A Policy-Guided Imitation Approach for Offline Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Offline reinforcement learning (RL) methods can generally be categorized into two types: RL-based and Imitation-based. RL-based methods could in principle enjoy out-of-distribution generalization but suffer from erroneous off-policy evaluation. Imitation-based methods avoid off-policy evaluation but are too conservative to surpass the dataset. In this study, we propose an alternative approach, inheriting the training stability of imitation-style methods while still allowing logical out-of-distribution generalization. We decompose the conventional reward-maximizing policy in offline RL into a guide-policy and an execute-policy. During training, the guide-poicy and execute-policy are learned using only data from the dataset, in a supervised and decoupled manner. During evaluation, the guide-policy guides the execute-policy by telling where it should go so that the reward can be maximized, serving as the *Prophet*. By doing so, our algorithm allows *state-compositionality* from the dataset, rather than *action-compositionality* conducted in prior imitation-style methods. We dumb this new approach Policy-guided Offline RL (`POR`). `POR` demonstrates the state-of-the-art performance on D4RL, a standard benchmark for offline RL. We also highlight the benefits of `POR` in terms of improving with supplementary suboptimal data and easily adapting to new tasks by only changing the guide-poicy.

1 Introduction

Offline RL, also known as batch RL, allows learning policies from previously collected data, without online interactions [26, 28]. It is a promising area for bringing RL into real-world domains, such as robotics [18], healthcare [40] and industrial control [51]. In such scenarios, arbitrary exploration with untrained policies is costly or dangerous, but sufficient prior data is available. While most off-policy RL algorithms are applicable in the offline setting by filling the replay buffer with offline data, improving the policy beyond the level of the behavior policy requires querying the value function (i.e., Q function) about values of actions that were not seen in the dataset. Those out-of-distribution actions can be deemed as adversarial examples of the Q function [25], which cause extrapolation error of the Q-function. The error will be accumulated by the deadly triad issue [42], propagate across the state-action space through the iterative dynamic programming procedure [3], and can not be eliminated without requiring growing batch of online samples.

To allieviate this issue, prior model-free offline RL methods typically add a behavior regularization term to the policy improvement step, to limit how far it deviates from the behavior policy. This can be achieved explicitly by calculating some divergence metrics [23, 46, 33, 10], or implicitly by regularizing the learned value functions to assign low values to out-of-distribution actions [25, 21, 1, 2]. Nevertheless, this imposes a trade-off between accurate value estimation (more behavior regularization) and maximum policy improvement (less behavior regularization). To avoid such

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

problem, recently a branch of methods bypass querying the values of unseen actions by performing some kind of imitation learning on the dataset. This can be achieved by filtering trajectories based on their return [35, 7], or reweighting transitions based on how advantageous they could be under the behavior policy [3, 22], or just be conditioned on some variables without any dataset reweighting [6, 8]. Although imitation-style methods enjoy stable training process and are able to effectively perform multi-step dynamic programming by assigning the proper reweighting weight or conditioned variable [22, 8], they only allow *action-compositionality* from the dataset, lose the ability to surpass the dataset by out-of-distribution generalization, which only appears in RL-based methods.

In this work, we propose an alternative approach. We aim at inheriting the training stability of imitation-style methods while still allowing logical out-of-distribution generalization. To do so, we propose Policy-guided Offline RL (POR), an algorithm that employs two policies to solve tasks: a guide-policy and an execute-policy. The job of the guide-policy is to learn the optimal next state given the current state, and the job of the execute-policy is to learn how different actions can produce different next states, given the current state. By this manner, we decompose the original task of RL (i.e., reward-maximizing) into two distinct yet complementary tasks, and this decomposition makes each task much easier to be solved. During evaluation, the guide-policy serves as a guide for the execute-policy by telling it where to go so that the reward can be maximized. By doing so, our algorithm allows *state-compositionality* from the dataset, rather than *action-compositionality* conducted in prior work, which owns logical out-of-distribution generalization.

Our method is easy to implement by only adding the learning of a guide-policy, the training stage of the guide-policy and execute-policy are decoupled, using in-sample learning from the dataset. We test POR in widely-used D4RL offline RL benchmarks and demonstrates the state-of-the-art performance, especially on low-quality datasets that require out-of-distribution generalization to achieve a high score. We also show that by decoupling the learning of guide-policy and execute-policy, we can enhance the guide-policy with supplementary suboptimal data, or re-learn the guide-policy to adapt to new tasks with different reward functions, without changing the execute-policy.

2 Related Work

RL-based offline approach A large portion of offline RL methods are RL-based, RL-based methods typically augment existing off-policy methods (e.g., Q-learning or actor-critic) with a behavior regularization term. The primary ingredient of this class of methods is to propose various policy regularizers to ensure that the learned policy does not stray too far from the behavior policy. These regularizers can appear explicitly as divergence penalties [46, 23, 10], implicitly through weighted behavior cloning [44, 35, 33], or more directly through careful parameterization of the policy [11, 53]. Another way to apply behavior regularizers is via modification of the critic learning objective to incorporate some form of regularization to encourage staying near the behavioral distribution and being pessimistic about unknown state-action pairs [32, 25, 21]. There are also several works incorporating behavior regularization through the use of uncertainty. The uncertainty quantification can be done via Monte Carlo dropout [47] or explicit ensemble models [2, 50, 19]. Note that the behavior regularization weight is crucial in RL-based methods. With a small weight, RL-based methods could in principle enjoy out-of-distribution generalization since they perform true dynamic programming. However, a small weight will make erroneous off-policy evaluation due to distribution shift and the policy is extremely vulnerable to the value function misestimation.

