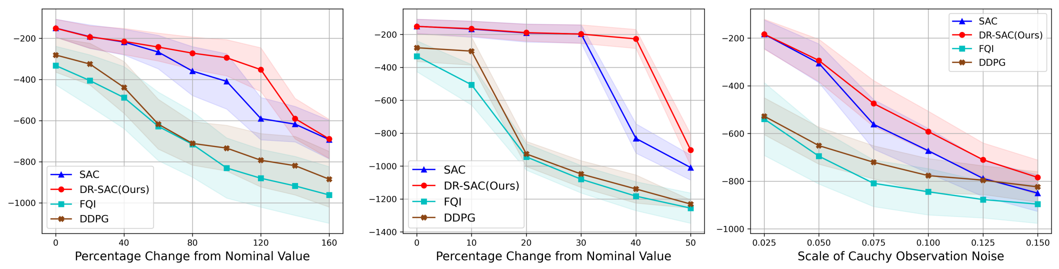
Environment	Behavior Policy Algorithm	Training Steps	Random-Action Probability ϵ
Pendulum	TD3	5×10^4	0.5
Cartpole	SAC	5×10^5	0.5
LunarLander	TD3	3×10^5	0.5
HalfCheetah	TD3	10^6	0.3
Reacher	TD3	10^6	0.3

C.2 EXTRA EXPERIMENT RESULTS

Pendulum In the *Pendulum* environment, we compare DR-SAC with SAC, FQI, and DDPG. All models are trained on the TD3-dataset. The robust algorithm RFQI does not perform well in this test, even when there is no perturbation. To evaluate the robustness of trained models, we change the environment parameters *length*, *mass*, and *gravity*, with nominal values as 1.0, 1.0 and 10.0 respectively. We grind search $\delta \in \{0.1, 0.2, \cdots, 1.0\}$ and find model under $\delta = 0.5$ have the best overall robustness.

DR-SAC shows consistent robustness improvement compared to all other algorithms. The performance under length perturbation is presented in Figure 1(a). In the mass perturbation test, DR-SAC has the best performance in all cases. For example, the average reward is over 40% higher than SAC when mass changes 120%. In Figure 2 (b), there is a notable gap between DR-SAC and SAC performance when gravity acceleration changes 40%.

To further show model performance under heavy-tailed perturbation, we also add Cauchy-distributed noise to state observations. The distribution of noise is defined as standard Cauchy distribution multiplied by a parameter *noise scale*. In Figure 2 (c), DR-SAC achieves consistent the best performance when *noise scale* increases.



(a) Mass Perturbation: pendulum mass change.

(b) Gravity Perturbation: gravity acceleration change.

(c) Observation Noise: Cauchy-distributed noise on nominal states.

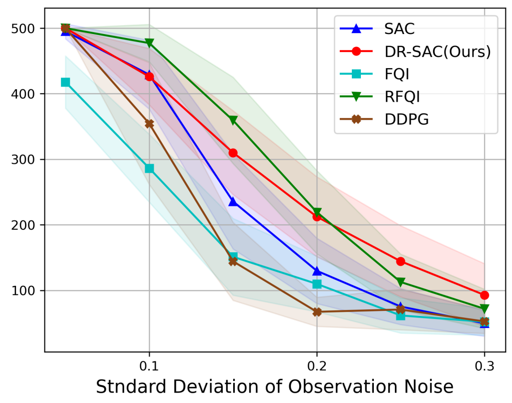
Figure 2: *Pendulum* results on TD3-dataset. The curves show the average reward of 50 episodes, shaded by ± 0.5 standard deviation.

Cartpole In the *Cartpole* environment, we compare the DR-SAC algorithm with non-robust algorithms SAC, DDPG, FQI, and robust algorithm RFQI. All algorithms are trained on the SAC-dataset. In our *Cartpole* environment, the force applied to the cart is continuous and determined by the actuator's action and parameter *force_mag*. The highest possible reward is 500 in each episode. To ensure fair comparison, all models are trained to have average rewards of 500 when no perturbation is added.

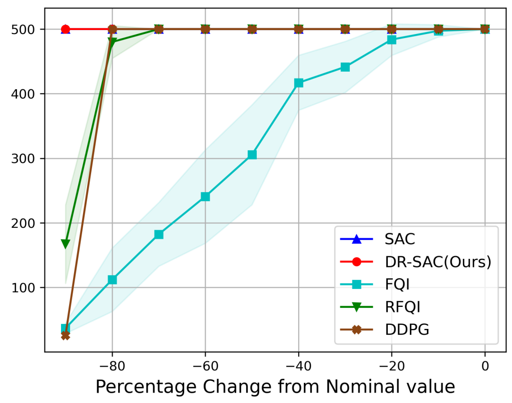
We test the robustness by introducing three changes to the environment: applying action perturbation, adding observation noise, and changing parameter *force_mag*. In the action perturbation test, the actuator takes random actions with different probabilities. In the observation perturbation test, noise with zero mean and different standard deviations is added to the nominal states in each step. The model parameter *force_mag* represents the unit force magnitude with the nominal value as 30.0. We grid search $\delta \in \{0.25, 0.5, 0.75, 1.0\}$ and find DR-SAC has the best performance when $\delta = 0.75$. We also use $\rho = 0.75$ to train the RFQI model.

