南京航空航天大學

Nanjing University of Aeronautics and Astronautics

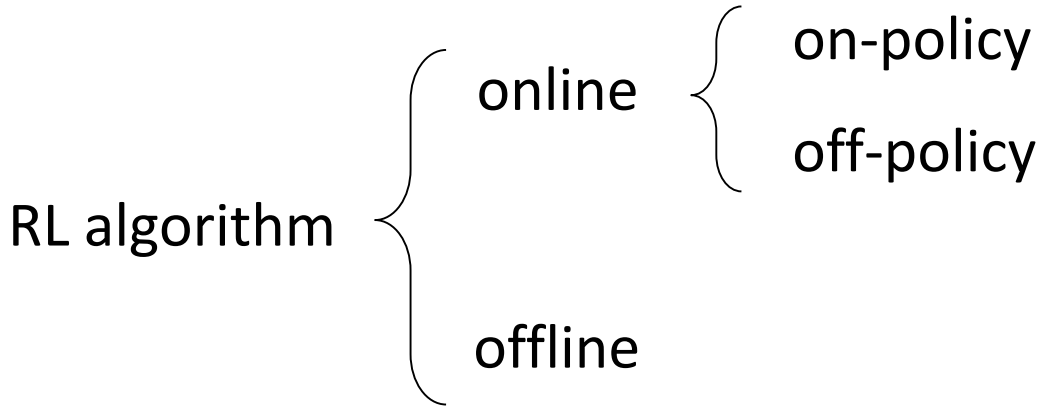
ParNeC 模式识别与神经计算研究组

PAttern Recognition and NEural Computing

Offline Reinforcement Learning With In-Sample Q-Learning

