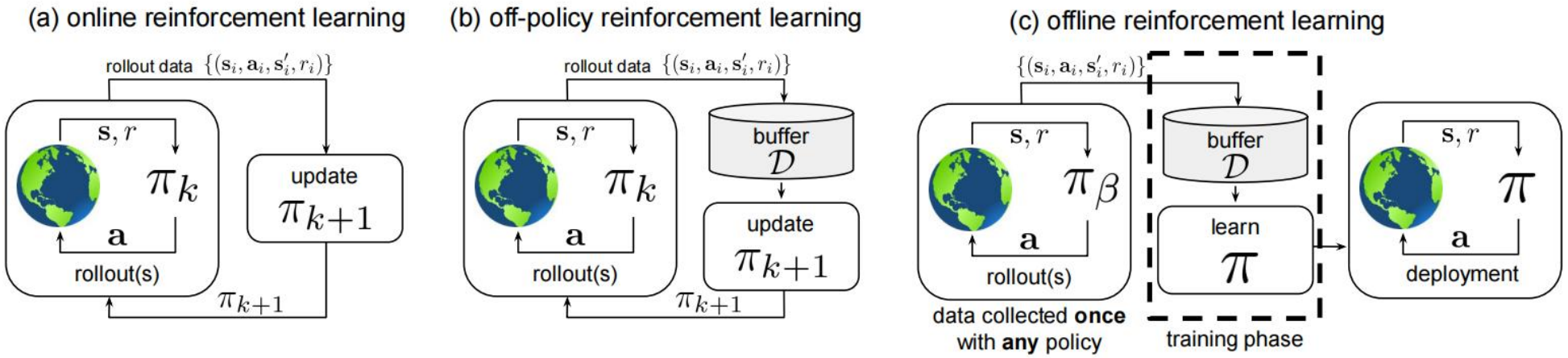
A Policy-Guided Imitation Approach for Offline Reinforcement Learning

Haoran Xu[♠]* **Li Jiang**[♣]* **Jianxiong Li**[♣] **Xianyuan Zhan**[♣,◇]
[♠]JD Technology, Beijing, China
[♣]Tsinghua University, Beijing, China
[◇]Shanghai AI Laboratory, Shanghai, China
{ryanxhr,jiangli3859}@gmail.com

