OFFLINE RL WITH NO OOD ACTIONS: IN-SAMPLE LEARNING VIA IMPLICIT VALUE REGULARIZATION

Haoran Xu[1]* **Li Jiang**[2] **Jianxiong Li**[1] **Zhuoran Yang**[3] **Zhaoran Wang**[4]
Victor Wai Kin Chan[2] **Xianyuan Zhan**[1][5]
[1]Institute for AI Industry Research (AIR), Tsinghua University
[2]Tsinghua-Berkeley Shenzhen Institute (TBSI), Tsinghua University
[3]Yale University [4]Northwestern University [5]Shanghai Artificial Intelligence Laboratory
{ryanxhr,jiangli3859,zhanxianyuan}@gmail.com

ABSTRACT

Most offline reinforcement learning (RL) methods suffer from the trade-off between improving the policy to surpass the behavior policy and constraining the policy to limit the deviation from the behavior policy as computing Q-values using out-of-distribution (OOD) actions will suffer from errors due to distributional shift. The recent proposed *In-sample Learning* paradigm (i.e., IQL), which improves the policy by quantile regression using only data samples, shows great promise because it learns an optimal policy without querying the value function of any unseen actions. However, it remains unclear how this type of method handles the distributional shift in learning the value function. In this work, we make a key finding that the in-sample learning paradigm arises under the *Implicit Value Regularization* (IVR) framework. This gives a deeper understanding of why the in-sample learning paradigm works, i.e., it applies implicit value regularization to the policy. Based on the IVR framework, we further propose two practical algorithms, Sparse Q-learning (SQL) and Exponential Q-learning (EQL), which adopt the same value regularization used in existing works, but in a complete in-sample manner. Compared with IQL, we find that our algorithms introduce sparsity in learning the value function, making them more robust in noisy data regimes. We also verify the effectiveness of SQL and EQL on D4RL benchmark datasets and show the benefits of in-sample learning by comparing them with CQL in small data regimes. Code is available at https://github.com/ryanxhr/IVR.

1 INTRODUCTION

Reinforcement learning (RL) is an increasingly important technology for developing highly capable AI systems, it has achieved great success in game-playing domains (Mnih et al., 2013; Silver et al., 2017). However, the fundamental online learning paradigm in RL is also one of the biggest obstacles to RL's widespread adoption, as interacting with the environment can be costly and dangerous in real-world settings. Offline RL, also known as batch RL, aims at solving the abovementioned problem by learning effective policies solely from offline data, without any additional online interactions. It is a promising area for bringing RL into real-world domains, such as robotics (Kalashnikov et al., 2021), healthcare (Tang & Wiens, 2021) and industrial control (Zhan et al., 2022). 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 entails querying the Q-function about values of actions produced by the policy, which are often not seen in the dataset. Those out-of-distribution actions can be deemed as adversarial examples of the Q-function, which cause extrapolation errors of the Q-function (Kumar et al., 2020). To alleviate this issue, prior model-free offline RL methods typically add pessimism to the learning objective, in order to be pessimistic about the distributional shift. Pessimism can be achieved by policy constraint, which constrains the policy to be close to the behavior policy (Kumar et al., 2019; Wu et al., 2019; Nair

[*]Work done while at JD Technology. Correspondence to Haoran Xu, Xianyuan Zhan.

et al., 2020; Fujimoto & Gu, 2021); or value regularization, which directly modifies the Q-function to be pessimistic (Kumar et al., 2020; Kostrikov et al., 2021a; An et al., 2021; Bai et al., 2021). Nevertheless, this imposes a trade-off between accurate value estimation (more regularization) and maximum policy performance (less regularization).

In this work, we find that we could alleviate the trade-off in *out-of-sample learning* by performing *implicit value regularization*, this bypasses querying the value function of any unseen actions, allows learning an optimal policy using *in-sample learning*[*]. More specifically, we propose the Implicit Value Regulazization (IVR) framework, in which a general form of behavior regularizers is added to the policy learning objective. Because of the regularization, the optimal policy in the IVR framework has a closed-form solution, which can be expressed by imposing weight on the behavior policy. The weight can be computed by a state-value function and an action-value function, the state-value function serves as a normalization term to make the optimal policy integrate to 1. It is usually intractable to find a closed form of the state-value function, however, we make a subtle mathematical transformation and show its equivalence to solving a convex optimization problem. In this manner, both of these two value functions can be learned by only dataset samples.

Note that the recently proposed method, IQL (Kostrikov et al., 2021b), although derived from a different view (i.e., approximate an upper expectile of dataset actions given a state), remains pretty close to the learning paradigm of our framework. Furthermore, our IVR framework explains why learning the state-value function is important in IQL and gives a deeper understanding of how IQL handles the distributional shift: it is doing implicit value regularization, with the hyperparameter τ to control the strength. This explains one disturbing issue of IQL, i.e., the role of τ does not have a perfect match between theory and practice. In theory, τ should be close to 1 to obtain an optimal policy while in practice a larger τ may give a worse result.