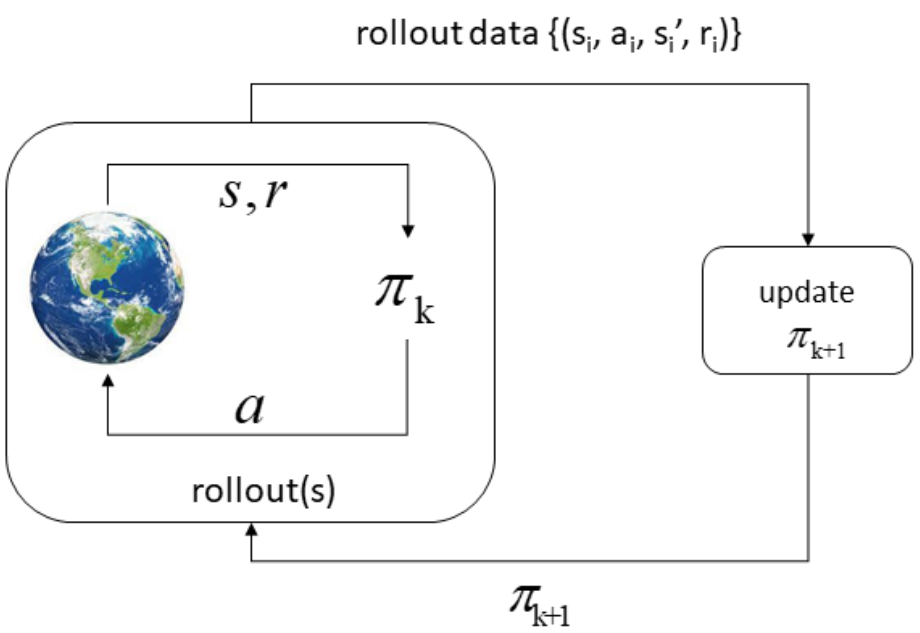
ICLR 2022

Background



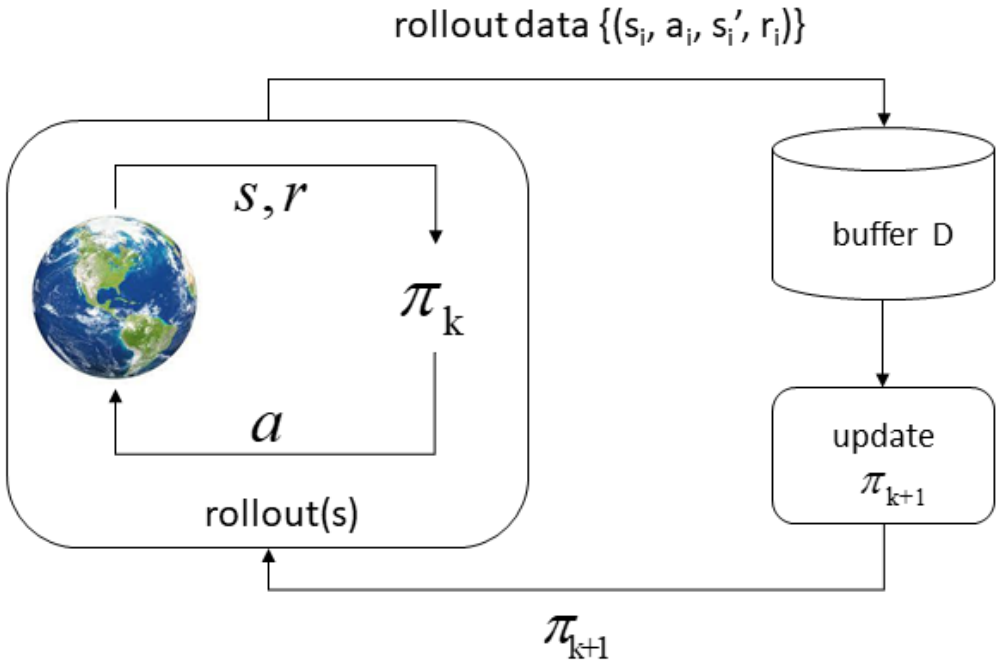
- on-policy: policy is updated with data collected by itself.
- off-policy: Behavior policy ≠ Policy used for action selection