Imitation-based offline approach Another line of methods, on the contrary, performing some kind of imitation learning on the dataset, without the need to do off-policy evaluation. When the dataset is good enough or contains high-performing trajectories, we can simply clone the actions observed in the dataset [36], or perform some kind of filtering or conditioning to extract useful transitions. For instance, recent work filters trajectories based on their return [7, 35], or directly filters individual transitions based on how advantageous these could be under the behavior policy and then clones them [3, 22, 4, 13]. Conditioned BC methods are based on the idea that every transition or trajectory is optimal when conditioned on the right variable [12, 8]. By this way, after conditioning, the data becomes optimal given the value of the conditioned variable. The conditioned variable could be the cumulative return [6, 16, 24], or the goal information if provided [30, 12, 8, 48]. A recent study [8] shows that goal-conditioning can be super useful in D4RL AntMaze datasets. However, this introduces the assumption that we have some prior information about the structure of the task, which is beyond the standard offline RL setup.

Our method can be viewed as a combination of RL-based and Imitation-based methods. We adopt an imitation-style manner to train the goal-policy and execute-policy. When evaluation, however, following the guide-policy, the execute-policy could produce out-of-distribution actions by leveraging the generalization capacity of the function approximator.

3 Preliminaries

Offline RL We consider the standard fully observed Markov Decision Process (MDP) setting [39]. An MDP can be represented as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, \rho, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T(\cdot|s,a)$ is the transition probability distribution function, $r(s,a)$ is the reward function, ρ is the initial state distribution and γ is the discount factor, we assume $\gamma \in (0,1)$ in this work. The goal of RL is to find a policy $\pi(a|s) : \mathcal{S} \times \mathcal{A} \to [0,1]$ that maximizes the expected cumulative discounted reward (or called return) along a trajectory as

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r\left(s_t, a_t\right) \middle| s_0 = s, a_0 = a, s_t \sim T\left(\cdot|s_{t-1}, a_{t-1}\right), a_t \sim \pi\left(\cdot|s_t\right) \text{ for } t \geq 1\right]. \quad (1)$$

In this work, we focus on the offline setting. The goal is to learn a policy from a fixed dataset $\mathcal{B} = \left\{\tau^i = \left(\left(s_0^i, a_0^i, s_0'^i, r_0^i\right), \left(s_1^i, a_1^i, s_1'^i, r_1^i\right), \cdots, \left(s_H^i, a_H^i, s_H'^i, r_H^i\right)\right)\right\}_{i=1}^{N}$ consisting of trajectories that are collected by different policies, where H is the time horizon. The dataset can be heterogenous and suboptimal, we denote the underlying behavior policy of $\mathcal{B}$ as μ, which represents the conditional distribution $p(a|s)$ observed in the dataset.

RL via Supervised Learning Conventional RL methods generally either compute the derivative of (1) with respect to the policy directly via policy gradient methods [37, 17], or estimate a value function or Q-function by means of TD learning [45], or both [38, 14]. In the *RL via Supervised Learning* framework, we avoid the complex and potentially high-variance policy gradient estimators, as well as the complexity of temporal difference learning. Instead, we perform conditioned behavior cloning on some extra information, such as goal state, cumulative return, or language description [30, 12]. When applying the framework to the offline RL setting, we can take the offline dataset $\mathcal{B}$ as input and find the outcome-conditioned policy π that optimizes

$$\max_{\pi} \mathbb{E}_{\tau\sim\mathcal{B}, t\sim\mathrm{Unif}(1,H), \omega\sim g(\cdot|\tau_{t:H})}\left[\log \pi\left(a_t|s_t, \omega\right)\right]. \quad (2)$$

where ω is an outcome conditioned on the remaining trajectory, i.e., $\omega \sim g\left(\cdot|\tau_{t:H}\right)$, and we use $\tau_{i:j} = (s_i, a_i, \ldots, s_j)$ to denote a fragment of the trajectory.

4 Policy-guided Offline Reinforcement Learning

We now continue to describe our proposed approach, POR. To have a clear understanding of the out-of-distribution generalization of POR, we begin with a motivating example which demonstrates that by allowing state-compositionality on the dataset, the agent is able to generalize to a much better policy than doing action-compositionality. Then we describe how we accomplish this idea in the more general case and introduce the algorithm details of POR. Finally we give a theoretical analysis of POR, we are interested to see how close the policy induced by POR and the optimal policy could be.

4.1 A Motivating Example

We first use a simple, didatic toy example to illustrate the importance of state-stitching for offline RL. Let's consider the navigation task shown in Figure 1, where the goal is to find the shortest path from the starting location (bottom-left corner) to the goal location (top-right corner) in the grid world. The state and action space in this task are discrete, the state consists of the (x, y) coordinate and the agent could choose to move to any of its nearby 8 grids (i.e., the action space is 8). The agent receives a reward of 0 for entering the goal state and -1 reward for all other transitions. The offline dataset (green-arrowed lines) is consisted of two suboptimal trajectories from the start location to the goal location, plus transitions of taking random actions at random states.