In the *Cartpole* environment, DR-SAC has the best overall performance under three types of perturbation. The performance under action perturbation is presented in Figure 1(b), and DR-SAC has substantially better performance compared to RFQI. In the observation noise perturbation test in Figure 3(a), DR-SAC has performance improvement over 75% compared to non-robust algorithms SAC and DDPG when the standard deviation of noise is 0.2 and 0.3.

LunarLander In the *LunarLander* environment, we compare DR-SAC with non-robust algorithms SAC, CQL, FQI, and robust algorithm RFQI. All algorithms are trained on the TD3-dataset. In the *LunarLander* environment, the lander has main and side engines, and the actuator can control the throttle of the main engine. We change environment parameters *engine_power* (main and side engine power) and *wind_power* (magnitude of linear wind) to validate algorithm robustness. We grid search $\delta \in \{0.25, 0.5, 0.75, 1.0\}$ and find DR-SAC has the best performance when $\delta = 0.25$. We also use $\rho = 0.25$ to train the RFQI model.



(a) Observation Perturbation: gaussian noise added to nominal states.



(b) "Force_mag" Perturbation: model parameter *force_mag* change.

Figure 3: *Cartpole* results on SAC-dataset. The curves show the average reward of 50 episodes, shaded by ± 0.5 standard deviation.

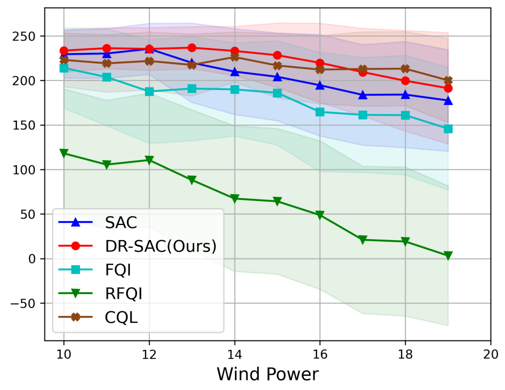


Figure 4: *LunarLander* results on TD3-dataset. The curves show the average reward of 50 episodes, shaded by ± 0.5 standard deviation.

Under all types of perturbations, DR-SAC shows superior robustness compared to other algorithms. The performance under *engine_power* perturbation is presented in Figure 1(c). In Figure 4, DR-SAC shows the highest average reward in most levels of wind perturbation. It is worth noting that the robust algorithm RFQI does not have an acceptable performance in this test, even compared to its non-robust counterpart FQI.

Reacher In the *Reacher* environment, we compare DR-SAC with non-robust algorithms SAC, FQI, CQL, and robust algorithm RFQI. All algorithms are trained on the TD3-dataset. In *Reacher* environment, the actuator controls a two-jointed robot arm to reach a target. We use *joint_damping* to denote the damping factor of both *joint0* and *joint1*, with default value as 1.0. We grind search $\delta \in \{0.1, 0.2, 0.3\}$ and find DR-SAC has the best performance when $\delta = 0.2$. We also use $\rho = 0.2$ to train the RFQI model.

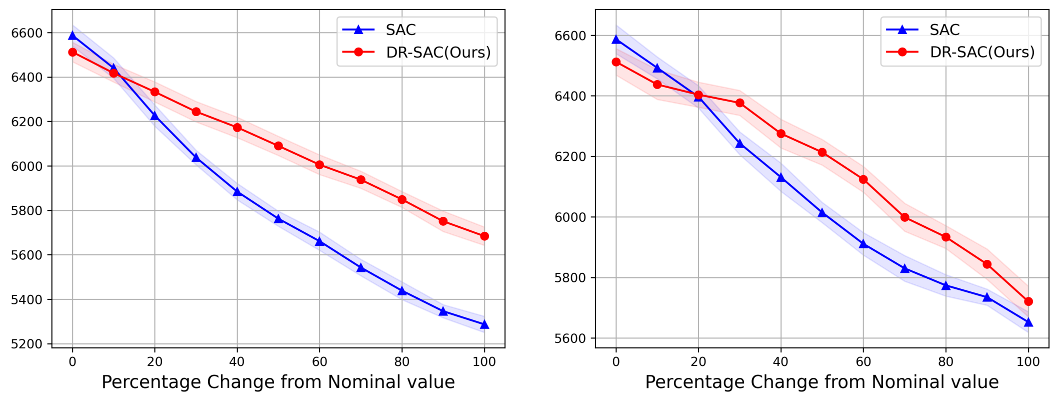
To test the robustness of all algorithms, we compare their performance after adding observation noise and changing parameters *joint_damping*. In the observation perturbation test, we add zero-mean Gaussian noise to the nominal state in dimensions $4 - 9$. The first 4 dimensions in state are trigonometric function values and are kept unperturbed. Performance under both perturbations is presented in Figure 1 (d) and (e). Moreover, in Figure 1 (e), the standard deviation regions were computed but omitted from the final plot because the overlapping shaded areas of multiple algorithms made the figure unreadable. We provide them in Table4.

