RORL: Robust Offline Reinforcement Learning via Conservative Smoothing

Anonymous Author(s)
Affiliation
Address
email

Abstract

Offline reinforcement learning (RL) provides a promising direction to exploit the massive amount of offline data for complex decision-making tasks. Due to the distribution shift issue, current offline RL algorithms are generally designed to be conservative for value estimation and action selection. However, such conservatism impairs the robustness of learned policies, leading to a significant change even for a small perturbation on observations. To trade off robustness and conservatism, we propose Robust Offline Reinforcement Learning (RORL) with a novel conservative smoothing technique. In RORL, we explicitly introduce regularization on the policy and the value function for states near the dataset and additional conservative value estimation on these OOD states. Theoretically, we show RORL enjoys a tighter suboptimality bound than recent theoretical results in linear MDPs. We demonstrate that RORL can achieve the state-of-the-art performance on the general offline RL benchmark and is considerably robust to adversarial observation perturbation.

1 Introduction

Over the past few years, deep reinforcement learning (RL) has been a vital tool for various decision-making tasks [30, 44, 42, 8] in a trial-and-error manner. A major limitation of current deep RL algorithms is that they require intense online interactions with the environment [26]. These data collecting processes can be costly and even prohibitive in many real-world scenarios such as robotics and health care. Offline RL [12, 24] is gaining more attention recently since it offers probabilities to learn reinforced decision-making strategies from fully offline datasets.

The main challenge of offline RL is the distribution shift between the offline dataset and the learned policy, which would lead to significant overestimation for the out-of-distribution (OOD) actions [12, 24]. To overcome such an issue, a series of offline RL works [51, 12, 58, 25, 2, 3] propose to celebrate conservatism, such as constraining the learned policy close to supported distribution or penalizing the Q-values of OOD actions. In addition, another stream of works builds upon model-based offline RL algorithms [61, 60, 50], which leverages the ensemble dynamics models to enforce pessimism via uncertainty penalizing or data generation.

However, conservatism is not the only concern when applying offline RL to the real world. Due to the sensor errors and model mismatch, the robustness of offline RL is crucial under the realistic engineering conditions, which has not been well studied yet. In online RL, a series of works have been studied to learn the optimal policy under worst-case perturbations of the observation [62, 35, 17] or environmental dynamics [49, 37, 38]. Yet, it is non-trivial to apply online robust RL techniques into the offline problems. The main challenge is that the perturbation of either states or actions may bring OOD data and extra overestimation for the value function. New techniques are needed to tackle the conservatism and robustness simultaneously in the offline RL.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

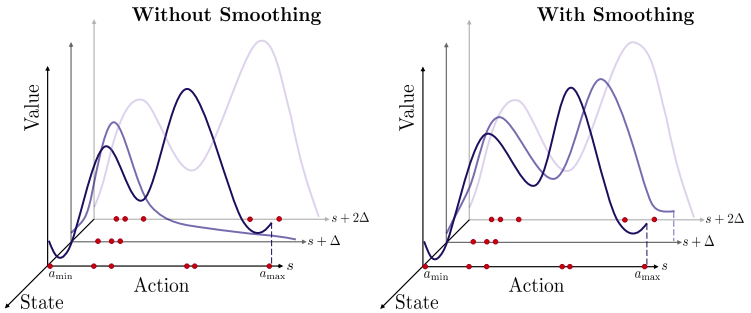


Figure 1: A schematic diagram of RORL. The red spots represent the offline data samples. Without state smoothing, the value function would change drastically over neighbor states and induce an unstable policy. Yet, the smoothness may also lead to value overestimation of dangerous areas. RORL trades off smoothness and possible overestimation as discussed in Sec 4.

.

This paper studies robust offline RL against adversarial observation perturbation, where the agent needs to learn the policy conservatively while handling the potential OOD observation with perturbation. We first demonstrate that current value-based offline RL algorithms lack the necessary smoothness for the policy, which is visualized in Figure 1. As an illustration, we show that a famous baseline method CQL [25] learns a non-smooth value function, leading to significant performance degradation for even a tiny scale perturbation on observation (see Section 3 for details). Notice that simply adopting the smoothing technique for existing methods may result in extra overestimation at the boundary of supported distribution and lead the agent toward the unsafe areas.

To this end, we propose Robust Offline Reinforcement Learning (RORL) with a novel conservative smoothing technique, which explicitly handles the overestimation of OOD state-action pairs. Specifically, we explicitly introduce smooth regularization on both the values and policies for states near the dataset support and conservatively estimate the values of these OOD states based on pessimistic bootstrapping. In addition, we theoretically prove that RORL yields a valid uncertainty quantifier in linear MDPs and enjoys a tighter suboptimality bound than previous work [3].

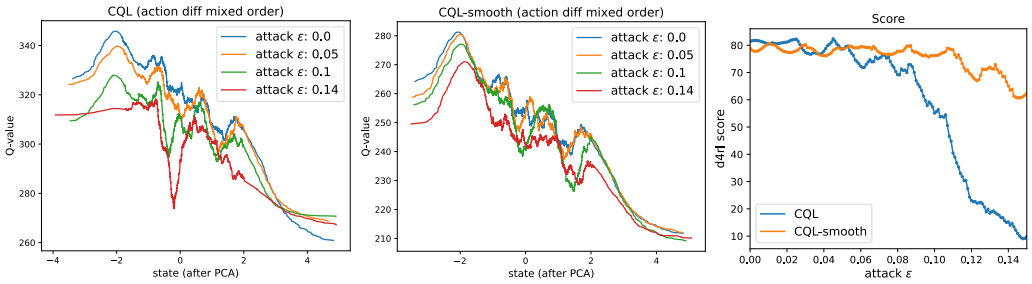
In our experiments[1], we demonstrate that RORL can achieve state-of-the-art (SOTA) performance in D4RL benchmark [9] for offline RL given a suitable adversarial perturbation scale during training. The surprising performance in the benchmark shows that the robust training improves the generalization ability and leads to performance improvement even in non-perturbed tasks. Meanwhile, compared with the current SOTA baseline [2], RORL is considerably more robust to adversarial perturbations on observations. We conduct the ablation study under different types of adversarial attacks on the observations, showing the consistently superior performance over several tasks.

