Recovering from Out-of-sample States via Inverse Dynamics in Offline Reinforcement Learning

Ke Jiang[1,2], **Jia-yu Yao**[3], **Xiaoyang Tan**[1,2*]
[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics
[2] MIIT Key Laboratory of Pattern Analysis and Machine Intelligence
[3] School of Electronic and Computer Engineering, Peking University
ke_jiang@nuaa.edu.cn, jiayu_yao@pku.edu.cn, x.tan@nuaa.edu.cn

Abstract

We deal with the *state distributional shift* problem commonly encountered in offline reinforcement learning during test, where the agent tends to take unreliable actions at out-of-sample (unseen) states. Our idea is to encourage the agent to follow the so called *state recovery* principle when taking actions, i.e., besides long-term return, the immediate consequences of the current action should also be taken into account and those capable of recovering the state distribution of the behavior policy are preferred. For this purpose, an inverse dynamics model is learned and employed to guide the state recovery behavior of the new policy. Theoretically, we show that the proposed method helps aligning the transited state distribution of the new policy with the offline dataset at out-of-sample states, without the need of explicitly predicting the transited state distribution, which is usually difficult in high-dimensional and complicated environments. The effectiveness and feasibility of the proposed method is demonstrated with the state-of-the-art performance on the general offline RL benchmarks.

1 Introduction

Reinforcement learning has made significant advances in recent years, but it has to collect experience actively to gain understanding of the underlying environments. However, such online interaction is not always practical, due to either the potential high cost of data collection procedure or its possible dangerous consequences in applications as autonomous driving or healthcare. To address these issues, offline reinforcement learning aims to learn a policy from offline datasets without doing any actual interaction with the environments [19, 13, 1].

However, directly deploying online RL algorithms, such as Deep Deterministic Policy Gradient [18], to learn the new policy from the offline dataset without proper constraints would highly likely suffer from action distributional shift due to the change in the actions generated by the new policy. This would result in the so called extrapolation error [8], i.e., the TD target could be wrongly estimated when querying those out-of-distribution (OOD) actions generated by the new policy. To address these issues, methods like Conservative Q-Learning (CQL) [16], TD3+BC [7] and Implicit Q-Learning (IQL) [14], treat the OOD actions as conterfactual queries and try to avoid performing such queries completely during learning, hence suppressing the Q-function extrapolation error. However, such pessimism for out-of-sample data could be too restricted and sample inefficient, as not all out-of-sample(unseen) states are not generalizable [20].

To effectively generalize to out-of-sample or even OOD states, it is necessary to constrain the behavior of agent at those states, otherwise the policy extrapolation error that occurs at test stage may drive the

*Corresponding Author

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

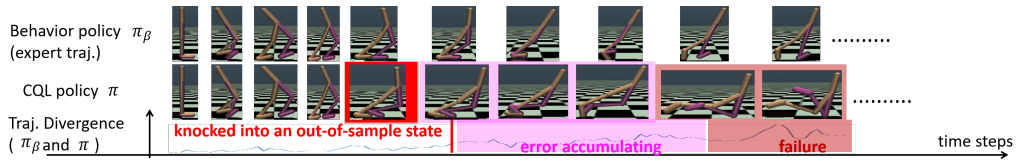


Figure 1: An example of CQL agent's failure for the accumulative error due to state distributional shift on a Walker2d robotic agent. The interval between every two images is ten steps.

agent's transited state distribution away from the offline dataset, referred as the *state distributional shift* problem - a problem has not been paid enough attention as most previous works focus on addressing the issues of OOD actions during training [16, 7]. The problem of *state distributional shift* is likely to arise in those extensive and non-stationary environments, where it is impossible for the agent to cover the entire state space during training. A brief example is illustrated in Figure 1, where a robotic agent, trained using offline RL frameworks like CQL, is knocked by some external force unintentionally and falls into an out-of-sample state during test, which leads to failure finally due to the unreliable actions taken at this out-of-sample(unseen) state [3] and the error confounding effects.

To deal with the above problem, a natural idea is to teach the agent to recover from these out-of-sample states to their familiar regions, i.e., the demonstration of offline datasets. For this purpose, we propose the Out-of-sample State Recovery (OSR) method to implement this *state recovery* principle. Our idea is to estimate the actions whose consequence would be in the support region of the offline dataset using inverse dynamics model (IDM) [2]. The estimated IDM can be interpreted as an extended policy with pre-defined consequences, hence being suitable for guiding the learning of the new policy. Theoretically, we show that the proposed OSR makes the transited state distribution of the learnt policy at out-of-sample states align well with the offline dataset, without the need of building any forward dynamics models. We also propose a modified OSR-v that suppresses the probability of selecting actions according to the risk of OOD consequences through imposing extra value-constraints when decision making. We experimentally[2] demonstrated that the proposed OSR outperforms several closely related state-of-the-art methods in offline AntMaze/MuJoCo control suites with various settings.

In what follows, after an introduction and a review of related works, Section 3 provides a concise introduction to the background of offline RL and our work. The OSR method is presented in detail with theoretical analysis of its effectiveness in Section 4.2. The variant of OSR, called OSR-v, is then introduced with theoretical analysis of its advantages in Section 4.3. In Section 4.4, the practical implementation of both methods is described, including the loss function used. Experimental results are presented in Section 5 to evaluate the effectiveness of both methods under various settings. Finally, the paper concludes with a summary of the findings and contributions.

2 Related work

Robust offline reinfocement learning. Robustness is critical for offline RL methods to implement in real-world applications. Recent work RORL [25] adds smoothing term to relax the conservatism of algorithms like Conservative Q-Learning, making the agent generalize to out-of-sample states. ROMI [23] introduce reverse mechanism to build links between isolated data clusters based on out-of-sample data. However, it is still possible that the agent would drift away from the dataset as the results of the well-known error compounding effects due to the myopic of the new policy, referred as *state distributional shift* problem, which is more important in practical applications. For example, in healthcare [12], it is critical for the agent to make decisions with safe consequences, i.e., generating reliable trajectories. Theoretically, it is shown that control the divergence of the learnt policy from the behavior policy helps to bound the *state distributional shift* [22], but the bound is actually very loose [15]. Recently model-based State Deviation Correction (SDC) [27] builds a dynamics model and a transition model to guide the agent to generate in-distribution trajectories, through which to enhance the agent's safety and controllability. However, building a high-capacity dynamics model is not always practical in complicated applications, highlighting the necessity to constraint the agent's behavior without constructing any forward model to predict the high-dimensional observations.

[2]Our code is available at https://github.com/Jack10843/OSR

Inverse dynamics model. An inverse dynamics model $I(a|s', s)$ predicts the action distribution that explains the transition between a given pair of states. Inverse dynamics models haven been applied to improving generalization to real-world problems [6], Markov representation learning [2], defining intrinsic rewards for exploration [5]. In our work, the inverse dynamics model is interpreted as a consequence-constraint policy that generates action distributions with predictable consequences, which in turn plays the role of supervision to the new policy.

3 Background

A reinforcement learning problem is usually modeled as a Markov Decision Process (MDP), which can be represented by a tuple of the form (S, A, P, R, γ), where S is the state space, A is the action space, P is the transition probability matrix, R and γ are the reward function and the discount factor. A policy is defined as $\pi : S \to A$ and trained to maximize the expected cumulative discounted reward in the MDP:

$$\max_{\pi} \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma R(s_t, \pi(a_t|s_t))\big] \tag{1}$$

In general, we define a Q-value function $Q^{\pi}(s, a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma R(s_t, \pi(a_t|s_t))|s, a]$ to represent the expected cumulative rewards. Q-learning is a classic method that trains the Q-value function by minimizing the Bellman error over Q [24]. In the setting of continuous action space, Q-learning methods use exact or an approximate maximization scheme, such as CEM [11] to recover the greedy policy, as follows,

$$\begin{aligned} Q &\leftarrow \arg\min_{Q} \mathbb{E}\big[R(s, a) + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')} Q(s', a') - Q(s, a)\big]^2 \\ \pi &\leftarrow \arg\max_{\pi} \mathbb{E}_s \mathbb{E}_{a \sim \pi(\cdot|s)} Q(s, a) \end{aligned} \tag{2}$$

In offline setting, Q-Learning algorithms learns a Q-value function $Q^{\pi}(s, a)$ and a policy π from a dataset $\mathcal{D}$, which is collected by a behavior policy π_{β}. Since there is always an action distributional shift of the new policy π and the behavior policy π_{β}, this basic recipe fails to estimate the Q-values for OOD state-action pairs. Conservative Q-Learning (CQL), as a representative OOD-constraint offline RL algorithm, tries to underestimate the Q-values for OOD state-action pairs to prevent the agent from extrapolation error [16]. The CQL term is as follows,

$$\min_{Q} \big[\mathbb{E}_{s \sim \mathcal{D}, a \sim \pi(\cdot|s)} Q(s, a) - \mathbb{E}_{s,a \sim \mathcal{D}} Q(s, a)\big] \tag{3}$$

However, at test stage, there still exists a static *state distributional shift* of the new policy π and the behavior policy π_{β}, which may accumulate the discrepancy between the agent and demonstration of the dataset. To migrate this, State Deviation Correction (SDC) [27] aims to train a policy choosing actions whose visited states are as closer to the dataset as possible. Specifically, the SDC trains a dynamics model M and a transition model U, to optimize the new policy, as follows,

$$\min_{\pi} \lambda \cdot \mathbb{E}_{s \sim \mathcal{D}} D\big(M(\cdot|\hat{s}, \pi(\cdot|\hat{s})), U(\cdot|s)\big) \tag{4}$$

where $\hat{s}$ is a perturbed version of the original state s, $\lambda > 0$ is the weight for the SDC regularization, and D is a distance measure, which is maximum mean discrepancy (MMD) in [27].

4 Out-of-sample state recovery using inverse dynamics

In this section, we first describe our Out-of-sample State Recovery (OSR) method in detail, as a policy regularization onto the actor loss, which is introduced in detail in section 4.2. Then we also propose a variant of OSR, OSR via value-constraints (OSR-v), for the purposes to reduce the hyperparameters and test the feasibility of practising *state recovery* principle in different modes, in section 4.3. Finally, we specifically describe how to implement OSR and OSR-v in practice in section 4.4.

4.1 Noise injection onto dataset

It is well-known that noise injection is helpful to address the covariate shift problem in deep learning [17]. While this technique is adopted in [27] to simulate the OOD states, in this work we interpret

the states perturbed with linear Gaussian noise as counter-examples on how to recover from those states into in-sample states. Given an offline dataset $\mathcal{D}$, which consists of quadruples (s, a, r, s'), we first perform data augmentation onto it to form a mixed dataset.

Specifically, given a quadruple (s, a, r, s'), we perturb the state s to obtain a noisy state,

$$\hat{s} = s + \beta \cdot \epsilon \tag{5}$$

where ϵ is sampled from the standard Gaussian distribution $\mathcal{N}(0, 1)$ and β is usually set to be a small constant. Besides, we could also utilize more complex methods to perturb the states, such as the adversarial attacks in [25]. In this way, we obtain a set of perturbed quadruples $(\hat{s}, a, r, s')$, and group them into a new perturbed dataset $\tilde{\mathcal{D}}$. Note that we do not perturb the next state s', to preserve the destination we wish the agent to jump from $\hat{s}$.

Finally we combine the two datasets $\mathcal{D}$ and $\tilde{\mathcal{D}}$ to form a new dataset $\mathcal{D}_{tot}$, i.e., $\mathcal{D}_{tot} = \mathcal{D} + \tilde{\mathcal{D}}$, where each element in D_{tot} is denoted as $(\tilde{s}, a, r, s')$. Next we describe how to train a inverse dynamics model to learn how to safely perform transition from $\{\tilde{s}\}$ to s'.

4.2 Learning to recover from out-of-sample states via policy-constraints

A inverse dynamics model (IDM), denoted as $I^{\pi_\beta}(a|s, s')$, is defined in terms of the behavior policy π_β, dynamics model $P(s'|s, a)$ and the transition function $P(s'|s, \pi_\beta)$ via Bayes' theorem, as follows,

$$I^{\pi_\beta}(a|s, s') = \frac{\pi_\beta(a|s)P(s'|s, a)}{P(s'|s, \pi_\beta)} \tag{6}$$

where $P(s'|s, \pi_\beta) = \sum_{a \in A} P(s'|s, a)\pi_\beta(a|s)$. Note that to learn a IDM model, we don't have to estimate the dynamic model of the environment but only needs to treat this task as a usual function approximation problem using a neural network. In particular, using the quadruples $(\tilde{s}, a, r, s')$ in $\mathcal{D}_{tot}$, the needed IDM $I^{\pi_\beta}(a|s, s')$ can be estimated using a probabilistic regression model with $\tilde{s}, s'$ as input and action a as output.