NeurIPS 2022

Background

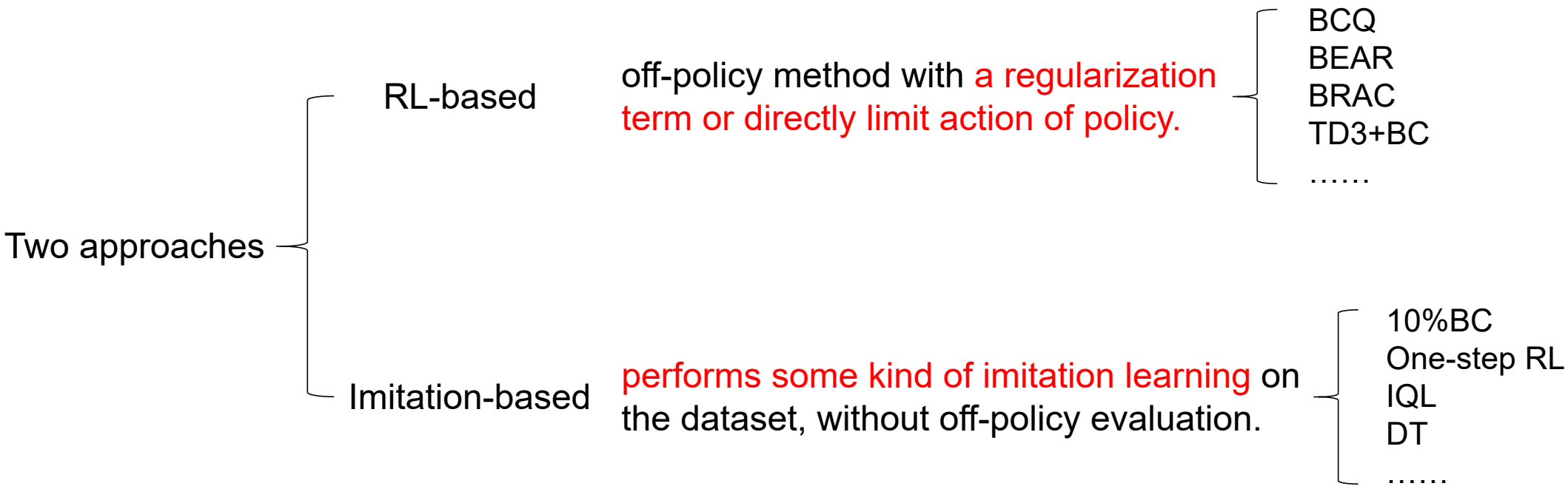


The problem of offline RL:

distributional shift

- state : state distributional shift occurs only deployment phase.
- action : action distributional shift occurs during both training phase and deployment phase.

Background



The core of both approaches is that the learned policy does not stray too far from the behavior policy.

The core difference of them is that RL-based methods learn a value function of policy while imitation-based methods don’t.

Motivation

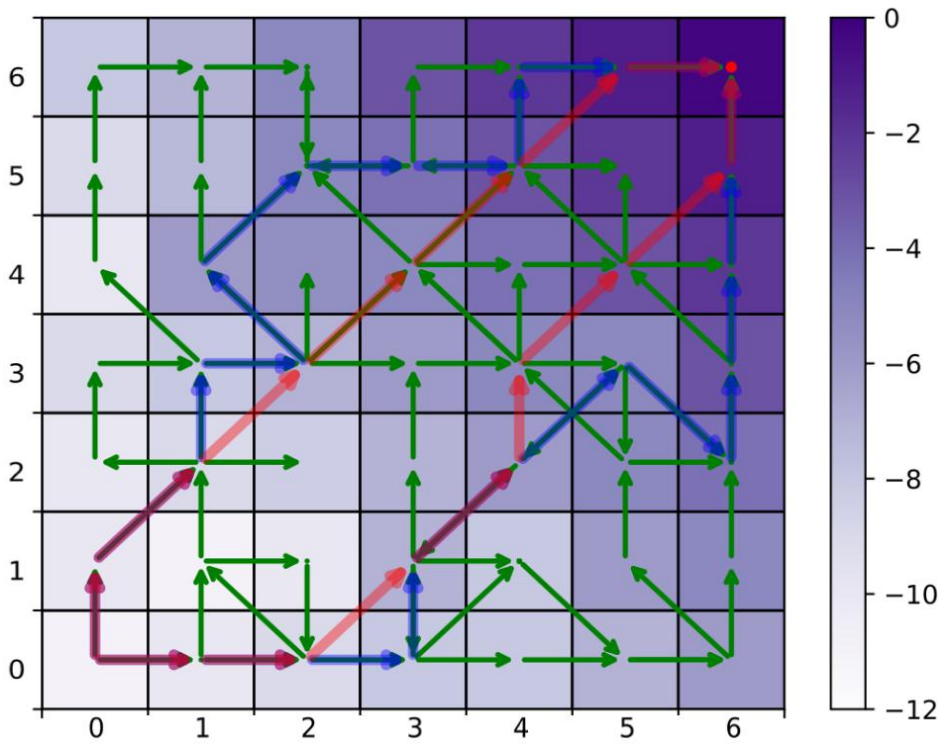
RL-based methods have the ability of out-of-distribution generalization, but is easily influenced by the value function misestimation.

Imitation-based methods are just the opposite, and more stable.

