Offline Reinforcement Learning with On-Policy Q-Function Regularization

Laixi Shi(✉)$^{1[0000-0003-4038-8620]}$, Robert Dadashi^{2}, Yuejie Chi$^{1[0000-0002-6766-5459]}$, Pablo Samuel Castro^{2}, and Matthieu Geist^{2}

^{1} Carnegie Mellon University, Pittsburgh PA, USA
{laixishi,yuejiechi}@cmu.edu
^{2} Google Research, Brain Team
{dadashi,psc,mfgeist}@google.com

Abstract. The core challenge of offline reinforcement learning (RL) is dealing with the (potentially catastrophic) extrapolation error induced by the distribution shift between the history dataset and the desired policy. A large portion of prior work tackles this challenge by implicitly/explicitly regularizing the learning policy towards the behavior policy, which is hard to estimate reliably in practice. In this work, we propose to regularize towards the Q-function of the behavior policy instead of the behavior policy itself, under the premise that the Q-function can be estimated more reliably and easily by a SARSA-style estimate and handles the extrapolation error more straightforwardly. We propose two algorithms taking advantage of the estimated Q-function through regularizations, and demonstrate they exhibit strong performance on the D4RL benchmarks.

Keywords: offline reinforcement learning · actor-critic · SARSA

1 Introduction

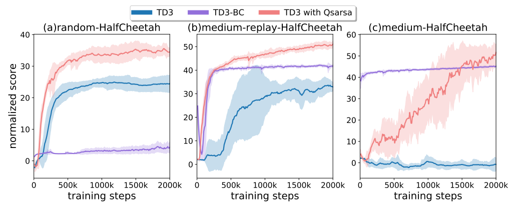

Fig. 1. A comparison between our Q_{sarsa}-regularized TD3 method (TD3 with Qsarsa), the vanilla TD3 method from online RL [11], and a strong baseline for offline RL TD3-BC [10] in the tasks of MuJoCo in D4RL.

Reinforcement learning (RL) has witnessed a surge of practical success recently, with widespread applications in games such as Go game and StarCraft II [30,31], control, autonomous driving, etc [1,25]. Online RL relies on sequentially collecting data through interactions with the environment. However, new sample collections or interactions might be infeasible due to privacy, safety, or cost, especially in real-world applications. To circumvent this, offline/batch RL seeks to learn from an existing dataset, without further interaction with the environment [26].

The history dataset can be regarded as generated by some unknown behavior policy π_{b}, which is not necessarily of the desired quality or has insufficient coverage over the state-action space. This results in one of the major challenges of offline RL: *distribution shift*. Here, the state-action distribution under the behavior policy may heavily differ from that produced by a more desirable policy. As a result, the policy evaluation process presents considerable extrapolation error over the *out-of-distribution* (OOD) regions (state-action pairs) that are insufficiently visited or even unseen in the history dataset. To address the extrapolation error, prior works mainly follow three principles: 1) *behavior regularization:* regularizing the learning policy to be close to the behavior policy or to imitate the (weighted) offline dataset directly [12,34,22,10,33]; 2) *pessimism:* considering conservative value/Q-function by penalizing the values over OOD state-action pairs [27,23,20,4]; 3) *in-sample:* learning without querying any OOD actions [20,13].

To achieve behavior regularization, a large portion of prior works choose to regularize the policy towards the behavior policy. As the behavior policy is unknown, they usually rely on access to an approximation of the behavior policy, or on access to a distance metric between any policy and the behavior policy [22,8]. However, either estimating the behavior policy as a multimodal conditional distribution or estimating the distance between any policy and π_{b} is a difficult task in practice, requiring tailored and complex designs such as conditional variational auto-encoders (CVAEs) [12,27] or restricted distance metrics [7]. To circumvent this, a natural question is: *Can we steer a learning policy towards the behavior policy without requiring access to the said policy?*

To answer this, we propose to rely on the Q-function of the behavior policy π_{b}, denoted as $Q^{\pi_{\mathsf{b}}}$, instead of relying directly on π_{b}. The Q-function $Q^{\pi_{\mathsf{b}}}$ can not only play a similar role as π_{b} in pushing the learning policy towards the behavior policy, but can also provide additional information such as the quality of the dataset/actions (e.g., larger $Q^{\pi_{\mathsf{b}}}$ represents better quality). In addition, harnessing $Q^{\pi_{\mathsf{b}}}$ in offline RL is promising and has several essential advantages compared to prior works relying on (an estimate of) the behavior policy π_{b}:

Estimating $Q^{\pi_{\mathsf{b}}}$ is easier than π_{b}. $Q^{\pi_{\mathsf{b}}}$ can be estimated by Q_{sarsa} via minimizing a SARSA-style objective, while π_{b} needs to be estimated as a multimodal conditional distribution, which is usually a more difficult problem.

Regularizing towards $Q^{\pi_{\mathsf{b}}}$ is more straightforward than π_{b}. The goal of imitating the behavior policy is usually to restrict extrapolation errors when evaluating the Q-function of the learning policy. Thus, it is more direct and efficient to regularize the Q-function of the learning policy towards $Q^{\pi_{\mathsf{b}}}$ than to do so indirectly in the bootstrapping step via a regularization towards π_{b}.

Despite these advantages, there are few works exploring the use of $Q^{\pi_{\mathsf{b}}}$ for offline RL regularization. Indeed, OnestepRL [3] is, to the best of our knowledge, the only work that promotes the usage of $Q^{\pi_{\mathsf{b}}}$ in continuous action tasks. However, Fig. 2 (more details are specified in Sec. 4.1) shows that although Q_{sarsa}, the estimate of $Q^{\pi_{\mathsf{b}}}$ using a SARSA-style objective, is one of the components in OnestepRL, it may not be the critical one that contributes to its success and hence, more investigations are warranted.

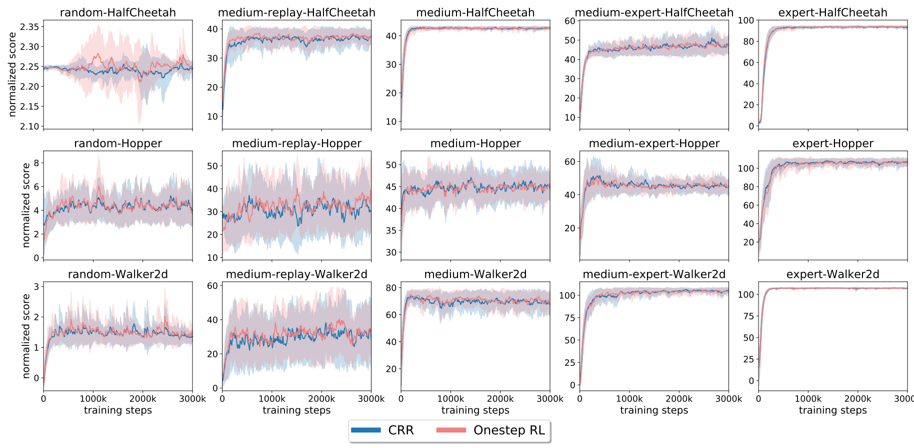

Fig. 2. The comparisions between OnestepRL [3] and CRR [33] over 15 tasks of MuJoCo in D4RL [9]. CRR and OnestepRL use the same actor with different critics (OnestepRL uses Q_{sarsa} while CRR not). The similar performance implies that the actor plays a more important role compared to the critic Q_{sarsa} in OnestepRL.