With the estimated IDM model available, we interpreted it as an extended policy with desired immediate consequence known, illustrating how to recover from out-of-samples into in-sample(safe) states, then use it to guide the learning of the new policy. In this work, we use the Kullback-Leibler(KL) divergence to measure the divergence between two distributions. In particular, we minimize the KL divergence between $I^{\pi_\beta}(a|\tilde{s}, s')$ and the new policy $\pi(a|\tilde{s})$, as follows,

$$\min_{\pi} \mathbb{E}_{\tilde{s} \sim \mathcal{D}_{tot}} KL\Big(\mathbb{E}_{s' \sim P(s'|s,\pi_\beta)} I^{\pi_\beta}(a|\tilde{s}, s') \Big\| \pi(a|\tilde{s})\Big) \tag{7}$$

where the $\tilde{s}$ is sampled from the mixed dataset $\mathcal{D}_{tot}$ and s' is its in-sample consecutive state. (7) is the OSR term aiming to play the same role of traditional model-based SDC, but without modeling the high-dimensional observations. Obviously, (7) utilizes a forward KL divergence, in which the difference between $I^{\pi_\beta}(a|\tilde{s}, s')$ and $\pi(a|\tilde{s})$ is weighted by $I^{\pi_\beta}(a|\tilde{s}, s')$. Minimizing this forward KL divergence is equivalent to maximizing the policy likelihood as, $\max_\pi \mathbb{E}_{\tilde{s} \sim \mathcal{D}_{tot}, s' \sim P(s'|s,\pi_\beta), a \sim I^{\pi_\beta}(a|\tilde{s}, s')}[\log \pi(a|\tilde{s})]$, which is easy to achieve by the definition of KL divergence. In this manner, the learnt policy π is able to recover the average behavior of the IDM $E_{s' \sim P(s'|s,\pi_\beta)} I^{\pi_\beta}(a|\tilde{s}, s')$ with greatest probability.

To further explain the feasibility of OSR term, Theorem 1 shows that the OSR term aligns the transited state distribution of the learnt policy with the offline dataset at out-of-sample states in the setting of bounded action space, which is enough to meet the requirements of most RL environments. First, two assumptions are given to serve Theorem 1 in Appendix A, where the assumptions give the continuity and positivity for the transition function of π_β, and the positivity for the new policy we train. Then,

Theorem 1. *Given two consecutive state s and s', the out-of-sample state $\hat{s}$ within the ϵ-neighbourhood of s and the behavior policy π_β. Given an inverse dynamics model $I^{\pi_\beta}(a|s, s')$. In bounded action space setting, the following two optimization formulas are equivalent,*

$$\min_{\pi} KL\Big(\mathbb{E}_{s' \sim P(s'|s,\pi_\beta)} I^{\pi_\beta}(a|\hat{s}, s') \Big\| \pi(a|\hat{s})\Big) \Longleftrightarrow \min_{\pi} KL\Big(P(s'|s, \pi_\beta) \Big\| P(s'|\hat{s}, \pi)\Big)$$

, that is, the two optimization formulas achieve the same policy.

The proof of Theorem 1 is conducted in Appendix B.1. It theoretically guarantees that OSR have the effect that the learnt policy is regularized to recover the transited state distribution $P(s'|\hat{s},\pi)$ at out-of-sample state $\hat{s}$ aligned well with the transited state distribution $P(s'|s,\pi_\beta)$ of the behavior policy π_β at the corresponding in-sample state s. For the fact that the offline dataset is generated via the behavior policy π_β, then $P(s'|s,\pi_\beta)$ is the in-distributional transited state distribution of the offline dataset. Therefore, we remark that OSR helps aligning the new policy's transited state distribution $P(s'|\hat{s},\pi)$ at out-of-sample state $\hat{s}$ with the offline dataset. Interestingly, we find that the term $\min_\pi KL\big(P(s'|s,\pi_\beta)\big\|P(s'|\hat{s},\pi)\big)$ could also be termed as a KL-divergence version of SDC introduced in (4).

4.3 Value-constraint out-of-sample state recovery: a variant

In this section, we propose Out-of-sample State Recovery via value-constraints (OSR-v) as a variant of OSR proposed before. The proposed OSR-v penalizes the Q-values of those actions that prefer transiting to OOD states at out-of-sample states, which drives the agent able to choose the actions that transits to in-sample states, recovering from out-of-sample states. In particular, given two consecutive states $\tilde{s}$ and s' sampled from the mixed dataset $\mathcal{D}_{tot}$ constructed before, we overestimate the Q-values of actions that transits to in-sample states, i.e., actions generated by the IDM $I^{\pi_\beta}(\cdot|\tilde{s},s')$, while suppressing the Q-values of actions with unknown consequences, i.e., actions generated by the new policy π, as follows,

$$\min_Q \mathbb{E}_{\tilde{s},s'\sim\mathcal{D}_{tot}}\Big(\mathbb{E}_{\hat{a}\sim\pi(\cdot|\tilde{s})}Q(\tilde{s},\hat{a}) - \mathbb{E}_{\hat{a}_{tar}\sim I^{\pi_\beta}(\cdot|\tilde{s},s')}Q(\tilde{s},\hat{a}_{tar})\Big) \tag{8}$$

where $\hat{a}_{tar}$ refers to the action sampled from IDM, which is referred as target action.

In practice, we observe that OSR-v enable the agent to follow the *state recovery* principle as well. In order to rigorously explain this phenomenon, we give Theorem 2 under the assumptions mentioned in Appendix A to show that performing OSR-v is equivalent to underestimating the value of OOD states while overestimating the value of in-sample states, which guides the agent prefer to transit to those in-sample(safe) states, avoiding error accumulated via *state distributional shift*.

Theorem 2. *Given two consecutive state s and s', the out-of-sample state $\hat{s}$ within the ϵ-neighbourhood of s and the behavior policy π_β. π is the new policy. Define $Q(s,a)$ as a approximal Q-value function and $V(s)$ as a approximal value function. Given an inverse dynamics model $I^{\pi_\beta}(a|s,s')$. Then the following two optimization formulas are approximately equivalent:*

$$\min_Q \mathbb{E}_{\hat{a}\sim\pi(\cdot|\hat{s})}Q(\hat{s},\hat{a}) - \mathbb{E}_{\hat{a}_{tar}\sim I^{\pi_\beta}(\cdot|\hat{s},s')}Q(\hat{s},\hat{a}_{tar}) \tag{9}$$

$$\overset{\sim}{\Leftrightarrow} \min_V \mathbb{E}_{s'\sim P(s'|\hat{s},\pi)}V(s') - \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)}V(s') \tag{10}$$

, that is, they achieve the same policy approximately. Then the difference of the two optimization formulas is no larger than $2\delta\sum_a \pi_\beta(a|\hat{s})Q(\hat{s},a)$.

The proof of Theorem 2 is conducted in Appendix B.4. Theorem 2 demonstrates the approximate equivalence of (9) and (10). There is actually a little gap between (9) and (10) which is caused by the existence of the continuous gap δ, so when δ is large, (9) and (10) are quite different. However, if we assume the infinite norm of reward function $\|R\|_\infty \le 1$, then $2\delta\sum_a \pi_\beta(a|\hat{s})Q(\hat{s},a) \le \frac{2\delta}{1-\gamma}$, where γ is the discount factor, and by the conservative selection of the magnitude of the noise ϵ, the δ would be a quite tiny value in practice.

The s' sampled from $P(s'|s,\pi_\beta)$, the state distribution aligned with the offline dataset $\mathcal{D}$, is in-sample, then overestimating of such s' increases the agent's preference to transiting to in-sample states, which is supported by the reasoning presented in Proposition 1, as follows,

Proposition 1. *Given an arbitrary state s and its target state s'. Q is an approximal Q-value function and V is its value function. π_Q is the learnt policy according to Q. Let $\pi_Q(a|s)$ is positive correlated to $Q(s,a)$, noted as $\pi_Q(a|s) \propto Q(s,a)$, and assume $\exists a, P(s'|s,a) > 0$, then $P(s'|s,\pi_Q) \propto V(s')$.*

The proof and more discussion of Proposition 1 is conducted in Appendix B.2. For the fact that out-of-sample(noisy) state $\hat{s}$ is close to s, the in-sample prior of s', so we assume $\exists a, P(s'|\hat{s},a) > 0$, and via Proposition 1, $\mathbb{E}_{s'\sim P(s'|s,\pi_\beta)}P(s'|\hat{s},\pi) \propto E_{s'\sim P(s'|s,\pi_\beta)}V(s')$. Therefore, the result of Theorem 2 and Proposition 1 guarantees that OSR-v can align the transited state distribution of the new policy with the offline dataset, avoiding compounding *state distributional shift* as well.

4.4 Implementation and algorithm summary

We implement our methods, OSR and OSR-v, introduced in previous sections based on an Conservative Q-Learning(CQL) framework to practice the *state recovery* principle in the setting of countinuous action space, without modeling the dynamics of the environment. To be specific, we would implement OSR in policy-constraint mode and OSR-v in value-constraint mode.

Policy-constraint mode (OSR). The policy-constraint based method, OSR proposed in (7), could be implemented as SR loss L_{sr} in (11), where the detailed derivation is presented in Appendix B.5.

$$L_{sr} = E_{\tilde{s},s' \sim \mathcal{D}_{tot}} KL\Big(I^{\pi_\beta}(a|\tilde{s},s') \Big\| \pi(a|\tilde{s})\Big) \tag{11}$$

Then the problem is to estimate the KL divergence in (11) under the condition of continuous action space. In general, we often model the policy as $\pi(a|\tilde{s}) = \mathcal{N}(\mu_\pi, \sigma_\pi; \tilde{s})$ and the IDM as $I^{\pi_\beta}(a|\tilde{s},s') = \mathcal{N}(\mu_I, \sigma_I; \tilde{s}, s')$, where $\mathcal{N}$ notes Gaussian distribution. First, we construct the SR loss for the policy network to estimate the KL divergence mentioned in (11).

There is an analytical solution for calculating the KL divergence between two Gaussian distributions, the Gaussian policy $\pi(a|\tilde{s}) = \mathcal{N}(\mu_\pi, \sigma_\pi; \tilde{s})$ and the Gaussian IDM $I^{\pi_\beta}(a|\tilde{s},s') = \mathcal{N}(\mu_I, \sigma_I; \tilde{s}, s')$, so we can transfer (11) as the following SR loss:

$$L_{sr} = \mathbb{E}_{\tilde{s},s' \sim \mathcal{D}_{tot}}\big[\sum_{d=1}^{D}(\log \frac{\sigma_I^d}{\sigma_\pi^d} + \frac{(\sigma_I^d)^2 + (\mu_I^d - \mu_\pi^d)^2}{2(\sigma_\pi^d)^2})\big] \tag{12}$$

where subscript d represents the value of d^{th} dimension of the $D - dimensional$ variable. After constructing the term of OSR in the settings of both discrete and continuous action space, we construct the actor loss function of the policy network as follows,

$$L_\pi = \mathbb{E}_{\tilde{s} \sim \mathcal{D}_{tot}, \hat{a} \sim \pi(\cdot|\tilde{s})}\Big[Q(\tilde{s},\hat{a})\Big] + \lambda L_{sr}$$

where λ is the balance-coefficient and the L_{sr} is as (12) in the setting of continuous action space. L_{sr} imply a behavior cloning from the target distribution estimated via the inverse dynamics model $I^{\pi_\beta}(\cdot|\tilde{s},s')$ to our new policy to enable the agent always transits to those in-sample states s' from no matter in-sample s or out-of-sample $\hat{s}$, i.e., $\tilde{s}$ from the mixed dataset $\mathcal{D}_{tot}$ mentioned in section 4.1. Then the critic loss function for the Q-networks is,

$$L_Q = E_{s,a,r,s' \sim \mathcal{D}}\Big[\alpha \cdot \big(\mathbb{E}_{\hat{a} \sim \pi(\hat{a}|s)} Q(s,\hat{a}) - Q(s,a)\big) + \big(r + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')} Q(s',a') - Q(s,a)\big)^2\Big]$$

where α is the balance-coefficient; The second term is an one-step Bellman error, as (2) shows.

Value-constraint mode (OSR-v). We note the term of (8) mentioned in section 4.3, as L_{sr-v}, to regularize the Q-networks. Then the critic loss function for the Q-networks is as follows,

$$\begin{aligned} L_Q &= \alpha \cdot L_{sr-v} + \mathbb{E}_{s,a,r,s' \sim \mathcal{D}}\Big[r + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')} Q(s',a') - Q(s,a)\Big] \\ where \quad L_{sr-v} &= \mathbb{E}_{\tilde{s},s' \sim \mathcal{D}_{tot}}\Big(\mathbb{E}_{\hat{a} \sim \pi(\cdot|\tilde{s})} Q(\tilde{s},\hat{a}) - \mathbb{E}_{\hat{a}_{tar} \sim I^{\pi_\beta}(\cdot|\tilde{s},s')} Q(\tilde{s},\hat{a}_{tar})\Big) \end{aligned}$$

where α is the balance-coefficient. We replace the CQL regularization, because L_{sr-v} would degrade to CQL regularization when it performs on the in-sample part(the original offline dataset $\mathcal{D}$) of the mixed dataset $\mathcal{D}_{tot}$, as Proposition 2 shows.