Table 4: Standard Deviation of Model Performance under Damping Perturbation in *Reacher*

Joint Damping Value	SAC	DR-SAC	FQI	RFQI	CQL
2.0	1.452	1.459	1.638	1.509	1.396
3.0	1.534	1.540	1.754	1.601	1.456
4.0	1.631	1.628	1.865	1.699	1.529
5.0	1.735	1.721	1.969	1.802	1.614
6.0	1.841	1.819	2.057	1.910	1.711
7.0	1.946	1.925	2.135	2.018	1.816

HalfCheetah In the *HalfCheetah* environment, we compare DR-SAC with SAC baseline only due to the unsatisfactory performance of FQI and RFQI. All algorithms are trained on the TD3-dataset. In the *HalfCheetah* environment, the actuator controls a cat-like robot consisting of 9 body parts and 8 joints to run. We use *front_stiff* and *front_damping* to denote the stiffness and damping factor of joint *fthigh*, *fshin*, and *ffoot*. Also, *back_stiff* and *back_damping* can be denoted in a similar way. The default value of these parameters can be found through the environmental assets of Gymnasium MuJoCo in `https://github.com/Farama-Foundation/Gymnasium/blob/main/gymnasium/envs/mujoco/assets/half_cheetah.xml`. We grid search $\delta \in \{0.1, 0.2, 0.3\}$ and find DR-SAC has the best performance when $\delta = 0.2$.

Performance of *back_damping* test is presented in Figure 1(f). Combining it with Figure 5, we can see DR-SAC has notable robustness improvement across all perturbation tests. For example, in *front_stiff* perturbation test, DR-SAC achieves an improvement as much as 10% when the change is 80%.



(a) Front Stiffness Perturbation (b) Front Damping Perturbation

Figure 5: *HalfCheetah* results on TD3-dataset. The curves show the average reward of 50 episodes, shaded by ± 0.5 standard deviation.

C.3 Ablation Study Details

C.3.1 Training Efficiency of DR-SAC

In this section, we want to show that DR-SAC with functional optimization finds a good balance between efficiency and accuracy. We compare training time and robustness of Algorithm 1, DR-SAC without functional optimization, and robust algorithm RFQI, to show our DR-SAC algorithm has the best overall performance.

Balance in Functional Approximation We first introduce DR-SAC algorithm without functional optimization. Most steps are the same as Algorithm 1, instead of following modifications. Step 5 in Algorithm 1 is removed. Q-network loss is replaced by

$$J_Q^{\text{DR\_acc}} = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[Q^\pi_{\mathcal{M}}(s,a) - \widetilde{\mathcal{T}}^\pi_\delta Q^\pi_{\mathcal{M}_\delta}(s,a)\right]^2, \tag{32}$$

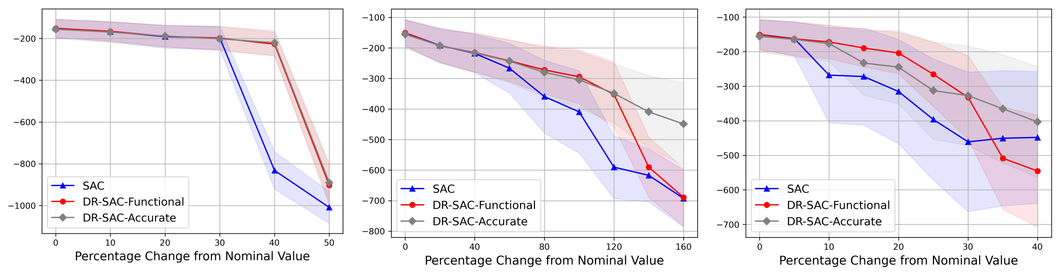
where $\widetilde{\mathcal{T}}_\delta^\pi$ is the empirical version of $\mathcal{T}_\delta^\pi$ by replacing $p_{s,a}^0$ with $\widetilde{p}_{s,a}^0$. We call this modified algorithm *DR-SAC-Accurate* and call Algorithm 1 *DR-SAC-Functional* in this section.

We train SAC, *DR-SAC-Functional*, and *DR-SAC-Accurate* algorithms in *Pendulum* environment. The optimization problem in Equation (10) is a problem over scalar $\beta > 0$ and solved via *Scipy* for each (s, a) pair. Table 5 shows the training steps and time for three algorithms. We see training time of *DR-SAC-Accurate* is over 150 times longer than standard SAC and over 50 times longer than *DR-SAC-Functional*. Considering *Pendulum* environment is relatively simple, *DR-SAC-Accurate* algorithm is hard to utilize in large-scale problems.

Table 5: Training steps and time for three algorithms in *Pendulum*

Algorithm	Training Steps	Training Time (Minute)
SAC	10k	1.7
DR-SAC-Functional	10k	4.7
DR-SAC-Accurate	8k	260

Moreover, we test the robustness of three algorithms by comparing their average reward under different perturbations. To be specific, we change *Pendulum* environment parameters: *length*, *mass*, and *gravity*. *DR-SAC-Functional* and *DR-SAC-Accurate* are trained with $\delta = 0.5$. Figure 6 shows that *DR-SAC-Functional* achieves comparable and even better performance under small-scale perturbation. For example, *DR-SAC-Functional* and *DR-SAC-Accurate* have almost the same performance under *gravity* perturbation in all test cases and *mass* perturbation test when change is less than 120%. In *length* perturbation test, *DR-SAC-Functional* has better performance when the change is less than 30%.



(a) Gravity Factor Perturbation. (b) Mass Factor Perturbation. (c) Length Factor Perturbation.