Based on the IVR framework, we further propose some practical algorithms. We find that the value regularization terms used in CQL (Kumar et al., 2020) and AWR (Peng et al., 2019) are two valid choices in our framework. However, when applying them to our framework, we get two complete in-sample learning algorithms. The resulting algorithms also bear similarities to IQL. However, we find that our algorithm introduces sparsity in learning the state-value function, which is missing in IQL. The sparsity term filters out those bad actions whose Q-values are below a threshold, which brings benefits when the quality of offline datasets is inferior. We verify the effectiveness of SQL on widely-used D4RL benchmark datasets and demonstrate the state-of-the-art performance, especially on suboptimal datasets in which value learning is necessary (e.g., AntMaze and Kitchen). We also show the benefits of sparsity in our algorithms by comparing with IQL in noisy data regimes and the robustness of in-sample learning by comparing with CQL in small data regimes.

To summarize, the contributions of this paper are as follows:

- We propose a general implicit value regularization framework, where different behavior regularizers can be included, all leading to a complete in-sample learning paradigm.
- Based on the proposed framework, we design two effective offline RL algorithms: Sparse Q-Learning (SQL) and Exponential Q-learning (EQL), which obtain SOTA results on benchmark datasets and show robustness in both noisy and small data regimes.

2 Related Work

To tackle the distributional shift problem, most model-free offline RL methods augment existing off-policy methods (e.g., Q-learning or actor-critic) with a behavior regularization term. Behavior regularization can appear explicitly as divergence penalties (Wu et al., 2019; Kumar et al., 2019; Fujimoto & Gu, 2021), implicitly through weighted behavior cloning (Wang et al., 2020; Nair et al., 2020), or more directly through careful parameterization of the policy (Fujimoto et al., 2018; Zhou et al., 2020). Another way to apply behavior regularization 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 (Nachum et al., 2019; Kumar et al., 2020; Kostrikov et al., 2021a; Xu et al., 2022c). There are also several works incorporating

[*] The core difference between in-sample learning and out-of-sample learning is that in-sample learning uses only dataset actions to learn the value function while out-of-sample learning uses actions produced by the policy.

behavior regularization through the use of uncertainty (Wu et al., 2021; An et al., 2021; Bai et al., 2021) or distance function (Li et al., 2023b).

However, in-distribution constraints used in these works might not be sufficient to avoid value function extrapolation errors. Another line of methods, on the contrary, avoid value function extrapolation by performing some kind of imitation learning on the dataset. When the dataset is good enough or contains high-performing trajectories, we can simply clone or filter dataset actions to extract useful transitions (Xu et al., 2022b; Chen et al., 2020), or directly filter individual transitions based on how advantageous they could be under the behavior policy and then clones them Brandfonbrener et al. (2021); Xu et al. (2021; 2022a). While alleviating extrapolation errors, these methods only perform single-step dynamic programming and lose the ability to "stitch" suboptimal trajectories by multi-step dynamic programming.

Our method can be viewed as a combination of these two methods while sharing the best of both worlds: SQL and EQL implicitly control the distributional shift and learns an optimal policy by in-sample generalization. SQL and EQL are less vulnerable to erroneous value estimation as in-sample actions induce less distributional shift than out-of-sample actions. Similar to our work, IQL approximates the optimum in-support policy by fitting the upper expectile of the behavior policy's action-value function, however, it is not motivated by remaining pessimistic to the distributional shift.

Our method adds a behavior regularization term to the RL learning objective. In online RL, there are also some works incorporating an entropy-regularized term into the learning objective (Haarnoja et al., 2018; Nachum et al., 2017; Lee et al., 2019; Neu et al., 2017; Geist et al., 2019; Ahmed et al., 2019), this brings multi-modality to the policy and is beneficial for the exploration. Note that the entropy-regularized term only involves the policy, it could be directly computed, resulting in a similar learning procedure as in SAC (Haarnoja et al., 2018). While our method considers the offline setting and provides a different learning procedure to solve the problem by jointly learning a state-value function and an action-value function.

3 PRELIMINARIES

We consider the RL problem presented as a Markov Decision Process (MDP) (Sutton et al., 1998), which is specified by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, r, \rho, \gamma \rangle$ consisting of a state space, an action space, a transition probability function, a reward function, an initial state distribution, and the discount factor. The goal of RL is to find a policy $\pi(a|s) : \mathcal{S} \times \mathcal{A} \to [0, 1]$ that maximizes the expected discounted cumulative 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. Unlike online RL methods, offline RL aims to learn an optimal policy from a fixed dataset $\mathcal{D}$ consisting of trajectories that are collected by different policies. The dataset can be heterogenous and suboptimal, we denote the underlying behavior policy of $\mathcal{D}$ as μ, which represents the conditional distribution $p(a|s)$ observed in the dataset.