2 Preliminaries

Offline RL and PBRL Considering an episodic MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, \gamma, \mathbb{P})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, T is the length of the episode, r is the reward function, $\mathbb{P}$ is the dynamics, and γ is the discount factor. In offline RL, the objective of the agent is to find an optimal policy by sampling experience from a fixed dataset $\mathcal{D} = \{(s_t^i, a_t^i, r_t^i, s_{t+1}^i)\}$. Nevertheless, directly applying off-policy algorithms in offline RL has distribution shift problem. In Q-learning, the value function evaluated on the greedy action a' in Bellman operator $\mathcal{T}Q = r + \gamma \mathbb{E}_{s'}[\max_{a'}(s', a')]$ tends to have extrapolation error since (s', a') has barely occurred in $\mathcal{D}$. There are mainly three kinds of methods to track the distribution shift problem in offline RL, including policy constraints, conservative value function, and uncertainty-based methods.

Pessimistic Bootstrapping for Offline RL (PBRL) [3] is an uncertainty-based method that uses bootstrapped Q-functions for uncertainty quantification and OOD sampling for regularization. Specifically, PBRL maintains K bootstrapped Q functions to quantify the epistemic uncertainty and performs pessimistic update to penalize Q functions with large uncertainties. The uncertainty is defined as the standard deviation among bootstrapped Q-functions. For each bootstrapped Q-function, the

[1] Our anonymized code is available at https://anonymous.4open.science/r/RORL-B144/



(a) Q-function of CQL (b) Q-function of CQL-smooth (c) Final performance

Figure 2: Figure (a) and (b) illustrate the Q-functions of $\hat{s}$ with adversarial noises in CQL and CQL-smooth, respectively. The same moving average factor is used in plotting both figures. Figure (c) shows the performance evaluation of CQL and CQL-smooth with different perturbation scale. We use 100 different $\epsilon \in [0.0, 0.15]$ for the evaluation.

Bellman target is defined as $\widehat{\mathcal{T}}Q(s,a) = r(s,a) + \gamma \widehat{\mathbb{E}}_{s' \sim P(\cdot|s,a), a' \sim \pi(\cdot|s')} \big[Q(s',a') - \beta u(s',a') \big]$. Under linear MDP assumptions, this uncertainty is equivalent to the LCB penalty and is provably efficient [21]. Furthermore, PBRL incorporates OOD sampling in training by sampling OOD actions to form (s, a^{ood}) pairs, where a^{ood} follows the learned policy. The learning target for (s, a^{ood}) is $\widehat{\mathcal{T}}^{\text{ood}}Q(s,a^{\text{ood}}) := Q(s,a^{\text{ood}}) - \beta u(s,a^{\text{ood}})$, which introduces uncertainty penalization to enforce pessimistic Q-functions for OOD actions.

Smooth Regularized RL Robust RL aims to learn a robust policy against the adversarial perturbed environment in online RL. SR^{2}L [43] enforces smoothness in both the policy and Q-functions. Specifically, SR^{2}L encourages the outputs of the policy and value function to not change much when injecting small perturbation to the state. For state s, SR^{2}L constructs a perturbation set $\mathbb{B}_d(s,\epsilon) = \{\hat{s} : d(s,\hat{s}) \le \epsilon\}$ and introduces a smoothness regularizer for policy as $\mathcal{R}_s^\pi = \mathbb{E}_{s\sim\rho^\pi} \max_{\hat{s}\in\mathbb{B}_d(s,\epsilon)} \mathcal{D}(\pi(\cdot|s) \| \pi(\cdot|\hat{s}))$, where $\mathcal{D}(\cdot\|\cdot)$ is a distance metric and the $\max$ operator gives an adversarial manner to choose $\hat{s}$. Similarly, the smoothness regularizer for the value function is defined as $\mathcal{R}_s^V = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi} \max_{\hat{s}\in\mathbb{B}_d(s,\epsilon)} (Q(s,a) - Q(\hat{s},a))^2$. The induced smoothness is shown to improve robustness against both random and adversarial perturbations.

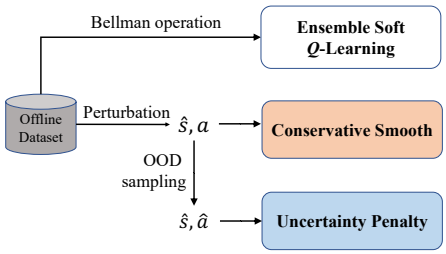
3 Robustness of Offline RL: A Motivating Example

We give a motivating example to illustrate the robustness of the popular CQL [25] policies. We introduce an adversarial attack on state s to obtain $\hat{s} = \arg\max_{\hat{s}\in\mathbb{B}_d(s,\epsilon)} D_{\text{J}}(\pi_\theta(\cdot|s) \| \pi_\theta(\cdot|\hat{s}))$, where $\mathbb{B}_d$ is the perturbation set to ensure $d(s,\hat{s}) \le \epsilon$ and the Jeffrey's divergence D_{J} for two distributions P, Q is defined by: $D_{\text{J}}(P\|Q) = \frac{1}{2}[D_{\text{KL}}(P\|Q) + D_{\text{KL}}(Q\|P)]$. To obtain $\hat{s}$, we take gradient assent with respect to the loss function $D_{\text{J}}(\pi_\theta(\cdot|s)\|\pi_\theta(\cdot|\hat{s}))$ and restrict the outputs to the $\mathbb{B}_d(s,\epsilon)$ set, where π_θ is a learned CQL policy and ϵ controls the strength of attack.

In the *walker-medium-v2* task from D4RL [9], we use various setups of ϵ for adversarial attack to evaluate the robustness of CQL policies. Specifically, we use $\epsilon \in \{0, 0.05, 0.1, 0.14\}$ to control the strengths of the attack, where we have $\hat{s} = s$ if $\epsilon = 0$. Given a specific ϵ, we sample N state-action pairs $\{(s_i, a_i)\}$ from the offline dataset, and then perform adversarial attack to obtain $\{(\hat{s}_i, a_i)\}$ and also the corresponding Q-values $\{Q_i(\hat{s}_i, a_i)\}$, where the Q-function is the trained CQL critic.

Figure 2(a) shows the relationship between $\hat{s}_i$ and the corresponding Q_i with different ϵ. To visualize $\hat{s}_i$, we perform PCA dimensional reduction [47] and choose one of the reduced dimensions to represent $\hat{s}_i$. With the increase of ϵ in the adversarial attack, the Q-curve has greater deviation compared to the curve with $\epsilon = 0$. The result signifies that the Q-function of CQL is not smooth in state space, which makes the adversarial noises easily affect the Q-function. As a comparison, we apply the proposed conservative smoothing loss in CQL training (i.e., *CQL-smooth*) and use the same evaluation method to obtain $\hat{s}_i$ and Q_i. According to the result in Figure 2(b), the value function becomes smoother.