We suggest that a proper use of Q^{π_b} has the potential to improve the performance in offline RL. As a warm-up study, by adding a slight regularization towards Q_{sarsa} (the estimate of Q^{π_b}) in TD3 [11] without other modifications, we already observe a strong performance boost (see Fig. 1). Motivated by this potential, in this work, we propose to use Q^{π_b} (estimated by Q_{sarsa}) as a promising regularization methodology for offline RL. We first demonstrate that Q_{sarsa} is a reasonable estimate for Q^{π_b} for *in-sample* state-action pairs and has controllable error over out-of-distribution regions. Equipped with these findings, we introduce two Q_{sarsa}-regularized algorithms based on TD3-BC, one of the state-of-the-art offline methods [10], and achieve strong performance on the D4RL benchmark [9], notably better than TD3-BC that we built on and OnestepRL.

2 Related works

We shall discuss several lines of research on offline RL, with an emphasis on the most related works that make use of behavior regularization.

Offline RL. The major challenge of offline RL is the extrapolation error of policy evaluation induced by the distribution shift between the history dataset and the desired learning policy [12,22]. As a result, a large portion of works follow the approximate dynamic programming framework (e.g., actor-critic) to balance learning beyond the behavior policy and handling the distribution shift. They usually resort to behavior regularization/constraints [14,12,22,34,33], pessimistic value/Q-function estimation for out-of-distribution actions [27,23,20,36,29], or learning without querying any OOD actions [20,13]. In addition, prior works also leverage other techniques such as sequential modeling [5,19], representation learning [24], and diffusion models as policies [32].

Offline RL with behavior regularization. In order to handle distribution shift, besides restricting the learning of policy almost to be in-distribution/in-sample [33,28] or leveraging imitation learning and working directly with the dataset [10,6], many prior works regularize the policy to an explicit estimation of the behavior policy [14,12,27,22,35,37] or characterizing the discrepancy between policies based on special distance or tailored approaches [22,8,7].

However, the estimation of either the behavior policy or the distance between policies is difficult in practice or requires special design. In this work, we propose the Q-function of the behavior policy as an essential and helpful component in offline RL, which can be estimated easily but has almost not been studied in the literature. Only a few works [28,33,15] consider the usage of the Q-function of the behavior policy, but usually combined with some special optimization objective for the actor (e.g., advantage-weighted regression) which may take the most credit, or focusing on distinct tasks such as discrete action tasks [15]. To fill the gap, this work evaluates the reliability of estimating the Q-function of the behavior policy, and makes use of this estimate to design methods achieving competitive performance over the baselines.

3 Problem formulation and notations

Discounted infinite-horizon MDP. In this paper, we consider a discounted infinite-horizon Markov Decision Process (MDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$. Here, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ represents the dynamics of this MDP (i.e., $P(\cdot \mid s, a)$ denote the transition probability from current state-action pair (s, a) to the next state), $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the immediate reward function, $\gamma \in [0, 1)$ is the discount factor. We denote a stationary policy, also called an action selection rule, as $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. The value function $V^\pi : \mathcal{S} \to \mathbb{R}$ and Q-value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ associated with policy π are defined as $V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s; \pi\right]$ and $Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a; \pi\right]$, where the expectation is taken over the sample trajectory $\{(s_t, a_t)\}_{t \geq 0}$ generated following that $a_t \sim \pi(\cdot \mid s_t)$ and $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ for all $t \geq 0$.

Offline/Batch dataset. We consider offline/batch RL, where we only have access to an offline dataset $\mathcal{D}$ consisting of N sample tuples $\{s_i, a_i, r_i, s'_i, a'_i\}_{i=1}^N$ generated following some behavior policy π_{b} over the targeted environment. The goal of offline RL is to learn an optimal policy $\pi^\star$ given dataset $\mathcal{D}$ which maximizes the long-term cumulative rewards, $\pi^\star = \operatorname{argmax}_\pi \mathbb{E}_{s \sim \rho}\left[V^\pi(s)\right]$, where ρ is the initial state distribution.

4 Introduction and evaluation of Q_{sarsa}

In this work, we propose to use an estimate of the Q-function of the behavior policy ($Q^{\pi_{\mathsf{b}}}$) for algorithm design. Since the investigation of exploiting $Q^{\pi_{\mathsf{b}}}$ or its estimate in offline RL methods is quite limited in the literature, we first

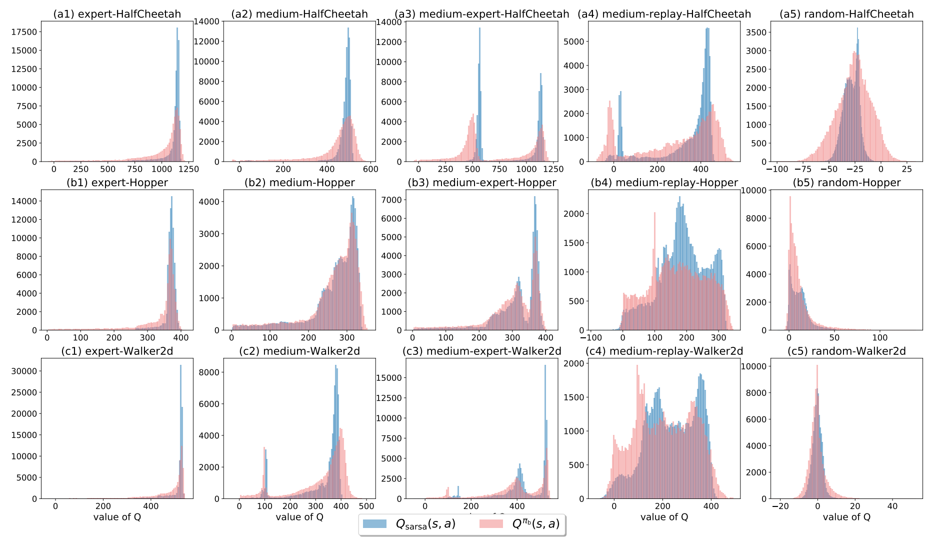

Fig. 3. The demonstration of Q_{sarsa} and $Q^{\pi_{\mathsf{b}}}$ over 15 tasks of MuJoCo in D4RL [9] with different tasks and diverse offline datasets. It shows that Q_{sarsa} estimate the value of $Q^{\pi_{\mathsf{b}}}$ reasonably regarding that their distribution largely overlap with each other (red and blue histogorams).

specify the form of the SARSA-style Q-estimate and then evaluate its reliability, characteristics, and possible benefits for prevalent offline algorithms.

The SARSA-style Q-estimate Q_{sarsa}. We estimate $Q^{\pi_{\mathsf{b}}}$ by the following SARSA-style optimization problem [3] and denote the output as Q_{sarsa}:

$$Q_{\mathsf{sarsa}} \leftarrow \min_{\theta_s} \mathbb{E}_{\mathcal{D}}[(r(s,a) + \gamma Q_{\bar{\theta}_s}(s',a') - Q_{\theta_s}(s,a))^2], \tag{1}$$

where θ_s (resp. $\overline{\theta}_s$) denotes the parameters of some neural network (resp. target network, a lagging version of θ_s), the expectation is w.r.t. $(s,a,r,s',a') \sim \mathcal{D}$, and $Q_{\mathsf{sarsa}} := Q_{\theta_s}$. Note that Q_{sarsa} is estimated by *in-sample* learning, which only queries the samples inside the history dataset, without involving any OOD regions.