Proposition 2. *When training on the original dataset $\mathcal{D}$, L_{sr-v} is equivalent to CQL regularization.*

The proof of Proposition 2 is conducted in Appendix B.3. Proposition 2 shows the feasibility of replacing the term of CQL by the OSR term, so that fewer hyperparameters are introduced. L_{sr-v} overestimates the actions that prefer transiting to in-sample sates to avoid error accumulation. Then the actor loss L_π we use in OSR-v is as follows,

$$L_\pi = \mathbb{E}_{\tilde{s} \sim \mathcal{D}_{tot}, \hat{a} \sim \pi(\cdot|\tilde{s})}\big[Q(\tilde{s},\hat{a})\big]$$

To sum up, both OSR and OSR-v optimize actor loss L_π to update the policy network π and critic loss L_Q to update our Q-networks, and output the learnt policy network π. The whole process of OSR is summarized in Algorithm 1 in Appendix C while the whole process of OSR-v is summarized as Algorithm 2 in Appendix C.

Table 1: Results of **OSR(ours), OSR-v(ours)**, SDC, RAMBO, MOPO, IQL and CQL on offline MuJoCo and AntMaze control tasks averaged over 4 seeds. * indicates the average without 'expert' datasets. The top-2 highest scores in each benchmark are bolded.

Task name	CQL	IQL	MOPO	RAMBO	SDC	OSR	OSR-v
Halfcheetah-r.	35.4	13.1	31.9	**40.0**	**36.2**	35.2± 1.2	35.9± 1.0
Walker2d-r.	7.0	5.4	13.3	11.5	**14.3**	13.5± 2.8	**14.1**± 3.9
Hopper-r.	10.8	7.9	13.0	**21.6**	10.6	10.3± 3.1	**13.5**± 3.4
Halfcheetah-m.	44.4	47.4	40.2	**77.6**	47.1	48.8± 0.2	**49.1**± 0.4
Walker2d-m.	79.2	78.3	26.5	**86.9**	81.1	**85.7**± 0.6	85.1± 0.7
Hopper-m.	58.0	66.2	14.0	**92.8**	91.3	83.1± 1.9	**95.2**± 1.1
Halfcheetah-m.-r.	46.2	44.2	**54.0**	**68.9**	47.3	46.8± 0.4	48.5± 0.6
Walker2d-m.-r.	26.7	73.8	**92.5**	85.0	30.3	**87.9**± 1.1	87.8± 1.0
Hopper-m.-r.	48.6	94.7	42.7	96.6	48.2	**96.7**± 1.7	**101.2**± 0.8
Halfcheetah-m.-e.	62.4	86.7	57.9	93.7	**101.3**	94.7± 0.9	**99.1**± 0.4
Walker2d-m.-e.	98.7	109.6	51.7	68.3	105.3	**114.3**± 1.5	**112.9**± 0.8
Hopper-m.-e.	111.0	91.5	55.0	83.3	112.9	**113.1**± 0.6	**113.2**± 0.6
Halfcheetah-e.	**104.8**	95.0	-	-	**106.6**	97.7± 0.7	102.1± 0.8
Walker2d-e.	**153.9**	109.4	-	-	108.3	110.3± 0.4	**111.4**± 0.8
Hopper-e.	109.9	109.9	-	-	112.6	**113.1**± 0.5	**112.9**± 0.4
MuJoCo-v2 Avg.	66.5	68.9	41.1*	68.5*	70.2	**76.7(69.2*)**	**78.9(71.5*)**
AntMaze-u.	74.0	87.5	0.0	25.0	89.0	**89.9**±1.9	**92.0**±0.8
AntMaze-m.-p.	61.2	71.2	0.0	16.4	**71.9**	66.0±2.5	**71.3**±1.3
AntMaze-l.-p.	15.8	**39.6**	0.0	0.0	37.2	37.9±2.7	**38.3**±2.5
AntMaze-u.-d.	**84.0**	62.2	0.0	0.0	57.3	**74.0**±2.8	69.0±2.5
AntMaze-m.-d.	53.7	70.0	0.0	23.2	**78.7**	**80.0**±2.7	77.0±2.4
AntMaze-l.-d.	14.9	**47.5**	0.0	2.4	33.2	**37.9**±1.5	34.0±2.2
AntMaze-v0 Avg.	50.6	63.0	0.0	22.3	61.2	**64.3**	**63.4**

5 Experiments

In experiments we aim to answer: 1) Does the proposed OSR help to achieve the state-of-the-art performance in offline RL? 2) Is the inverse dynamics model able to guide the policy recovering from out-of-sample states? 3) Is OSR able to improve the agent's robustness for out-of-sample situations? 4) Does OSR generalize well if the offline dataset only covers limited valuable area of the state space. Our experiments are organized as follows: Firstly, we conduct a comparative study on the MuJoCo and AntMaze benchmarks in the D4RL datasets for different versions of our method; then we design an out-of-sample MuJoCo setting to evaluate the generalization ability of CQL, SDC, the proposed OSR and OSR-v, and explore the behavior of the inverse dynamics model(IDM) and OSR policy; finally, we test the efficacy of OSR when presented with limited valuable data. A brief introduction of our code is available in Appendix D.9. **MuJoCo.** There are three types of high-dimensional control environments representing different robots in D4RL: Hopper, Halfcheetah and Walker2d, and five kinds of datasets: 'random', 'medium', 'medium-replay', 'medium-expert' and 'expert'. The 'random' is generated by a random policy and the 'medium' is collected by an early-stopped SAC [9] policy. The 'medium-replay' collects the data in the replay buffer of the 'medium' policy. The 'expert' is produced by a completely trained SAC. The 'medium-expert' is a mixture of 'medium' and 'expert' in half. **AntMaze.** AntMaze in D4RL is a maze-like environment where an ant agent must navigate through obstacles to reach a goal location. There are three different layouts of maze: 'umaze', 'medium' and 'large', and different dataset types: 'fixed', 'play' and 'diverse' which has different start and goal locations used to collect the dataset. **Hyperparameter Details.** We base the hyperparameters for OSR on those used in CQL [16]. We observe the performance of OSR is influenced by the noise magnitude β and SR weighting λ. *Therefore, we conduct a sensitivity analysis of OSR in Appendix D.1.*

Other experiments, including Sensitive Analysis, Ablation study, Comparison Study of implementations with Q-ensemble trick and so on, are attached in Appendix D.

5.1 Comparative study on offline MuJoCo/AntMaze suite

In this section, we compare our methods with several significant methods, including MOPO [26], RAMBO [21], IQL [14], CQL [16] and SDC [27], on the D4RL dataset in the MuJoCo benchmarks. Part of the results for the comparative methods are obtained by [27, 21]. All these methods are implemented without Q-ensemble trick [4] and results are shown in Table 1, where the proposed OSR and OSR-v achieve the similar or better performance than other methods on most tasks and the best average score, and we also remark that OSR and OSR-v generalize significantly better on the 'medium-expert' datasets where limited high-valuable data is available. More training details of OSR/OSR-v could be achieved in Appendix D.3. Besides, we also implement OSR with Q-ensemble trick of 10 Q-functions [4], termed as OSR-10, to compare with Q-ensemble based methods like RORL [25], SAC-10 and EDAC-10 [4], in Appendix D.4. From the results above, we conclude that OSR helps to achieve the state-of-the-art performance in offline RL.

5.2 Out-of-sample MuJoCo setting

To evaluate whether the inverse dynamics model and the proposed OSR policy enable the agent to recover from out-of-sample situations, we design Out-of-sample MuJoCo (OOSMuJoCo) benchmarks by adding different steps of random actions (Gaussian noise with magnitude of 1e-3) to simulate external force to knock the agent into out-of-sample states on Halfcheetah, Walker2d and Hopper. There are 3 levels of external force: slight(s) takes 5 steps of random actions, moderate(m) takes 10 steps and large(l) takes 20 steps. The visualization of OOSMuJoCo is shown in Figure 2.

Figure 2: The visualizations of the 9 OOSMuJoCo benchmarks. The red arrows represent the direction the agents move after external force, and the agents fall into out-of-sample states then.

We evaluate the performance of policies trained by CQL, SDC, OSR and OSR-v on 'medium-expert' datasets in the three benchmarks. The score and performance decrease of these policies across the 9 OOSMuJoCo benchmarks are shown in Table 2, where the performance decrease is measured as the percentage reduction in scores from OOSMuJoCo compared to the results of standard MuJoCo environments in Table 1. Our results indicate that the proposed OSR/OSR-v experiences significantly less performance degradation than the other two methods across most of the OOSMuJoCo benchmarks, demonstrating better robustness against perturbation for out-of-sample situations in the complicated and non-stationary real-world environments.

Table 2: Results of CQL, SDC, OSR/OSR-v in OOSMuJoCo setting on the normalized return and decrease metric averaged over 4 seeds. The hightest score and lowest decrease are bolded.

Task name	CQL score	dec.(%)	SDC score	dec.(%)	OSR score	dec.(%)	OSR-v score	dec.(%)
Halfcheetah-OOS-s.	56.2	10.0	**100.9**	0.4	94.1	**0.6**	98.4	0.7
Halfcheetah-OOS-m.	52.9	15.3	**99.8**	1.5	92.7	2.1	98.4	**0.7**
Halfcheetah-OOS-l.	42.1	32.5	83.3	17.8	**91.7**	**3.2**	79.8	19.5
Walker2d-OOS-s.	81.1	47.3	102.7	2.5	**113.3**	**0.9**	110.3	2.3
Walker2d-OOS-m.	77.4	21.6	102.2	2.9	**113.2**	**1.0**	110.9	1.8
Walker2d-OOS-l.	52.0	47.3	95.6	9.2	**110.1**	**3.7**	106.7	5.5
Hopper-OOS-s.	110.4	**0.6**	112.0	**0.6**	111.4	1.5	**112.5**	**0.6**
Hopper-OOS-m.	97.3	12.3	110.8	1.9	109.2	3.4	**112.2**	**0.9**
Hopper-OOS-l.	54.4	51.0	89.2	20.9	**106.1**	**6.2**	105.6	6.7

Besides, as a toy example, we visualize the Halfcheetah agent within an expert trajectory produced by the behavior policy π_β on the offline dataset $\mathcal{D}$, as is shown in Figure 3, then we also visualize the trajectories generated by the policy of OSR and the inverse dynamics model(IDM) guided from

the behavior policy, where we set the target state of IDM to the corresponding state in the expert trajectory. We add external force to knock the agent into an out-of-sample state and we observe that the agents equiped with IDM and OSR policy are both able to recover from the out-of-sample states. The state recovery processes of other 2 benchmarks is shown in Appendix D.5.2 and the visualizations of state distribution is shown in Appendix D.5.1. The results above demonstrate that the the inverse dynamics model is able to guide the policy recovering from out-of-sample state and OSR is able to improve the agent's robustness for out-of-sample situations.

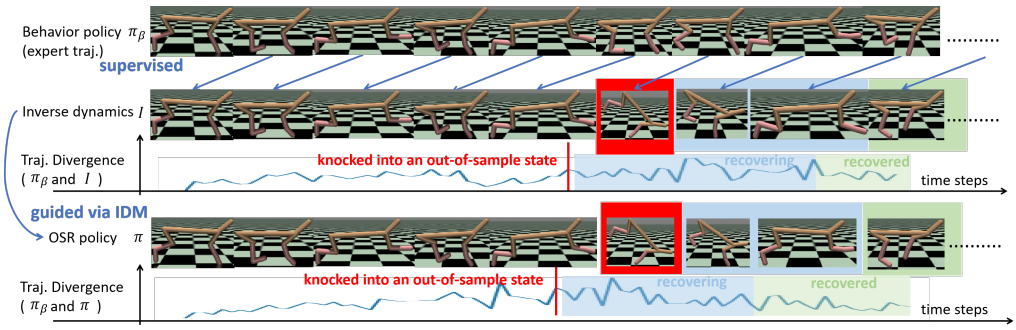


Figure 3: The state recovery process via IDM and OSR on the 'Halfcheetah-OOS-large' benchmark. The interval between every two images is ten steps.

5.3 Limited valuable data setting

In this section, we test our method's feasibility when only limited valuable data is available for training, which is a novel task with the mixture of the expert dataset and random dataset with different ratios [27], aiming to evaluate the robustness of ORL algorithms on the condition that the offline dataset only covers limited valuable state space. In this paper, the proportions of random samples are 0.5, 0.6, 0.7, 0.8 and 0.9 and the ratios of expert samples decline, for Halfcheetah, Hopper and Walker2d. All datasets contain 1,000,000 samples.

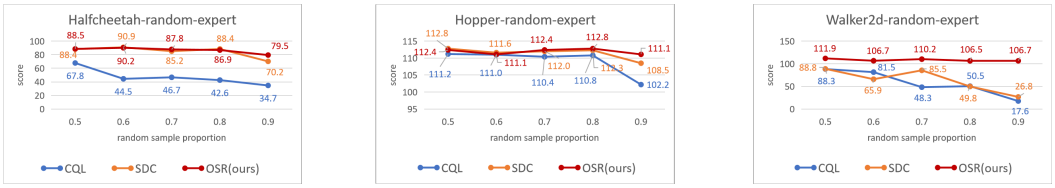


Figure 4: The curve of CQL, SDC and OSR in the setting of limited valuable data.

We compare the proposed OSR with CQL and SDC in a limited valuable data setting. The results, shown in Figure 4, demonstrate that all methods experience performance degradation when expert samples are limited. However, our proposed method, OSR, achieves the lowest performance decrease with the raise of the random sample ratio among the three methods in most tasks, which suggests that our method is less sensitive to the quality of training data. The results above mean that even if the dataset covers only limited valuable area of the extensive state space, the proposed OSR still works.