POR: a combination of RL-based and imitation-based methods

A state-stitching method with state value function

Motivation



green-arrowed lines : the offline dataset

red-arrowed lines : the trajectories obtained from state-stitching method

blue-arrowed lines : the trajectories obtained from action-stitching method

This shows the advantage of state-stitching methods

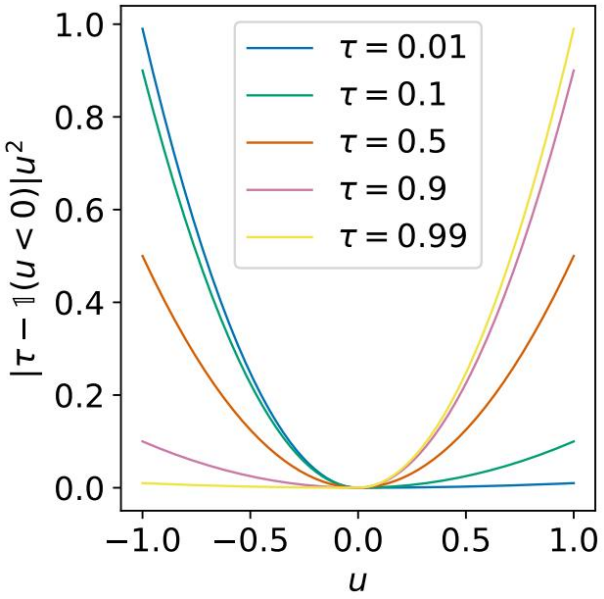
The main idea of POR:
the guide-policy tells the agent which state to go, the execute-policy takes the action

Method

Learning the state value function:

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

When $\tau = 1/2$, this operator is equal to Bellman expectation operator, while when $\tau \to 1$, this operator approaches Bellman optimality operator.



when $\tau > 1/2$, the loss gives more weight to larger values

Purpose： make state value function more distinct for different states

Learning the guide-policy:

$$\max_{\omega} \mathbb{E}_{(s,s')\sim\mathcal{D}} \left[V_\phi(g_\omega(s)) + \alpha \log g_\omega(s'|s) \right]$$

In addition to maximizing the guide-policy with respect to V (s), consider whether it can be done

An alternative learning objective that eliminates the need of α :

$$\max_{\omega} \mathbb{E}_{(s,s')\sim\mathcal{D}} \left[e^{(r+\gamma V_{\phi'}(s') - V_\phi(s))} \log g_\omega(s'|s) \right]$$

Learning the execute-policy:

learning objective during training:

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

During evaluation:

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

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

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

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*

$$\|\pi(s, g(s)) - a^*\| \leq \underbrace{(L_1 + L_2)\|g(s) - s'\|}_{l_1} + \underbrace{\|a_g - a^*\|}_{l_2} + \underbrace{\epsilon}_{l_3} \tag{8}$$

Proof:

$$\begin{aligned}
\|\pi(s, g(s)) - a^*\| &= \|\pi(s, g(s)) - \pi(s, s') + \pi(s, s') - a + a - a_g + a_g - a^*\| \\
&\leq \|\pi(s, g(s)) - \pi(s, s')\| + \|\pi(s, s') - a\| + \|a - a_g\| + \|a_g - a^*\| \ \text{(Triangle)} \\
&\leq L_2\|g(s) - s'\| + \epsilon + L_1\|g(s) - s'\| + \|a_g - a^*\| \ \text{(Assumption1\&2)} \\
&\leq \underbrace{(L_1 + L_2)\|g(s) - s'\|}_{l_1} + \underbrace{\|a_g - a^*\|}_{l_2} + \underbrace{\epsilon}_{l_3}
\end{aligned}$$

A stronger conclusion:

$$J(\pi^*) - J(\pi) \leq \sup_{s,a,h}(l_1 + l_2 + l_3)T$$

Method

Algorithm 1 Policy Guided Offline RL

Require: $\mathcal{D}$, τ, α (optional).