In addition, we show how the adversarial attack affects the final performance of offline RL policies. We use $\epsilon \in [0, 0.15]$ to evaluate both the original CQL policies (i.e., *CQL*) and CQL with conservative



Algorithm 1: RORL

while *not converged* **do**
- Sample mini-batch transitions (s, a, r, s') from $\mathcal{D}$.
- Perform extended soft Q-learning to train multiple Q-functions $\{Q_1, \ldots, Q_K\}$.
- Sample $\hat{s}$ from $\mathbb{B}_d(s, \epsilon)$ to obtain $(\hat{s}, a)$ pairs.
- Perform adversarial smoothing for the Q-network.
- Calculate uncertainty to penalize $(\hat{s}, \hat{a})$ for conservatism.
- Combining terms to train the critic with Eq. (5).
- Train the policy with smooth constraint as Eq. (6).

Figure 3: **RORL Algorithm**: RORL trains multiple soft Q-functions for uncertainty quantification. The conservative smoothing loss is calculated for $(\hat{s}, a)$ with perturbed state. We perform uncertainty penalization for $(\hat{s}, \hat{a})$ with OOD actions.

smoothing loss (i.e., *CQL-smooth*) in adversarial attack. Figure 2(c) shows the performance with different settings of ϵ. We find that our smooth constraints significantly improve the robustness of CQL, especially for large adversarial noises.

4 Robust Offline RL via Conservative Smoothing

In RORL, we develop smooth regularization on both the policy and the value function for states near the dataset. The smooth constraints make the policy and Q-function robust to perturbation in the dataset (for offline training) and interactive samples (for online evaluation). Nevertheless, the smoothness may lead to value overestimation of the edge of the supported dataset. To address this problem, we adopt bootstrapped Q-functions [33] for uncertainty quantification and sample OOD actions for penalization. RORL obtains conservative and smooth value estimation on both OOD states and actions, which improves the generalization ability of offline RL algorithms. The overall architecture of RORL is given in Figure 3.

Robust Q-function We sample three sets of state-action pairs and apply different loss functions to obtain a conservative smooth policy. Specifically, for a (s, a) pair sampled from $\mathcal{D}$, we construct a perturbation set $\mathbb{B}_d(s, \epsilon)$ to obtain $(\hat{s}, a)$ pairs, where $\hat{s} \in \mathbb{B}_d(s, \epsilon)$ and ϵ. Then we perform OOD sampling by using the current policy π_θ to obtain $(\hat{s}, \hat{a})$ pairs, where $\hat{a} \sim \pi_\theta(\hat{s})$. RORL contains K Q-functions trained by Soft Q-learning as critics. We denote the parameters of i-th Q-function and the target Q-function as ϕ_i and ϕ_i', respectively. In the following, we give different learning targets for (s, a), $(\hat{s}, a)$, and $(\hat{s}, \hat{a})$ pairs, respectively.

First, for a (s, a) pair sampled from $\mathcal{D}$, we apply extended soft Q-learning to obtain the target as

$$\widehat{\mathcal{T}} Q_{\phi_i}(s, a) := r(s, a) + \gamma \widehat{\mathbb{E}}_{a' \sim \pi_\theta(\cdot|s)} \big[\min_{j=1,\ldots,K} Q_{\phi_j'}(s', a') - c \cdot \log \pi_\theta(a'|s') \big], \tag{1}$$

where the next-Q function takes minimum value among the target Q-functions.

Then, for a $(\hat{s}, a)$ pair with a perturbed state, we enforce smoothness in Q-function by minimizing the Q-value difference between $Q(s, a)$ and $Q(\hat{s}, a)$. In particular, we choose an adversarial $\hat{s} \in \mathbb{B}_d(s, \epsilon)$ that maximizes $\mathcal{L}(Q(\hat{s}, a), Q(s, a))$, and then minimize this loss function with the adversarial $\hat{s}$, where $\mathcal{L}$ is a distance measure. Intuitively, we want that the Q-function is smooth with the most difficult (i.e., adversarial) attack in $\mathbb{B}_d(s, \epsilon)$, where ϵ controls the perturbation scale. To solve such a min-max problem, we maximize $\mathcal{L}(Q(\hat{s}, a), Q(s, a))$ to obtain $\hat{s}$, and then minimize the loss function with the selected $\hat{s}$.

$$\mathcal{L}_{\text{smooth}}(s, a; \phi_i) = \max_{\hat{s} \in \mathbb{B}_d(s, \epsilon)} \mathcal{L}\big(Q_{\phi_i}(\hat{s}, a), Q_{\phi_i}(s, a)\big) \tag{2}$$

where $\mathcal{L}(\cdot, \cdot)$ is the smooth loss function, and we denote $\delta(s, \hat{s}, a) = Q_{\phi_i}(\hat{s}, a) - Q_{\phi_i}(s, a)$. We remark that if $\delta(s, \hat{s}, a) > 0$, the perturbed state may induce an overestimated Q-value that we need to smooth. In contrast, if $\delta(s, \hat{s}, a) < 0$, the perturbed Q-function is underestimated, which does not cause a serious problem in offline RL. As a result, we use different weights for $\delta(s, \hat{s}, a)_+$ and $\delta(s, \hat{s}, a)_-$. The definition of $\mathcal{L}(\cdot, \cdot)$ is give as follows,

$$\mathcal{L}\big(Q_{\phi_i}(\hat{s}, a), Q_{\phi_i}(s, a)\big) = (1 - \tau)\delta(s, \hat{s}, a)_+^2 + \tau \delta(s, \hat{s}, a)_-^2, \tag{3}$$

where we choose $\tau \leq 0.5$, $x_+ = \max(x, 0)$ and $x_- = \min(x, 0)$. In this term, we does not introduce OOD action $\hat{a}$ for smoothing since the actions are desired to be close to the behavior actions in the dataset for areas near the offline data.