We use the grid color (more saturated to purple means higher value) to show the value function $V(s)$ computed from the offline dataset. We compare two methods that use the value function differently. At a state $s_b \sim \mathcal{B}$, the first method chooses the action a_b in the dataset that leads to s'_b with the highest $V(s'_b)$; the second method chooses the action a that leads to s' in the dataset with the highest $V(s')$, note that a is not necessarily to be within the dataset. The first method is doing action "stitching" in the dataset while the second method is doing state "stitching". Although both methods are able to "stitch" suboptimal trajectories together from the dataset and successfully travel from the start location to the goal location, the action-stitching method takes more steps (11 steps) compared with the state-stitching method (7 or 8 steps), as shown in Figure 1.

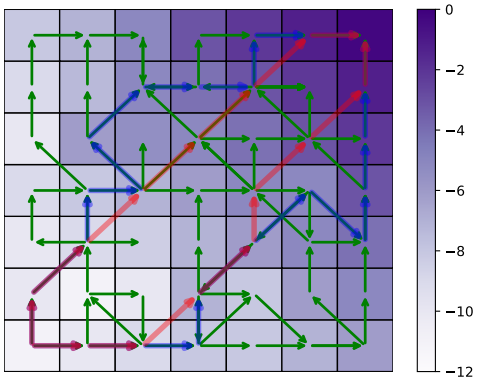


Figure 1: A toy example in the grid world that requires navigation from the start location (bottom-left corner) to the goal location (top-right corner). The offline dataset is colored with green, the trajectories obtained from action-stitching method and state-stitching method are colored with blue and red, respectively. Note that both action-stitching and state-stitching method have two equal-valued action choice at the start location (↑ and →), resulting in two different trajectories in both blue and red lines.

The acction-stitching method introduced above is actually an abstract methodology of Imitation-based methods, which imitate the transitons in the dataset unequally with different weight choices. However, these methods are too conservative to find the optimal trajectory, especially when there does not exist one in the offline dataset. To generate better-than-dataset trajectories (e.g., red-arrowed lines), the agent needs to take logical out-of-distribution actions.

Note that in the toy example, we give the agent the freedom to choose **any** action that leads to the highest-value state in the dataset. This allows the most out-of-distribution generalization and is realizable in some simple tasks (e.g., discrete state and action space) with high action coverage, for instance, in the toy example, given historical (s, a, s') results at $(5, 4)$, the agent can successfully recognize that taking upper-right will arrive at $(6, 5)$. However, in the more general case, especially the continuous state and action space setting, this may make the agent do erronumously-generalized actions as the same state can hardly be observed twice. To perform logical out-of-distribution generalization, we need a *Prophet*, i.e., the guide-policy, to help the agent by telling it which state it should (high reward) and can (logical generalization) go to. It turns out that the involvement of the guide-policy also brings some theoritical meaning, we will discuss that in Section 4.4.

4.2 Learning the Guide-Policy

Recall that the goal of the guide-policy is to guide the execute policy about which state s it should go. To accomplish that, we need to train a value function V. Value function can be learned using only (s, s') samples in the offline dataset, and the training of V doesn't suffer from overestimation because there's no out-of-distribution actions contained. However, this only learn the value of behavior policy μ [3]. To approximate the optimal value function, we adopt tricks from recent work by giving the ℓ_2 loss with different weight using expectile regression [22], yielding the following asymmetric ℓ_2 loss

$$\min_V \ \mathbb{E}_{(s,r,s')\sim\mathcal{B}} \left[L_2^\tau \left(r + \gamma V_{\phi'}(s') - V_\phi(s) \right) \right], \text{ where } L_2^\tau(u) = |\tau - \mathbb{1}(u < 0)| u^2. \tag{3}$$

It can be seen that when $\tau = 1/2$, this operator is reduced to Bellman expectation operator, while when $\tau \to 1$, this operator approaches Bellman optimality operator. Note that our learning objective bears similarity with IQL [22], however, here we only need to learn a value function while IQL needs an additional Q-function to extract the policy using weighted behavior cloning.

Simply maximizing the guide-policy with respect to $V(s)$ will result in a state that the execute-policy may make erronumous generalization. To allieviate this issue, we add a behavior cloning term to the learning objective of the guide-policy. Denote the guide-policy as $g_\omega(s) : \mathcal{S} \to \mathcal{S}$, the learning objective of g is given by

$$\max_g \mathbb{E}_{(s,s')\sim\mathcal{B}} \left[V_\phi(g_\omega(s)) - \alpha(s' - g_\omega(s))^2 \right], \tag{4}$$

where the weight α serves as the trade-off between guiding to space with high-reward and space that the execute-policy ought to have a correct generalization.