// Training
Initialize V_ϕ, $V_{\phi'}$, g_ω, π_θ
for $t = 1, 2, \cdots, N$ **do**
    Sample transitions $(s, r, s') \sim \mathcal{D}$
    Update V_ϕ by Eq.(3) → $\min_\phi \mathbb{E}_{(s,r,s')\sim\mathcal{D}} \left[L_2^\tau \left(r + \gamma V_{\phi'}(s') - V_\phi(s) \right) \right]$, where $L_2^\tau(u) = |\tau - \mathbb{1}(u < 0)|u^2$
    Update g_ω by Eq.(4) or Eq.(5) → $\max_\omega \mathbb{E}_{(s,s')\sim\mathcal{D}} \left[V_\phi(g_\omega(s)) + \alpha \log g_\omega(s'|s) \right]$ / $\max_\omega \mathbb{E}_{(s,s')\sim\mathcal{D}} \left[e^{(r+\gamma V_{\phi'}(s') - V_\phi(s))} \log g_\omega(s'|s) \right]$
    Update $V_{\phi'}$ by $\phi' \leftarrow \lambda\phi + (1-\lambda)\phi'$
end for
for $t = 1, 2, \cdots, M$ **do**
    Sample transitions $(s, a, s') \sim \mathcal{D}$
    Update π_θ by Eq.(6) → $\max_\theta \mathbb{E}_{(s,a,s')\in\mathcal{D}} \left[\log \pi_\theta(a|s, s') \right]$
end for
// Evaluation
Get initial state s, set d as False
while not d **do**
    Get action a form Eq.(7) → $a = \arg\max_a \pi_\theta(a|s, g_\omega(s))$
    Roll out a and get (s', r, d)
    Set $s = s'$
end while

Note that the training of guide-policy and execute-policy are fully decoupled

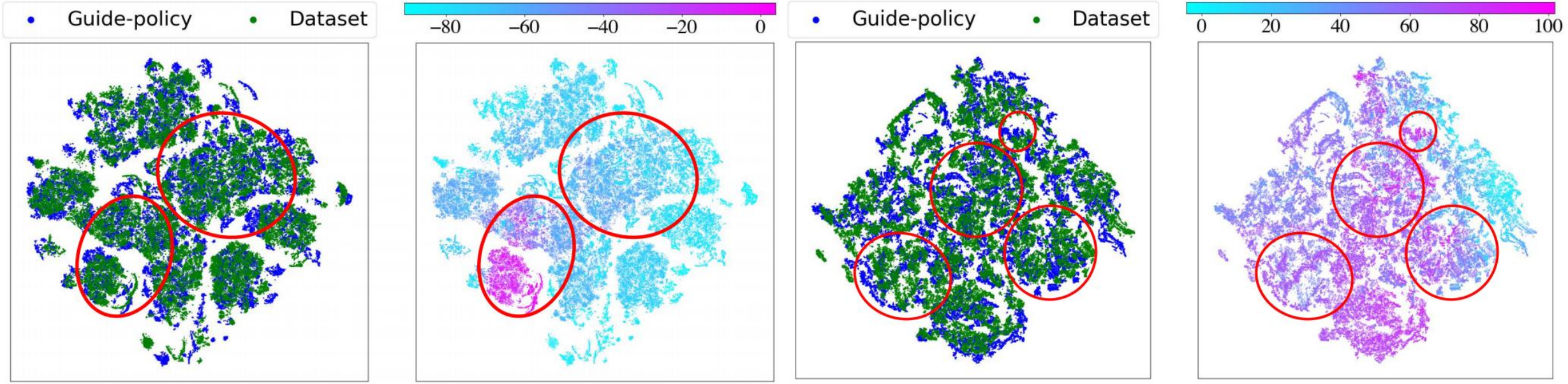
Experiment

D4RL Dataset	Weighted BC			Conditioned BC			Best RL Baseline
	10%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

POR performs competitively to the best performance of prior methods in high-quality datasets, while achieving much better performance than other methods in low-quality datasets