6 Conclusion

In this paper, we proposed a simple yet effective method, named Out-of-sample State Recovery (OSR), for offline reinforcement learning. The proposed method follows the *state recovery* principle to deal with the *state distributional shift* issue commonly encountered in offline RL. Unlike previous methods, we address this in a way that does not need to explicitly model the transition of the underlying environment by treating the inverse dynamics model as a guidance with desired immediate consequence for the new policy. Our theoretical results show that the proposed OSR is equivalent to aligning the transition distribution of the learnt policy with the offline dataset. The proposed method is evaluated on several offline AntMaze/MuJoCo settings and achieved the SOTA performance.

Acknowledgement

This work is partially supported by National Key R&D program of China (2021ZD0113203), National Science Foundation of China (61976115).

References

[1] Mohammad Mehdi Afsar, Trafford Crump, and Behrouz H. Far. Reinforcement learning based recommender systems: A survey. *CoRR*, abs/2101.06286, 2021.

[2] Cameron Allen, Neev Parikh, Omer Gottesman, and George Konidaris. Learning markov state abstractions for deep reinforcement learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 8229–8241, 2021.

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016.

[4] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 7436–7447, 2021.

[5] Jongwook Choi, Yijie Guo, Marcin Moczulski, Junhyuk Oh, Neal Wu, Mohammad Norouzi, and Honglak Lee. Contingency-aware exploration in reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[6] Paul F. Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *CoRR*, abs/1610.03518, 2016.

[7] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 20132–20145, 2021.

[8] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. PMLR, 2019.

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. PMLR, 2018.

[10] Geoffrey E. Hinton and Sam T. Roweis. Stochastic neighbor embedding. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 833–840. MIT Press, 2002.

[11] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pages 651–673. PMLR, 2018.

[12] Hao-Cheng Kao, Kai-Fu Tang, and Edward Y. Chang. Context-aware symptom checking for disease diagnosis using hierarchical reinforcement learning. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2305–2313. AAAI Press, 2018.

[13] B. Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Kumar Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Trans. Intell. Transp. Syst.*, 23(6):4909–4926, 2022.

[14] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[15] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11761–11771, 2019.

[16] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[17] Michael Laskey, Jonathan Lee, Roy Fox, Anca D. Dragan, and Ken Goldberg. DART: noise injection for robust imitation learning. In *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, volume 78 of *Proceedings of Machine Learning Research*, pages 143–156. PMLR, 2017.

[18] 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. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[19] Andrew Lobbezoo, Yanjun Qian, and Hyock-Ju Kwon. Reinforcement learning for pick and place operations in robotics: A survey. *Robotics*, 10(3):105, 2021.

[20] Michael W. McCracken. Robust out-of-sample inference. *Journal of Econometrics*, 99(2):195–223, 2000.

[21] Marc Rigter, Bruno Lacerda, and Nick Hawes. RAMBO-RL: robust adversarial model-based offline reinforcement learning. *CoRR*, abs/2204.12581, 2022.

[22] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1889–1897. JMLR.org, 2015.

[23] Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *arXiv e-prints*, 2021.

[24] Christopher J. C. H. Watkins and Peter Dayan. Technical note q-learning. *Mach. Learn.*, 8:279–292, 1992.

[25] Rui Yang, Chenjia Bai, Xiaoteng Ma, Zhaoran Wang, Chongjie Zhang, and Lei Han. RORL: robust offline reinforcement learning via conservative smoothing. *CoRR*, abs/2206.02829, 2022.

[26] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y. Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: model-based offline policy optimization. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[27] Hongchang Zhang, Jianzhun Shao, Yuhang Jiang, Shuncheng He, Guanwen Zhang, and Xiangyang Ji. State deviation correction for offline reinforcement learning. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, pages 9022–9030. AAAI Press, 2022.

Appendix

A Two assumptions

Assumption 1. *The transition function $P(s'|s, \pi_\beta)$ could be considered as a function according to variable s, that is $f(s)$. Then we assume this function is continuous:*

$$\begin{aligned}\forall \epsilon, \exists \delta, s.t. \|s - \hat{s}\| \leq \epsilon \Rightarrow \\ \|P(s'|s, \pi_\beta) - P(s'|\hat{s}, \pi_\beta)\| \leq \delta\end{aligned}$$

and $\forall s, s'$, we have $P(s'|s, \pi_\beta) > 0$.

By Assumption 1, we could easily infer:

$$\begin{aligned}\forall \epsilon, \exists \delta, s.t. \|s - \hat{s}\| \leq \epsilon \Rightarrow \\ 1 - \delta \leq \frac{P(s'|s, \pi_\beta)}{P(s'|\hat{s}, \pi_\beta)} \leq 1 + \delta\end{aligned}$$

Assumption 2. *Assume the new policy we train is a soft exploration policy, which choose every action a from the action set A with probability ϵ_π at least at each state s in S, i.e., $\forall s \in S, a \in A, \pi(a|s) \geq \epsilon_\pi > 0$.*

The rationality of this comes from the boundedness of action space A. For example, one typical way we model the policy π is to construct a Gaussian distribution. Noting the upper bound of the action space A is a_{upp} and the lower bound is a_{low}, then $\epsilon_\pi = \min(\pi(a_{upp}|s), \pi(a_{low}|s)) > 0$.

B Proofs

B.1 Proof of Theorem 1

Theorem 1. Given two consecutive state s and s', the out-of-sample state $\hat{s}$ within the ϵ-neighbourhood of s and the behavior policy π_β. Given an inverse dynamics model $I^{\pi_\beta}(a|s, s')$. In bounded action space setting, the following two optimization formulas are equivalent,

$$\min_\pi KL\left(\mathbb{E}_{s' \sim P(s'|s, \pi_\beta)} I^{\pi_\beta}(a|\hat{s}, s') \middle\| \pi(a|\hat{s})\right) \tag{13}$$

$$\iff \min_\pi KL\left(P(s'|s, \pi_\beta) \middle\| P(s'|\hat{s}, \pi)\right) \tag{14}$$

, that is, the two optimization formulas achieve the same policy.

Proof.

$$
\begin{aligned}
&\min_{\pi} KL\left(\mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} I^{\pi_\beta}(a|\hat{s},s') \middle\| \pi(a|\hat{s})\right) \\
\Leftrightarrow &\min_{\pi} \sum_a \sum_{s'} P(s'|s,\pi_\beta) \frac{\pi_\beta(a|\hat{s}) P(s'|\hat{s},a)}{P(s'|\hat{s},\pi_\beta)} \log \sum_{\tilde{s}'} P(s'|s,\pi_\beta) \frac{\pi_\beta(a|\hat{s}) P(\tilde{s}'|\hat{s},a)}{P(\tilde{s}'|\hat{s},\pi_\beta)\pi(a|\hat{s})} \\
\Leftrightarrow &\min_{\pi} \sum_a \sum_{s'} P(s'|s,\pi_\beta) \frac{\pi_\beta(a|\hat{s}) P(s'|\hat{s},a)}{P(s'|\hat{s},\pi_\beta)} \log \frac{1}{\pi(a|\hat{s})} \\
\leq &\min_{\pi} (1+\delta) \sum_a \sum_{s'} \pi_\beta(a|\hat{s}) P(s'|\hat{s},a) \log \frac{1}{\pi(a|\hat{s})} \qquad (Assumption 1) \\
\Leftrightarrow &\min_{\pi} \sum_a \sum_{s'} \pi_\beta(a|\hat{s}) P(s'|\hat{s},a) \log \frac{\epsilon_\pi P(s'|\hat{s},\pi_\beta)}{\pi(a|\hat{s})} \\
\leq &\min_{\pi} \sum_a \sum_{s'} \pi_\beta(a|\hat{s}) P(s'|\hat{s},a) \log \frac{P(s'|\hat{s},\pi_\beta)}{\sum_a \pi(a|\hat{s}) P(s'|\hat{s},a)} \qquad (Assumption 2) \\
\Leftrightarrow &\min_{\pi} \sum_{s'} P(s'|\hat{s},\pi_\beta) \log \frac{P(s'|\hat{s},\pi_\beta)}{P(s'|\hat{s},\pi)} \\
\leq &\min_{\pi} \sum_{s'} \frac{P(s'|s,\pi_\beta)}{1-\delta} \log \frac{\frac{P(s'|s,\pi_\beta)}{1-\delta}}{P(s'|\hat{s},\pi)} \qquad (Assumption 1) \\
\Leftrightarrow &\min_{\pi} \sum_{s'} P(s'|s,\pi_\beta) \log \frac{P(s'|s,\pi_\beta)}{P(s'|\hat{s},\pi)} \\
\Leftrightarrow &\min_{\pi} KL\left(P(s'|s,\pi_\beta) \middle\| P(s'|\hat{s},\pi)\right)
\end{aligned} \tag{15}
$$

On the other hand:

$$
\begin{aligned}
&\min_{\pi} KL\left(\mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} I^{\pi_\beta}(a|\hat{s},s') \middle\| \pi(a|\hat{s})\right) \\
\Leftrightarrow &\min_{\pi} \sum_a \sum_{s'} P(s'|s,\pi_\beta) \frac{\pi_\beta(a|\hat{s}) P(s'|\hat{s},a)}{P(s'|\hat{s},\pi_\beta)} \log \frac{1}{\pi(a|\hat{s})} \\
\geq &\min_{\pi} (1-\delta) \sum_a \sum_{s'} \pi_\beta(a|\hat{s}) P(s'|\hat{s},a) \log \frac{1}{\sum_a \pi(a|\hat{s}) P(s'|\hat{s},a)} \qquad (Assumption 1) \\
\Leftrightarrow &\min_{\pi} \sum_{s'} P(s'|\hat{s},\pi) \log \frac{P(s'|\hat{s},\pi_\beta)}{P(s'|\hat{s},\pi)} \\
\geq &\min_{\pi} \sum_{s'} \frac{P(s'|s,\pi)}{1+\delta} \log \frac{\frac{P(s'|s,\pi_\beta)}{1+\delta}}{P(s'|\hat{s},\pi)} \qquad (Assumption 1) \\
\Leftrightarrow &\min_{\pi} \sum_{s'} P(s'|s,\pi_\beta) \log \frac{P(s'|s,\pi_\beta)}{P(s'|\hat{s},\pi)} \\
\Leftrightarrow &\min_{\pi} KL\left(P(s'|s,\pi_\beta) \middle\| P(s'|\hat{s},\pi)\right)
\end{aligned} \tag{16}
$$

Combining (15) and (16), we complete the proof. □

B.2 Proof of Proposition 1

Proposition 1. Given an arbitrary state s and its target state s'. Q is an approximal Q-value function and V is its value function. π_Q is the learnt policy according to Q. Let $\pi_Q(a|s)$ is positive correlated to $Q(s,a)$, noted as $\pi_Q(a|s) \propto Q(s,a)$, and assume $\exists a, P(s'|s,a) > 0$, then $P(s'|s,\pi_Q) \propto V(s')$.

Proof.

$$\pi_Q(a|s) \propto Q(s,a) \tag{17}$$

$$\Rightarrow \pi_Q(a|s) \cdot \sum_{s'} P(s'|s,a) \propto R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s') \tag{18}$$

$$\Rightarrow \sum_{s'} \big[\pi_Q(a|s) \cdot P(s'|s,a)\big] \propto \sum_{s'} P(s'|s,a)\big[R(s,a) + \gamma V(s')\big] \tag{19}$$

$$\Rightarrow \big[\pi_Q(a|s) \cdot P(s'|s,a)\big] \propto P(s'|s,a)\big[R(s,a) + \gamma V(s')\big] \tag{20}$$

$$\Rightarrow \sum_{a} \big[\pi_Q(a|s) \cdot P(s'|s,a)\big] \propto \sum_{a} P(s'|s,a)R(s,a) + V(s') \cdot \gamma \sum_{a} P(s'|s,a) \tag{21}$$

$$\Rightarrow P(s'|s,\pi_Q) \propto C_1(s,s') + V(s') \cdot C_2(s,s',\gamma) \tag{22}$$

$$\Rightarrow P(s'|s,\pi_Q) \propto V(s') \tag{23}$$

complete the proof. □

Proposition 1 shows that from any state s, overestimating the value of a target state s' would enhance the potentiality for the agent to transit to this target s'(except the condition that s' is unreachable from s, i.e., $\forall a, P(s'|s,a) = 0$). In this paper, the transition we wish the agent to take is from the noisy $\hat{s}$ of s to the s' which is known reachable from s. For the reason that $\hat{s}$ is not very far away from s, so we can assume s' is reachable from $\hat{s}$, and by Proposition 1, we conclude that $\mathbb{E}_{s' \sim P(s'|s,\pi_\beta)} P(s'|\hat{s},\pi) \propto E_{s' \sim P(s'|s,\pi_\beta)} V(s')$. For the fact that $P(s'|s,\pi_\beta)$ is the transited state distribution conditioned on s that aligned with the offline dataset $\mathcal{D}$, we conclude overestimating of in-sample s' increases the agent's preference to transiting to in-sample states.

B.3 Proof of Proposition 2

Proposition 2. When training on the original dataset $\mathcal{D}$, L_{sr-v} is equivalent to CQL regularization.