4.3 Learning the Execute-Policy: Training and Evaluation

After learning the guide-policy, now we turn to the execute-policy. Since the job of the execute-policy is to having a strong generalization ability, we adopt the *RL via Supervised Learning* framework by conditioning the execute-policy on s' that encounted in the dataset. To be more specific, denote the execute-policy as $\pi_\theta(s, s') : \mathcal{S} \times \mathcal{S} \to [0, 1]$. During training, π performs supervised learning by maximizing the likelihood of the actions given the states and next states, yielding the following objective

$$\max_{\pi} \ \mathbb{E}_{(s,a,s')\in\mathcal{B}} \left[\log \pi_\theta(a|s, s')\right]. \tag{5}$$

During evaluation, given a state s, the final action is determined by both the guide-policy and the execute-policy, by

$$a = \arg\max_{a} \ \pi_\theta(a|s, g_\omega(s)). \tag{6}$$

Algorithm 1 Policy Guided Offline RL

Require: $\mathcal{B}$, τ, α.
// Training
Initialize V_ϕ, $V_{\phi'}$, g_ω, π_θ
for $t = 1, 2, \cdots, N$ **do**
  Sample transitions $(s, r, s') \sim \mathcal{B}$
  Update V_ϕ by Eq.(3)
  Update g_ω by Eq.(4)
  Update $V_{\phi'}$ by $\phi' \leftarrow \lambda\phi + (1-\lambda)\phi'$
end for
for $t = 1, 2, \cdots, M$ **do**
  Sample transitions $(s, a, s') \sim \mathcal{B}$
  Update π_θ by Eq.(5)
end for
// Evaluation
Get initial state s, set d as False
while not d **do**
  Get action a form Eq.(5)
  Roll out a and get (s', r, d)
  Set $s = s'$
end while

Note that our learning objective (5) differs from previous *RL via Supervised Learning* methods in that we only use the next state s' as the conditioned variable, there's no need to estimate the cumulative return [6] and it also works in settings where we don't know the goal information [8]. Also, during evaluation, previous methods show that the choice of conditioned variable is crucial important as little changes will cause significant performance difference [6]. In our method, owing to the existence of the guide-policy, the optimal conditioned variable can be automatically generated.

Our final algorithm, POR, consists of three supervised stages: learning V, learning g and learning π. We summarize the training and evaluation procedure of POR in Algorithm 1. Note that the training of g and π are fully decoupled, this brings POR some nice properties to improve with supplementary data or transfer to new tasks, we will further investigate it in the experiments.

4.4 Analysis

In this section, we give a theoritical analysis of POR. Concretely, we aim to 1) give the lower bound of the performance difference between POR and the optimal policy π^*, 2) analyze how the guide-policy g will influence this bound.

We begin by introducing the notation and assumptions used in our analysis. We denote $P(s, s') : \mathcal{S} \times \mathcal{S} \to \mathcal{A}$ as the *inverse transition operator*. We denote a_g and a as the ground truth of $\pi(s, g(s))$ and $\pi(s, s')$, respectively. We also denote $\epsilon := \sup_{(s,a,s')\in\mathcal{B}} \|\pi(s, s') - a\|$ as the upper bound of the approximation error in the dataset during training. We then introduce the following two assumptions.

Assumption 1 *(Non-lazy MDP) The inverse transition operator $P(s, s')$ is L_1-Lipschitz continuous, i.e., $\|P(s_1, s_1') - P(s_2, s_2')\| \le L_1\|(s_1, s_1') - (s_2, s_2')\|$.*

Assumption 2 *(Lipschitz continuous function approximators) The execute-policy we trained is L_2-Lipschitz continuous, i.e., $\|\pi(s_1, s_1') - \pi(s_2, s_2')\| \le L_2\|(s_1, s_1') - (s_2, s_2')\|$.*

Assumption 1 holds when actions do have effects on state transitions (i.e. if $a \neq 0$, then $\|s' - s\| \ge \epsilon_s$>0). We refer the MDP that satisfies Assumption 1 as *non-lazy MDP* and others as *lazy MDP*. It should be mentioned that most real-world tasks belong to the *non-lazy MDP* case, it is meaningless to study under the *lazy MDP* case becase in *lazy MDP*, action changes will have little effect on state transitions, making the policy learning in vain. Assumption 2 is a mild assumption that is frequently utilized in plenty of works [43, 31, 29].

Then in Theorem 1, we show that under this two assumptions, the gap between the optimal action and the action output by POR can be bounded.

Theorem 1 *(Single step gap to optimal action). The single step gap between optimal action and the action induced by our method can be bounded as*

$$\|\pi(s, g(s)) - a^*\| \leq \underbrace{(L_1 + L_2)\|g(s) - s'\|}_{l_1} + \underbrace{\|a_g - a^*\|}_{l_2} + \underbrace{\epsilon}_{l_3} \tag{7}$$

Theorem 1 states that single step optimal gap is related to three parts: l_1, l_2, l_3. l_1 is the generalization performance factor. It can be found that a small value of $\|g(s) - s'\|$ enables a small l_1, which indicates the necessity to constrain the guide-policy to stay close to the dataset. l_2 denotes the suboptimality constant. Intuitively, a loosened constraint on the guide-policy may induce a small suboptimality constant but on the other hand, may suffer from the risk of a high value of l_1. The guide-policy g serves as a trade-off between l_1 and l_2, we could balance the constraint strength and suboptimality by adjusting α in Eq.(4) to obtain the lowest bound. l_3 is related to the approximation error on the training data, which can be reduced via improving the representation ability of function approximators but may suffer from a high L_2 due to the overfitting caused by over-parameterization. l_1 also depends on the Lipschitz constant L_1 and L_2. Note that a smooth *inverse transition operator* will result in a small value of L_1 and L_2 (because $\pi(s, s')$ is trained to approximate $P(s, s')$), this means our method may achieve better performance under a smoother inverse transition.