Figure 6: *Pendulum* results on TD3-dataset. Curves show average reward of 50 episodes, shaded by ± 0.5 standard deviation. Algorithms are SAC, DR-SAC with and without functional approximation.

Efficiency Comparison with RFQI In Section 4.2, existing DR-RL algorithm RFQI also shows comparable performance under some perturbations. In this paragraph, we want to show that DR-SAC requires much less training time than RFQI, improving its applicability to large scale problems. Table 1 lists the training time of SAC, DR-SAC, FQI, and RFQI algorithms in three testing environments. DR-SAC is demonstrated to be well-trained in at most 20% time required by RFQI. Compared with each non-robust baseline, the training time of DR-SAC is at most 360% of SAC, while RFQI requires $1000 - 1300\%$ more training time than FQI. In Figure 7, we provide a plot of performance changes against the training time in the *Reacher* environment, where RFQI is shown to be under-trained when the curve of DR-SAC converges.

Moreover, this efficiency improvement does not solely arise from the functional approximation step, but also from the inherent optimization efficiency in the loss function structure. The RFQI algorithm considers the RMDP framework with uncertainty sets defined by the TV distance and is empirically built on the BCQ algorithm. In RFQI, there exists a step similar to (20) to find the optimal functional under empirical measurement. Experimental results show that the efficiency gap arises from the number of GD steps in solving this optimization problem. RFQI sets the default GD steps as 1000 while DR-SAC achieves comparable robustness performance with only 5 steps. To further investigate, we vary the GD steps in RFQI to 5, 10 and 100 in the *LunarLander* environment and report the model

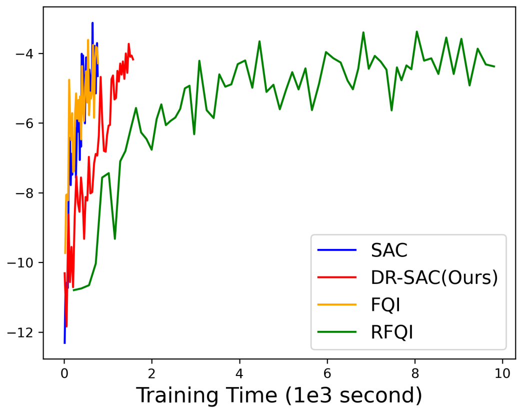


Figure 7: Average Reward of 20 Episodes over Training Time in *Reacher* Environment.

performance in the unperturbed environment. As shown in Table 6, performance drops sharply when RFQI uses fewer GD steps, indicating that the loss function structure in RFQI inherently leads to slower convergence and requires more optimization steps. In our framework, the choice of actor-critic based non-robust baseline, KL divergence induced uncertainty set and generative modeling in nominal distribution estimation together yields a more optimization-friendly formulation, contributing to our method's practical efficiency.

Table 6: GD steps, training time and performance in *LunarLander*

Algorithm	DR-SAC	RFQI	RFQI	RFQI	RFQI (Used)
GD Steps	5	5	10	100	1000
Training Time (min)	36	12	21	139	238
Performance	240.0	175.9	181.9	192.9	201.2

C.3.2 Robustness of VAE Model

A consistent challenge in DR-RL algorithm design is that unknown nominal distributions $p^0_{s,a}$ often appear in the loss function. In Section 3.2 and Appendix A.1, we review methods used in other model-free DR-RL algorithms and motivate the necessity of generative models in our setting. Although generative models inevitable introduce additional estimation error when constructing empirical measures $\tilde{p}^0_{s,a}$, our ablation studies demonstrate that DR-SAC is largely insensitive to the VAE modeling, therefore improving its applicability. In the *Pendulum* environment, where the state and action space dimensions are 3 and 1 respectively, we train DR-SAC with VAEs of latent dimensions $1, 5, 10, 20, 50$ and evaluate performance under perturbed pendulum mass. As shown in Figure 8, DR-SAC maintains superior robustness over the SAC baseline as long as the latent dimension lies within a reasonable range (between 5 and 20 in our experiments).

To demonstrate the choice of VAE over other generative models, we implemented Diffusion Probabilistic Models and Normalizing Flows as alternatives to the VAE in DR-SAC and conducted ablation studies on *Pendulum* and *Cartpole*. Model performance is provided in Figure 9 and 10. DR-SAC with Diffusion models achieved comparable robustness to the VAE in *Pendulum*. However, Flow-based models showed unstable performance even in unperturbed *Pendulum*environment. Crucially, the efficiency of sampling process with Diffusion models is a major bottleneck. Diffusion-based training is at least 4.5 times slower than VAE-based training.