Experiment

Validation Experiments on Out-of-Distribution Generalization



(a) antmaze-large-diverse

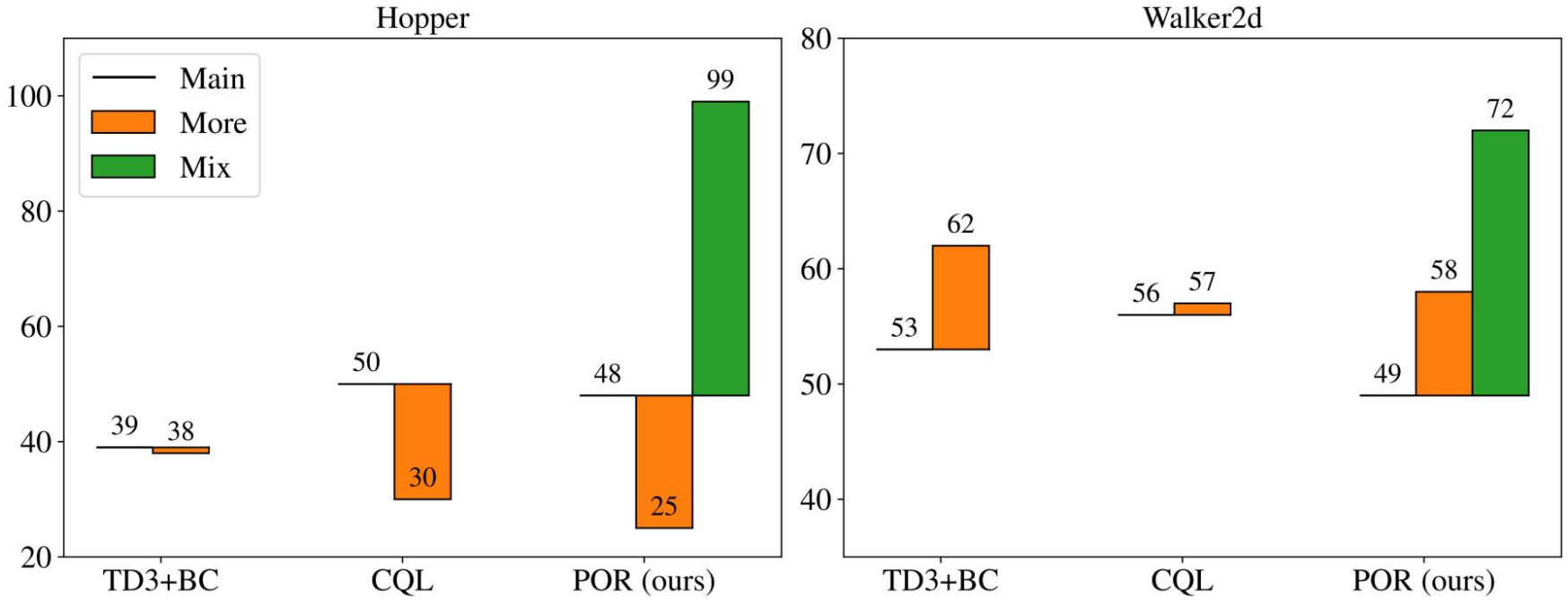
(b) hopper-medium-replay

The distribution of $g(s)$ bears some similarity to the distribution of s'

At the non-overlapped area, $g(s)$ has a much higher value than s',
which proves the useful generalization of $g(s)$

Experiment

Improve guide-policy by additional suboptimal data



$\mathcal{D}_O$: a small yet high-quality dataset(original)
$\mathcal{D}_e$: a supplementary large dataset
main : only use $\mathcal{D}_O$ to train
more : use $\mathcal{D}_O \cup \mathcal{D}_e$ to train
mix : keep the execute-policy learned from $\mathcal{D}_e$, use $\mathcal{D}_O \cup \mathcal{D}_e$ to re-train the guide-policy