Based on Theorem 1, we can also give the performance bound of our method by replace the ϵ in Theorem 3.2 in [12] with the RHS of Eq.(7): $J(\pi^*) - J(\pi) \leq \sup_{s,a,h}(l_1 + l_2 + l_3)T$, where T is the horion of each episode, h is the time step.

5 Experiments

We present empirical evaluations of `POR` in this section. We first evaluate `POR` against other baseline algorithms on D4RL [9] benchmark datasets. We then explore deeper on the guide-policy about the benfits from decoupled training process. We finally establish ablation studies on the execute-policy.

5.1 Comparative Experiments on D4RL Benchmark Datasets

We first evaluate our approach on D4RL MuJoCo and AntMaze datasets [9]. Notice that most of MuJoCo datasets (except `random` and `medium` datastes) are consisted of a large fraction of near-optimal trajectories, while AntMaze datasets contain few or no near-optimal trajectories. Hence those low-quality datasets can serve as a good testbed for quantifying the *compositionality* ability.

We compare `POR` with both RL-based and Imitation-based baselines. For RL-based baselines, we select the best one from 5 state-of-the-art algorithms, including BCQ [11], BEAR [23], BRAC [46], CQL [25] and TD3+BC [10]. For Imitation-based baselines, we further split them into *Weighted BC* methods and *Conditioned BC* methods. Weighted BC methods include Filter Behavior Cloning (Filt. BC) that filters 10% of best trajectories in terms of the total trajectory reward, One-step RL [3] and IQL [21] that use different weight to do behavior cloning. Conditioned BC methods include DT [6] and RvS [8]. DT is built with Transformer and conditioned on the cumulative return. RvS well studied the effect of different conditional information on different tasks. The score of RvS is chosen by the higher score between RvS-R and Rvs-G, which use the reward-to-go and **oracle** goal information as the conditioned variable, respectively. Note that we also categorize our method `POR` into conditioned BC methods because the execute-policy is learned in a similar manner with other conditioned BC methods. Note that the reuslts of Filter BC are from our own implementation, and the results of IQL on MuJoCo random tasks are re-runed using the author-provided implementation. Other results are taken directly from their corresponding papers. To enable fairness and consistency in the evaluation process, we train our algorithm for 1 million time steps and evaluate every 5000 time steps that consisting of 10 episodes. Full experimental details are included in Appendix C.2 and the learning curve can be found in Appendix E.

The results are shown in Table 1, in MuJoCo locomotion tasks, `POR` performs competitively to the best performance of prior methods in high-quality datasets, while achieves much better performance than other methods in low-quality datasets (e.g., `random` and `medium` datasets). On the more challenging AntMaze tasks, `POR` outperforms all other baselines by a large margin. It even surpassed RvS that uses the oracle goal information to learn the policy. We conjecture that the success in all those dataset should be credited to the out-of-distribution generalization ability of `POR`.

Table 1: Averaged normalized scores of POR against other baselines. The scores are taken over the final 10 evaluations with 5 seeds. In the *Best RL Baseline* column, the algorithm with the best performance among 5 RL-based algorithms (BCQ [11], BEAR [23], BRAC [46], CQL [25], TD3+BC [10]) is presented, taken from their original papers. POR achieved highest scores in 12 out of 18 tasks.

D4RL Dataset	Weighted BC			Conditioned BC			Best RL Baseline
	Filt. BC	One-step	IQL	DT	RvS	POR (ours)	
antmaze-u	62.8	64.3	87.5±2.6	59.2	65.4	90.6 ±7.1	89.0[BCQ]
antmaze-u-d	50.2	60.7	66.2±13.8	53.0	60.9	71.3 ±12.1	61.0[BEAR]
antmaze-m-p	5.4	0.3	71.2±7.3	0	58.1	84.6 ±5.6	68.0[CQL]
antmaze-m-d	9.8	0	70.0±10.9	0	67.3	79.2 ±3.1	68.0[CQL]
antmaze-l-p	0	0	39.6±5.8	0	32.4	58.0 ±12.4	18.8[CQL]
antmaze-l-d	0	0	47.5±9.5	0	36.9	73.4 ±8.5	45.6[CQL]
antmaze mean	21.4	20.8	63.6 ±8.3	18.7	53.5	76.2 ±8.1	58.4
halfcheetah-r	5.4	3.7±0.2	11.2±2.9	-	-	29 ±0.7	20.0[CQL]
hopper-r	4.2	5.2±0.2	7.9±0.4	-	-	12±2.1	14.2 [BEAR]
walker2d-r	6.7	5.6±0.6	5.9±0.5	-	-	6.3±0.3	8.3 [CQL]
halfcheetah-m	42.5	48.4±0.1	47.4±0.2	42.6±0.1	41.6	48.8 ±0.5	48.3[TD3+BC]
hopper-m	56.9	59.6±2.5	66.2±5.7	67.6±1.0	60.2	98.2 ±1.6	59.3[TD3+BC]
walker2d-m	75.0	81.8 ±2.2	78.3±8.7	74.0±1.4	71.7	81.1±2.3	83.7[TD3+BC]
halfcheetah-m-r	40.6	38.1±1.3	44.2±1.2	36.6±0.8	38.0	43.5±0.9	45.5 [CQL]
hopper-m-r	75.9	97.5±0.7	94.7±8.6	82.7±7.0	73.5	98.9 ±2.1	95.0[CQL]
walker2d-m-r	62.5	49.5±12.0	73.8+7.1	66.6±3.0	60.6	76.6 ±6.9	81.8[TD3+BC]
halfcheetah-m-e	92.9	93.4±1.6	86.7±5.3	86.8±1.3	92.2	94.7 ±2.2	91.6[CQL]
hopper-m-e	110.9	103.3±1.9	91.5±14.3	107.6 ±1.8	101.7	90.0±12.1	105.4[CQL]
walker2d-m-e	109.0	113.0 ±0.4	109.6±1.0	108.1±0.2	106.0	109.1±0.7	110.1[TD3+BC]
locomotion mean	55.5	58.2 ±2.0	57.6 ±6.6	56 ±1.4	53.7	65.6 ±2.7	63.5