- offline: utilize previously collected data, without additional online data collection

Background



on-policy methods

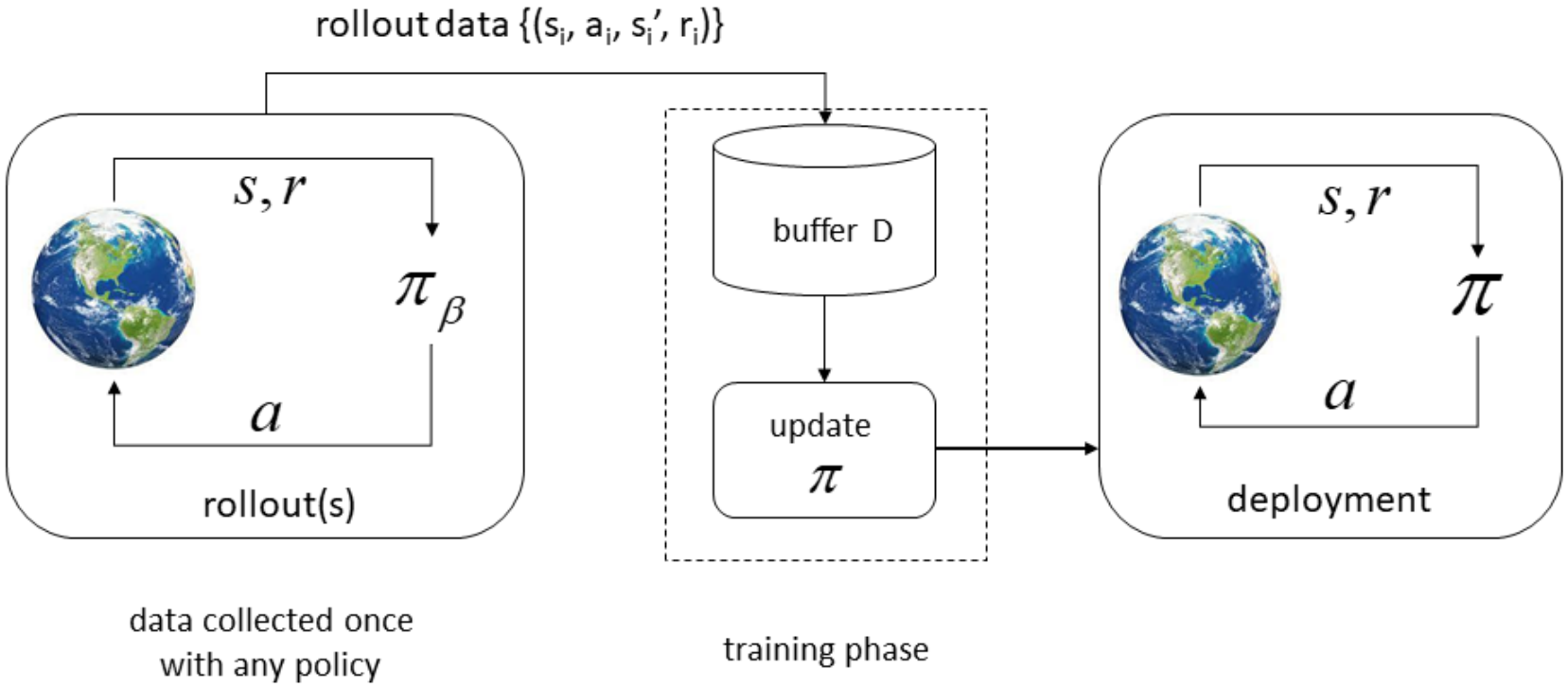
Examples: Policy Iteration, Sarsa, PPO, TRPO