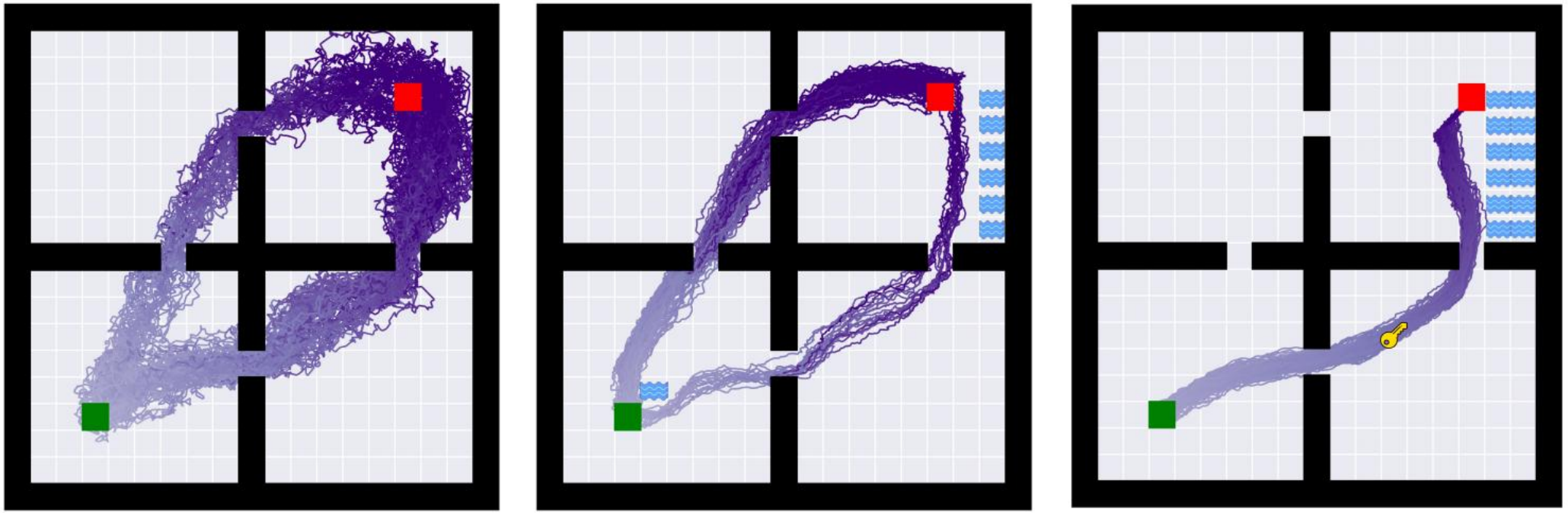
One possibility : More trajectories make $V(s)$ more exact accurate.

Experiment

Task A

Task B

Task C



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

First train g and π to solve task A, then in task B and task C, re-train g and reuse π in task A

This result shows the strong adaptation ablity of the guide-policy g for different tasks, as well as the generalization ability of the execute-policy π across different tasks.

Why One-step and IQL are imitation-based methods?

The key to judging whether an algorithm is a imitation-based method is the existence of policy evaluation.

CQL:

Algorithm 1 Conservative Q-Learning (both variants)

1: Initialize Q-function, Q_θ, and optionally a policy, π_ϕ.
2: **for** step t in $\{1, \ldots, \mathrm{N}\}$ **do**
3: Train the Q-function using G_Q gradient steps on objective from Equation 4
$\theta_t := \theta_{t-1} - \eta_Q \nabla_\theta \mathrm{CQL}(\mathcal{R})(\theta)$
(Use $\mathcal{B}^*$ for Q-learning, $\mathcal{B}^{\pi_{\phi_t}}$ for actor-critic)
4: (only with actor-critic) Improve policy π_ϕ via G_π gradient steps on ϕ with SAC-style entropy regularization:
$\phi_t := \phi_{t-1} + \eta_\pi \mathbb{E}_{\mathbf{s}\sim\mathcal{D}, \mathbf{a}\sim\pi_\phi(\cdot|\mathbf{s})}[Q_\theta(\mathbf{s}, \mathbf{a}) - \log \pi_\phi(\mathbf{a}|\mathbf{s})]$
5: **end for**

IQL:

Algorithm 1 Implicit Q-learning

Initialize parameters ψ, θ, $\hat{\theta}$, ϕ.
TD learning (IQL):
for each gradient step **do**
$\psi \leftarrow \psi - \lambda_V \nabla_\psi L_V(\psi)$
$\theta \leftarrow \theta - \lambda_Q \nabla_\theta L_Q(\theta)$
$\hat{\theta} \leftarrow (1-\alpha)\hat{\theta} + \alpha\theta$
end for
Policy extraction (AWR):
for each gradient step **do**
$\phi \leftarrow \phi - \lambda_\pi \nabla_\phi L_\pi(\phi)$
end for

Thanks