4.1 Evaluation of Q_{sarsa}

An ablation for OnestepRL. First, to verify the benefit of using Q_{sarsa} directly as the critic in the actor-critic framework, proposed in OnestepRL [3], we conduct an ablation study by replacing the critic Q_{sarsa} by the critic trained by vanilla temporal difference learning widely used in online RL (i.e., CRR [33]) and show the comparisons in Fig. 2. Fig. 2 show the ablation study results over 15 MuJoCo tasks which include the main portion of the tasks considered in OnestepRL [3]. Note that to ensure a fair comparison, the actor component is the same for both methods — advantage-weighted regression/exponentially weighted imitation [3,33]. Observing that the two methods perform almost the same in Fig. 2, it indicates that the key to the success of OnestepRL may be the objective function

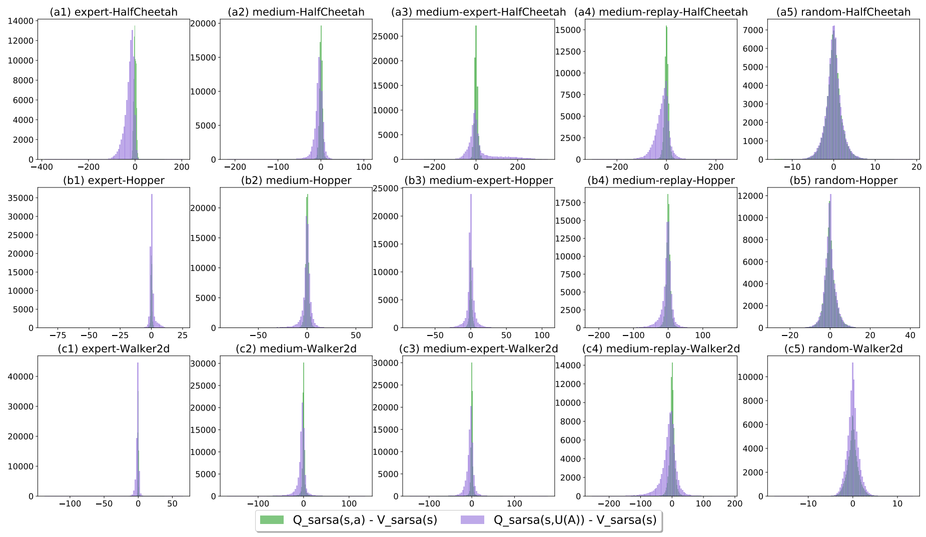

Fig. 4. The illustration of the histograms of the advantage $Q_{\mathsf{sarsa}}(s,a) - V_{\mathsf{sarsa}}(s)$ for *in-sample* state-action pairs $(s,a) \in \mathcal{D}$ and also the advantage $Q_{\mathsf{sarsa}}(s,U(A)) - V_{\mathsf{sarsa}}(s)$ for OOD state-action pairs $(s,U(A)) \notin \mathcal{D}$ over 15 tasks of MuJoCo in D4RL [9]. It indicates that Q_{sarsa} usually xihas no/small OOD error, since Q_{sarsa} over OOD state-action (purple histograms) won't exceed the value of in-distribution Q_{sarsa} (green histograms) too often or too much.

of the actor which restricts to *in-sample* learning but not Q_{sarsa}, since replacing Q_{sarsa} does not impact the performance.

Q_{sarsa} estimates $Q^{\pi_{\mathsf{b}}}$ reasonably well for $(s,a) \in \mathcal{D}$. The goal of Q_{sarsa} is to estimate the ground truth $Q^{\pi_{\mathsf{b}}}$. With this in mind, taking the MuJoCo tasks for instance, we provide the histograms of $Q_{\mathsf{sarsa}}(s,a)$ (in blue) and also $Q^{\pi_{\mathsf{b}}}(s,a)$ estimated by reward-to-go[3] (in pink) in Fig. 3 for $(s,a) \in \mathcal{D}$ in different levels of history datasets. It shows that the distribution of the value of $Q_{\mathsf{sarsa}}(s,a)$ almost coincide with $Q^{\pi_{\mathsf{b}}}(s,a)$, indicating that Q_{sarsa} estimates $Q^{\pi_{\mathsf{b}}}$ accurately to some degree. The slight difference between Q_{sarsa} (blue histograms) and $Q^{\pi_{\mathsf{b}}}$ (red histograms) might result from that Q_{sarsa} is estimating the Q-value of the behavior policy — may be a mixing of a set of policies (such as in the *medium-replay* case), while each point of the red histograms from $Q^{\pi_{\mathsf{b}}}(s,a)$ is actually the Q-value of one specific policy from the policy set but not the mixing policy (behavior policy).

In addition, the evaluation of Q_{sarsa} in Fig. 3 also provides useful intuition for the distribution of the dataset. For instance, the distribution of the *expert* datasets in Fig. 3(a1)-(c1) concentrate around the region with larger value/Q-function value since they are generated by some expert policy, while the *medium-replay*

[3] Note that $Q^{\pi_{\mathsf{b}}}$ is unknown. So we utilize the reward-to-go function [19] starting from any state-action pair $(s,a) \in \mathcal{D}$ as $Q^{\pi_{\mathsf{b}}}(s,a)$, i.e., $Q^{\pi_{\mathsf{b}}}(s,a) := \sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'})$ with $(s_t, a_t) = (s,a)$. The estimation can be filled by the trajectories in the entire dataset $\mathcal{D}$ with simple Monte Carlo return estimate, the same as the estimation of the value function used by [28].

datasets in Fig. 3(a4)-(c4) have more diverse data over the entire space of the tasks (with more diverse Q-function values) due to they being sampled from some replay buffers generated by different policies.

Q_{sarsa} has controllable out-of-distribution (OOD) error. Before continuing, we first introduce a value function $V_{\mathsf{sarsa}} : \mathcal{S} \to \mathbb{R}$, where $V_{\mathsf{sarsa}}(s) := \mathbb{E}_{a \in \pi_{\mathsf{b}}(s)} \left[Q_{\mathsf{sarsa}}(s, a) \right]$ for all $s \in \mathcal{S}$, learned by minimizing the following problem $\min_V \mathbb{E}_{(s,a) \in \mathcal{D}} \left[|V(s) - Q_{\mathsf{sarsa}}(s, a)|^2 \right]$. As mentioned in the introduction (Sec. 1), extrapolation error of the Q-value estimation, especially the overestimation over the out-of-distribution (OOD) state-action pairs is the essential challenge for offline RL problems. To evaluate if Q_{sarsa} overestimate the Q-value in OOD regions, in Fig. 4, we demonstrate the deviations from $Q_{\mathsf{sarsa}}(s, a)$ to the expectation of $Q_{\mathsf{sarsa}}(s, a)$ ($V_{\mathsf{sarsa}}(s)$) for state action pairs (s, a) over *in-distribution* region $\mathcal{D}$ (green histograms) and also OOD region ($(s, U(A))$ in purple histograms). In particular, $U(A)$ denotes the uniform distribution over the action space $\mathcal{A}$.

Fig. 4 indicates that $Q_{\mathsf{sarsa}}(s, a)$ **has controllable OOD error**. Specifically, the relative $Q_{\mathsf{sarsa}}(s, a) - V_{\mathsf{sarsa}}(s)$ value over OOD regions (purple histograms) generally has similar or smaller estimation value with comparisons to the maximum advantages over *in-sample* regions, i.e., $\max_{(s,a) \in \mathcal{D}} \left(Q_{\mathsf{sarsa}}(s, a) - V_{\mathsf{sarsa}}(s) \right)$ (green histograms). It implies that Q_{sarsa} usually has no/only slight overestimation over OOD regions and probably has some implicit pessimistic regularization effects.