Proof. The proof is performed on the equivalence of OSR and CQL regularizations given an in-sample state s:

$$\begin{aligned}
&\min_Q \mathbb{E}_{a \sim \pi(a|s)} Q(s,a) - \mathbb{E}_{s' \sim P(s'|s,\pi_\beta)} \mathbb{E}_{a \sim I^{\pi_\beta}(a|s,s')} Q(s,a) \\
\Leftrightarrow &\min_Q \mathbb{E}_{a \sim \pi(a|s)} Q(s,a) - \sum_{s'} P(s'|s,\pi_\beta) \sum_{a} \frac{\pi_\beta(a|s) P(s'|s,a)}{P(s'|s,\pi_\beta)} Q(s,a) \\
\Leftrightarrow &\min_Q \mathbb{E}_{a \sim \pi(a|s)} Q(s,a) - \sum_{a} \pi_\beta(a|s) Q(s,a) \\
\Leftrightarrow &\min_Q \mathbb{E}_{a \sim \pi(a|s)} Q(s,a) - \mathbb{E}_{a \sim \pi_\beta(a|s)} Q(s,a) \\
\Leftrightarrow &\quad CQL regularization
\end{aligned}$$

Complete the proof. □

B.4 Proof of Theorem 2

Theorem 2. Given two consecutive state s and s', the out-of-sample state $\hat{s}$ within the ϵ-neighbourhood of s and the behavior policy π_β. π is the new policy. Define $Q(s,a)$ as a approximal Q-value function and $V(s)$ as a approximal value function. Given an inverse dynamics model $I^{\pi_\beta}(a|s,s')$. Then the following two optimization formulas are approximately equivalent:

$$\min_Q E_{\hat{a} \sim \pi(\cdot|\hat{s})} Q(\hat{s},\hat{a}) - \mathbb{E}_{\hat{a}_{tar} \sim I^{\pi_\beta}(\cdot|\hat{s},s')} Q(\hat{s},\hat{a}_{tar}) \tag{24}$$

$$\widetilde{\Leftrightarrow} \min_V \mathbb{E}_{s' \sim P(s'|\hat{s},\pi)} V(s') - \mathbb{E}_{s' \sim P(s'|s,\pi_\beta)} V(s') \tag{25}$$

, that is, they achieve the same policy approximately. Then the difference of the two optimization formulas is no larger than $2\delta \sum_a \pi_\beta(a|\hat{s}) Q(\hat{s},a)$.

Proof. First we derive the upper bound of (24):

$$\begin{aligned}
&\min_Q E_{\hat{a}\sim\pi(\cdot|\hat{s})}Q(\hat{s},\hat{a}) - \mathbb{E}_{\hat{a}_{tar}\sim I^{\pi_\beta}(\cdot|\hat{s},s')}Q(\hat{s},\hat{a}_{tar}) \\
\Leftrightarrow &\min_Q \sum_a \pi(a|\hat{s})Q(\hat{s},a) - \sum_{s'} P(s'|s,\pi_\beta)\sum_a \frac{\pi_\beta(a|\hat{s})P(s'|\hat{s},a)}{P(s'|\hat{s},\pi_\beta)}Q(\hat{s},a) \\
\Leftrightarrow &\min_Q \sum_a \left[\pi(a|\hat{s}) - \sum_{s'} P(s'|s,\pi_\beta)\frac{\pi_\beta(a|\hat{s})P(s'|\hat{s},a)}{P(s'|\hat{s},\pi_\beta)}\right]Q(\hat{s},a) \\
\leq &\min_Q \sum_a \left[\pi(a|\hat{s}) - (1-\delta)\pi_\beta(a|\hat{s})\right]Q(\hat{s},a) \qquad (Assumption1)
\end{aligned}$$

Similarly, we have the lower bound of (24):

$$\begin{aligned}
&\min_Q E_{\hat{a}\sim\pi(\cdot|\hat{s})}Q(\hat{s},\hat{a}) - \mathbb{E}_{\hat{a}_{tar}\sim I^{\pi_\beta}(\cdot|\hat{s},s')}Q(\hat{s},\hat{a}_{tar}) \\
\geq &\min_Q \sum_a \left[\pi(a|\hat{s}) - (1+\delta)\pi_\beta(a|\hat{s})\right]Q(\hat{s},a)
\end{aligned}$$

Then we derive the upper bound of (25):

$$\begin{aligned}
&\min_V \mathbb{E}_{s'\sim P(s'|\hat{s},\pi)}V(s') - \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)}V(s') \\
\Leftrightarrow &\min_V \sum_{s'} P(s'|\hat{s},\pi)V(s') - \sum_{s'} P(s'|s,\pi_\beta)V(s') \\
\leq &\min_V \sum_{s'} P(s'|\hat{s},\pi)V(s') - \sum_{s'}(1-\delta)P(s'|\hat{s},\pi_\beta)V(s') \qquad (Assumption1) \\
\Leftrightarrow &\min_V \sum_{s'}\sum_a \pi(a|\hat{s})P(s'|\hat{s},a)V(s') - \sum_{s'}(1-\delta)\sum_a \pi_\beta(a|\hat{s})P(s'|\hat{s},a)V(s') \\
\Leftrightarrow &\min_V \sum_a \left[\pi(a|\hat{s}) - (1-\delta)\pi_\beta(a|\hat{s})\right]\cdot\left[\gamma P(s'|\hat{s},a)V(s') + R(\hat{s},a) - R(\hat{s},a)\right] \qquad (BellmanEquation) \\
\Leftrightarrow &\min_Q \sum_a \left[\pi(a|\hat{s}) - (1-\delta)\pi_\beta(a|\hat{s})\right]\left[Q(\hat{s},a) - R(\hat{s},a)\right] \\
\Leftrightarrow &\min_Q \sum_a \left[\pi(a|\hat{s}) - (1-\delta)\pi_\beta(a|\hat{s})\right]Q(\hat{s},a)
\end{aligned}$$

Similarly, we have the lower bound of (25):

$$\begin{aligned}
&\min_V \mathbb{E}_{s'\sim P(s'|\hat{s},\pi)}V(s') - \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)}V(s') \\
\geq &\min_Q \sum_a \left[\pi(a|\hat{s}) - (1+\delta)\pi_\beta(a|\hat{s})\right]Q(\hat{s},a)
\end{aligned}$$

Therefore, for the face that (24) and (25) have the same upper and lower bounds, we conclude that

$$\min_Q E_{\hat{a}\sim\pi(\cdot|\hat{s})}Q(\hat{s},\hat{a}) - \mathbb{E}_{\hat{a}_{tar}\sim I^{\pi_\beta}(\cdot|\hat{s},s')}Q(\hat{s},\hat{a}_{tar}) \tag{26}$$

is approximately equivalent to:

$$\min_V \mathbb{E}_{s'\sim P(s'|\hat{s},\pi)}V(s') - \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)}V(s') \tag{27}$$

for the fact that they would achieve the similar policy. Then we subtract the upper bound $\sum_a \left[\pi(a|\hat{s}) - (1-\delta)\pi_\beta(a|\hat{s})\right]Q(\hat{s},a)$ with the lower bound $\sum_a \left[\pi(a|\hat{s}) - (1+\delta)\pi_\beta(a|\hat{s})\right]Q(\hat{s},a)$, and we can get the bound of the difference between (24) and (25), which is $2\delta\sum_a \pi_\beta(a|\hat{s})Q(\hat{s},a)$.
Complete the proof.

□

B.5 Explanation of the difference between the theory and implementation

First, we add the original s to the tuple $(\tilde{s}, s')$ in the D_{tot} for clearer derivation, as $(s, \tilde{s}, s')$. And it looks like that the proposed optimization objective (Eq.7) is not equivalent to its actual implementation (Eq.11), i.e.,

$$\mathbb{E}_{(s,\tilde{s})\sim D_{tot}} D_{KL}\left[\mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} I^{\pi_\beta}(a|\tilde{s}, s') \middle\| \pi(a|\tilde{s})\right] \neq \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} D_{KL}\left[I^{\pi_\beta}(a|\tilde{s}, s') \middle\| \pi(a|\tilde{s})\right]$$

, where $P(s'|s, \pi_\beta)$ is the transition distribution.

However, these two optimization problems are actually equivalent, in the sense that they induce the same solution,

$$\begin{aligned}
&\arg\min_{\pi} \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} D_{KL}\left[\mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} I^{\pi_\beta}(a|\tilde{s}, s') \middle\| \pi(a|\tilde{s})\right] \\
=&\arg\min_{\pi} \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} D_{KL}\left[I^{\pi_\beta}(a|\tilde{s}, s') \middle\| \pi(a|\tilde{s})\right]
\end{aligned}$$

In what below, we give the detailed derivation,

$$\begin{aligned}
&\arg\min_{\pi} \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} D_{KL}\left[\mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} I^{\pi_\beta}(a|\tilde{s}, s') \middle\| \pi(a|\tilde{s})\right] \\
=&\arg\min_{\pi} \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \sum_a \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} I^{\pi_\beta}(a|\tilde{s}, s') \log \frac{\mathbb{E}_{s''\sim P(s''|s,\pi_\beta)} I^{\pi_\beta}(a|\tilde{s}, s'')}{\pi(a|\tilde{s})} \\
=&\arg\min_{\pi} \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} \sum_a I^{\pi_\beta}(a|\tilde{s}, s') \log \frac{\mathbb{E}_{s''\sim P(s''|s,\pi_\beta)} I^{\pi_\beta}(a|\tilde{s}, s'')}{\pi(a|\tilde{s})} \\
=&\arg\min_{\pi} \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} \sum_a I^{\pi_\beta}(a|\tilde{s}, s') \log \mathbb{E}_{s''\sim P(s''|s,\pi_\beta)} I^{\pi_\beta}(a|\tilde{s}, s'') \\
&\qquad + \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} \sum_a I^{\pi_\beta}(a|\tilde{s}, s') \log \frac{1}{\pi(a|\tilde{s})}
\end{aligned} \tag{28}$$

Note the term $\mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} \sum_a I^{\pi_\beta}(a|\tilde{s}, s') \log \mathbb{E}_{s''\sim P(s''|s,\pi_\beta)} I^{\pi_\beta}(a|\tilde{s}, s'')$ in Eq.(28) is a constant w.r.t. π, hence we can remove it as follows,

(28)

$$= \arg\min_{\pi} \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} \sum_a I^{\pi_\beta}(a|\tilde{s}, s') \log \frac{1}{\pi(a|\tilde{s})} \tag{29}$$

We remark that $\mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} \sum_a I^{\pi_\beta}(a|\tilde{s}, s') \log I^{\pi_\beta}(a|\tilde{s}, s')$ is a constant w.r.t. π, so we can add it onto Eq.(29) as follows,

(29)

$$\begin{aligned}
&= \arg\min_{\pi} \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} \sum_a I^{\pi_\beta}(a|\tilde{s}, s') \log \frac{1}{\pi(a|\tilde{s})} \\
&\qquad + \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} \sum_a I^{\pi_\beta}(a|\tilde{s}, s') \log I^{\pi_\beta}(a|\tilde{s}, s') \\
&= \arg\min_{\pi} \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} \sum_a I^{\pi_\beta}(a|\tilde{s}, s') \log \frac{I^{\pi_\beta}(a|\tilde{s}, s')}{\pi(a|\tilde{s})} \\
&= \arg\min_{\pi} \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} D_{KL}\left[I^{\pi_\beta}(a|\tilde{s}, s') \middle\| \pi(a|\tilde{s})\right]
\end{aligned} \tag{30}$$

And then we can remove the expectation w.r.t. s' in Eq.(30) with Monte Calro approximation with the sample number N as 1,

$$\begin{aligned}
&\arg\min_{\pi} \mathbb{E}_{(s,\tilde{s})\sim D_{tot}} \mathbb{E}_{s'\sim P(s'|s,\pi_\beta)} D_{KL}\left[I^{\pi_\beta}(a|\tilde{s},s') \middle\| \pi(a|\tilde{s})\right] \\
\approx &\arg\min_{\pi} \mathbb{E}_{\tilde{s}\sim D_{tot}} \frac{1}{N}\sum_{i=1}^{N} D_{KL}\left[I^{\pi_\beta}(a|\tilde{s},s'_i) \middle\| \pi(a|\tilde{s})\right] \\
\approx &\arg\min_{\pi} \mathbb{E}_{\tilde{s}\sim D_{tot}} D_{KL}\left[I^{\pi_\beta}(a|\tilde{s},s') \middle\| \pi(a|\tilde{s})\right] \qquad (31)
\end{aligned}$$

where Eq.(31) is the Eq.(11) in our paper.

C Alogrithms

The pseudoccode of OSR is listed in Algorithm 1, and the pseudoccode of OSR-v is listed in Algorithm 2.