Finally, to prevent overestimation of OOD states and actions, we use bootstrapped uncertainty $u(\hat{s}, \hat{a})$ as a penalty for $Q(\hat{s}, \hat{a})$, where $\hat{a} \sim \pi(\hat{s})$ is an OOD action sampled from the current policy π. We remark that a similar OOD sampling is also used in PBRL [3]. *The difference is that PBRL only penalizes the OOD actions, while RORL penalizes both the OOD states and actions to provide conservatism for unfamiliar areas.* The ensemble technique in RL comes from Bootstrapped DQN [33]. The ensemble forms an estimation of the Q-posterior, which yields diverse predictions on areas with scarce data. $u(\hat{s}, \hat{a})$ quantifies the deviation of a datapoint from the offline dataset [2, 3]. We follow PBRL and use a loss function as:

$$\mathcal{L}_{\text{ood}}(s; \phi_i) = \mathbb{E}_{\hat{s}\sim\mathbb{B}_d(s,\epsilon), \hat{a}\sim\pi_\theta(\hat{s})} \big(\widehat{\mathcal{T}}_{\text{ood}} Q_{\phi_i}(\hat{s}, \hat{a}) - Q_{\phi_i}(\hat{s}, \hat{a})\big)^2, \tag{4}$$

where the pseudo-target for the OOD datapoints is computed as: $\widehat{\mathcal{T}}_{\text{ood}} Q_{\phi_i}(\hat{s}, \hat{a}) := Q_{\phi_i}(\hat{s}, \hat{a}) - u(\hat{s}, \hat{a})$, where $u(\hat{s}, \hat{a}) := \sqrt{\frac{1}{K}\sum_{k=1}^{K} \big(Q_{\phi_i}(\hat{s}, \hat{a}) - \bar{Q}_\phi(\hat{s}, \hat{a})\big)^2}$, The bootstrapped uncertainty $u(\hat{s}, \hat{a})$ is defined as the standard deviation among the Q-ensemble.

Combining the loss functions above, RORL has the following loss function for each Q_{ϕ_i}:

$$\min_{\phi_i} \mathbb{E}_{s,a,r,s'\sim\mathcal{D}} \Big[\big(\widehat{\mathcal{T}} Q_{\phi_i}(s, a) - Q_{\phi_i}(s, a)\big)^2 + \alpha \mathcal{L}_{\text{smooth}}(s, a; \phi_i) + \beta \mathcal{L}_{\text{ood}}(s; \phi_i) \Big], \tag{5}$$

Robust Policy We learn a robust policy by using a smooth constraint to make the policy change less with perturbations. Similarly, we choose an adversarial state $\hat{s} \in \mathbb{B}_d(s, \epsilon)$ that maximizes $D_{\text{J}}\big(\pi_\theta(\cdot|s) \| \pi_\theta(\cdot|\hat{s})\big)$, and then minimize the policy difference between $\pi_\theta(\cdot|s)$ and $\pi_\theta(\cdot|\hat{s})$. To conclude, we minimize the following loss function for the policy,

$$\min_\theta \Big[\mathbb{E}_{s,a,r,s'\sim\mathcal{D}} \big[- \min_{j=1,\ldots,K} Q_{\phi_j}(s, a) + \alpha_2 \max_{\hat{s}\in\mathbb{B}_d(s,\epsilon)} D_{\text{J}}\big(\pi_\theta(\cdot|s) \| \pi_\theta(\cdot|\hat{s})\big) + c \log \pi_\theta(a|s) \big] \Big]. \tag{6}$$

where the first term aims to maximize the minimum of the ensemble Q-functions to obtain a conservative policy, and the third term is regularization of entropy.

5 Theoretical Analysis

We analyze the simplified learning objective of RORL in linear MDPs [20, 21], where the feature map of the state-action pair takes the form of $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, and the transition kernel and the reward function are assumed to be linear in ϕ. The parameter $\widetilde{w}_t$ of RORL can be solved in closed form following the least squares value iteration algorithm (LSVI), which minimizes the following loss function.

$$\widetilde{w}_t^i = \min_{w\in\mathcal{R}^d} \Big[\sum_{i=1}^{m} \big(y_t^i - Q_w(s_t^i, a_t^i)\big)^2 + \sum_{i=1}^{m} \frac{1}{|\mathbb{B}_d(s_t^i, \epsilon)|} \sum_{\hat{s}_t^i \in \mathcal{D}_{\text{ood}}(s_t^i)} \big(Q_w(s_t^i, a_t^i) - Q_w(\hat{s}_t^i, a_t^i)\big)^2 + \sum_{(\hat{s},\hat{a},\hat{y})\sim\mathcal{D}_{\text{ood}}} \big(\hat{y} - Q_w(\hat{s}, \hat{a})\big)^2 \Big], \tag{7}$$

where we have $Q_w(s_t^i, a_t^i)) = \phi(s_t^i, a_t^i)^\top w$ since the Q-function is also linear in ϕ. The first term in Eq. (7) is the ordinary TD-error, where we consider the setting of $\gamma = 1$ and the Q-target is $y_t^i = r(s_t^i, a_t^i) + V_{t+1}(s_{t+1}^i)$. The second term is the proposed conservative smoothing loss. Specifically, $\hat{s}_t^i \sim \mathcal{D}_{\text{ood}}(s_t^i)$ are sampled from a l_∞ ball of center s_t^i and norm $\epsilon > 0$, which can be formulated as $\hat{s} \sim \mathbb{B}_d(s_t^i, \epsilon)$. The third term is the additional OOD-sampling loss [3], which enforces conservatism for OOD actions. Differently, we use perturbed states sampled from $\mathcal{D}_{\text{ood}} = \bigcup_{i=1}^{m} \mathcal{D}_{\text{ood}}(s_t^i)$ rather than states from dataset in PBRL. The OOD action $\hat{a}$ is sampled from policy π. The explicit solution of Eq. (7) takes the following form by following LSVI:

$$\widetilde{w}_t^i = \widetilde{\Lambda}_t^{-1} \Big(\sum_{i=1}^{m} \phi(s_t^i, a_t^i) y_t^i + \sum_{(\hat{s},\hat{a},\hat{y})\sim\mathcal{D}_{\text{ood}}} \phi(\hat{s}, \hat{a}) \hat{y} \Big), \tag{8}$$

where the covariance matrix $\widetilde{\Lambda}_t$ is defined as

$$\widetilde{\Lambda}_t = \sum_{i=1}^{m} \phi(s_t^i, a_t^i)\phi(s_t^i, a_t^i)^\top + \sum_{(\hat{s},\hat{a})\sim\mathcal{D}_{\text{ood}}} \phi(\hat{s}_t, \hat{a}_t)\phi(\hat{s}_t, \hat{a}_t)^\top + \sum_{i=1}^{m} \frac{1}{|\mathbb{B}_d(s_t^i, \epsilon)|} \sum_{\hat{s}_t^i \sim \mathcal{D}_{\text{ood}}(s_t^i)} [\phi(\hat{s}_t^i, a_t^i) - \phi(s_t^i, a_t^i)]\big[\phi(\hat{s}_t^i, a_t^i) - \phi(s_t^i, a_t^i)\big]^\top. \tag{9}$$