5.2 Validation Experiments on Out-of-Distribution Generalization

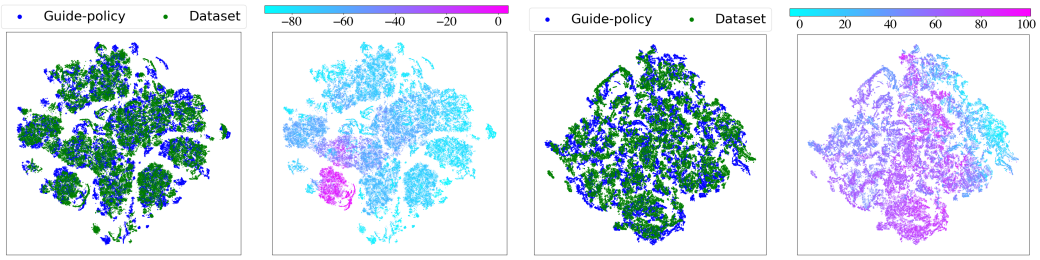
In this section, we try to validate our hypothesis about the out-of-distribution generalization of POR. We are especially interested to see how the guide-policy learned in those high-dimensional tasks in D4RL. To do so, we compare the distribution of $g(s)$ generated by the guide-policy with the distribution of s' from the offline dataset. We choose `antmaze-large-diverse` and `hopper-medium-replay`, two datasets for evaluation. For each dataset, we randomly collect $100,000$ (s, s') transition tuples. To visualize the results clearly, we plot the distribution of $g(s)$ and s' with t-Distributed Stochastic Neighbor Embedding (t-SNE) [15], we also visualize the true values of all states using the optimal value function $V^*(s)$.

The results are shown in Figure 2, it can be found that in both tasks, the distribution of $g(s)$ bears some similarity to the distribution of s'. However, when looking at the non-overlapped area of $g(s)$ and s', we found that $g(s)$ has a much higher value than s', especially in the `antmaze-large-diverse` dataset. Those out-of-distribution and high-value area are exactly where POR generalized to, note that the high-value benefit of the single-step generalization could also be accumulated and propagated through time steps, resulting a much better policy.

5.3 Investigations on the Guide-Policy

In this section, we investigate what is the potential benefit of decoupling the training process of the guide-policy and execute-policy. We found that as the execute-policy is not task-specific and only cares about the effect of actions to different states, we can reuse its powerful generalization ability when encounted with additional suboptimal data or adapting to a new task. We only need to re-learn the guide-policy as it is task-specific.

Improve guide-policy by additional suboptimal data We first study the offline learning setting where the agent has only access to a small high-quality dataset $\mathcal{D}_e$, but also has a large offline dataset $\mathcal{D}_o$ sampled from one or multiple behavior policies that could be highly sub-optimal. This setting is more realistic as low-quality datasets appear more often in real-world tasks [49].



(a) antmaze-large-diverse (b) hopper-medium-replay

Figure 2: Visualizations of the distribution of states generated by the guide-policy and states in the dataset, on `antmaze-large-diverse` and `hopper-medium-replay` datasets. The left pannel of each subfigure shows the $100,000$ data points of $g(s)$ (blue) and s' (green). The right pannel of each subfigure shows the corresponding value of each state. The distribution of $g(s)$ and s' bear some similarity but $g(s)$ is more prone to generate high-value states, especially on `antmaze-large-diverse` dataset.

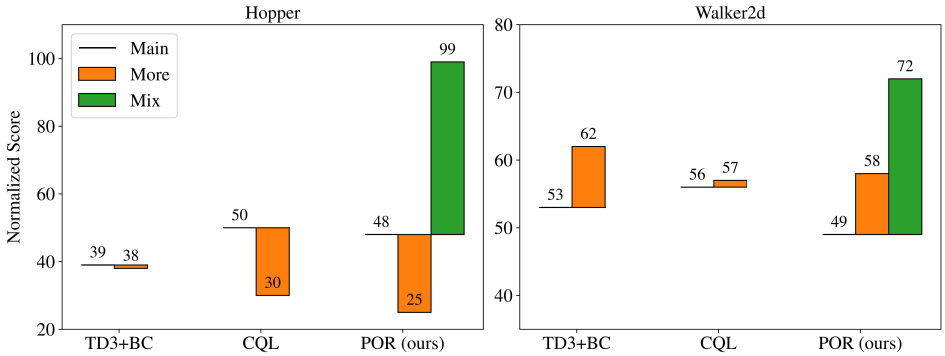
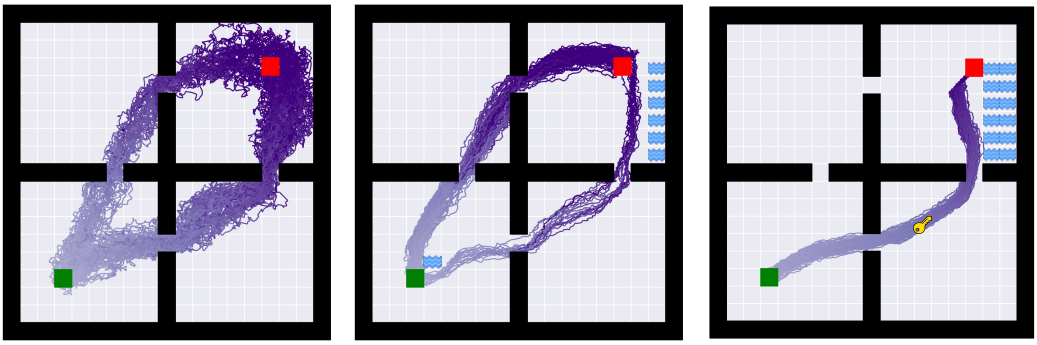