Algorithm 1 Policy-constraint Out-of-sample State Recovery (OSR)

Input: mixed offline dataset $\mathcal{D}_{tot}$, maximal update iterations T,
Parameter: policy network π, Q-networks Q_1, Q_2, inverse dynamics model I,
Output: learnt policy network π

1: Initialize the policy network, Q-networks and the inverse dynamics model.
2: Let $t = 0$.
3: **while** $t < T$ **do**
4: Sample mini-batch of N samples $(\tilde{s}, a, r, s')$ from $\mathcal{D}_{tot}$.
5: Train the inverse dynamics model I.
6: Feed $\tilde{s}$ to the policy network π and get μ_π, σ_π.
7: Get $\hat{a} = \mu_\pi + \sigma_\pi \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$.
8: Feed $\tilde{s}$ and s' to the inverse dynamics model I and get μ_I, σ_I.
9: Update the policy network π according to,

$$\begin{aligned}
L_{sr} &= \mathbb{E}_{\tilde{s},s'\sim\mathcal{D}_{tot}}\big[\log\frac{|\sigma_I|}{|\sigma_\pi|} + tr(\sigma_\pi^{-1}\sigma_I) + (\mu_I - \mu_\pi)^T\sigma_\pi^{-1}(\mu_I - \mu_\pi)\big] \\
L_\pi &= \mathbb{E}_{\tilde{s},s'\sim\mathcal{D}_{tot},\hat{a}\sim\pi(\cdot|\tilde{s})}\Big[Q(\tilde{s},\hat{a})\Big] + \lambda L_{sr}.
\end{aligned}$$

10: Update the Q-networks according to,

$$L_Q = E_{s,a,r,s'\sim\mathcal{D}}\Big[\alpha\cdot\big(\mathbb{E}_{\hat{a}\sim\pi(\hat{a}|s)}Q(s,\hat{a}) - Q(s,a)\big) + \big(r + \gamma\mathbb{E}_{a'\sim\pi(\cdot|s')}Q(s',a') - Q(s,a)\big)^2\Big].$$

11: **end while**
12: **return** learnt policy network π.

Algorithm 2 Value-constraint Out-of-sample State Recovery (OSR-v)

Input: offline dataset $\mathcal{D}_{tot}$, maximal update iterations T,
Parameter: policy network π, Q-networks Q_1, Q_2, inverse dynamics model I,
Output: learnt policy network π

1: Initialize the policy network, Q-networks and the inverse dynamics model.
2: Let $t = 0$.
3: **while** $t < T$ **do**
4: Sample mini-batch of N samples $(\tilde{s}, a, r, s')$ from $\mathcal{D}_{tot}$.
5: Train the inverse dynamics model I.
6: Feed $\tilde{s}$ to the policy network π and get $\hat{a}$.
7: Feed $\tilde{s}$ and s' to the inverse dynamics model I and get $\hat{a}_{tar}$.
8: Update the policy network π according to,

$$L_\pi = \mathbb{E}_{\tilde{s}\sim\mathcal{D}_{tot}, \hat{a}\sim\pi(\cdot|\tilde{s})}\left[Q(\tilde{s}, \hat{a})\right]$$

9: Update the Q-networks according to,

$$L_{sr-v} = \mathbb{E}_{\tilde{s},s'\sim\mathcal{D}_{tot}}\left(\mathbb{E}_{\hat{a}\sim\pi(\cdot|\tilde{s})}Q(\tilde{s}, \hat{a}) - \mathbb{E}_{\hat{a}_{tar}\sim I^{\pi_\beta}(\cdot|\tilde{s},s')}Q(\tilde{s}, \hat{a}_{tar})\right)$$

$$L_Q = \alpha \cdot L_{sr-v} + \mathbb{E}_{s,a,r,s'\sim\mathcal{D}}\left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$$

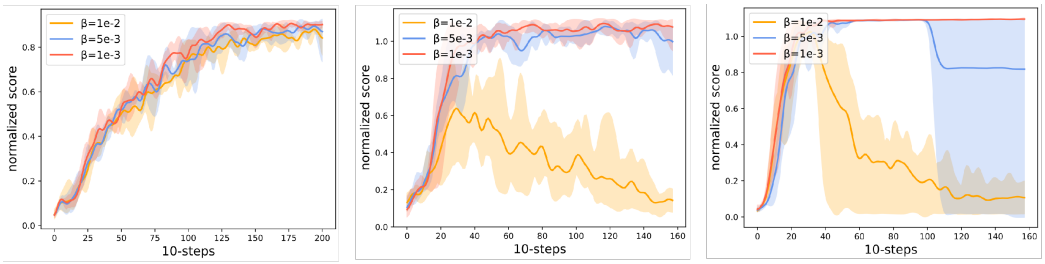
10: **end while**
11: **return** learnt policy network π.

D More experimental results

D.1 Sensitive analysis over hyperparameters

In this section, we perform sensitive analysis on two key hyperparameters: the balance-coefficient λ and the noise magnitude β to evaluate how these hyperparameters influence the performance of OSR.

Sensitive analysis on β. The β is the hyperparameter that control the magnitude of the noise that add to our mixed dataset $\mathcal{D}_{tot}$ for training. Its influence to OSR is as shown in Figure 5, where the curve is smoothed by a Gaussian filter. The performance of the learnt policy is best when the β is 1e-3; takes second place when the β equals to 5e-3; is worst when the β is 1e-2. From the results, we conclude that when β is too large, the agent would fail to be well trained over some or all seeds. Therefore, we ought to conservatively choose the value of β and experience suggests the β should be no bigger than 2e-3 for 'walker2d' benchmark, 5e-3 for 'hopper' and '3e-3' for 'halfcheetah'.

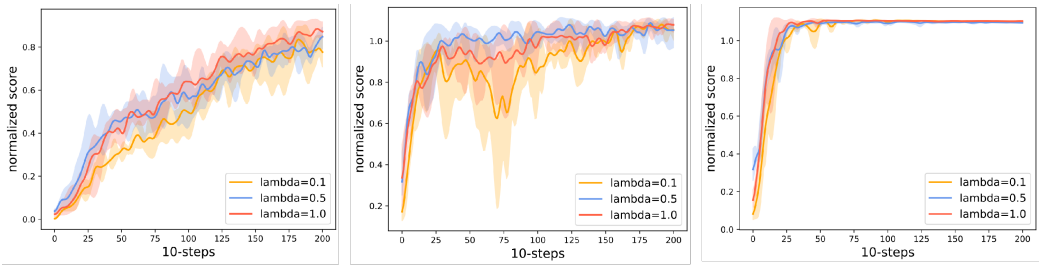


(a) (b) (c)

Figure 5: The sensitive results of β. (a) is on 'halfcheetah-medium-expert', (b) is on 'hopper-medium-expert' and (c) is on 'walker2d-medium-expert'.

Sensitive analysis on λ. The λ is the weight for the OSR regularization that affects the magnitude of conservatism of OSR. The sensitivity of this hyperparameter to the performance of OSR is as

shown in Figure 6, where the curve is smoothed by a Gaussian filter. The learnt policy performs best when λ equals to 1.0; slight poor when λ is 0.5; worst when λ is 0.1. Therefore, we can conclude that if λ is too small, OSR may fail to train the policy, and experience suggests that the λ should range from 0.5 to 1.0.



(a) (b) (c)

Figure 6: The sensitive results of λ. (a) is on 'halfcheetah-medium-expert', (b) is on 'hopper-medium-expert' and (c) is on 'walker2d-medium-expert'.

To further evaluate how the hyperparameters λ and β affect the performance, we have attached more results of sensitive analysis based on the normalized score metrics as follows,

Table 3: Sensitive analysis on 'Halfcheetah-medium-expert'

λ \ β	**1e-5**	**1e-4**	**1e-3**	**1e-2**	**1e-1**
0(CQL)	62.4	62.4	62.4	62.4	62.4
0.01	58.7	59.3	64.6	62.1	46.5
0.1	54.2	76.4	83.4	63.7	50.3
0.5	75.4	87.9	**94.6**	82.3	33.6
1.0	73.2	89.2	92.4	46.7	34.4
10.0	64.7	67.9	76.5	43.9	32.7

Table 4: Sensitive analysis on 'Hopper-medium-expert'

λ \ β	**1e-5**	**1e-4**	**1e-3**	**1e-2**	**1e-1**
0(CQL)	111.0	111.0	111.0	111.0	111.0
0.01	109.3	110.2	111.6	46.3	20.6
0.1	111.4	112.1	111.3	29.1	18.7
0.5	111.5	112.3	**112.9**	17.1	20.4
1.0	112.1	111.7	**113.0**	17.4	14.5
10.0	98.3	70.8	69.6	22.6	13.3

Table 5: Sensitive analysis on 'Walker2d-medium-expert'

λ \ β	**1e-5**	**1e-4**	**1e-3**	**1e-2**	**1e-1**
0(CQL)	98.7	98.7	98.7	98.7	98.7
0.01	101.1	102.5	104.6	94.1	33.3
0.1	103.2	108.9	112.4	97.6	16.6
0.5	109.4	112.6	**114.1**	83.3	16.8
1.0	108.2	110.1	**113.8**	84.7	20.1
10.0	101.7	100.8	89.1	72.8	17.1

From the results, we observe that we should be cautious in choosing a β that is not very large; otherwise, it could lead to failure. We remark that it is better to choose the appropriate hyperparameters in the neighborhood of the bold data listed in the table above.

D.2 Ablation study

D.2.1 Ablation study on the CQL term

We have conducted a more comprehensive ablation study on three environments, shown in the Table below, where each row gives the normalized scores of a specific compared ablated method.

Table 6: Ablation study on the CQL term

	Halfcheetah-m.-e.	**Hopper-m.-e.**	**Walker2d-m.-e.**
QL	9.8	0.3	0.2
QL+BC	41.2	44.7	73.6
QL+IDM	47.5	53.9	80.2
CQL	62.4	98.7	111.0
CQL+BC	85.7	111.8	104.3
CQL+IDM(OSR)	**94.7**	**114.3**	**113.1**

Based on the above results, we have the following observations:

1.the IDM model is useful - we can see that the performance ranking is: OSR > CQL+BC > CQL , where BC denotes traditional behavior cloning (using behavior policy). The IDM can be thought of as a behavior cloning but taking the consequence of an action into consideration when cloning that action.

2.regulating the Q function search space is important for the generalization capability of an offline RL agent - note that if we replace CQL with standard QL in OSR, we have the following performance ranking: OSR>QL+IDM>QL+BC, which shows that the CQL-stlye value function learning is important both for IDM and BC. One possible reason for this is that the agent is trained under the actor-critic framework where both value function and policy play a role, and most importantly, in the setting of offline RL, conservative learning like CQL is critical for suppressing extrapolation error and overestimation, as pointed out by the reviewer.

3.IDM and CQL are complementary to each other - the IDM provides a way to guide the agent to navigate to safer regions, which allows the agent to learn more smart behavior when encountering unfamiliar states - in the sense that the desired behaviors of an agent should be rational (i.e., being less likely to be punished by the objective function of CQL) not only under normal in-distribution states but under difficult unseen situations as well (this latter point is less studied in current literature) . The IDM can also be thought of as a mechanism to control the training procedure of CQL, such that it will behave less 'overly-conservatively' during learning, potentially improving its generalization capability.

D.2.2 Ablation study on the noise injection

We have conducted a more comprehensive ablation study, and the results are shown in the Table below,

Table 7: Ablation study on the noise injection

	Halfcheetah-m.-e.	**Hopper-m.-e.**	**Walker2d-m.-e.**
CQL	62.4	98.7	111.0
CQL+BC	85.7	111.8	104.3
CQL+BC(s.)	91.8	111.2	109.2
CQL+BC(s.a.)	88.3	111.4	108.9
OSR(β=0)	92.3	111.8	110.1
OSR	**94.7**	**114.3**	**113.1**

First, we run our OSR algorithm with the noisy level being zero and compare it with CQL - an OSR baseline without the inverse dynamic model (IDM). The results indicate that our OSR (β=0) (the 5th row) outperforms CQL (the 1st row) significantly on the 3 tasks. This demonstrates the significance of the inverse dynamics model (IDM) in helping the agent to navigate to more safe regions. Similar performance improvement can be observed if we replace the IDM model with the behavior policy (see the CQL + BC setting, the second row), but its effect is not so significant as our OSR method.

Then we introduce the state noise into our base model OSR (β=0). The results show that this further improves the performance of the OSR by 2-3 (last row), and adding noise improves the performance of CQL + BC (the 3rd and the 4th row) as well except on the Hopper task, while the proposed OSR(>0) method consistently improves the performance over all the three environments.

D.3 Training details

In this section, we introduce our training details, including: 1) the hyperparameters and how they were choosen; 2) the structure of the neural networks we use: the Q-networks, inverse dynamics model network and policy network; 3) the total amount of compute and the type of resources used.

D.3.1 Hyperparameters

In Table 8 and 9, we give the hyperparameters used by OSR and OSR-v to generate Table 1 results. In fact, we have already evaluated the influence of the hyperparameters noise magnitude β and SDC weight λ in Appendix D.1.

Table 8: The hyperparameters used by OSR to generate Table 1 results.

Task name	noise magnitude β	OSR weight λ	CQL weight α
Halfcheetah-random	1e-3	0.5	10.0
Walker2d-random	1e-3	0.5	5.0
Hopper-random	1e-3	0.5	5.0
Halfcheetah-medium	3e-3	0.5	10.0
Walker2d-medium	2e-3	0.5	5.0
Hopper-medium	5e-3	0.5	5.0
Halfcheetah-medium-r.	3e-3	0.5	10.0
Walker2d-medium-r.	2e-3	0.5	5.0
Hopper-medium-r.	5e-3	0.5	5.0
Halfcheetah-medium-e.	3e-3	0.5	10.0
Walker2d-medium-e.	2e-3	0.5	5.0
Hopper-medium-e.	5e-3	0.5	5.0
Halfcheetah-expert	3e-3	0.5	10.0
Walker2d-expert	2e-3	0.5	5.0
Hopper-expert	5e-3	0.5	5.0
AntMaze-umaze	1e-3	0.5	5.0
AntMaze-medium-p.	1e-3	0.5	5.0
AntMaze-large-p.	1e-3	0.5	5.0
AntMaze-umaze-d.	1e-3	0.5	5.0
AntMaze-medium-d.	1e-3	0.5	5.0
AntMaze-large-d.	1e-3	0.5	5.0

Table 9: The hyperparameters used by OSR-v to generate Table 1 results.