5 Offline RL with Q-sarsa (Qsarsa-AC)

Inspired by the potential of Q_{sarsa} shown in Sec. 4, we propose a new algorithm called **Qsarsa-AC** by taking advantage of Q_{sarsa} and designing regularizations based on it. To directly and clearly evaluate our changes around Q_{sarsa}, **Qsarsa-AC** is prototyped against one of the state-of-the-art offline methods — TD3-BC [10] due to the simplicity of its algorithmic design. Specifically, among current offline state-of-the-art methods, we observe that TD3-BC has the minimal adjustments — by adding a BC term in the actor — compared to the existing off-policy method (used in online RL) [11]. As a result, it becomes much more straightforward and clear to evaluate the additional benefits, if any, indeed stem from our regularizations assisted by Q_{sarsa}, rather than other components of the framework.

Before proceeding, we introduce several useful notations. We denote the parameter of two critic networks (resp. target critic networks) as $\{\theta_i\}_{i \in \{0,1\}}$ (resp. $\{\overline{\theta}_i\}_{i \in \{0,1\}}$). Similarly, ϕ (resp. $\overline{\phi}$) represents the parameter of the policy (actor) network (resp. policy target network).

5.1 Critic with Q_{sarsa}

Before introducing our method, we recall the optimization problem of the two critics in prior arts TD3/TD3-BC [11,10], i.e., minimizing the temporal difference

(TD) error: with $y(s_j, a_j) = r(s_j, a_j) + \gamma \min_{i\in\{1,2\}} Q_{\overline{\theta}_i}(s'_j, \widetilde{a}'_j)$,

$$\theta_i \leftarrow \min \underbrace{B^{-1} \sum_{\mathcal{B}} |y(s_j, a_j) - Q_{\theta_i}(s_j, a_j)|^2,}_{\text{TD-error}} \quad i \in \{1, 2\}, \tag{2}$$

where $\mathcal{B} = \{s_j, a_j, r_j, s'_j, a'_j\}_{i=1}^{B}$ is a batch of data sampled from $\mathcal{D}$ with size B, $\widetilde{a}'_j$ is the perturbed action [11,10] chosen by current target policy given state s'_j, i.e., $\widetilde{a}'_j \approx \pi_{\overline{\phi}}(s'_j)$.

With the above in mind, we propose the following critic object of **Qsarsa-AC** by adding two natural regularization terms based on Q_{sarsa}: for $i \in \{1, 2\}$,

$$\theta_i \leftarrow \min \text{TD-error} + \lambda B^{-1} \sum_{\mathcal{B}} \Big\{ \underbrace{\big[Q_{\theta_i}(s_j, a_j) - Q_{\mathsf{sarsa}}(s_j, a_j)\big]^2}_{\textsf{InD-sarsa}}$$
$$+ \underbrace{w(s'_j, \widetilde{a}'_j) \cdot \big[Q_{\theta_i}(s'_j, \widetilde{a}'_j) - Q_{\text{sarsa}}(s'_j, \widetilde{a}'_j)\big]^2}_{\textsf{OOD-sarsa}} \Big\}, \tag{3}$$

where λ is a constant factor and $w(\cdot)$ is a mask function to be specified shortly.

Here, we choose the classic mean-squared error for both InD-sarsa and OOD-sarsa regularization terms. We introduce the role and intuition of the additional two regularizations separately. **(i) InD-sarsa: in-distribution regularization.** We highlight that this term regularizes the online learning Q-function (critic) Q_{θ_i} towards Q_{sarsa} only counting on *in-sample* state-action pairs (s_j, a_j) from the offline dataset (i.e., $(s_j, a_j) \in \mathcal{D}$). Recalling that for all $(s_j, a_j) \in \mathcal{D}$, $Q_{\text{sarsa}}(s_j, a_j) \approx Q^{\pi_{\mathsf{b}}}(s_j, a_j)$ (see Fig. 3) is a reasonable target, this *in-sample* regularization plays the role of pushing Q_{θ_i} to $Q^{\pi_{\mathsf{b}}}$ to avoid the overestimation of Q_{θ_i}. Note that this regularization is unlikely to bring in OOD errors for Q_{θ_i} since it only acts on state-action pairs inside the dataset ($(s_j, a_j) \in \mathcal{D}$). **(ii) OOD-sarsa: out-of-distribution regularization.** In contrast to InD-sarsa, this term pushes the Q-function Q_{θ_i} towards Q_{sarsa} by acting on OOD regions (i.e., $(s'_j, \widetilde{a}'_j)$ perhaps not in $\mathcal{D}$). It is used to restrict/reduce the overestimation error of Q_{θ_i} over the OOD regions in order to circumvent the extrapolation error challenges.

Specifically, recall that the bootstrapping term $Q_{\overline{\theta}_i}(s'_j, \widetilde{a}'_j)$ in (2) plays an essential role in estimating the Q-value, which has potentially catastrophic extrapolation error. The reason is that the state-action pair $(s'_j, \widetilde{a}'_j)$ may appear scarcely or be unseen in the offline dataset, yielding large OOD errors since $Q_{\theta_i}(s'_j, \widetilde{a}'_j)$ and/or $Q_{\overline{\theta}_i}(s'_j, \widetilde{a}'_j)$ may not be sufficiently optimized during training. To address this, OOD-sarsa directly regularizes $Q_{\theta_i}(s'_j, \widetilde{a}'_j)$ towards a more stable Q-function $Q_{\mathsf{sarsa}}(s'_j, \widetilde{a}'_j)$, which generally does not incur large OOD errors (see Sec. 4), to restrict the overestimation error of the bootstrapping term. Last but not least, we specify the choice of $w(\cdot, \cdot)$, which is a hard mask to prevent the regularization term from being contaminated by extra OOD errors. Specifically, we remove *bad state-action pairs* in case $Q_{\mathsf{sarsa}}(s'_j, \widetilde{a}'_j)$ has relatively large OOD error: $w(\cdot, \cdot)$ is set as

$$w(s'_j, \widetilde{a}'_j) = \mathbb{1}\Big(Q_{\mathsf{sarsa}}(s'_j, \widetilde{a}'_j) > V_{\mathsf{sarsa}}(s'_j) - |Q_{\mathsf{sarsa}}(s'_j, a'_j) - V_{\mathsf{sarsa}}(s'_j)|\Big). \tag{4}$$

Table 1. The normalized score (see Appendix A) of the final 10 evaluations over 5 random seeds. The highest performing scores are highlighted as pink and ± captures the standard deviation over seeds. For all the baselines, we re-run the implementations in the Acme framework [18] to keep an identical evaluation process for all methods. It shows that our two methods **Qsarsa-AC** and **Qsarsa-AC2** achieve better performance over the 15 tasks of MuJoCo benchmarks in D4RL [9].