RL methods based on approximate dynamic programming (both online and offline) typically maintain an action-value function (Q-function) and, optionally, a state-value function (V-function), refered as $Q(s, a)$ and $V(s)$ respectively (Haarnoja et al., 2017; Nachum et al., 2017; Kumar et al., 2020; Kostrikov et al., 2021b). These two value functions are learned by encouraging them to satisfy single-step Bellman consistencies. Define a collection of policy evaluation operator (of different policy $\mathbf{x}$) on Q and V as

$$(\mathcal{T}^{\mathbf{x}}Q)(s, a) := r(s, a) + \gamma \mathbb{E}_{s'|s,a}\mathbb{E}_{a' \sim \mathbf{x}}\left[Q(s', a')\right]$$
$$(\mathcal{T}^{\mathbf{x}}V)(s) := \mathbb{E}_{a \sim \pi}\left[r(s, a) + \gamma \mathbb{E}_{s'|s,a}\left[V(s')\right]\right],$$

then Q and V are learned by $\min_Q J(Q) = \frac{1}{2}\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[(\mathcal{T}^{\mathbf{x}}Q - Q)(s, a)^2\right]$ and $\min_V J(V) = \frac{1}{2}\mathbb{E}_{s\sim\mathcal{D}}\left[(\mathcal{T}^{\mathbf{x}}V - V)(s)^2\right]$, respectively. Note that $\mathbf{x}$ could be the learned policy π or the behavior policy μ, if $\mathbf{x} = \mu$, then $a \sim \mu$ and $a' \sim \mu$ are equal to $a \sim \mathcal{D}$ and $a' \sim \mathcal{D}$, respectively. In offline RL, since $\mathcal{D}$ typically does not contain all possible transitions (s, a, s'), one actually uses an empirical policy evaluation operator that only backs up a single s' sample, we denote this operator as $\hat{\mathcal{T}}^{\mathbf{x}}$.

In-sample Learning via Expectile Regression Instead of adding explicit regularization to the policy evaluation operator to avoid out-of-distribution actions, IQL uses only in-sample actions to learn the optimal Q-function. IQL uses an asymmetric ℓ_2 loss (i.e., expectile regression) to learn the V-function, which can be seen as an estimate of the maximum Q-value over actions that are in the dataset support, thus allowing implicit Q-learning:

$$\min_V \mathbb{E}_{(s,a)\sim\mathcal{D}}\Big[\big|\tau - \mathbb{1}\big(Q(s,a)-V(s)<0\big)\big|\big(Q(s,a)-V(s)\big)^2\Big] \tag{2}$$

$$\min_Q \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\Big[\big(r(s,a)+\gamma V(s')-Q(s,a)\big)^2\Big],$$

where $\mathbb{1}$ is the indicator function. After learning Q and V, IQL extracts the policy by advantage-weighted regression (Peters et al., 2010; Peng et al., 2019; Nair et al., 2020):

$$\min_\pi \mathbb{E}_{(s,a)\sim\mathcal{D}}\big[\exp\left(\beta\left(Q(s,a)-V(s)\right)\right)\log\pi(a|s)\big]. \tag{3}$$

While IQL achieves superior D4RL benchmark results, several issues remain unsolved:

- The hyperparameter τ has a gap between theory and practice: in theory τ should be close to 1 to obtain an optimal policy while in practice a larger τ may give a worse result.
- In IQL the value function is estimating the optimal policy instead of the behavior policy, how does IQL handle the distributional shift issue?
- Why should the policy be extracted by advantage-weighted regression, does this technique guarantee the same optimal policy as the one implied in the learned optimal Q-function?

4 Offline RL with Implicit Value Regularization

In this section, we introduce a framework where a general form of value regularization can be implicitly applied. We begin with a special MDP where a behavior regularizer is added to the reward, we conduct a full mathematical analysis of this regularized MDP and give the solution of it under certain assumptions, which results in a complete in-sample learning paradigm. We then instantiate a practical algorithm from this framework and give a thorough analysis and discussion of it.