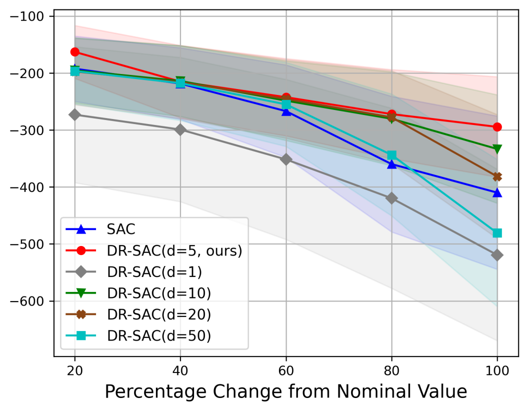
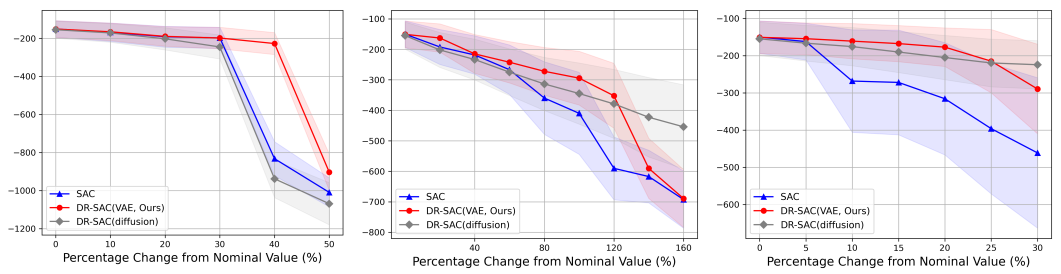
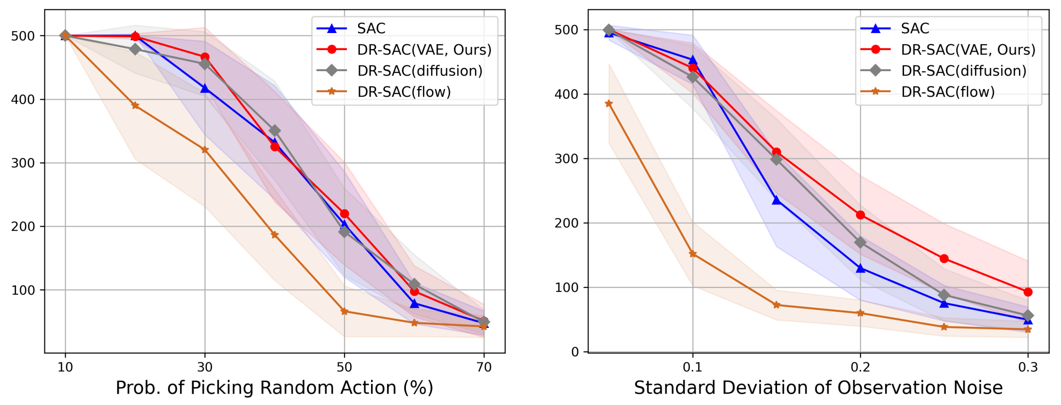


Figure 8: *Pendulum* results on TD3-dataset with mass perturbation and different VAE latent dimensions. The curves show the average reward of 50 episodes, shaded by ±0.5 standard deviation.



(a) Gravity Factor Perturbation. (b) Mass Factor Perturbation. (c) Length Factor Perturbation.

Figure 9: *Pendulum* results with different generative models on TD3-dataset. Curves show average reward of 50 episodes, shaded by ±0.5 standard deviation.



(a) Observation Perturbation: gaussian noise added to nominal states.

(b) "Force_mag" Perturbation: model parameter *force_mag* change.

Figure 10: *Cartpole* results with different generative models on SAC-dataset. The curves show the average reward of 50 episodes, shaded by ±0.5 standard deviation.

C.3.3 Usage of V-Network

In this section, we demonstrate that keeping the V-network in the SAC algorithm reduces the sensitivity on dataset distribution. As introduced in Appendix C.1, offline datasets in this work are generated by first training a behavior policy and applying the epsilon-greedy method to collect data. Experimental results shows that SAC without the V-network exhibits unstable performance when the behavior policy differs across datasets.

Our experiments are conducted in the *Pendulum* environment. We generate two datasets with behavior policy trained by an online version of SAC and TD3, denoted as SAC-dataset and TD3-dataset, respectively. Figure 11 presents the average reward of 20 episodes against training steps in four scenarios: SAC-dataset vs. TD3-dataset, SAC algorithm with vs. without V-network. Removing the V-network shows minor influence on offline SAC learning using SAC-dataset. However, for TD3-dataset, SAC with V-network achieves a stable average reward around -150 quickly, but the average reward of SAC without V-network fluctuates intensely and never exceeds -200. This validates that SAC with a V-network is less sensitive to behavior policy and dataset distribution.

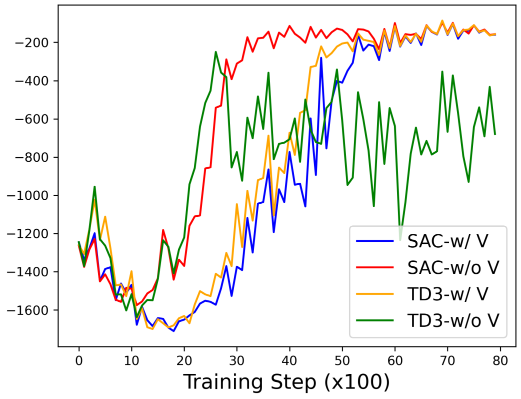


Figure 11: Average Reward of 20 Episodes over Training Step in *Pendulum* Environment.