		BC	**CRR**	**OnestepRL**	**CQL**	**TD3-BC**	**Qsarsa-AC** (Ours)	**Qsarsa-AC2** (Ours)
Random	HalfCheetah	2.3 ± 0.0	2.2 ± 0.1	2.3 ± 0.0	25.6 ± 1.8	6.0 ± 1.6	30.2 ± 1.3	30.1 ± 1.3
	Hopper	4.1 ± 1.9	3.9 ± 1.3	3.6 ± 1.0	8.7 ± 1.3	12.7 ± 9.3	7.5 ± 0.9	8.1 ± 1.7
	Walker2d	1.3 ± 0.2	1.4 ± 0.2	1.6 ± 0.3	0.3 ± 0.5	5.4 ± 6.3	5.9 ± 8.1	4.5 ± 4.9
Medium Replay	HalfCheetah	34.4 ± 3.1	37.3 ± 2.1	38.1 ± 2.1	-2.4 ± 1.9	42.5 ± 0.7	42.2 ± 0.7	54.4 ± 1.8
	Hopper	30.0 ± 9.5	29.9 ± 7.8	43.2 ± 9.2	93.0 ± 2.4	39.3 ± 8.4	92.6 ± 10.2	68.1 ± 19.7
	Walker2d	21.3 ± 17.1	21.1 ± 12.2	28.6 ± 12.3	13.9 ± 13.0	67.6 ± 6.4	83.7 ± 9.0	80.0 ± 11.4
Medium	HalfCheetah	42.5 ± 2.0	42.7 ± 0.7	42.8 ± 0.7	48.7 ± 0.4	45.4 ± 0.4	44.1 ± 0.5	56.4 ± 0.8
	Hopper	41.1 ± 10.2	43.3 ± 2.0	45.5 ± 3.2	55.6 ± 6.2	46.0 ± 3.3	78.6 ± 15.7	66.6 ± 24.3
	Walker2d	67.4 ± 12.6	65.6 ± 11.5	70.7 ± 10.1	76.6 ± 6.6	82.5 ± 0.9	79.0 ± 3.1	83.3 ± 3.5
Medium Expert	HalfCheetah	50.2 ± 6.9	49.6 ± 8.2	47.9 ± 6.3	68.4 ± 16.0	91.8 ± 1.4	91.3 ± 0.9	90.9 ± 1.0
	Hopper	48.7 ± 6.8	47.5 ± 3.2	45.6 ± 4.3	92.7 ± 16.1	83.6 ± 17.6	56.9 ± 21.3	73.6 ± 25.5
	Walker2d	98.0 ± 14.3	105.6 ± 2.6	105.5 ± 3.4	106.2 ± 1.4	106.4 ± 5.2	105.2 ± 6.1	107.5 ± 1.2
Expert	HalfCheetah	91.3 ± 2.4	93.9 ± 0.8	93.8 ± 1.2	72.6 ± 37.7	94.6 ± 0.4	93.8 ± 1.0	93.7 ± 1.0
	Hopper	105.3 ± 6.6	107.3 ± 4.9	105.9 ± 5.9	75.6 ± 26.9	104.2 ± 5.5	95.0 ± 21.1	97.8 ± 17.0
	Walker2d	107.5 ± 0.3	107.2 ± 0.6	107.3 ± 0.3	106.0 ± 5.0	108.1 ± 0.3	107.0 ± 0.6	107.0 ± 0.8
	Total	745.3 ± 94.1	758.5 ± 58.1	782.4 ± 60.3	841.5 ± 137.2	936.2 ± 67.8	1013.2 ± 100.3	1021.8 ± 116.0

5.2 Actor with Q_{sarsa}

Recall the ideal optimization problem for the learning policy (actor) in TD3-BC [10]: $\max_\pi \mathbb{E}_{\mathcal{D}}[\frac{\alpha}{\mathbb{E}_{\mathcal{D}}[Q(s,a)]} \cdot Q(s,\pi(s)) - (\pi(s)-a)^2]$, which directly adds a behavior cloning (BC) regularization towards the distribution of the dataset with a universal weight α for any offline dataset and any state-action pair. In this work, armed with Q_{sarsa}, we propose the following optimization problem referring to Q_{sarsa}-determined point-wise weights $f(Q_{\mathsf{sarsa}}, s, a)$ for BC instead of the fixed universal α:

$$\max_\pi \mathbb{E}_{\mathcal{D}}\left[\frac{Q(s,\pi(s))}{\mathbb{E}_{\mathcal{D}}[Q(s,a)]} - f(Q_{\mathsf{sarsa}}, s, a)(\pi(s)-a)^2\right], \tag{5}$$

where $f(Q_{\mathsf{sarsa}}, s, a) := p_{s,a}(Q_{\mathsf{sarsa}})/g(Q_{\mathsf{sarsa}})$ is constructed by two terms given below. **(i) Global weight $g(Q_{\mathsf{sarsa}})$ for BC.** $g(Q_{\mathsf{sarsa}})$ serves as a global quantification of the dataset quality and determines the foundation of the weights on BC (keeping the same for all $(s,a) \in \mathcal{D}$). Intuitively, when the dataset is generated by some high-quality policies (e.g., the *expert* dataset), we are supposed to imitate the policy and put more weights on the BC regularization (bigger $1/g(Q_{\mathsf{sarsa}})$), otherwise smaller weights on BC. Towards this, we use $Q_{\mathsf{sarsa}}^{\mathsf{mean}} := \left|\mathbb{E}_{\mathcal{D}}[Q_{\mathsf{sarsa}}(s,a)]\right|$ as a global quantification for the behavior policy, which leads to

$$g(Q_{\mathsf{sarsa}}) = \mathrm{Clip}\left[\alpha \exp\left(\frac{\tau_1}{Q_{\mathsf{sarsa}}^{\mathsf{mean}}}\right), (1, 10^6)\right]. \tag{6}$$

Here, the clipping function Clip$(\cdot)$ is just to normalize $g(Q_{\mathsf{sarsa}})$ between 1 and a huge number (e.g., 10^6) to forbid it from being too big, which can cause numerical problems. **(ii) Point-wise weight $p_{s,a}(Q_{\mathsf{sarsa}})$ for BC.** $p_{s,a}(Q_{\mathsf{sarsa}}) \in [0,1]$ is typically a point-wise normalized weight for different (s,a) pairs, formed as

$$p_{s,a}(Q_{\mathsf{sarsa}}) = \max\left(\exp\left(\frac{\tau_2\left(Q_{\mathsf{sarsa}}(s,a) - V_{\mathsf{sarsa}}(s)\right)}{\left|Q_{\mathsf{sarsa}}(s,a)\right|}\right), 1\right). \tag{7}$$

In particular, $p_{s,a}(Q_{\mathsf{sarsa}})$ puts larger weights on BC for *high-quality* state-action pair (s,a) which deserves the learning policy to visit, otherwise reduces the weights for BC regularization. The quality of the state-action pairs is determined by the advantage $Q_{\mathsf{sarsa}}(s,a) - V_{\mathsf{sarsa}}(s)$ normalized by $Q_{\mathsf{sarsa}}(s,a)$.

5.3 A variant Qsarsa-AC2

Given that the global weight $g(Q_{\mathsf{sarsa}})$ aims to evaluate the quality of the given dataset over some specific task, it is supposed to depend on the relative value of $Q^{\pi_{\mathsf{b}}}$ w.r.t. the optimal Q-function $Q^\star$ over this task. However, $g(Q_{\mathsf{sarsa}})$ in (6) is calculated by the absolute value of $Q_{\mathsf{sarsa}} \approx Q^{\pi_{\mathsf{b}}}$ without considering that $Q^\star$ may vary in different tasks (for example, $Q^\star$ of hopper or walker2d are different). So supposing that we have access to $\max_{s,a} Q^\star(s,a)$ for different tasks (can be approximated by the maximum of the reward function), we propose **Qsarsa-AC2** as a variant of **Qsarsa-AC** which only has a different form of $g(Q_{\mathsf{sarsa}})$ compared to **Qsarsa-AC** as follows:

$$g(Q_{\mathsf{sarsa}}) = \text{Clip}\left[\alpha \exp\left(\frac{\tau_3 \max_{s,a} Q^\star(s,a)}{Q_{\mathsf{sarsa}}^{\mathsf{mean}}}\right), (1, 10^6)\right], \tag{8}$$

where $\max_{s,a} Q^\star(s,a)$ is estimated by $\frac{\max_{s,a} r_{\text{expert}}(s,a)}{(1-\gamma)}$ with $r_{\text{expert}}(s,a)$, the reward data from the *expert* dataset.