4.1 Behavior-regularized MDPs

Like entropy-regularized RL adds an entropy regularizer to the reward (Haarnoja et al., 2018), in this paper we consider imposing a general behavior regularization term to objective (1) and solve the following *behavior-regularized* MDP problem

$$\max_\pi \mathbb{E}\Bigg[\sum_{t=0}^{\infty}\gamma^t\Big(r(s_t,a_t)-\alpha\cdot f\Big(\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}\Big)\Big)\Bigg], \tag{4}$$

where $f(\cdot)$ is a regularization function. It is known that in entropy-regularized RL the regularization gives smoothness of the Bellman operator (Ahmed et al., 2019; Chow et al., 2018), e.g., from greedy max to softmax over the whole action space when the regularization is Shannon entropy. While in our new learning objective (4), we find that the smoothness will transfer the greedy max from policy π to a softened max (depending on f) over behavior policy μ, this enables an in-sample learning scheme, which is appealing in the offline RL setting.

In the behavior-regularized MDP, we have a modified policy evaluation operator $\mathcal{T}_f^\pi$ given by

$$(\mathcal{T}_f^\pi)Q(s,a) := r(s,a)+\gamma\mathbb{E}_{s'|s,a}\left[V(s')\right]$$

where

$$V(s)=\mathbb{E}_{a\sim\pi}\Big[Q(s,a)-\alpha f\Big(\frac{\pi(a|s)}{\mu(a|s)}\Big)\Big].$$

The policy learning objective can also be expressed as $\max_\pi \mathbb{E}_{s\sim\mathcal{D}}\left[V(s)\right]$. Compared with the origin policy evaluation operator $\mathcal{T}^\pi$, $\mathcal{T}_f^\pi$ is actually applying a value regularization to the Q-function. However, the regularization term is hard to compute because the behavior policy μ is unknown. Although we can use Fenchel-duality (Boyd et al., 2004) to get a sampled-based estimation if f belongs to the f-divergence (Wu et al., 2019), this unnecessarily brings a min-max optimization problem, which is hard to solve and results in a poor performance in practice (Nachum et al., 2019).

4.2 Assumptions and Solutions

We now show that we can get the optimal value function Q^* and V^* without knowing μ. First, in order to make the learning problems (4) analyzable, two basic assumptions are required as follows:

Assumption 1. *Assume $\pi(a|s) > 0 \Rightarrow \mu(a|s) > 0$ so that π/μ is well-defined.*

Assumption 2. *Assume the function $f(x)$ satisfies the following conditions on $(0, \infty)$: (1) $f(1) = 0$; (2) $h_f(x) = xf(x)$ is strictly convex; (3) $f(x)$ is differentiable.*

The assumptions of $f(1) = 0$ and $xf(x)$ strictly convex make the regularization term be positive due to the Jensen's inequality as $\mathbb{E}_\mu\big[\frac{\pi}{\mu}f\big(\frac{\pi}{\mu}\big)\big] \geq 1f(1) = 0$. This guarantees that the regularization term is minimized only when $\pi = \mu$. Because $h_f(x)$ is strictly convex, its derivative, $h'_f(x) = f(x) + xf'(x)$ is a strictly increasing function and thus $(h'_f)^{-1}(x)$ exists. For simplicity, we denote $g_f(x) = (h'_f)^{-1}(x)$. The assumption of differentiability facilitates theoretic analysis and benefits practical implementation due to the widely used automatic derivation in deep learning.

Under these two assumptions, we can get the following two theorems:

Theorem 1. *In the behavior-regularized MDP, any optimal policy π^* and its optimal value function Q^* and V^* satisfy the following optimality condition for all states and actions:*

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[V^*(s')\right]$$

$$\pi^*(a|s) = \mu(a|s) \cdot \max\left\{g_f\Big(\frac{Q^*(s,a) - U^*(s)}{\alpha}\Big), 0\right\} \tag{5}$$

$$V^*(s) = U^*(s) + \alpha \mathbb{E}_{a\sim\mu}\left[\Big(\frac{\pi^*(a|s)}{\mu(a|s)}\Big)^2 f'\Big(\frac{\pi^*(a|s)}{\mu(a|s)}\Big)\right] \tag{6}$$

where $U^*(s)$ is a normalization term so that $\sum_{a\in\mathcal{A}} \pi^*(a|s) = 1$.

The proof is provided in Appendix C.1. The proof depends on the KKT condition where the derivative of a Lagrangian objective function with respect to policy $\pi(a|s)$ becomes zero at the optimal solution. Note that the resulting formulation of Q^* and V^* only involves U^* and action samples from μ. $U^*(s)$ can be uniquely solved from the equation obtained by plugging Eq.(5) into $\sum_{a\in\mathcal{A}} \pi^*(a|s) = 1$, which also only uses actions sampled from μ. In other words, now the learning of Q^* and V^* can be realized in an in-sample manner.