D REGRET BOUND

Definition D.1. The distributionally robust regret $R_{\mathcal{M}_\delta}(\pi)$ of a policy $\pi \in \Pi$ is defined as:

$$R_{\mathcal{M}_\delta}(\pi) := \left\|V^\star_{\mathcal{M}_\delta} - V^\pi_{\mathcal{M}_\delta}\right\|_\infty .$$

For any policy π, the soft value and soft Q-functions satisfy:

$$V^\pi_{\mathcal{M}_\delta}(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q^\pi_{\mathcal{M}_\delta}(s,a) - \alpha \log \pi(a \mid s)\right].$$

The following inequality holds:

$$\left\|V^\star_{\mathcal{M}_\delta} - V^\pi_{\mathcal{M}_\delta}\right\|_\infty \le \left\|Q^\star_{\mathcal{M}_\delta} - Q^\pi_{\mathcal{M}_\delta}\right\|_\infty .$$

Based on the estimator $\hat{p}^0_{s,a}$, we define the corresponding estimate DR soft value function

$$\hat{V}^\pi_{\mathcal{M}_\delta}(s) = \inf_{\mathbf{p}\in\hat{\mathcal{P}}(\delta)} \mathbb{E}_{\mathbf{p}}\left[\sum_{t=1}^{\infty} \gamma^{t-1}\left(r_t + \alpha\cdot\mathcal{H}\left(\pi\left(s_t\right)\right)\right) \mid \pi, s_1 = s\right],$$

where $\hat{\mathcal{P}}_{s,a}(\delta) := \left\{p_{s,a} \in \Delta(|\mathcal{S}|) : D_{\mathrm{KL}}\left(p_{s,a}\|\hat{p}^0_{s,a}\right) \le \delta\right\}$. Similarly, the estimate DR soft Q-function is given by

$$\hat{Q}^\pi_{\mathcal{M}_\delta}(s,a) = \inf_{\mathbf{p}\in\hat{\mathcal{P}}(\delta)} \mathbb{E}_{\mathbf{p}}\left[r_1 + \sum_{t=2}^{\infty} \gamma^{t-1}\left(r_t + \alpha\cdot\mathcal{H}\left(\pi\left(s_t\right)\right)\right) \mid \pi, s_1 = s, a_1 = a\right].$$

Define $\hat{V}^\star_{\mathcal{M}_\delta} = \max_{\pi\in\Pi} \hat{V}^\pi_{\mathcal{M}_\delta}$ and $\hat{\pi}^\star_{\mathcal{M}_\delta} \in \operatorname{argmax}_{\pi\in\Pi} \hat{V}^\pi_{\mathcal{M}_\delta}$.

The estimate $\hat{\mathcal{T}}^\pi_\delta$ is defined as

$$\hat{\mathcal{T}}^\pi_\delta Q(s,a) = \mathbb{E}[r] + \gamma\cdot\sup_{\beta\ge 0}\left\{-\beta\log\left(\mathbb{E}_{\hat{p}^0_{s,a}}\left[\exp\left(\frac{-V\left(s'\right)}{\beta}\right)\right]\right) - \beta\delta\right\},$$

Assumption D.2. Assume $\mathrm{KL}(p^0_{s,a}\|\hat{p}^0_{s,a}) \le \varepsilon_1^2$ and $\mathrm{supp}(p^0_{s,a}) = \mathrm{supp}(\hat{p}^0_{s,a})$.

By Pinsker's inequality, $\mathrm{TV}(p^0_{s,a}, \hat{p}^0_{s,a}) \le \frac{1}{2}\sqrt{\mathrm{KL}(p^0_{s,a}\|\hat{p}^0_{s,a})} \le \frac{1}{2}\varepsilon_1$.

Bound of $\|\hat{\mathcal{T}}^\pi_\delta Q(s,a) - \mathcal{T}^\pi_\delta Q(s,a)\|$.

Lemma D.3. *Under Assumption D.2,*

$$\|\hat{\mathcal{T}}^\pi_\delta Q(s,a) - \mathcal{T}^\pi_\delta Q(s,a)\| \le 2\gamma\varepsilon_1 \frac{R_{\max} + \alpha\log|A|}{(1-\gamma)\delta} e^{(R_{\max}+\alpha\log|A|)/(1-\gamma)\underline{\beta}}.$$

Proof. As we defined in Section 3.2,

$$f((s,a),\beta) := -\beta\log\left(\mathbb{E}_{p^0_{s,a}}\left[e^{-V(s')/\beta}\right]\right) - \beta\delta, \quad \hat{f}((s,a),\beta) := -\beta\log\left(\mathbb{E}_{\hat{p}^0_{s,a}}\left[e^{-V(s')/\beta}\right]\right) - \beta\delta.$$

From (Xu 2023, Proposition 5), the maximums of $f((s,a),\beta)$ and $\hat{f}((s,a),\beta)$ are achieved at $\beta^\star, \hat{\beta}^\star \in [0, V_{\max}/\delta]$, that is,

$$\mathcal{T}^\pi_\delta Q(s,a) = \mathbb{E}[r] + \gamma\sup_{\beta\ge 0} f((s,a),\beta) = \mathbb{E}[r] + \gamma \sup_{\beta\in[0,V_{\max}/\delta]} f((s,a),\beta),$$
$$\hat{\mathcal{T}}^\pi_\delta Q(s,a) = \mathbb{E}[r] + \gamma\sup_{\beta\ge 0} \hat{f}((s,a),\beta) = \mathbb{E}[r] + \gamma \sup_{\beta\in[0,V_{\max}/\delta]} \hat{f}((s,a),\beta).$$