We denote the first term and the second term as $\widetilde{\Lambda}^{\text{in}}$ and $\widetilde{\Lambda}_t^{\text{ood}}$, which represent the covariance matrices induced by the offline samples and OOD samples, respectively. Nevertheless, in linear MDPs, it is difficult to ensure the covariance $\widetilde{\Lambda}^{\text{in}} + \widetilde{\Lambda}_t^{\text{ood}} \succeq \lambda I$, since it requires that the embeddings of the samples are isotropic to make the eigenvalues of the corresponding covariance matrix lower bounded. This condition holds if we can sample embeddings uniformly from the whole embedding space. However, since the offline dataset has limited coverage in the state-action space and the OOD samples come from a limited l_∞-ball around the offline data, $\widetilde{\Lambda}^{\text{in}} + \widetilde{\Lambda}_t^{\text{ood}}$ cannot be guaranteed to be positive definite. PBRL [3] uses the assumption of $\widetilde{\Lambda}_t^{\text{ood}} \succeq \lambda I$, while it is unachievable empirically. In RORL, we solve this problem by introducing an additional conservative smoothing loss, which induces a covariance matrix as $\widetilde{\Lambda}_t^{\text{ood\_diff}} = \sum_{i=1}^{m} \frac{1}{|\mathbb{B}_d(s_t^i, \epsilon)|} \sum_{\hat{s}_t^i \sim \mathcal{D}_{\text{ood}}(s_t^i)} [\phi(\hat{s}_t^i, a_t^i) - \phi(s_t^i, a_t^i)][\phi(\hat{s}_t^i, a_t^i) - \phi(s_t^i, a_t^i)]^\top$ (i.e., the third term in Eq. (9)). The following theorem gives the guarantees of $\widetilde{\Lambda}_t^{\text{ood\_diff}} \succeq \lambda I$.

Theorem 1 *Assume $\exists i \in [1, m]$ the vector group of all $\hat{s}_t^i \sim \mathcal{D}_{ood}(s_t^i)$: $\{\phi(\hat{s}_t^i, a_t^i) - \phi(s_t^i, a_t^i)\}$ be full rank, then the covariance matrix $\widetilde{\Lambda}_t^{\text{ood\_diff}}$ is positive-definite: $\widetilde{\Lambda}_t^{\text{ood\_diff}} \succeq \lambda \cdot \mathrm{I}$ where $\lambda > 0$.*

Recall the covariance matrix of PBRL is $\widetilde{\Lambda}_t^{\text{PBRL}} = \widetilde{\Lambda}_t^{\text{in}} + \widetilde{\Lambda}_t^{\text{ood}}$, and RORL has a covariance matrix as $\widetilde{\Lambda}_t = \widetilde{\Lambda}_t^{\text{PBRL}} + \widetilde{\Lambda}_t^{\text{ood\_diff}}$, we have the following corollary based on Theorem 1.

Corollary 1 *Under the linear MDP assumptions, we have $\widetilde{\Lambda}_t \succeq \widetilde{\Lambda}_t^{\text{PBRL}}$. Further, the covariance matrix $\widetilde{\Lambda}_t$ of RORL is positive-definite: $\widetilde{\Lambda}_t \succeq \lambda \cdot \mathrm{I}$, where $\lambda > 0$.*

Recent theoretical analysis shows that an appropriate uncertainty quantification is essential to provable efficiency in offline RL [21, 56, 3]. Pessimistic Value Iteration [21] defines a general ξ-uncertainty quantifier as the penalty and achieves provable efficient pessimism in offline RL. In linear MDPs, Lower Confidence Bound (LCB)-penalty [1, 20] is known to be a ξ-uncertainty quantifier for appropriately selected β_t as $\Gamma^{\text{lcb}}(s_t, a_t) = \beta_t \cdot \big[\phi(s_t, a_t)^\top \Lambda_t^{-1} \phi(s_t, a_t)\big]^{1/2}$. Following the analysis of PBRL [3], since the bootstrapped uncertainty is an estimation to the LCB-penalty and the OOD sampling provides a covariance matrix of $\widetilde{\Lambda}_t^{\text{PBRL}}$, the proposed RORL also form a valid ξ-uncertainty quantifier with the covariance matrix $\widetilde{\Lambda}_t \succeq \lambda \cdot \mathrm{I}$ given in Corollary 1. This allows us to further characterize the optimality gap based on the pessimistic value iteration [21, 3]. We have the following suboptimality gap under linear MDP assumptions.

Corollary 2 $\text{SubOpt}(\pi^*, \hat{\pi}) \leq \sum_{t=1}^{T} \mathbb{E}_{\pi^*}\big[\Gamma_t^{\text{lcb}}(s_t, a_t)\big] < \sum_{t=1}^{T} \mathbb{E}_{\pi^*}\big[\Gamma_t^{\text{lcb\_PBRL}}(s_t, a_t)\big]$.

We refer to Appendix A for detailed proof. The second inequality is directly induced by $\Gamma_t^{\text{lcb}}(s_t, a_t) < \Gamma_t^{\text{lcb\_PBRL}}(s_t, a_t)$. Therefore, RORL enjoys a tighter suboptimality bound than PBRL [3].

6 Experiments

We evaluate our method on D4RL benchmark [10] with various continuous-control tasks and datasets. We compare RORL with several offline RL algorithms, including (i) BC that performs behavior cloning, (ii) CQL [25] that learns conservative value function for OOD actions, (iii) EDAC [2] that learns a diversified Q-ensemble to enforce conservatism, and (iv) PBRL [3] that performs uncertainty penalization and OOD sampling. Among these methods, EDAC [2] and PBRL [3] are related to RORL since all these methods apply Q-ensemble for conservatism. EDAC needs much more Q-networks (i.e., 10$\sim$50) for hopper tasks than PBRL and RORL that use 10 Q-networks. (v) We also include a basic SAC-10 algorithm as a baseline [2], which is an extension of SAC with 10 Q-functions. To assign uniform adversarial attack budget on each dimension of observations, we normalize the observations for SAC-10, EDAC and RORL. More details are provided in Appendix B.