Theorem 1 also shows how the behavior regularization influences the optimality condition. If we choose f such that there exists some x that $g_f(x) < 0$, then it can be shown from Eq.(5) that the optimal policy π^* will be sparse by assigning zero probability to the actions whose Q-values $Q^*(s,a)$ are below the threshold $U^*(s) + \alpha h'_f(0)$ and assigns positive probability to near optimal actions in proportion to their Q-values (since $g_f(x)$ is increasing). Note that π^* could also have no sparsity, for example, if we choose $f = \log(x)$, then $g_f = \exp(x-1)$ will give all elements non-zero values.

Theorem 2. *Define $\mathcal{T}^*_f$ the case where π in $\mathcal{T}^\pi_f$ is the optimal policy π^*, then $\mathcal{T}^*_f$ is a γ-contraction.*

The proof is provided in Appendix C.2. This theorem means that by applying $Q^{k+1} = \mathcal{T}^*_f Q^k$ repeatedly, then sequence Q^k will converge to the Q-value of the optimal policy π^* when $k \to \infty$.

After giving the closed-form solution of the optimal value function. We now aim to instantiate a practical algorithm. In offline RL, in order to completely avoid out-of-distribution actions, we want a *zero-forcing* support constraint, i.e., $\mu(a|s) = 0 \Rightarrow \pi(a|s) = 0$. This reminds us of the class of α-divergence (Boyd et al., 2004), which is a subset of f-divergence and takes the following form ($\alpha \in \mathbb{R}\backslash\{0,1\}$):

$$D_\alpha(\mu, \pi) = \frac{1}{\alpha(\alpha-1)}\mathbb{E}_\pi\left[\Big(\frac{\pi}{\mu}\Big)^{-\alpha} - 1\right].$$

α-divergence is known to be mode-seeking if one chooses $\alpha \leq 0$. Note that the Reverse KL divergence is the limit of $D_\alpha(\mu, \pi)$ when $\alpha \to 0$. We can also obtain Helinger distance and Neyman χ^2-divergence as $\alpha = 1/2$ and $\alpha = -1$, respectively. One interesting property of α-divergence is that $D_\alpha(\mu, \pi) = D_{1-\alpha}(\pi, \mu)$.

4.3 SPARSE Q-LEARNING (SQL)

We first consider the case where $\alpha = -1$, which we find is the regularization term CQL adds to the policy evaluation operator (according to Appendix C in CQL): $Q(s,a) = \mathcal{T}^\pi Q(s,a) - \beta[\frac{\pi(a|s)}{\mu(a|s)} - 1]$. In this case, we have $f(x) = x - 1$ and $g_f(x) = \frac{1}{2}x + \frac{1}{2}$. Plug them into Eq.(5) and Eq.(6) in Theorem 1, we get the following formulation:

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[V^*\left(s'\right)\right] \tag{7}$$

$$\pi^*(a|s) = \mu(a|s) \cdot \max\left\{\frac{1}{2} + \frac{Q^*(s,a) - U^*(s)}{2\alpha}, 0\right\} \tag{8}$$

$$V^*(s) = U^*(s) + \alpha \mathbb{E}_{a\sim\mu}\left[\left(\frac{\pi^*(a|s)}{\mu(a|s)}\right)^2\right], \tag{9}$$

where $U^*(s)$ needs to satisfy the following equation to make π^* integrate to 1:

$$\mathbb{E}_{a\sim\mu}\left[\max\left\{\frac{1}{2} + \frac{Q^*(s,a) - U^*(s)}{2\alpha}, 0\right\}\right] = 1 \tag{10}$$

It is usually intractable to get the closed-form solution of $U^*(s)$ from Eq.(10), however, here we make a mathematical transformation and show its equivalence to solving a convex optimization problem.

Lemma 1. *We can get $U^*(s)$ by solving the following optimization problem:*

$$\min_U \; \mathbb{E}_{a\sim\mu}\left[\mathbb{1}\left(\frac{1}{2} + \frac{Q^*(s,a) - U(s)}{2\alpha} > 0\right)\left(\frac{1}{2} + \frac{Q^*(s,a) - U(s)}{2\alpha}\right)^2\right] + \frac{U(s)}{\alpha} \tag{11}$$

The proof can be easily got if we set the derivative of the objective to 0 with respect to $U(s)$, which is exactly Eq.(10). Now we obtain a learning scheme to get Q^*, U^* and V^* by iteratively updating Q, U and V following Eq.(9), objective (11) and Eq.(7), respectively. We refer to this learning scheme as SQL-U, however, SQL-U needs to train three networks, which is a bit computationally expensive.