Hence,

$$\left|\hat{\mathcal{T}}^\pi_\delta Q(s,a) - \mathcal{T}^\pi_\delta Q(s,a)\right| \le \gamma \sup_{\beta\in[0,V_{\max}/\delta]} \left|\hat{f}((s,a),\beta) - f((s,a),\beta)\right|.$$

Note that $\mathrm{supp}(p^0_{s,a}) = \mathrm{supp}(\hat{p}^0_{s,a})$, which implies that $F_p(0) = \operatorname{essinf}_{s'\sim p^0_{s,a}} V(s') = \operatorname{essinf}_{s'\sim\hat{p}^0_{s,a}} V(s') = F_{\hat{p}}(0)$. Now, we can assume that the optimal $\beta^\star, \hat{\beta}^\star$ is achieved in $[\underline{\beta}, V_{\max}/\delta]$, where $\underline{\beta} = \min\{\beta^\star/2, \hat{\beta}^\star/2, 1/2\}$.

Then, we aim to bound the supremum of $\left|\hat{f}((s,a),\beta) - f((s,a),\beta)\right|$ over the interval $[\underline{\beta}, V_{\max}/\delta]$. Since $\log x \le x - 1$ when $x \ge 1$, we have

$$\begin{aligned}|\hat{f}((s,a),\beta) - f((s,a),\beta)| &= \beta\left|\log\left(\mathbb{E}_{\hat{p}^0_{s,a}}\left[e^{-V(s')/\beta}\right]\right) - \log\left(\mathbb{E}_{p^0_{s,a}}\left[e^{-V(s')/\beta}\right]\right)\right| \\ &\le \frac{V_{\max}}{\delta}\frac{\left|\mathbb{E}_{\hat{p}^0_{s,a}}\left[e^{-V(s')/\beta}\right] - \mathbb{E}_{p^0_{s,a}}\left[e^{-V(s')/\beta}\right]\right|}{\min\left\{\mathbb{E}_{\hat{p}^0_{s,a}}\left[e^{-V(s')/\beta}\right], \mathbb{E}_{p^0_{s,a}}\left[e^{-V(s')/\beta}\right]\right\}}.\end{aligned}$$

Since $\mathrm{TV}(p^0_{s,a}, \hat{p}^0_{s,a}) \le \varepsilon_1$ and $V_{\min} \ge 0$, we have

$$\begin{aligned}&\left|\mathbb{E}_{\hat{p}^0_{s,a}}\left[e^{-V(s')/\beta}\right] - \mathbb{E}_{p^0_{s,a}}\left[e^{-V(s')/\beta}\right]\right| = \left|\sum_{s'\in S}\left(\hat{p}^0_{s,a}(s') - p^0_{s,a}(s')\right)e^{-V(s')/\beta}\right| \\ &\le \sum_{s'\in S}\left|\hat{p}^0_{s,a}(s') - p^0_{s,a}(s')\right|e^{-V_{\min}/\beta} \le 2\mathrm{TV}(p^0_{s,a}, \hat{p}^0_{s,a}) \le \varepsilon_1.\end{aligned}$$

In addition,

$$\min\left\{\mathbb{E}_{\hat{p}^0_{s,a}}\left[e^{-V(s')/\beta}\right], \mathbb{E}_{p^0_{s,a}}\left[e^{-V(s')/\beta}\right]\right\} \ge e^{-V_{\max}/\underline{\beta}}.$$

Thus, we obtain

$$|\hat{f}((s,a),\beta) - f((s,a),\beta)| \le \varepsilon_1\frac{V_{\max}}{\delta}e^{V_{\max}/\underline{\beta}}, \quad \text{where } V_{\max} = \frac{R_{\max} + \alpha\log|A|}{1-\gamma}.$$

Combining this with the earlier inequality gives

$$\left\|\hat{\mathcal{T}}^{\pi}_{\delta}Q(s,a) - \mathcal{T}^{\pi}_{\delta}Q(s,a)\right\| \le \gamma\varepsilon_1\frac{R_{\max} + \alpha\log|A|}{(1-\gamma)\delta}e^{(R_{\max}+\alpha\log|A|)/(1-\gamma)\underline{\beta}} := \varepsilon_2.$$

$\square$

Bound of $\|\hat{Q}^{\pi}_{\mathcal{M}_\delta} - Q^{\pi}_{\mathcal{M}_\delta}\|$. For any $\pi \in \Pi$, let $Q^{k+1} = \mathcal{T}^{\pi}_{\delta}Q^k$, $\hat{Q}^{k+1} = \hat{\mathcal{T}}^{\pi}_{\delta}\hat{Q}^k$, and $\hat{Q}^0 = Q^0$. By Proposition 3.4, we know that Q^k will converge to the DR soft Q-value $Q^{\pi}_{\mathcal{M}_\delta}$, which is the fixed point of T^{π}_{δ}. That is, $T^{\pi}_{\delta}Q^{\pi}_{\mathcal{M}_\delta} = Q^{\pi}_{\mathcal{M}_\delta}$ and $Q^k \to Q^{\pi}_{\mathcal{M}_\delta}$. Similarly, there exists a fixed point of $\hat{T}^{\pi}_{\delta}$ such that $\hat{T}^{\pi}_{\delta}\hat{Q}^{\pi}_{\mathcal{M}_\delta} = \hat{Q}^{\pi}_{\mathcal{M}_\delta}$ and $\hat{Q}^k \to \hat{Q}^{\pi}_{\mathcal{M}_\delta}$. Then