off-policy methods

Examples: Q- learning, DQN, DDPG

Background

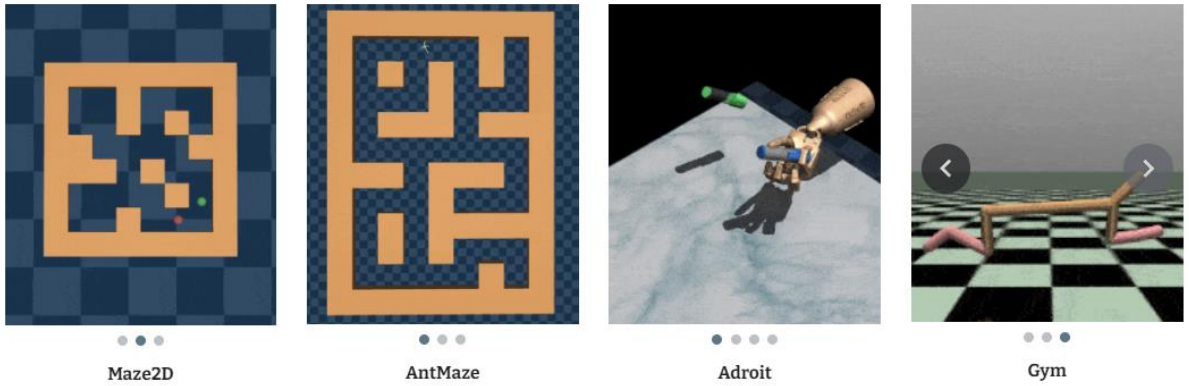


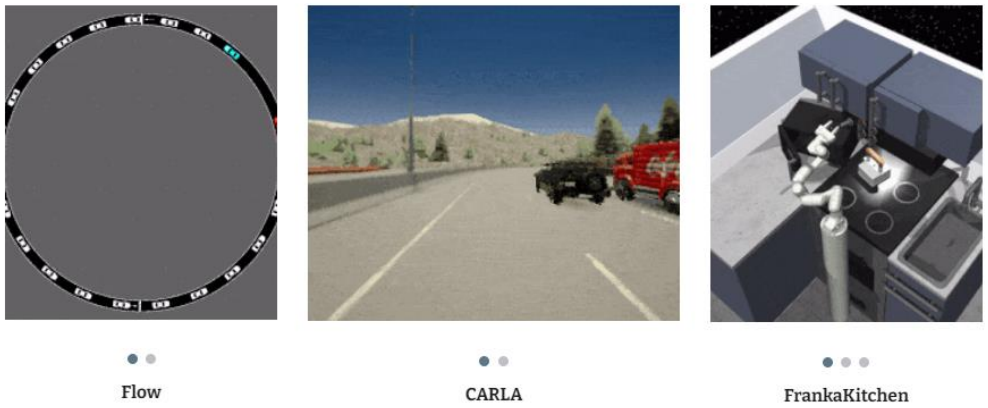
offline methods

Motivation

- offline RL is very appealing in a range of real-world domains.
- real-world exploration with untrained policies is costly or dangerous, but prior data is available.
- prior methods generally by constraining the policy or regularizing the learned value function to avoid out-of-distribution actions.

Offline Dataset Introduction





D4RL 离线强化学习数据集

特点：

- 无指向性和多任务数据
- 窄的数据分布
- 不是RL policy产生的数据
- 包含次优轨迹的数据

https://sites.google.com/view/d4rl/home

Preliminaries

- goal: to obtain a policy that maximizes the cumulative discounted returns.

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 \sim p_0(\cdot), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)\right]$$

- offline RL:

$$L_{TD}(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[(r(s,a) + \gamma \max_{a'} Q_{\hat{\theta}}(s', a') - Q_{\theta}(s,a))^2]$$

Expectile Regression

Practical methods for estimating various statistics of a random variable have been thoroughly studies in applied statistics and econometrics. The $\tau \in (0, 1)$ expectile of some random variable X is defined as a solution to the asymmetric least squares problem:

$$\arg\min_{m_\tau} \mathbb{E}_{x\sim X}[L_2^\tau(x - m_\tau)], \text{ where } L_2^\tau(u) = |\tau - \mathbb{1}(u < 0)|u^2$$

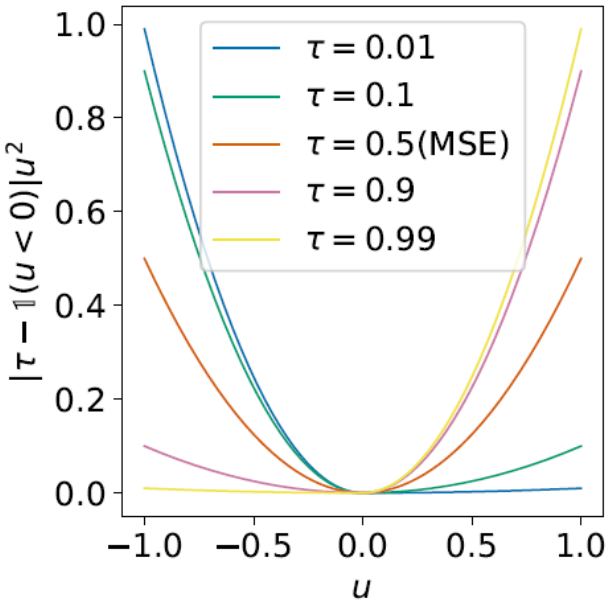


Figure 1: expectile regression.
tau = 0.5 corresponds to the standard MSE loss,
tau= 0.9 gives more weight to positives differences.