Note that the term $\mathbb{E}_{a\sim\mu}\left[\left(\frac{\pi^*(a|s)}{\mu(a|s)}\right)^2\right]$ in Eq.(9) is equal to $\mathbb{E}_{a\sim\pi^*}\left[\frac{\pi^*(a|s)}{\mu(a|s)}\right]$, as π^* is optimized to become mode-seeking, for actions sampled from π^*, its probability $\pi^*(a|s)$ should be close to the probability under the behavior policy, $\mu(a|s)$. Note that for actions sampled from μ, $\pi^*(a|s)$ and $\mu(a|s)$ may have a large difference because $\pi^*(a|s)$ may be 0.

Hence in SQL we **make an approximation** by assuming $\mathbb{E}_{a\sim\pi^*}\left[\frac{\pi^*(a|s)}{\mu(a|s)}\right] = 1$, this removes one network as $U^* = V^* - \alpha$. Replacing U^* with V^*, we get the following learning scheme that only needs to learn V and Q iteratively to get V^* and Q^*:

$$\min_V \; \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\mathbb{1}\left(1 + \frac{Q(s,a) - V(s)}{2\alpha} > 0\right)\left(1 + \frac{Q(s,a) - V(s)}{2\alpha}\right)^2 + \frac{V(s)}{\alpha}\right] \tag{12}$$

$$\min_Q \; \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(r(s,a) + \gamma V(s') - Q(s,a)\right)^2\right] \tag{13}$$

After getting V and Q, following the formulation of π^* in Eq.(8), we can get the learning objective of policy π by minimizing the KL-divergence between π and π^*:

$$\max_\pi \; \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\mathbb{1}\left(1 + \frac{Q(s,a) - V(s)}{2\alpha} > 0\right)\left(1 + \frac{Q(s,a) - V(s)}{2\alpha}\right)\log \pi(a|s)\right]. \tag{14}$$

4.4 EXPONENTIAL Q-LEARNING (EQL)

Now let's consider another choice, $\alpha \to 0$ which is the Reverse KL divergence. Note that AWR also uses Reverse KL divergence, however, it applies it to the policy improvement step and needs to sample actions from the policy when learning the value function. In this case, we get $f(x) = \log(x)$ and $g_f(x) = \exp(x-1)$. Plug them into Eq.(5) and Eq.(6) in Theorem 1, we have

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[V^*\left(s'\right)\right]$$

$$\pi^*(a|s) = \mu(a|s) \cdot \exp\left(\frac{Q^*(s,a) - U^*(s)}{\alpha} - 1\right)$$
$$V^*(s) = U^*(s) + \alpha \mathbb{E}_{a\sim\mu}\left[\left(\frac{\pi^*(a|s)}{\mu(a|s)}\right)^2 \frac{\mu(a|s)}{\pi^*(a|s)}\right],$$

note that $\mathbb{E}_{a\sim\mu}[(\frac{\pi^*(a|s)}{\mu(a|s)})^2 \frac{\mu(a|s)}{\pi^*(a|s)}]$ is equal to 1, so we get $V^*(s) = U^*(s) + \alpha$, this eliminates the existence of U^* **without any approximation**. Replacing U^* with V^*, we get the following formulation:

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[V^*(s')\right]$$
$$\pi^*(a|s) = \mu(a|s) \cdot \exp\left(\frac{Q^*(s,a) - V^*(s)}{\alpha}\right)$$

Note that π^* should be integrated to 1, we use the same mathematical transformation did in SQL and get the closed-form solution of $V^*(s)$ by solving the following convex optimization problem.

Lemma 2. *We can get $V^*(s)$ by solving the following optimization problem:*

$$\min_V \ \mathbb{E}_{a\sim\mu}\left[\exp\left(\frac{Q^*(s,a) - V(s)}{\alpha}\right)\right] + \frac{V(s)}{\alpha}$$

Now the final learning objective of Q, V and π is:

$$\min_V \ \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp\left(\frac{Q(s,a) - V(s)}{\alpha}\right) + \frac{V(s)}{\alpha}\right] \tag{15}$$

$$\min_Q \ \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(r(s,a) + \gamma V(s') - Q(s,a)\right)^2\right] \tag{16}$$

$$\max_\pi \ \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp\left(\frac{Q(s,a) - V(s)}{\alpha}\right) \log \pi(a|s)\right], \tag{17}$$

we name this algorithm as EQL (Exponential Q-Learning) because there is an exponential term in the learning objective.

To summarize, our final algorithm, SQL and EQL, consist of three supervised stages: learning V, learning Q, and learning π. We use target networks for Q-functions and use clipped double Q-learning (take the minimum of two Q-functions) in learning V and π. We summarize the training procedure in Algorithm 1.

4.5 DISCUSSIONS

SQL and EQL establishes the connection with several prior works such as CQL, IQL and AWR.