Task name	noise magnitude β	CQL-OSR weight α
Halfcheetah-random	1e-3	10.0
Walker2d-random	1e-3	5.0
Hopper-random	1e-3	5.0
Halfcheetah-medium	3e-3	10.0
Walker2d-medium	2e-3	5.0
Hopper-medium	5e-3	5.0
Halfcheetah-medium-r.	3e-3	10.0
Walker2d-medium-r.	2e-3	5.0
Hopper-medium-r.	5e-3	5.0
Halfcheetah-medium-e.	3e-3	10.0
Walker2d-medium-e.	2e-3	5.0
Hopper-medium-e.	5e-3	5.0
Halfcheetah-expert	3e-3	10.0
Walker2d-expert	2e-3	5.0
Hopper-expert	5e-3	5.0
AntMaze-umaze	1e-3	5.0
AntMaze-medium-p.	1e-3	5.0
AntMaze-large-p.	1e-3	5.0
AntMaze-umaze-d.	1e-3	5.0
AntMaze-medium-d.	1e-3	5.0
AntMaze-large-d.	1e-3	5.0

D.3.2 Neural network structures

In this section, we introduce the structure of the networks we use in this paper: policy network, Q network and the inverse dynamics model network.

The structure of the policy network and Q networks is as shown in Table 10, where 's_dim' is the dimension of states and 'a_dim' is the dimension of actions. 'h_dim' is the dimension of the hidden layers, which is usually 256 in our experiments. The policy network is a Guassian policy and the Q networks includes two Q function networks and two target Q function networks.

Table 10: The structure of the policy net and the Q networks.

policy net	Q net
Linear(s_dim, 256)	Linear(s_dim, h_dim)
Relu()	Relu()
Linear(h_dim, h_dim)	Linear(h_dim, h_dim)
Relu()	Relu()
Linear(h_dim, a_dim)	Linear(h_dim, 1)

The structure of the inverse dynamics network is as shown in Table 11, which is a conditional variational auto-encoder. 's_dim' is the dimension of states, 'a_dim' is the dimension of actions and 'h_dim' is the dimension of the hidden variables. 'z_dim' is the dimension of the Gaussian hidden variables in conditional variational auto-encoder.

D.3.3 Compute resources

We conducted our experiments using a server equipped with one Intel Xeon Gold 6226R CPU, with 16 cores and 32 threads, and 128GB of DDR4 memory. We used a NVIDIA RTX3090 GPU with 24GB of memory for our deep learning experiments. All computations were performed using Python 3.7 and the PyTorch 1.8.1 deep learning framework.

Table 11: The structure of the inverse dynamics model network.

inverse dynamics model net
Linear(2 * s_dim, h_dim)
Linear(h_dim, h_dim)
Linear(h_dim, h_dim)
Linear(h_dim, z_dim) | Linear(h_dim, z_dim)
Linear(2 * s_dim + z_dim, h_dim)
Linear(h_dim, h_dim)
Linear(h_dim, a_dim)

D.4 Comparison study of methods with Q-ensemble trick

Besides, we also implement OSR with Q-ensemble trick [4], termed as OSR-10, like RORL [25], SAC-10 and EDAC-10 [4], with 10 Q-functions. The results are shown in Figure 12 in Appendix D.4. We observe that the Halfcheetah task improves obviously in SOTA results when using OSR-10, potentially due to its heightened sensitivity to out-of-sample situations.

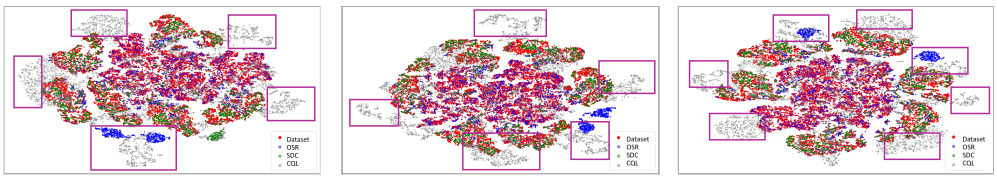
Table 12: Results of **OSR-10(ours)** and SAC-10, EDAC-10, RORL on 9 MuJoCo benchmarks. Part of the results are reported in the RORL paper. The top-2 highest scores in each benchmark are bolded.

Methos	Halfcheetah			Walker2d			Hopper		
	medium	medium-r.	medium-e.	medium	medium-r.	medium-e.	medium	medium-r.	medium-e.
RORL	**66.8±0.7**	61.9±1.5	**107.8±1.1**	**102.4±1.4**	90.4±0.5	**121.2±1.5**	**104.8±0.1**	102.8±0.5	**112.7±0.2**
SAC-10	64.9±1.3	**63.2±0.6**	107.1±2.0	46.7±45.3	89.6±3.1	116.7±1.9	0.8±0.2	**102.9±0.9**	6.1±7.7
EDAC-10	64.1±1.1	60.1±0.3	107.2±1.0	87.6±11.0	**94.0±1.2**	115.4±0.5	103.6±0.2	102.8±0.3	58.1±22.3
OSR-10	**67.1±0.9**	**64.7±1.7**	**108.7±0.9**	**102.0±0.8**	**93.8±1.3**	**123.4±2.2**	**105.5±0.6**	**103.1±1.2**	**113.2±2.7**

D.5 More experimental details on OOSMuJoCo

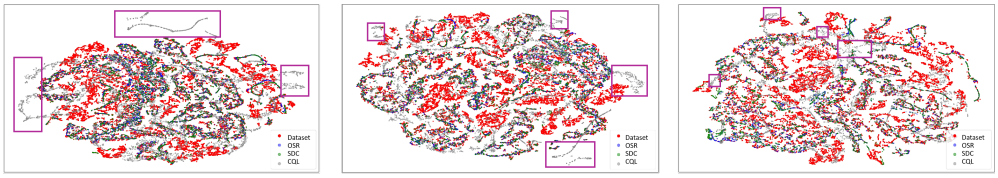
D.5.1 State distributional visualized results

To assess the feasibility of our proposed approach, we used t-Distributed Stochastic Neighbour Embedding (t-SNE) [10] to visualize the state distributions of several benchmarks: Halfcheetah-OOS, Hopper-OOS and Walker2d-OOS, as shown in Figure 8, 9 and 7. The results indicate that both SDC and our proposed method, OSR, generate significantly fewer out-of-distribution (OOD) samples, that is, the states generated by SDC and OSR lay in the support of the dataset. In contrast, traditional CQL tends to deviate from the dataset when making decisions at out-of-sample states.



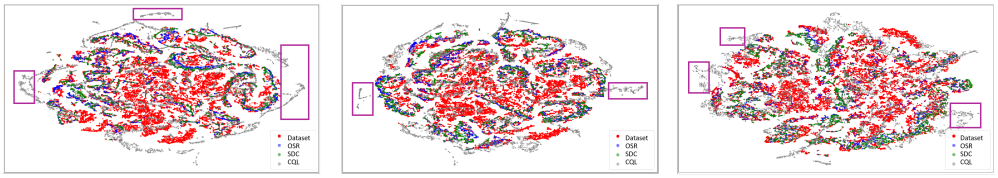
OOS-slight OOS-moderate OOS-large

Figure 7: The visualizations of the state distributions of experiments on the 'Halfcheetah-OOS' with three magnitudes. The purple boxes indicate some typical out-of-sample trajectories.



OOS-slight OOS-moderate OOS-large

Figure 8: The visualizations of the state distributions of experiments on the 'Hopper-OOS' with three magnitudes. The purple boxes indicate some typical out-of-sample trajectories.



OOS-slight OOS-moderate OOS-large

Figure 9: The visualizations of the state distributions of experiments on the 'Walker2d-OOS' with three magnitudes. The purple boxes indicate some typical out-of-sample trajectories.

D.5.2 Other state recovery processes

We also visualize the Hopper and Walker2d agents respectively within an expert trajectory produced by the behavior policy π_β on the offline dataset $\mathcal{D}$, as is shown in Figure 10 and 11(every 10 steps apart), then we visualize the trajectories by the inverse dynamics model(IDM) guided by the expert trajectory and the new policy of OSR. We add external force to knock the agent into an out-of-sample state and we observe that the agents with IDM and the new policy are all able to recover from the out-of-sample states.

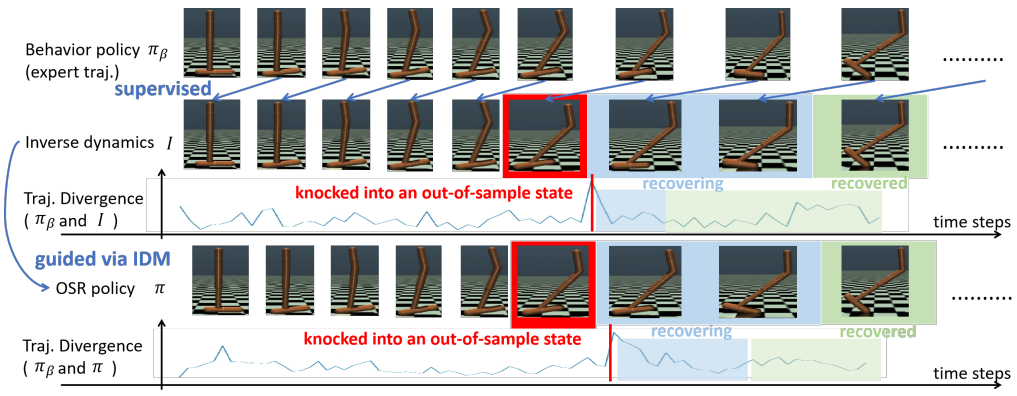


Figure 10: The state recovery process via IDM and OSR on the 'Hopper-OOS-large' benchmark. The interval between every two images is ten steps.

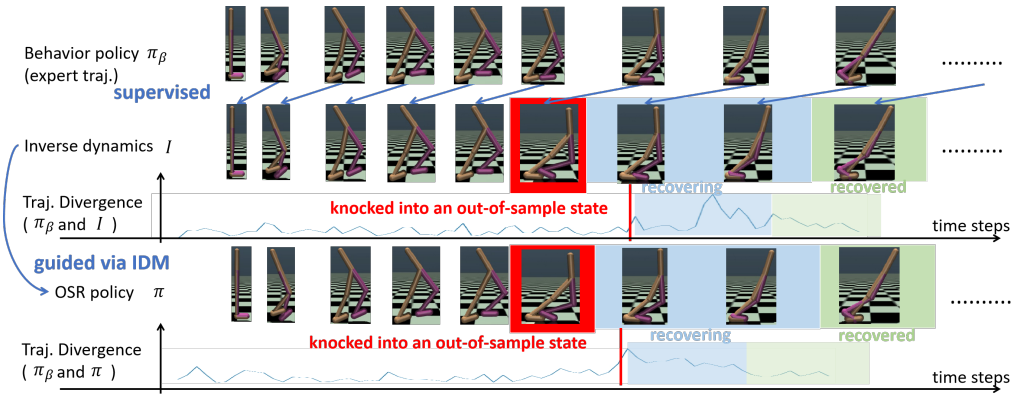


Figure 11: The state recovery process via IDM and OSR on the 'Walker2d-OOS-large' benchmark. The interval between every two images is ten steps.

In order to observe the process of state recovery more clearly, we visualized the motion of each step of the three agents during the state recovery period via the learnt policy of OSR, as is shown in Figure 12. We remark that OSR is able to help the agents to recover from the out-of-sample states, which is quite important for real-world applications.

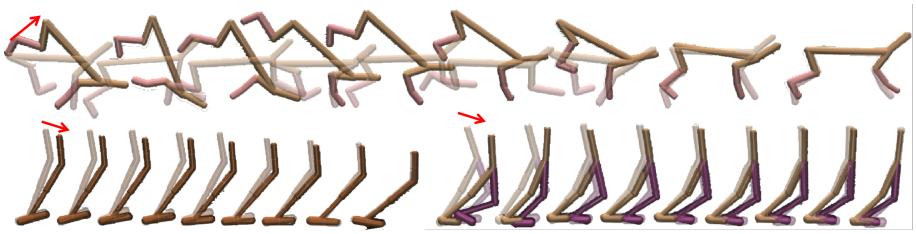


Figure 12: The state recovery processes of three agents: Halfcheetah, Hopper and Walker2d. The red arrows represent the external forces that knock the agents falling into out-of-sample states.

D.5.3 Comparison with other robust offline RL framework

We have compared our method with RORL [25] on the out-of-sample MuJoCo benchmark, and the results are given below; for comparison, we also give the results obtained without using the out-of-samples on these benchmarks. From the table, we observe that RORL generalizes well at out-of-sample states with slight noise, but its performance may suffer a lot under large perturbation. For example, while its normalized score on Halfcheetah-OOS-s. is 103.8, its performance drops significantly to 55.6 on the more challenging Halfcheetah-OOS-l. benchmark. On the contrary, our method performs much more stable across all the environments. In the attached PDF file, we give some visualization of the trajectories generated by both methods in some OOS benchmarks.