In-Sample Q-learning

aim to entirely avoid querying out-of-sample actions:

$$L(\theta) = \mathbb{E}_{(s,a,s',a')\sim\mathcal{D}}[(r(s,a) + \gamma Q_{\hat{\theta}}(s',a') - Q_\theta(s,a))^2]$$

if the dataset has unlimited capacity and no sampling error, the optimal parameters should satisfy

$$Q_{\theta^*}(s,a) \approx r(s,a) + \gamma \mathbb{E}_{\substack{s'\sim p(\cdot|s,a) \\ a'\sim\pi_\beta(\cdot|s)}}[Q_{\hat{\theta}}(s',a')]$$

Learning the value function with expectile regression

- use expectile regression to modify the policy evaluation objective

$$L(\theta) = \mathbb{E}_{(s,a,s',a') \sim \mathcal{D}}[L_2^\tau(r(s,a) + \gamma Q_{\hat{\theta}}(s',a') - Q_\theta(s,a))]$$

- "lucky" sample: a large target value might not necessarily reflect the existence of a single action that achieves that value

$$L_V(\psi) = \mathbb{E}_{(s,a)\ \sim \mathcal{D}}[L_2^\tau(Q_{\hat{\theta}}(s,a) - V_\psi(s))]$$

$$L_Q(\theta) = \mathbb{E}_{(s,a,s')\ \sim \mathcal{D}}[(r(s,a) + \gamma V_\psi(s') - Q_\theta(s,a))^2]$$

Algorithm

Algorithm 1 In-Sample Q-learning

Initialize parameters ψ, θ, $\hat{\theta}$, ϕ.

TD learning (IQL):

for each gradient step **do**

$\psi \leftarrow \psi - \lambda_V \nabla_\psi L_V(\psi)$ $\Longrightarrow$ $L_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[L_2^\tau(Q_{\hat{\theta}}(s,a) - V_\psi(s))]$

$\theta \leftarrow \theta - \lambda_Q \nabla_\theta L_Q(\theta)$ $\Longrightarrow$ $L_Q(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[(r(s,a) + \gamma V_\psi(s') - Q_\theta(s,a))^2]$

$\hat{\theta} \leftarrow (1-\alpha)\hat{\theta} + \alpha\theta$

end for

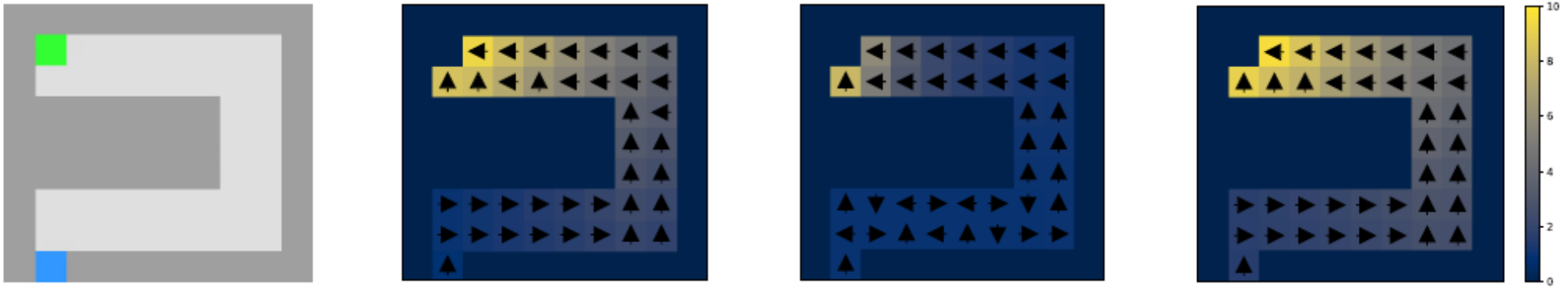
Policy extraction (AWR):

for each gradient step **do**

$\phi \leftarrow \phi - \lambda_\pi \nabla_\phi L_\pi(\phi)$ $\Longrightarrow$ $L_\pi(\phi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[\exp(\beta(Q_{\hat{\theta}}(s,a) - V_\psi(s)))\log \pi_\phi(a|s)]$

end for

Experiment



(a) toy maze MDP (b) true optimal $V^\star$ (c) One-step Policy Eval. (d) IQL

Figure 2: Evaluation of our algorithm on a toy umaze environment (a). When the static dataset is heavily corrupted by suboptimal actions, one-step policy evaluation results in a value function that degrades to zero far from the rewarding states too quickly (c). Our algorithm aims to learn a near-optimal value function, combining the best properties of SARSA-style evaluation with the ability to perform multi-step dynamic programming, leading to value functions that are much closer to optimality (shown in (b)) and producing a much better policy (d).