Algorithm 1 Sparse or Exponential Q-Learning

```
Require: $\mathcal{D}$, $\alpha$.
Initialize $Q_\phi$, $Q_{\phi'}$, $V_\psi$, $\pi_\theta$
for $t = 1, 2, \cdots, N$ do
    Sample transitions $(s, a, r, s') \sim \mathcal{D}$
    Update $V_\psi$ by Eq.(12) or Eq.(15) using $V_\psi$, $Q_{\phi'}$
    Update $Q_\phi$ by Eq.(13) or Eq.(16) using $V_\psi$, $Q_\phi$
    Update $Q_{\phi'}$ by $\phi' \leftarrow \lambda\phi + (1-\lambda)\phi'$
    Update $\pi_\theta$ by Eq.(14) or Eq.(17) using $V_\psi$, $Q_{\phi'}$
end for
```

Like CQL pushes down policy Q-values and pushes up dataset Q-values, in SQL and EQL, the first term in Eq.(12) and Eq.(15) pushes up V-values if $Q - V > 0$ while the second term pushes down V-values, and α trades off these two terms. SQL incorporates the same inherent conservatism as CQL by adding the χ^2-divergence to the policy evaluation operator. However, SQL learns the value function using only dataset samples while CQL needs to sample actions from the policy. In this sense, SQL is an "implicit" version of CQL that avoids any out-of-distribution action. Like AWR, EQL applies the KL-divergence, but implicitly in the policy evaluation step. In this sense, EQL is an "implicit" version of AWR that avoids any OOD action.

Like IQL, SQL and EQL learn both V-function and Q-function. However, IQL appears to be a heuristic approach and the learning objective of V-function in IQL has a drawback. We compute the derivative of the V-function learning objective with respect to the residual $(Q - V)$ in SQL and IQL (see Figure 2 in Appendix A). We find that SQL keeps the derivative unchanged when the residual

Table 1: Averaged normalized scores of SQL against other baselines. The scores are taken over the final 10 evaluations with 5 seeds. SQL or EQL achieves the highest scores in 14 out of 18 tasks.

Dataset	BC	10%BC	BCQ	DT	One-step	TD3+BC	CQL	IQL	SQL	EQL
halfcheetah-m	42.6	42.5	47.0	42.6	48.4	48.3	44.0 ±0.8	47.4 ±0.2	48.3±0.2	47.2±0.3
hopper-m	52.9	56.9	56.7	67.6	59.6	59.3	58.5 ±2.1	66.3 ±5.7	75.5 ±3.4	70.6±2.6
walker2d-m	75.3	75.0	72.6	74.0	81.8	83.7	72.5 ±0.8	72.5 ±8.7	84.2 ±4.6	83.2±4.4
halfcheetah-m-r	36.6	40.6	40.4	36.6	38.1	44.6	45.5 ±0.5	44.2 ±1.2	44.8±0.7	44.5±0.5
hopper-m-r	18.1	75.9	53.3	82.7	97.5	60.9	95.0 ±6.4	95.2 ±8.6	101.7 ±3.3	98.1±3.6
walker2d-m-r	26.0	62.5	52.1	66.6	49.5	81.8	77.2±5.5	76.1 ±7.3	77.2±3.8	81.6±4.2
halfcheetah-m-e	55.2	92.9	89.1	86.8	93.4	90.7	90.7±4.3	86.7±5.3	94.0±0.4	94.6 ±0.5
hopper-m-e	52.5	110.9	81.8	107.6	103.3	98.0	105.4±6.8	101.5 ±7.3	111.8 ±2.2	111.5±2.1
walker2d-m-e	107.5	109.0	109.0	108.1	113.0	110.1	109.6±0.7	110.6±1.0	110.0±0.8	110.2±0.8
antmaze-u	54.6	62.8	78.9	59.2	64.3	78.6	84.8±2.3	85.5 ±1.9	92.2±1.4	93.2 ±2.2
antmaze-u-d	45.6	50.2	55.0	53.0	60.7	71.4	43.4±5.4	66.7 ±4.0	74.0 ±2.3	70.4±2.7
antmaze-m-p	0	5.4	0	0.0	0.3	10.6	65.2±4.8	72.2 ±5.3	80.2 ±3.7	77.5±4.3
antmaze-m-d	0	9.8	0	0.0	0.0	3.0	54.0±11.7	71.0 ±3.2	75.1 ±4.2	74.0±3.7
antmaze-l-p	0	0.0	6.7	0.0	0.0	0.2	38.4±12.3	39.6 ±4.5	50.2 ±4.8	45.6±4.2
antmaze-l-d	0	6.0	2.2	0.0	0.0	0.0	31.6±9.5	47.5 ±4.4	52.3 ±5.2	49.5±4.7
kitchen-c	33.8	-	-	-	-	-	43.8 ±11.2	61.4 ±9.5	76.4 ±8.7	70.3±7.1
kitchen-p	33.9	-	-	-	-	-	49.8±10.1	46.1 ±8.5	72.5 ±7.4	70.5±8.8
kitchen-m	47.5	-	-	-	-	-	51.0±6.5	52.8 ±4.5	67.4 ±5.4	61.6±5.2