$$\begin{aligned}\|\hat{Q}^{\pi}_{\mathcal{M}_\delta} - Q^{\pi}_{\mathcal{M}_\delta}\| &= \|\hat{T}^{\pi}_{\delta}\hat{Q}^{\pi}_{\mathcal{M}_\delta} - T^{\pi}_{\delta}Q^{\pi}_{\mathcal{M}_\delta}\| \\ &= \|\hat{T}^{\pi}_{\delta}\hat{Q}^{\pi}_{\mathcal{M}_\delta} - T^{\pi}_{\delta}\hat{Q}^{\pi}_{\mathcal{M}_\delta} + T^{\pi}_{\delta}\hat{Q}^{\pi}_{\mathcal{M}_\delta} - T^{\pi}_{\delta}Q^{\pi}_{\mathcal{M}_\delta}\| \\ &\le \varepsilon_2 + \gamma\|\hat{Q}^{\pi}_{\mathcal{M}_\delta} - Q^{\pi}_{\mathcal{M}_\delta}\| \\ \implies \|\hat{Q}^{\pi}_{\mathcal{M}_\delta} - Q^{\pi}_{\mathcal{M}_\delta}\| &\le \frac{\varepsilon_2}{1-\gamma}.\end{aligned}$$

Regret bound. We define the updating policy as

$$\hat{\pi}_{k+1} = \operatorname*{argmin}_{\pi\in\Pi} D_{\mathrm{KL}}\left(\pi(\cdot\mid s)\middle\|\frac{\exp\left(\frac{1}{\alpha}\hat{Q}^{\hat{\pi}_k}_{\mathcal{M}_\delta}(s,\cdot)\right)}{Z^{\hat{\pi}_k}(s)}\right), k = 0, 1, \cdots$$

By Proposition 3.6, the policy sequence $\{\hat{\pi}^k\}$ converges to the optimal policy $\hat{\pi}^\star_{\mathcal{M}_\delta}$ under the estimate DR soft policy iteration as $k \to \infty$.

For each state $s \in \mathcal{S}$, we have $V^\star_{\mathcal{M}_\delta}(s) - V^{\hat{\pi}^\star_{\mathcal{M}_\delta}}_{\mathcal{M}_\delta}(s) \ge 0$. By definition, $\hat{V}^\star_{\mathcal{M}_\delta}(s) = \hat{V}^{\hat{\pi}^\star_{\mathcal{M}_\delta}}_{\mathcal{M}_\delta}(s)$. Then, we have

$$\begin{aligned}V^\star_{\mathcal{M}_\delta}(s) - V^{\hat{\pi}^\star_{\mathcal{M}_\delta}}_{\mathcal{M}_\delta}(s) &\le \left|V^\star_{\mathcal{M}_\delta}(s) - \hat{V}^\star_{\mathcal{M}_\delta}(s)\right| + \left|\hat{V}^{\hat{\pi}^\star_{\mathcal{M}_\delta}}_{\mathcal{M}_\delta}(s) - V^{\hat{\pi}^\star_{\mathcal{M}_\delta}}_{\mathcal{M}_\delta}(s)\right| \\ &= \left|\sup_\pi V^\pi_{\mathcal{M}_\delta}(s) - \sup_\pi \hat{V}^\pi_{\mathcal{M}_\delta}(s)\right| + \left|\hat{V}^{\hat{\pi}^\star_{\mathcal{M}_\delta}}_{\mathcal{M}_\delta}(s) - V^{\hat{\pi}^\star_{\mathcal{M}_\delta}}_{\mathcal{M}_\delta}(s)\right| \\ &\le \sup_\pi\left|V^\pi_{\mathcal{M}_\delta}(s) - \hat{V}^\pi_{\mathcal{M}_\delta}(s)\right| + \left|\hat{V}^{\hat{\pi}^\star_{\mathcal{M}_\delta}}_{\mathcal{M}_\delta}(s) - V^{\hat{\pi}^\star_{\mathcal{M}_\delta}}_{\mathcal{M}_\delta}(s)\right| \\ &\le 2\sup_\pi\left|V^\pi_{\mathcal{M}_\delta}(s) - \hat{V}^\pi_{\mathcal{M}_\delta}(s)\right|\end{aligned}$$

Thus,

$$R_{\mathcal{M}_\delta}(\hat{\pi}^\star_{\mathcal{M}_\delta}) = \left\| V^\star_{\mathcal{M}_\delta} - V^{\hat{\pi}^\star_{\mathcal{M}_\delta}}_{\mathcal{M}_\delta} \right\|_\infty \leq 2 \sup_\pi \left\| V^\pi_{\mathcal{M}_\delta} - \hat{V}^\pi_{\mathcal{M}_\delta} \right\|_\infty \leq 2 \sup_\pi \left\| Q^\pi_{\mathcal{M}_\delta} - \hat{Q}^\pi_{\mathcal{M}_\delta} \right\|_\infty \leq \frac{2\varepsilon_2}{1-\gamma}.$$