Experiment

Table 1: Averaged normalized scores on MuJoCo locomotion and Ant Maze tasks. Our method outperforms prior methods on the challenging Ant Maze tasks, which require dynamic programming, and is competitive with the best prior methods on the locomotion tasks.

Dataset	BC	10%BC	BCQ	DT	ABM	AWAC	Onestep RL	TD3+BC	CQL	IQL (Ours)
halfcheetah-m-v2	42.6	42.5	**47.0**	42.6±0.1	53.6	43.5	**48.4±0.1**	**48.3±0.3**	44.0±5.4	**47.4±0.2**
hopper-m-v2	52.9	56.9	56.7	**67.6±1.0**	0.7	57.0	59.6±2.5	59.3±4.2	58.5±2.1	**66.2±5.7**
walker2d-m-v2	75.3	75.0	72.6	74.0±1.4	0.5	72.4	**81.8±2.2**	83.7±2.1	72.5±0.8	78.3± 8.7
halfcheetah-m-r-v2	36.6	40.6	40.4	36.6±0.8	**50.5**	40.5	38.1±1.3	**44.6±0.5**	**45.5±0.5**	**44.2±1.2**
hopper-m-r-v2	18.1	75.9	53.3	82.7±7.0	49.6	37.2	**97.5±0.7**	60.9±18.8	**95.0±6.4**	**94.7±8.6**
walker2d-m-r-v2	26.0	62.5	52.1	66.6±3.0	53.8	27.0	49.5±12.0	**81.8±5.5**	77.2±5.5	73.8±7.1
halfcheetah-m-e-v2	55.2	**92.9**	**89.1**	86.8±1.3	18.5	42.8	**93.4±1.6**	**90.7±4.3**	**91.6±2.8**	86.7±5.3
hopper-m-e-v2	52.5	**110.9**	81.8	**107.6±1.8**	0.7	55.8	103.3±1.9	98.0±9.4	**105.4±6.8**	91.5±14.3
walker2d-m-e-v2	**107.5**	**109.0**	**109.5**	**108.1±0.2**	3.5	74.5	**113.0±0.4**	**110.1±0.5**	**108.8±0.7**	**109.6±1.0**
locomotion-v2 total	466.7	**666.2**	602.5	**672.6±16.6**	231.4	450.7	**684.6±22.7**	**677.4±44.5**	**698.5±31.0**	**692.4±52.1**
antmaze-u-v0	54.6	62.8	**89.8**	59.2	59.9	56.7	64.3	78.6	74.0	**87.5 ± 2.6**
antmaze-u-d-v0	45.6	50.2	**83.0**	53.0	48.7	49.3	60.7	71.4	**84.0**	62.2 ± 13.8
antmaze-m-p-v0	0.0	5.4	15.0	0.0	0.0	0.0	0.3	10.6	61.2	**71.2 ± 7.3**
antmaze-m-d-v0	0.0	9.8	0.0	0.0	0.5	0.7	0.0	3.0	53.7	**70.0 ± 10.9**
antmaze-l-p-v0	0.0	0.0	0.0	0.0	0.	0.0	0.0	0.2	15.8	**39.6±5.8**
antmaze-l-d-v0	0.0	6.0	0.0	0.0	0.0	1.0	0.0	0.0	14.9	**47.5±9.5**
antmaze-v0 total	100.2	134.2	187.8	112.2	109.1	107.7	125.3	163.8	303.6	**378.0±49.9**
total	566.9	800.4	790.3	784.8	340.5	558.4	809.9	841.2	1002.1	**1070.4±102.0**
kitchen-v0 total	**154.5**	-	-	-	-	-	-	-	144.6	**159.8±22.6**
adroit-v0 total	104.5	-	-	-	-	-	-	-	93.6	**118.1±30.7**
total+kitchen+adroit	825.9	-	-	-	-	-	-	-	1240.3	**1348.3±155.3**
runtime	10m	10m		960m	20m	20m*	20m	80m	20m	

*: Note that it is challenging to compare one-step and multi-step methods directly. Also, Brandfonbrener et al. (2021) reports results for a set of hyperparameters, such as batch and network size, that is significantly different from other methods. We report results for the original hyperparameters and runtime for a comparable set of hyperparameters.