6 Experimental evaluation

We first introduce the evaluation of our methods with comparisons to state-of-the-art baselines over the D4RL MuJoCo benchmarks [9], followed by ablation experiments to offer a more detailed analysis for the components based on Q_{sarsa}.

6.1 Settings and baselines

Experimental settings. To evaluate the performance of the proposed methods, we conduct experiments on the D4RL benchmark of OpenAI gym MuJoCo tasks [9], constructed with various domains and dataset quality. Specifically, we conduct experiments on the recently released '-v2' version for MuJoCo in D4RL consisting of 5 different levels of offline datasets (*random*, *medium-replay*, *medium*,

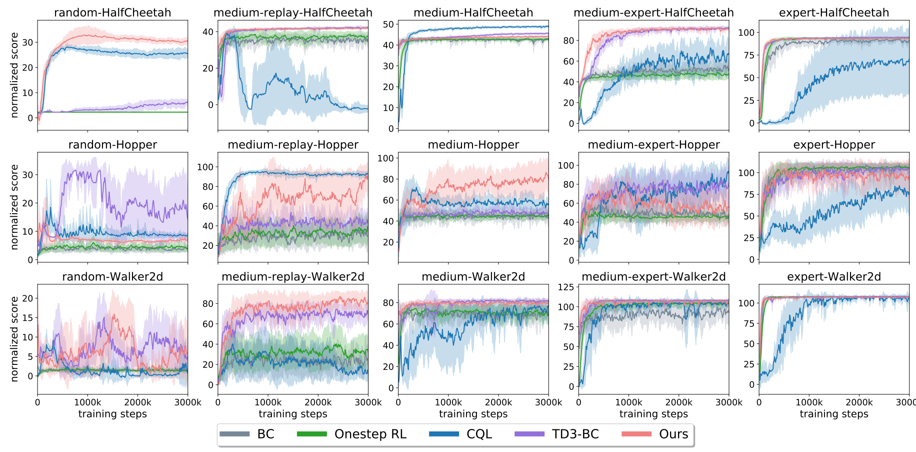

Fig. 5. Normalized scores (see Appendix A) of the evaluations during the training process (5 seeds).

medium-expert, and *expert*) over 3 different environments, in total 15 tasks. All the baselines and our methods are trained for 3M steps.

Baselines. Besides behavior cloning (BC), we compare our performance to several state-of-the-art offline methods, namely CRR [33], OnestepRL [3], CQL [23], and TD3-BC [10]. We highlight that OnestepRL and TD3-BC are the most related baselines: 1) OnestepRL is the only prior work that exploit Q_{sarsa} in offline methods by directly setting Q_{sarsa} as the critic, whereas we adopt slight regularizations with the aid of Q_{sarsa}; 2) our methods are designed with TD3-BC as the prototype with additional Q_{sarsa}-assisted regularizations. For all the baselines, we use the implementations from the Acme framework [18] which maintains the designing details of the original papers with tuned hyper-parameters.

Implementations and hyperparameters. Recall that the core of the proposed methods is Q_{sarsa} — an estimate of Q^{π_b} — can be learned from (1). Hence, the training of our methods is divided into two phases: 1) learning Q_{sarsa} by solving (1) for 1M steps; 2) following the learning process of TD3-BC with a fixed Q_{sarsa} for 3M training steps.

The hyperparameters of our methods defined in Sec. 5 are tuned in a small finite set using 5 random seeds that are different from those used in the final results reported in Table 1. In particular, we use $\lambda = 0.03$ for the critic and $\alpha = 10^{-4}$, $\tau_1 = 3000, \tau_2 = 4, \tau_3 = 4.5$ for the actor. We remark that τ_1 is chosen according to the range of the value of Q_{sarsa} as $\tau_1 \approx 2 \cdot \max_{s,a} Q_{\text{sarsa}}(s, a)$ in MuJoCo. For the critic, we tune λ in (3) within the set $\{0.01, 0.03, 0.05\}$. For the actor, τ_2 is tuned inside $\{0.4, 2, 4\}$, and τ_1 and τ_3 are tuned across $\{2500, 3000, 3500\}$ and $\{4, 4.5, 5\}$ with a fixed $\alpha = 10^{-4}$.

6.2 Results

Evaluation of Q_{sarsa}. We evaluate our two methods **Qsarsa-AC** and **Qsarsa-AC2** with comparison to the baselines over the 15 tasks using 5 random seeds; the

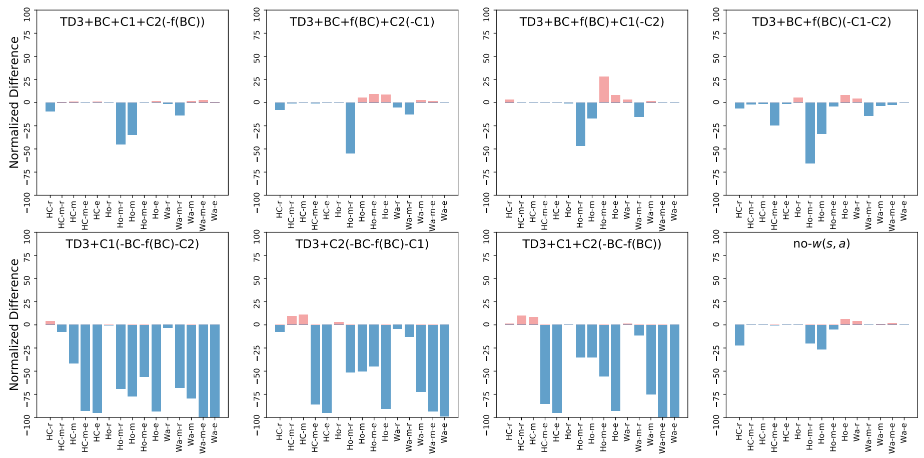

Fig. 6. The ablation study of **Qsarsa-AC** (denoted as TD3 + BC + f(BC) + C1 + C2), where f(BC) means the weight $f(Q_{\mathsf{sarsa}}, s, a)$ for BC in (5), and C1 (resp. C2) denotes the 1^{st} (resp. 2^{nd}) regularization term designed for the critic in (3). Here, the *Normalized Difference* is calculated as the difference of the normalized score (see Appendix A) between one ablation and the full algorithm **Qsarsa-AC** (the mean score of 10 evaluations over 5 seeds).