Table 13: Comparison with RORL on the out-of-sample MuJoCo benchmark

	RORL-OOS	RORL-w/o OOS	OSR-OOS	OSR-w/o OOS
Halfcheetah-OOS-s.	103.8	107.8	94.1	94.7
Halfcheetah-OOS-m.	79.8	107.8	92.7	94.7
Halfcheetah-OOS-l.	55.6	107.8	91.7	94.7
Hopper-OOS-s.	111.5	121.2	113.3	114.3
Hopper-OOS-m.	89.5	121.2	113.2	114.3
Hopper-OOS-l.	66.5	121.2	110.1	114.3
Walker2d-OOD-s.	117.8	112.7	111.4	113.1
Walker2d-OOS-m.	82.2	112.7	109.2	113.1
Walker2d-OOS-l.	46.5	112.7	106.1	113.1

D.6 Comparison with offline RL works with reverse model

we have compared our method with ROMI [23] - an offline RL method based on RDM, and the results are given below. This shows that although, on average, both methods perform comparably across the environments (average score: 68.9 (OSR) vs. 68.2 (ROMI)), our method performs much better in most 'medium' and 'medium-expert' benchmarks (e.g., Halfcheetah-m.-e., Hopper-m., Hopper-m.e., Walker2d-m.e.) than ROMI, indicating the advantage of using IDM if the underlying dataset covering a portion of high-value areas. However, our method may suffer from very noisy datasets (e.g., Hopper-r.), in which case RDM works better. This suggests that it could be beneficial to combine the advantages of both models for an even more robust offline RL.

Table 14: Comparison with ROMI on the standard MuJoCo benchmark

	ROMI	OSR(ours)
Halfcheetah-m.	**49.1**	48.8
Halfcheetah-m.-r.	**47.0**	46.8
Halfcheetah-m.-e.	86.8	**94.7**
Halfcheetah-r.	24.5	**35.2**
Hopper-m.	72.3	**83.1**
Hopper-m.-r.	**98.1**	96.7
Hopper-m.-e.	111.4	**113.1**
Hopper-r.	**30.2**	10.3
Walker2d-m.	84.3	**85.7**
Walker2d-m.-r.	**109.7**	87.9
Walker2d-m.-e.	109.7	**114.3**
Walker2d-r.	7.5	**13.5**

D.7 Comparison on the MuJoCo benchmark with adversarial attacks

We have trained our method OSR-10 based on the large perturbation scales in the following table (same as Table 5 in [25]), referred as OSR-10-large-noise (OSR-10-l.), on the three 'medium' MuJoCo datasets, while OSR-10 and RORL utilize the perturbation with smaller scales metioned before.

Table 15: Hyperparameters of OSR-10-large-noise (OSR-10-l.)

	Halfcheetah-medium	**Hopper-medium**	**Walker2d-medium**
Scalar ϵ_{OOD} of OOD Loss	0.00	0.02	0.03
Scalar ϵ_{OSR} of OSR Loss	0.05	0.005	0.07

We run OSR-10-l. and RORL-l. (RORL trained on the perturbation scales in the Table 5 of [25] on 'medium' as well) in both the adversarial environments introduced in [25] and the OOS MuJoCo benchmarks proposed in our paper. The results of both methods in the adversarial environments are listed in the table below, where we set the scale of perturbation of the adversarial environments as 0.15, and we also provide the two methods' results in the clean environments for comparison (3rd and 6th rows).

The results show that the peformance of RORL and OSR-10 improves by 13.1% and 12.1% respectively afer adding larger scale of perturbation in training. This indicates that increasing the difficulty of training data to some degree is indeed helpful in enhancing the generalization capbility of the learned agent, hence improving its robustness against adversarial attacks. It's our future work to further investigate the performance boundary of applying this trick.

Table 16: Comparison on the MuJoCo benchmark with adversarial attacks with RORL

	RORL-OOS	RORL-l.-OOS	RORL-l.-w/o OOS	OSR-10-OOS	OSR-10-l.-OOS	OSR-10-l.-w/o OOS
Halfcheetah+random	53.4	58.4	61.2	55.2	59.3	63.3
Halfcheetah+action diff	51.7	52.1	61.2	53.3	54.6	63.3
Halfcheetah+min Q	43.9	46.4	61.2	50.6	55.0	63.3
Hopper+random	67.8	82.8	102.8	70.6	87.9	103.5
Hopper+action diff	62.3	77.3	102.8	63.8	76.2	103.5
Hopper+min Q	41.3	46.2	102.8	44.9	49.4	103.5
Walker2d+random	76.7	93.1	97.4	79.4	91.7	100.7
Walker2d+action diff	76.4	89.2	97.4	79.1	92.8	100.7
Walker2d+min Q	68.6	72.5	97.4	78.9	81.6	100.7

D.8 Experiments on out-of-distribution MuJoCo benchmark

To assess the generalizability of our method, we need to first construct a dataset with more realistic out-of-sample states to test the learnt model. For this we employ a modified generative adversarial network (GAN), which is optimized as follows,

$$\min_{G}\max_{D}[E_{s\sim P(s)}[\log D(s)] + E_{P_G(s')}[\log(1-D(s'))]] + \alpha \cdot E_{P_{G(s)}} H[\pi_\beta(\cdot|s)] \tag{32}$$

where G is generator, D is the discriminator, $P(s)$ is the real-word state distribution (the dataset) and $P_G(s)$ is the state distribution generated via G. H is the entropy function. In words, the above objective aims to generate real-world samples (i..e., consistent with the in-distribution data) that confuse the behavior policy π_β most; hence its output could be considered as kind of realistic out-of-sample states.

In evaluation, we initialize the MuJoCo environments with the generated out-of-sample states and assess the generalizability of the learnt agent on them. The results, based on the average of normalized scores and recovery rate, are as follows:

Table 17: Normalized scores/recovery rate on the generated out-of-distribution MuJoCo states

	Halfcheetah	**Hopper**	**Walker2d**
CQL	20.4/33.8%	36.5/39.1%	13.1/12.6%
OSR	**40.1/69.9%**	**72.0/88.8%**	**43.3/32.4%**

The above results indicate that our OSR effectively guides the agent to recover from most real-world out-of-sample situations (nearly 70%) in the Halfcheetah and Hopper tasks, despite the mismatch between the constructed (Gaussian) noisy dataset and the actual (GAN-based) out-of-sample states. Although the Walker2D task seems challenging to it, our OSR method performs significantly better than CQL, where a risk-guiding mechanism like ours is lacking.

To gain further insights, we provide visualizations of typical out-of-distribution (OOD) states in the three tasks, along with the corresponding trajectories of OSR, as is shown in Figure 13.

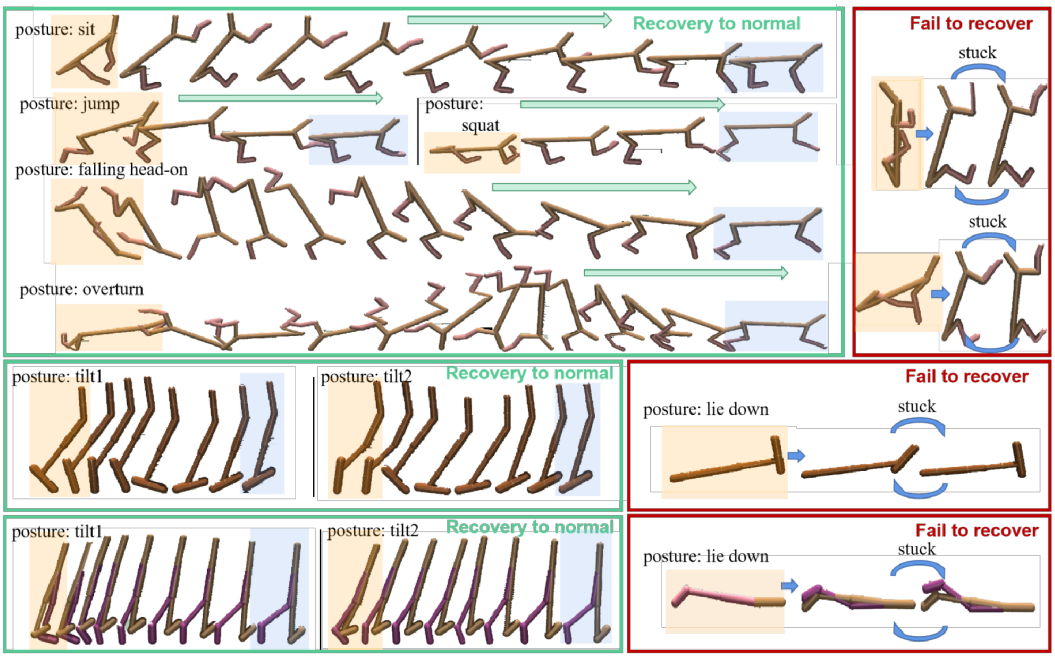


Figure 13: The visualization of OSR's state recovery process on the real-world out-of-sample (OOS) Halfcheetah, Hopper and Walker2d benchmarks, where the initial states are generated by the modified generative adversarial network. The states in orange boxes represent the generated OOS states, and those in blue boxes are the recovered normal states. The trajectories in the left green box are samples successfully recovered while in the right red box are samples failed to recover.

D.9 Code

We build our OSR and OSR-v based on the CQL project from github[3]. The reasons why we choose YOUNG-GENG's CQL project instead of the official version[4] are as follows: 1) The official version CQL code perform incorrect gradient calculation with old version of pytorch(see issue #5); 2)In the project by yong-geng, a Gaussian policy is utilized, which meets the needs of our method.

We build our OSR-10 based on the RORL project from github[5].

E Discussion

The assumptions. The theoretical framework presented in this paper is based on two key assumptions. Assumption 1 assumes that the transition of the behavior policy is 'smooth-transitioned', meaning that it is insensitive to disturbances in observations. However, in practical applications, the behavior policy may not have such a property, which could limit the generalizability of the proposed method to some datasets. Therefore, it is important to carefully consider the effects of non-smooth transitions when applying the proposed method in practice. Assumption 2, on the other hand, assumes that the new policy is modeled as a Gaussian distribution. While this assumption is reasonable in many settings, it may not hold if the new policy is modeled in a different way, such as a deterministic policy, then the validity of Assumption 2 should be carefully evaluated based on the specific modeling assumptions used in each application. Therefore, it is important to note that the limitations of the proposed method are closely tied to the assumptions made in the theoretical framework.

[3]Project of CQL by YOUNG-GENG: https://github.com/young-geng/CQL
[4]Project of CQL by AVIRALKUMAR2907: https://github.com/aviralkumar2907/CQL
[5]Project of CQL by YangRui2015: https://github.com/YangRui2015/RORL

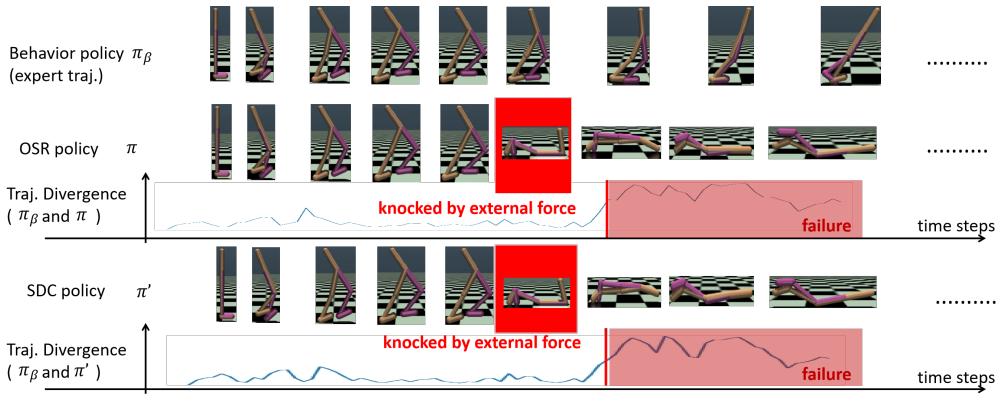


Figure 14: The state recovery process via IDM and OSR on the Walker2d-OOS benchmark with vary large knock(100 steps of random actions). The interval between every two images is ten steps.

Not all kind of out-of-sample situations could be recovered. In a related work by [27], the authors identify three key factors that can lead to *state distributional shift* problem in offline reinforcement learning: initial state difference, dynamics bias, and approximation error. Initial state difference refers to the difference between the initial state distributions of the offline dataset and the actual environment, and dynamics bias arises due to the discrepancy between the empirical distribution of the dynamics of the offline dataset and the true dynamics. Approximation error of the neural network can also contribute to *state distributional shift*. However, it is important to note that in scenarios where the initial state difference and dynamics bias drive the agent to a state that is quite far from the dataset, neither SDC nor our proposed method, OSR, can enable the agent to recover to the dataset, for the lack of the information for the inference of such a recovery, as is shown in Figure 14. Then additional strategies may be required to address these challenges. Nonetheless, both SDC and OSR can effectively address the problems of approximation error and some perturbed environments, where the out-of-sample states are located within the β neighborhood of the dataset.