Experiment

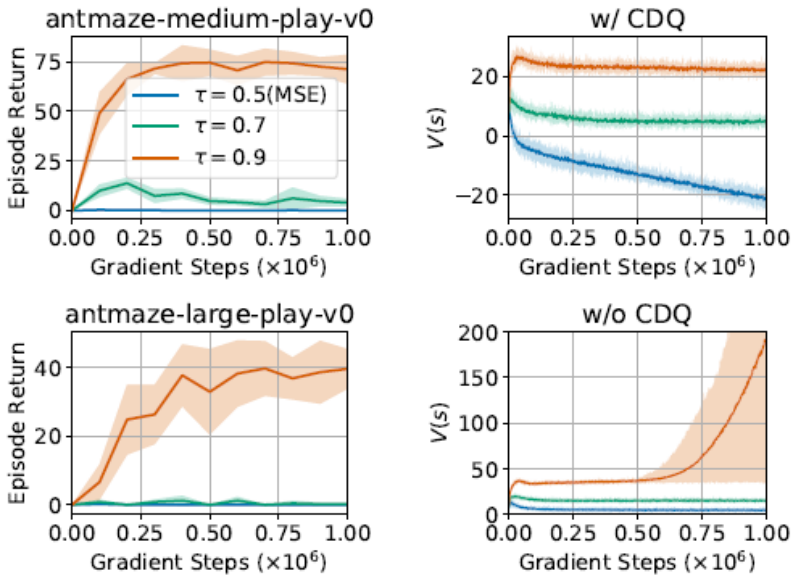


Figure 3: **Left**: Estimating a larger expectile τ is crucial for antmaze tasks that require dynamical programming ('stitching') (left). **Right:** Clipped double Q-Learning is crucial for learning values for $\tau = 0.9$ (right).

Dataset	AWAC	CQL	IQL (Ours)
antmaze-umaze-v0	56.7 → 59.0	70.1 → **99.4**	**86.7** → **96.0**
antmaze-umaze-diverse-v0	49.3 → 49.0	31.1 → **99.4**	**75.0** → 84.0
antmaze-medium-play-v0	0.0 → 0.0	23.0 → 0.0	**72.0** → **95.0**
antmaze-medium-diverse-v0	0.7 → 0.3	23.0 → 32.3	**68.3** → **92.0**
antmaze-large-play-v0	0.0 → 0.0	1.0 → 0.0	**25.5** → **46.0**
antmaze-large-diverse-v0	1.0 → 0.0	1.0 → 0.0	**42.6** → **60.7**
antmaze-v0 total	107.7 → 108.3	151.5 → 231.1	**370.1** → **473.7**
pen-binary-v0	**44.6** → **70.3**	31.2 → 9.9	37.4 → 60.7
door-binary-v0	**1.3** → **30.1**	0.2 → 0.0	0.7 → **32.3**
relocate-binary-v0	**0.8** → 2.7	0.1 → 0.0	0.0 → **31.0**
hand-v0 total	**46.7** → 103.1	31.5 → 9.9	38.1 → **124.0**
total	154.4 → 211.4	182.8 → 241.0	**408.2** → **597.7**

Table 2: Online finetuning results showing the initial performance after offline RL, and performance after 1M steps of online RL. In all tasks, IQL is able to finetune to a significantly higher performance than the offline initialization, with final performance that is comparable to or better than the best of either AWAC or CQL on all tasks except pen-binary-v0.

THANKS