results are reported in Table 1. There are three key observations. **(i)** Q_{sarsa} **brings benefits to existing offline methods.** Recall that our proposed methods are built on the framework of TD3-BC. The last 3 columns of Table 1 illustrate the comparisons between our two methods and TD3-BC. It indicates that the proposed **Qsarsa-AC** and **Qsarsa-AC2** methods, which take advantage of the information of Q_{sarsa} (see Sec. 5), can bring additional benefits to the existing offline framework TD3-BC and outperform all the conducted baselines. It is also promising to integrate Q_{sarsa} to other offline methods using approximate dynamic programming framework such as SAC [16] and IQL [21]. We also show that our methods has competitive results against some additional state-of-the-art methods (such as IQL and DT) in Appendix Table 2 since these baselines are not closely related. **(ii) Directly setting** Q_{sarsa} **as the critic has no benefits.** To evaluate whether Q_{sarsa} play an important role in the success of OnestepRL [3] more systematically, we also involve CRR as an ablation baseline to OnestepRL. As introduced in Sec. 4, we use variants of OnestepRL and CRR with the same actor loss (exponentially weighted imitation [3]), so that the only difference between OnestepRL and CRR is that OnestepRL uses Q_{sarsa} as the critic, while CRR uses the critic learned by TD error (usually used in online RL). The similar scores in Table. 1 and also the normalized training scores in Fig. 2 show that there is almost no difference between CRR and OnestepRL, indicating that the success of OnestepRL may be attributed to the actor loss. **(iii) Regularization strength for** Q_{sarsa}**.** Based on Table 1, it is noted that our methods achieve better performance with very small weights on the critic regularization terms ($\lambda = 0.03$ in (3)). Q_{sarsa} typically doesn't influence the actor widely since it only determines the weights of the BC term. This observation implies that some small regularizations based on the Q-function of π_{b} (Q_{sarsa}) may already achieve the goal of addressing the extrapolation and OOD error in offline RL.

Ablation study. We also perform an ablation study for the components of our method **Qsarsa-AC**, illustrated in Fig. 6. Recall that the main components of **Qsarsa-AC** include $f(Q_{\mathsf{sarsa}}, s, a)$ (denoted as f(BC)) for the actor and two regularizations for the critic — InD-sarsa (represented as C1) and OOD-sarsa (represented as C2). In Fig. 6, the first row shows that if we keep TD3-BC as the baseline, removing any of the three components based on Q_{sarsa}, especially removing the two critic regularizations together, leads to evident dropping of the performance. It indicates that the information of Q_{sarsa} inside the components brings significant benefits to the existing TD3-BC framework. We further evaluate the ablations of removing BC in the actor (as well as f(BC)), which lead to dramatic dropping of the performance, shown in the second row of Fig. 6. The performance drop is reasonable since the weights for the critic regularizations (C1 and C2) designed in (3) are tuned based on TD3-BC with BC in hand. Finally, the ablation of removing the mask $w(\cdot)$ inside the OOD-sarsa regularization in (3) implies that OOD error of Q_{sarsa} may happen and harm the performance without $w(\cdot)$, but not often.

7 Conclusion

We propose to integrate a SARSA-style estimate of the Q-function of the behavior policy into offline RL for better performance. Given the limited use of the Q-function of the behavior policy in the current literature, we first evaluate the SARSA-style Q-estimate to establish its reliability in estimating the Q-function and potential to restrict OOD errors. We then propose two methods by taking advantage of the SARSA-style Q-estimate based on TD3-BC, one of the offline state-of-the-art methods. Our proposed methods achieve strong performance in D4RL MuJoCo benchmarks and outperform the baselines. It is our hope to inspire future works that exploit the benefit of Q/value-functions of the behavior policy in more offline methods, such as using different regularization loss functions beyond ℓ_2, combining it with other regularization techniques, and in different schemes in the approximate dynamic programming framework, even sequential modeling.

A Appendix

The comparisons to additional baselines. Here, we provide the performance comparisons to two additional strong baselines DT [5] and IQL [21]. IQL is a well-known strong baseline using the same approximate dynamic programming framework as our methods, while DT resort to a different framework — sequential modeling. In Table A, we directly report the scores in the original paper of DT [5] and the reported scores for IQL in a prior work [27]. Table A shows that our methods can not only bring significant benefits for some existing offline methods (such as TD3-BC) but also achieve competitive performance with comparisons to those strong offline methods.

Table 2. This table displays the comparisons with two additional offline state-of-the-arts DT [5] and IQL [21] over the 15 tasks of MuJoCo benchmarks in D4RL [9].

		DT	**IQL**	**Qsarsa-AC** (Ours)	**Qsarsa-AC2** (Ours)
Random	HalfCheetah	—	13.1 ± 1.1	30.2 ± 1.3	30.1 ± 1.3
	Hopper	—	7.9 ± 0.2	7.5 ± 0.9	8.1 ± 1.7
	Walker2d	—	5.4 ± 1.2	5.9 ± 8.1	4.5 ± 4.9
Medium Replay	HalfCheetah	36.6 ± 0.8	44.2 ± 1.2	42.2 ± 0.7	54.4 ± 1.8
	Hopper	82.7 ± 7.0	94.7 ± 8.6	92.6 ± 10.2	68.1 ± 19.7
	Walker2d	66.6 ± 3.0	73.8 ± 7.1	83.7 ± 9.0	80.0 ± 11.4
Medium	HalfCheetah	42.6 ± 0.1	47.4 ± 0.2	44.1 ± 0.5	56.4 ± 0.8
	Hopper	67.6 ± 1.0	66.2 ± 5.7	78.6 ± 15.7	66.6 ± 24.3
	Walker2d	74.0 ± 1.4	78.3 ± 8.7	79.0 ± 3.1	83.3 ± 3.5
Medium Expert	HalfCheetah	86.8 ± 1.3	86.7 ± 5.3	91.3 ± 0.9	90.9 ± 1.0
	Hopper	107.6 ± 1.8	91.5 ± 14.3	56.9 ± 21.3	73.6 ± 25.5
	Walker2d	108.1 ± 0.2	110.1 ± 0.5	105.2 ± 6.1	107.5 ± 1.2
Expert	HalfCheetah	—	95.0 ± 0.5	93.8 ± 1.0	93.7 ± 1.0
	Hopper	—	109.4 ± 0.5	95.0 ± 21.1	97.8 ± 17.0
	Walker2d	—	109.0 ± 1.2	107.0 ± 0.6	107.0 ± 0.8
Total (without random & expert)		672.6	692.8	673.6	680.8
Total		—	1032.7	1013.2	1021.8

Performance metric With an output score of some method evaluated on MuJoCo in D4RL, we use the widely used *normalized performance score* as the performance metric: normalized score $:= \frac{\text{score}-\text{random score}}{\text{expert score}-\text{random score}} \cdot 100$, where the expert score (resp. random score) represents the performance of an expert policy (resp. a random policy). Here, noting that different offline datasets (such as *expert* dataset, *medium* dataset) for the same task enjoy a same expert/random score; for self-consistency, we recall the expert scores and random scores in different tasks in Table 3.

	expert score	**random score**
HalfCheetah	12135.0	−280.18
Hopper	3234.3	−20.27
Walker2d	4592.3	1.63

Table 3. The expert scores and random scores of different MuJoCo tasks in D4RL [9].

Auxiliary implementation details For our two methods **Qsarsa-AC** and **Qsarsa-AC2**, the models of the policies (actors) and the Q-functions (critics) are the same as the ones in TD3-BC [10]. The models for Q_{sarsa} (including the networks parameterized by θ_s and $\overline{\theta}_s$) are MLPs with ReLU activations and with 2 hidden layers of width 1024. The training of the Q_{sarsa} network parameterized by θ_s is completed in the first training phase using Adam with initial learning rate 10^{-4} and batch size as 512. The target of Q_{sarsa} is updated smoothly with $\tau = 0.005$, i.e., $\overline{\theta}_s \leftarrow (1-\tau)\overline{\theta}_s + \tau\theta_s$.