Table 1: Normalized average returns on Gym tasks, averaged over 4 random seeds. Part of the results are reported in the EDAC paper.

Task Name	BC	CQL	PBRL	SAC-10 (Reproduced)	EDAC (Paper)	RORL (Ours)
halfcheetah-random	2.2±0.0	**31.3±3.5**	11.0±5.8	**29.0±1.5**	**28.4±1.0**	**28.6±0.6**
halfcheetah-medium	43.2±0.6	46.9±0.4	57.9 ±1.5	**64.9±1.3**	**65.9±0.6**	**65.6±0.8**
halfcheetah-medium-expert	44.0±1.6	95.0±1.4	92.3±1.1	**107.1±2.0**	**106.3±1.9**	**107.8±0.7**
halfcheetah-medium-replay	37.6±2.1	45.3±0.3	45.1±8.0	**63.2±0.6**	**61.3±1.9**	**61.5±0.6**
hopper-random	3.7±0.6	5.3±0.6	26.8±9.3	**25.9±9.6**	**25.3±10.4**	**27.2±7.5**
hopper-medium	54.1±3.8	61.9±6.4	75.3±31.2	0.8±0.2	101.6±0.6	**105.0±0.3**
hopper-medium-expert	53.9±4.7	96.9±15.1	**110.8±0.8**	6.1±7.7	**110.7±0.1**	**112.7±0.2**
hopper-medium-replay	16.6±4.8	86.3±7.3	**100.6±1.0**	**102.9±0.9**	**101.0±0.5**	**102.1±0.4**
walker2d-random	1.3±0.1	5.4±1.7	8.1±4.4	1.5±1.1	16.6±7.0	**21.4±0.2**
walker2d-medium	70.9±11.0	79.5±3.2	89.6±0.7	46.7±45.3	92.5±0.8	**102.4±1.4**
walker2d-medium-expert	90.1±13.2	109.1±0.2	110.1±0.3	116.7±1.9	114.7±0.9	**121.2±1.5**
walker2d-medium-replay	20.3±9.8	76.8±10.0	77.7±14.5	**89.6±3.1**	**87.1±2.3**	**90.4 ± 0.5**
Average	36.5	61.6	67.1	54.5	**76.0**	**78.8**

6.1 Benchmark Results

We evaluate each method on Gym domain that includes three environments (Half Cheetah, Hopper, and Walker2d) with four non-expert datasets (random, medium, medium-replay, and medium-expert). The medium-replay dataset contains experiences collected in training a medium-level policy. The random or medium dataset is generated by a single random or medium policy. The medium-expert dataset is a mixture of medium and expert datasets. We set small perturbation scale ϵ to 0.001/0.005/0.01 for halfcheetah/hopper/walker2d tasks respectively when training RORL and do not include observation perturbation in the testing time.

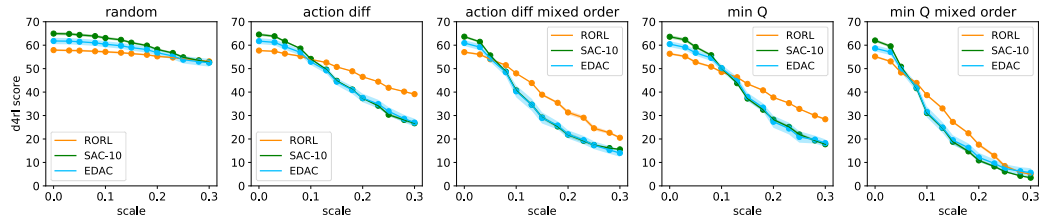
Table 1 reports the performance of the average normalized score with standard deviation. (i) SAC-10 is unstable in several Walker2d and Hopper tasks since the ensemble number is relatively small to provide reliable uncertainties. (ii) EDAC solves this problem by using diversity constraints while still requiring 10$\sim$50 Q-networks to obtain reasonable performance. In contrast, RORL only uses 10 ensemble Q-networks to outperform EDAC with 10$\sim$50 networks. (iii) PBRL chooses an alternative OOD-sampling technique to reduce the ensemble numbers. According to the result, RORL significantly outperforms PBRL with the same ensemble number. The reason is RORL additionally uses conservative smoothing loss for both OOD states and actions, which improves the generalization ability of the policy and Q-functions. We remark that RORL significantly improves over the current SOTA results on walker2d and hopper tasks, which might be because the two tasks require a more precise balancing of conservatism and generalization for better performance.

6.2 Adversarial Attack

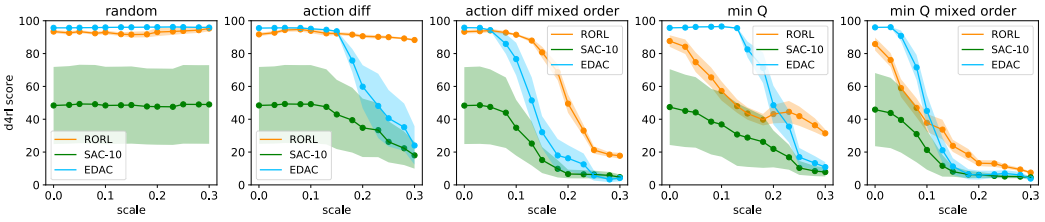
We adopt three attack methods, namely *random*, *action diff*, and *min Q* following prior works [62, 37]. Given perturbation scale ϵ, the later two methods perform adversarial perturbation on observations and are given access to the agent's policy and value functions.

- *random* uniformly samples perturbed states in an l_∞ ball of norm ϵ.
- *action diff* is an effective attack based on the agent's policy and is proved to maximize an upper bound on the performance difference between perturbed and unperturbed environments [62]. It directly finds states in an l_∞ ball of norm ϵ to minimize: $\min_{\hat{s}\in\mathbb{B}_d(s,\epsilon)} -D_{\mathrm{J}}\big(\pi_\theta(\cdot|s)\|\pi_\theta(\cdot|\hat{s})\big)$.
- *min Q* requires both the agent's policy and value function to perform a relatively stronger attack. The attacker finds a perturbed state to minimize the expected return of taking an action from that state: $\min_{\hat{s}\in\mathbb{B}_d(s,\epsilon)} Q(s,\pi_\theta(\hat{s}))$. For ensemble-based algorithms, Q is set as the mean of ensemble Q functions.