Figure 3: Normalize scores of different algorithms with different training schemes. While the "main" and "more" training scheme can be applied to all methods, the "mix" training scheme is only available to `POR` because we decouple the training process of g and π. With the "mix" training scheme, `POR` outperforms all other algorithms by a large margin, in both `hopper` and `walker2d` environments.

Naïvely, we can treat $\mathcal{D}_e$ or combine $\mathcal{D}_e$ and $\mathcal{D}_o$ as the offline dataset $\mathcal{B}$, then run any off-the-shelf offline RL algorithm. We label this two approach as "main" and "more". However, as shown in Figure 3, the performance of TD3+BC and CQL are low, indicating that both "main" and "more" are insufficient to learn a good policy. The reason why "main" fails is because the resulting policy may not be able to generalize due to the limited size. While "more" can generalize better than "main" due to access to a much larger dataset, the policy will be negatively impacted by the low-quality data in $\mathcal{D}_o$ as both TD3+BC and CQL contain the behavior regularization term.

Due to the availability of decoupled training, `POR` is able to use a different training scheme, we can only use $\mathcal{D}_e$ to train the execute-policy but use the combination of $\mathcal{D}_e$ and $\mathcal{D}_o$ to train the guide-policy. The (optimal) value function in $\mathcal{D}_e$ will become better when combining with $\mathcal{D}_o$, as the value function will be more accurate and better generalized with more data coverage (see *exploration* in online RL as an example). We lable this unique training procedure as "mix". It can be shown in Figure 3 that, by doing so, `POR` could have a super large improvement over "main" and "more". "mix" could bring `POR` up tp 99, close to the upper limit of the performance in the `hopper` environment, with only limited data. In the `walker2d` environment, `POR` with "mix" also outperforms all other baselines.

Change guide-policy in new tasks Then, we study whether we can use the guide-policy to do a lightweight adaptation to new tasks, without changing the execute-policy. We conduct experiments on the continuous variant of the classic four-room environment [16]. The training data consists of trajectories collected by a goal-reaching controller with the start and end location sampled randomly at non-wall locations, mixed with data collected by a random policy. In this environment, we introduce three different tasks: Task A (`Four-room`) needs the agent to travel from the start location to the goal location, besides that, Task B (`Four-room-river`) and task C (`Four-room-key`) require the agent to bypass the river or get a key before arriving at the goal location.



(a) Four-room (b) Four-room-river (c) Four-room-key

Figure 4: Rollout trajectories of POR with the same pretrained execute-policy but different guide policies in different four-room tasks, with color becoming more saturated as time progresses. The execute-policy is pretrained in task A and remains unchanged in task B and C. All three tasks require the agent to find a path from the start to the goal. Besides, Task B requires the agent not to fall into the river and task C requires the agent to get the key as well as avoiding the river before arriving at the goal location.

We first train the guide-policy and the execute-policy π to solve task A. In task B and task C, we re-train the guide-policy using the corresponding reward function in that task, and reuse the execute-policy π from Task A. In Figure 4, we can see that by doing so, POR is able to successfully solve task B and C. In details, in task B, POR is aware of the river and correctly bypass it. In task C, POR only chooses to pass the bottem door, this is owing to the awareness of the importance of obtaining the key before reaching the goal. This result shows the strong adaptation ablity of the guide-policy for different tasks, as well as the generlization ability of the execute-policy across different tasks.

5.4 Ablation Study on the Execute-Policy

We also present ablation study on the execute-policy, with the aim to answer the following questions: 1) How does network capacity affect the performance of the execute-poliy? 2) How does the performance change if we use partial "good" data, instead of full data, to train the execute policy?

Table 2: Ablation study of the execute-policy on network capacity and dataset size.

	Big Network	Small Network
Full $\mathcal{B}$	A: 76.2, M: 65.7	A: 63.8, M: 57.0
Partial $\mathcal{B}$	A: 52.4, M: 60.9	A: 41.5, M: 56.7

To answer the first question, we compare with two choices of network size: a big network with $(1024, 1024)$ hidden units and a small network with $(256, 256)$ hidden units. To answer the second question, we compare with the choice that only select a subset of the dataset to train the execute policy. Concretely, we choose the top $X\%$ trajectories in the dataset, ordered by episode returns. We sweep X over $[10, 25, 40]$ and choose the best score. We give the mean scores of AntMaze (A) and MuJoCo (M) datasets in Table 2. It can be seen that adopting a big network consistently gives a better performance, which is also found in [8]. Using partial data will also result in a less-performed policy, especially on AntMaze datasets. Note that in MuJoCo datasets, the performance didn't drop too much. This is because MuJoCo datasets don't require the compositionality ability, using partial trajectories could already achieve high scores.