Acknowledgements Part of this work was completed when L. Shi was an intern at Google Research, Brain Team. The work of L. Shi and Y. Chi is supported in part by the grants NSF CCF-2106778 and CNS-2148212. L. Shi is also gratefully supported by the Leo Finzi Memorial Fellowship, Wei Shen and Xuehong Zhang Presidential Fellowship, and Liang Ji-Dian Graduate Fellowship at Carnegie Mellon University. The authors would like to thank Alexis Jacq for reviewing an early version of the paper. The authors would like to thank the anonymous reviewers for valuable feedback and suggestions. We would also like to thank the Python and RL community for useful tools that are widely used in this work, including Acme [18], Numpy [17], and JAX [2].

Ethical statement Offline RL methods may bring benefits for social application scenarios when collecting new data is infeasible due to cost, privacy or safety. For example, learning to diagnose from historical medical records or designing recommendations given existing clicking records of some advertisements. For negative social impact, offline methods may enable big data discriminatory pricing to yield unfair market or improve the recommendation techniques to make more people to be addicted to the social media. However, our proposed methods is more related to introducing scientific thoughts and investigations, which do not target such possible applications. Additionally, this work will only use public benchmarks and data, so no personal data will be acquired or inferred.

References

1. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: A brief survey of deep reinforcement learning. arXiv preprint arXiv:1708.05866 (2017)
2. Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., et al.: Jax: Autograd and xla. Astrophysics Source Code Library pp. ascl–2111 (2021)
3. Brandfonbrener, D., Whitney, W., Ranganath, R., Bruna, J.: Offline rl without off-policy evaluation. Advances in neural information processing systems **34**, 4933–4946 (2021)
4. Buckman, J., Gelada, C., Bellemare, M.G.: The importance of pessimism in fixed-dataset policy optimization. arXiv preprint arXiv:2009.06799 (2020)
5. Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., Mordatch, I.: Decision transformer: Reinforcement learning via sequence modeling. Advances in neural information processing systems **34**, 15084–15097 (2021)
6. Chen, X., Zhou, Z., Wang, Z., Wang, C., Wu, Y., Ross, K.: Bail: Best-action imitation learning for batch deep reinforcement learning. Advances in Neural Information Processing Systems **33**, 18353–18363 (2020)
7. Dadashi, R., Rezaeifar, S., Vieillard, N., Hussenot, L., Pietquin, O., Geist, M.: Offline reinforcement learning with pseudometric learning. In: International Conference on Machine Learning. pp. 2307–2318. PMLR (2021)
8. Fakoor, R., Mueller, J.W., Asadi, K., Chaudhari, P., Smola, A.J.: Continuous doubly constrained batch reinforcement learning. Advances in Neural Information Processing Systems **34**, 11260–11273 (2021)
9. Fu, J., Kumar, A., Nachum, O., Tucker, G., Levine, S.: D4rl: Datasets for deep data-driven reinforcement learning. arXiv preprint arXiv:2004.07219 (2020)
10. Fujimoto, S., Gu, S.S.: A minimalist approach to offline reinforcement learning. Advances in neural information processing systems **34**, 20132–20145 (2021)
11. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning. pp. 1587–1596. PMLR (2018)
12. Fujimoto, S., Meger, D., Precup, D.: Off-policy deep reinforcement learning without exploration. In: International Conference on Machine Learning. pp. 2052–2062. PMLR (2019)
13. Garg, D., Hejna, J., Geist, M., Ermon, S.: Extreme q-learning: Maxent rl without entropy. arXiv preprint arXiv:2301.02328 (2023)
14. Ghasemipour, S.K.S., Schuurmans, D., Gu, S.S.: Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In: International Conference on Machine Learning. pp. 3682–3691. PMLR (2021)
15. Gulcehre, C., Colmenarejo, S.G., Wang, Z., Sygnowski, J., Paine, T., Zolna, K., Chen, Y., Hoffman, M., Pascanu, R., de Freitas, N.: Regularized behavior value estimation. arXiv preprint arXiv:2103.09575 (2021)
16. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. pp. 1861–1870. PMLR (2018)
17. Harris, C.R., Millman, K.J., Van Der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., et al.: Array programming with numpy. Nature **585**(7825), 357–362 (2020)

18. Hoffman, M., Shahriari, B., Aslanides, J., Barth-Maron, G., Behbahani, F., Norman, T., Abdolmaleki, A., Cassirer, A., Yang, F., Baumli, K., et al.: Acme: A research framework for distributed reinforcement learning. arXiv preprint arXiv:2006.00979 (2020)
19. Janner, M., Li, Q., Levine, S.: Offline reinforcement learning as one big sequence modeling problem. Advances in neural information processing systems **34**, 1273–1286 (2021)
20. Kostrikov, I., Fergus, R., Tompson, J., Nachum, O.: Offline reinforcement learning with fisher divergence critic regularization. In: International Conference on Machine Learning. pp. 5774–5783. PMLR (2021)
21. Kostrikov, I., Nair, A., Levine, S.: Offline reinforcement learning with implicit q-learning. arXiv preprint arXiv:2110.06169 (2021)
22. Kumar, A., Fu, J., Soh, M., Tucker, G., Levine, S.: Stabilizing off-policy q-learning via bootstrapping error reduction. Advances in Neural Information Processing Systems **32** (2019)
23. Kumar, A., Zhou, A., Tucker, G., Levine, S.: Conservative q-learning for offline reinforcement learning. arXiv preprint arXiv:2006.04779 (2020)
24. Lee, B.J., Lee, J., Kim, K.E.: Representation balancing offline model-based reinforcement learning. In: International Conference on Learning Representations (2020)
25. Levine, S.: Reinforcement learning and control as probabilistic inference: Tutorial and review. arXiv preprint arXiv:1805.00909 (2018)
26. Levine, S., Kumar, A., Tucker, G., Fu, J.: Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643 (2020)
27. Lyu, J., Ma, X., Li, X., Lu, Z.: Mildly conservative q-learning for offline reinforcement learning. arXiv preprint arXiv:2206.04745 (2022)
28. Peng, X.B., Kumar, A., Zhang, G., Levine, S.: Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. arXiv preprint arXiv:1910.00177 (2019)
29. Rezaeifar, S., Dadashi, R., Vieillard, N., Hussenot, L., Bachem, O., Pietquin, O., Geist, M.: Offline reinforcement learning as anti-exploration. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 8106–8114 (2022)
30. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of go without human knowledge. nature **550**(7676), 354–359 (2017)
31. Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., et al.: Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (2019)
32. Wang, Z., Hunt, J.J., Zhou, M.: Diffusion policies as an expressive policy class for offline reinforcement learning. arXiv preprint arXiv:2208.06193 (2022)
33. Wang, Z., Novikov, A., Zolna, K., Springenberg, J.T., Reed, S., Shahriari, B., Siegel, N., Merel, J., Gulcehre, C., Heess, N., et al.: Critic regularized regression. arXiv preprint arXiv:2006.15134 (2020)
34. Wu, Y., Tucker, G., Nachum, O.: Behavior regularized offline reinforcement learning. arXiv preprint arXiv:1911.11361 (2019)
35. Yang, S., Wang, Z., Zheng, H., Feng, Y., Zhou, M.: A regularized implicit policy for offline reinforcement learning. arXiv preprint arXiv:2202.09673 (2022)
36. Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., Finn, C.: Combo: Conservative offline model-based policy optimization. Advances in neural information processing systems **34**, 28954–28967 (2021)

37. Zhang, G., Kashima, H.: Behavior estimation from multi-source data for offline reinforcement learning. arXiv preprint arXiv:2211.16078 (2022)