In our experiments, the two objectives of *action diff* and *min Q* are optimized via two ways. Specifically, we optimize the objectives through:



(a) Performance under attack on halfcheetah-medium-v2 dataset



(b) Performance under attack on walker2d-medium-v2 dataset

Figure 4: Figures (a) and (b) illustrate the performance of RORL, SAC-10, and EDAC under different types of attack and attack scales range $[0, 0.3]$. The curves are averaged over 3 seeds and smoothed with a window size of 3. The shaded region represents half a standard deviation.

(1) selecting the best state from uniformly sampled 50 states, which has the advantage of simplicity and little computation cost.

(2) uniformly sampling 20 initial states and performing gradient decent for 10 steps with a step size of $\frac{1}{10}\epsilon$ from each initial state to find the best perturbed state. Note that we need to clip the perturbed states within the l_∞ ball at the end of each optimization step. We name this optimization method 'mixed order' optimization.

We compare RORL with ensemble-based baselines EDAC and SAC-10 on halfcheetah-medium-v2 and walker2d-medium-v2 datasets. To handle large adversarial noise, we set the perturbation scale of the policy and Q functions as [0.1, 0.05] for halfcheetah-medium-v2 and [0.1, 0.03] for walker2d-medium-v2 in RORL's training phase. More detailed description can be found in Appendix B. The results are shown in Figure 4. In the results, RORL exhibits improved robustness than other baselines under adversarial attacks and decreases the slowest as ϵ increases. On the other hand, we find that random attack is not effective for ensemble-based offline RL algorithms, and the 'mixed order' attack brings more significant performance drop than vanilla zero-order optimization.

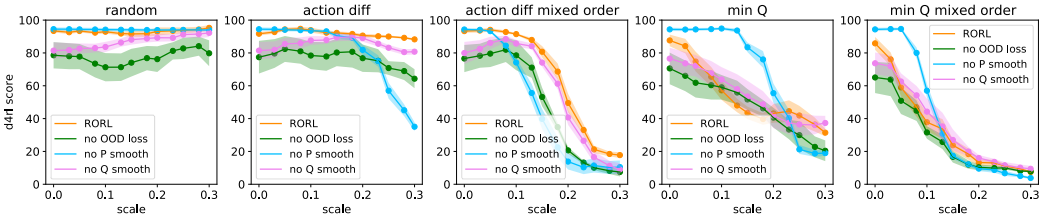


Figure 5: Ablation studies on walker2d-medium-v2 with varying perturbation scale. The curve is averaged across 3 random seeds and smoothed with a window size of 3. The shaded region represents half a standard deviation.

6.3 Ablations

We conducted ablation studies on walker2d-medium-v2 dataset to evaluate the importance of the policy smoothing term, the Q function smoothing term and the penalty for OOD states in Figure 5. From the results, we can conclude that each loss contributes to the performance of RORL under adversarial observation attacks. The policy smoothing loss is essential when the perturbation scale is large, while the performance of 'no P smooth' is higher than RORL when the scale is smaller than 0.1. Besides, learning without OOD loss performs worse than RORL at almost all the perturbation scales.

In addition, Q smooth loss has the minimal impact, which is reasonable since the basic algorithm SAC-10 is based on 10 ensemble Q networks. Our smoothing technique may have a more significant impact on baselines with fewer Q networks, such as CQL in Section 3.

7 Related Works

Offline RL Research related to offline RL has experienced explosive growth in recent years. In model-free domain, offline RL methods focus on correcting the extrapolation error [12] in the off-policy algorithms. The natural idea is to regularize the learned policy near the dataset distribution [51, 54, 31, 52, 58, 11]. For example, MARVIL reweights the policy with exponential advantage, which implicitly guarantees the policy within the KL-divergence neighborhood of the behavior policy. Another stream of model-free methods prevents the selection of OOD actions by penalizing their Q-value [24, 25, 2, 7]. With the ensemble Q networks and the additional loss term to diversify their gradients, EDAC [2] achieves SOTA performance in the D4RL benchmark. Instead of diversifying gradients, PBRL [3] proposes explicit training of the OOD actions, which achieves comparable performance with fewer ensemble networks. Inspired by EDAC and PBRL, we build our work upon ensemble networks, focusing more on smoothness over the state space.

Besides the surprising empirical results, theoretical analysis of offline reinforcement learning algorithms is of increasing interest [6, 21, 39, 56, 59]. Though the assumptions for the dataset vary in the different papers, they all suggest that pessimism and conservatism are necessary for offline RL. Our paper can be viewed as a robust extension to the previous work [21] (see Section 5 for details).

Robust RL The research line of robust RL can be traced back to H_∞-control theory [55, 4], where policies are optimized to be well-performed in the worst possible deterministic environment. Depending on the definition, there are different streams of research on robust RL. As the extension of robust control to MDPs, Robust MDPs (RMDPs) [32, 18, 41, 16] are proposed to formulate the perturbation of transition probabilities for MDPs. Though some recent analyses with theoretical guarantees come out under specific assumptions for RMDP [64, 57, 27], there is currently no practical algorithm to solve RMDP in a large-scale problem, expect some linear approximation attempt [46]. In online RL, domain randomization [48, 29] assumes the model uncertainty can be predefined in data collection by changing the setup of a simulator. However, it is not practical for offline RL. Robust Adversarial Reinforcement Learning (RARL) [37] and Noisy Robust Markov Decision Process (NR-MDP) [22] study the robust RL with the perturbed actions, showing that the policy robustness to adversarial or noisy actions can also induce robustness for model parameter changes.

The most related work to ours is SR^{2}L [43], which shows policy smoothing can lead to significant performance improvements in the online setting. In contrast, we focus on the offline setting and tackle the potential overestimation of perturbed states. Another related work is S4RL [45], where the authors study different data augmentation methods to smooth observations in offline RL. Their result supports the necessity of state smoothing. More related works are given in Appendix D.