6 Conclusions and Limitations

In this work, we propose a novel offline RL approach, POR, which leverages the training stability of imitation-style methods while still encourages logical out-of-distribution generlization. POR allows *state-compositionality* rather than *action-compositionality* from the dataset. Through theoretical analysis and extensive experiments, we show that POR outperforms prior methods on a variety of datasets, especially those with low quality. We also emprically demonstrate the additional benfits of POR in terms of improving with supplementary suboptimal data and easily adapting to new tasks. One limitation of POR is that the prediction of the guide-policy may be biased when the state space is high-dimensional. However, this can be allieviated with the help of reprensentation learning [27, 52].

References

[1] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in Neural Information Processing Systems*, 34, 2021.

[2] Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhihong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. *ArXiv preprint*, abs/2202.11566, 2022.

[3] David Brandfonbrener, William F Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *ArXiv preprint*, abs/2106.08909, 2021.

[4] David Brandfonbrener, William F Whitney, Rajesh Ranganath, and Joan Bruna. Quantile filtered imitation learning. *ArXiv preprint*, abs/2112.00950, 2021.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *ArXiv preprint*, abs/1606.01540, 2016.

[6] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021.

[7] Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith Ross. BAIL: best-action imitation learning for batch deep reinforcement learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[8] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *ArXiv preprint*, abs/2112.10751, 2021.

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

[10] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *ArXiv preprint*, abs/2106.06860, 2021.

[11] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *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*. PMLR, 2018.

[12] Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Manon Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[13] Caglar Gulcehre, Sergio Gómez Colmenarejo, Ziyu Wang, Jakub Sygnowski, Thomas Paine, Konrad Zolna, Yutian Chen, Matthew Hoffman, Razvan Pascanu, and Nando de Freitas. Regularized behavior value estimation. *ArXiv preprint*, abs/2103.09575, 2021.

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *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*. PMLR, 2018.

[15] Geoffrey E. Hinton and Sam T. Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*. MIT Press, 2002.

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

[17] Sham M. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*. MIT Press, 2001.

[18] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *ArXiv preprint*, abs/2104.08212, 2021.

[19] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[21] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*. PMLR, 2021.

[22] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *ArXiv preprint*, abs/2110.06169, 2021.

[23] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019.

[24] Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. *ArXiv preprint*, abs/1912.13465, 2019.

[25] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[26] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*. Springer, 2012.

[27] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*. PMLR, 2020.

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

[29] Hsueh-Ti Derek Liu, Francis Williams, Alec Jacobson, Sanja Fidler, and Or Litany. Learning smooth neural functions via lipschitz regularization. *ArXiv preprint*, abs/2202.08345, 2022.

[30] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on robot learning*. PMLR, 2020.

[31] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[32] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *ArXiv preprint*, abs/1912.02074, 2019.

[33] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *ArXiv preprint*, abs/2006.09359, 2020.

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019.

[35] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *ArXiv preprint*, abs/1910.00177, 2019.

[36] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, 1989.

[37] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *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*. JMLR.org, 2015.

[38] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2014.

[39] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[40] Shengpu Tang and Jenna Wiens. Model selection for offline reinforcement learning: Practical considerations for healthcare settings. In *Machine Learning for Healthcare Conference*. PMLR, 2021.

[41] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012.

[42] Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *ArXiv preprint*, abs/1812.02648, 2018.

[43] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018.

[44] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh Merel, Jost Tobias Springenberg, Scott E. Reed, Bobak Shahriari, Noah Y. Siegel, Çaglar Gülçehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[45] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4), 1992.

[46] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *ArXiv preprint*, abs/1911.11361, 2019.

[47] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua M. Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*. PMLR, 2021.

[48] Rui Yang, Yiming Lu, Wenzhe Li, Hao Sun, Meng Fang, Yali Du, Xiu Li, Lei Han, and Chongjie Zhang. Rethinking goal-conditioned supervised learning and its connection to offline rl. *ArXiv preprint*, abs/2202.04478, 2022.

[49] Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Chelsea Finn, and Sergey Levine. How to leverage unlabeled data in offline reinforcement learning. *ArXiv preprint*, abs/2202.01741, 2022.

[50] 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 *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[51] Xianyuan Zhan, Haoran Xu, Yue Zhang, Yusen Huo, Xiangyu Zhu, Honglei Yin, and Yu Zheng. Deepthermal: Combustion optimization for thermal power generating units using offline reinforcement learning. *ArXiv preprint*, abs/2102.11492, 2021.

[52] Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[53] Wenxuan Zhou, Sujay Bajracharya, and David Held. Latent action space for offline reinforcement learning. In *Conference on Robot Learning*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix B.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplement.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix C
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] In all related figures.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Apendix C
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Data from [9]
   (b) Did you mention the license of the assets? [Yes] The license is Apache 2.0.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See supplement for code and four room dataset.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] Data is simulated.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] Data is simulated.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]