8 Conclusion

We propose Robust Offline Reinforcement Learning (RORL) to trade-off conservatism and robustness for offline RL. To achieve that, we introduce the conservative smoothing for the perturbed states while actively underestimating their values based on pessimistic bootstrapping to keep conservative. We show that RORL can achieve comparable performance with fewer ensemble Q networks than the previous methods in the offline RL benchmark. The result supports that our proposed conservative smoothing techniques benefit general offline RL by improving the generalization ability. In addition, we demonstrate that RORL is also robust to adversarial perturbations across the different types of attacks. We hope our work can promote the application of offline RL under real-world engineering conditions.

The main limitation of our method is that the adversarial state sampling slows down the computing process, which may be improved in future work.

References

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in neural information processing systems*, volume 24, pages 2312–2320, 2011.

[2] 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, 2021.

[3] Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhi-Hong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *International Conference on Learning Representations*, 2022.

[4] Tamer Başar and Pierre Bernhard. *H-infinity optimal control and related minimax design problems: a dynamic game approach*. Springer Science & Business Media, 2008.

[5] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 262–275. Springer, 2017.

[6] Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, pages 1042–1051. PMLR, 2019.

[7] Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. Adversarially trained actor critic for offline reinforcement learning. *arXiv preprint arXiv:2202.02446*, 2022.

[8] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[9] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[10] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[11] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[12] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *ICML*, 2019.

[13] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. In *International Conference on Learning Representations*, 2019.

[14] Florin Gogianu, Tudor Berariu, Mihaela C Rosca, Claudia Clopath, Lucian Busoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning: an optimisation perspective. In *International Conference on Machine Learning*, pages 3734–3744. PMLR, 2021.

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[16] Chin Pang Ho, Marek Petrik, and Wolfram Wiesemann. Fast Bellman Updates for Robust MDPs. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1979–1988. PMLR.

[17] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

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

[19] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34, 2021.

[20] Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143. PMLR, 2020.

[21] Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021.

[22] Parameswaran Kamalaruban, Yu-Ting Huang, Ya-Ping Hsieh, Paul Rolland, Cheng Shi, and Volkan Cevher. Robust reinforcement learning via adversarial training with langevin dynamics. *Advances in Neural Information Processing Systems*, 33:8127–8138, 2020.

[23] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. In *NeurIPS*, 2020.

[24] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *NeurIPS*, 2019.

[25] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *NeurIPS*, 2020.

[26] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[27] Jialian Li, Tongzheng Ren, Dong Yan, Hang Su, and Jun Zhu. Policy learning for robust markov decision process with a mismatched generative model. *arXiv preprint arXiv:2203.06587*, 2022.

[28] Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Jerry Zhu. Policy poisoning in batch reinforcement learning and control. *Advances in Neural Information Processing Systems*, 32, 2019.

[29] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

[30] 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.

[31] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[32] Arnab Nilim and Laurent Ghaoui. Robustness in markov decision problems with uncertain transition matrices. *Advances in neural information processing systems*, 16, 2003.

[33] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *NeurIPS*, 2016.

[34] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 1(2):3, 2016.

[35] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*, 2017.

[36] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2040–2042, 2018.

[37] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[38] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

[39] Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *Advances in Neural Information Processing Systems*, 34, 2021.

[40] Mihaela Rosca, Theophane Weber, Arthur Gretton, and Shakir Mohamed. A case for new neural network smoothness constraints. *NeurIPS 2020 Workshop on ICBINB*, 2020.

[41] Aurko Roy, Huan Xu, and Sebastian Pokutta. Reinforcement learning under model mismatch. *Advances in neural information processing systems*, 30, 2017.

[42] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[43] Qianli Shen, Yan Li, Haoming Jiang, Zhaoran Wang, and Tuo Zhao. Deep reinforcement learning with robust and smooth policy. In *International Conference on Machine Learning*, pages 8707–8718. PMLR, 2020.

[44] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[45] Samarth Sinha, Ajay Mandlekar, and Animesh Garg. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *Conference on Robot Learning*, pages 907–917. PMLR, 2022.

[46] Aviv Tamar, Huan Xu, and Shie Mannor. Scaling up robust mdps by reinforcement learning. *arXiv preprint arXiv:1306.6189*, 2013.

[47] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[48] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[49] Eugene Vinitsky, Yuqing Du, Kanaad Parvate, Kathy Jang, Pieter Abbeel, and Alexandre Bayen. Robust reinforcement learning using adversarial populations. *arXiv preprint arXiv:2008.01825*, 2020.

[50] Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34, 2021.

[51] Qing Wang, Jiechao Xiong, Lei Han, Han Liu, Tong Zhang, et al. Exponentially weighted imitation learning for batched historical data. *Advances in Neural Information Processing Systems*, 31, 2018.

[52] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020.

[53] Fan Wu, Linyi Li, Chejian Xu, Huan Zhang, Bhavya Kailkhura, Krishnaram Kenthapadi, Ding Zhao, and Bo Li. Copa: Certifying robust policies for offline reinforcement learning against poisoning attacks. *arXiv preprint arXiv:2203.08398*, 2022.

[54] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[55] Lihua Xie and Carlos E de Souza. Robust h/sub infinity/control for linear systems with norm-bounded time-varying uncertainty. In *29th IEEE Conference on Decision and Control*, pages 1034–1035. IEEE, 1990.

[56] Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. Bellman-consistent pessimism for offline reinforcement learning. *Advances in neural information processing systems*, 34, 2021.

[57] Wenhao Yang, Liangyu Zhang, and Zhihua Zhang. Towards theoretical understandings of robust markov decision processes: Sample complexity and asymptotics. *arXiv preprint arXiv:2105.03863*, 2021.

[58] Yiqin Yang, Xiaoteng Ma, Li Chenghao, Zewu Zheng, Qiyuan Zhang, Gao Huang, Jun Yang, and Qianchuan Zhao. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[59] Ming Yin, Yaqi Duan, Mengdi Wang, and Yu-Xiang Wang. Near-optimal offline reinforcement learning with linear representation: Leveraging variance information with pessimism. *arXiv preprint arXiv:2203.05804*, 2022.

[60] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

[61] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In *NeurIPS*, 2020.

[62] 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.

[63] Xuezhou Zhang, Yiding Chen, Xiaojin Zhu, and Wen Sun. Robust policy gradient against strong data corruption. In *International Conference on Machine Learning*, pages 12391–12401. PMLR, 2021.

[64] 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*, pages 3331–3339. PMLR, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Sec 1 and Appendix B.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Sec 6.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix B.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cited D4RL [10] and EDAC[2] for their datasets and code.
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We included our code in the anonymized link.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Opensource code and dataset.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]