is below a threshold, while IQL doesn't. In IQL, the derivative keeps decreasing as the residual becomes more negative, hence, the V-function will be over-underestimated by those bad actions whose Q-value is extremely small. Note that SQL and EQL will assign a zero or exponential small probability mass to those bad actions according to Eq.(14) and Eq.(17), the sparsity is incorporated due to the mode-seeking behavior of χ^2-divergence and KL-divergence.

Also, IQL needs two hyperparameters (τ and β) while SQL only needs one (α). The two hyperparameters in IQL may not align well because they represent two different regularizations. Note that objective (17) is exactly how IQL extracts the policy! However, the corresponding optimal V-function learning objective (15) is not objective (2). This reveals that the policy extraction part in IQL gets a different policy from the one implied in the optimal Q-function.

5 EXPERIMENTS

We present empirical evaluations of SQL and EQL in this section. We first evaluate SQL and EQL against other baseline algorithms on benchmark offline RL datasets. We then show the benefits of sparsity introduced in SQL and EQL by comparing them with IQL in noisy data regimes. We finally show the robustness of SQL and EQL by comparing them with CQL in small data regimes.

5.1 BENCHMARK DATASETS

We first evaluate our approach on D4RL datasets (Fu et al., 2020). It is worth mentioning that Antmaze and Kitchen datasets include few or no near-optimal trajectories, and highly require learning a value function to obtain effective policies via "stitching". We compare SQL with prior state-of-the-art offline RL methods, including BC (Pomerleau, 1989), 10%BC (Chen et al., 2021), BCQ (Fujimoto et al., 2018), DT (Chen et al., 2021), TD3+BC (Fujimoto & Gu, 2021), One-step RL (Brandfonbrener et al., 2021), CQL (Kumar et al., 2020), and IQL (Kostrikov et al., 2021a). Aggregated results are displayed in Table 1. In MuJoCo tasks, where performance is already saturated, SQL and EQL show competitive results to the best performance of prior methods. In more challenging AntMaze and Kitchen tasks, SQL and EQL outperform all other baselines by a large margin. This shows the effectiveness of value learning in SQL and EQL. We show learning curves and performance profiles generated by the rliable library (Agarwal et al., 2021) in Appendix D.

We then compare our approach with other baselines on high-dimensional image-based Atari datasets in RL Unplugged (Gulcehre et al., 2020). Our approach also achieves superior performance on these datasets, we show aggregated results, performance profiles and experimental details in Appendix D.

5.2 Noisy Data Regime

In this section, we try to validate our hypothesis that the sparsity term our algorithm introduced in learning the value function will benefit when the datasets contain a large portion of noisy transitions. To do so, we make a "mixed" dataset by combining random datasets and expert dataset with different expert ratios. We test the performance of SQL, EQL and IQL under different mixing ratios in Fig. 1.

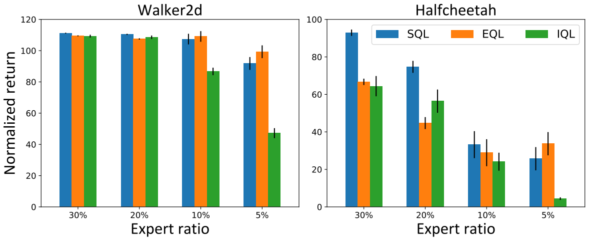

Figure 1: Performance of different methods in noisy data regimes.

It is shown that SQL and EQL outperforms IQL under all settings. The performance of IQL is vulnerable to the expert ratio, it has a sharp decrease from 30% to 1% while SQL and EQL still retain the expert performance. For example, in walker2d, SQL and EQL reaches near 100 performance when the expert ratio is only 5%; in halfcheetah, IQL is affected even with a high expert ratio (30%).

5.3 Small Data Regime

In this section, we try to explore the benefits of in-sample learning over out-of-sample learning. We are interested to see whether in-sample learning brings more robustness than out-of-sample learning when the dataset size is small or the dataset diversity of some states is small, which are challenges one might encounter when using offline RL algorithms on real-world data.

Table 2: The normalized return (NR) and Bellman error (BR) of CQL, SQL and EQL in small data regimes.

Dataset (AntMaze)		CQL		SQL		EQL	
		NR	BE	NR	BE	NR	BE
Medium	Vanilla	65.2	13.1	75.1	1.6	74.0	2.2
	Easy	48.2	14.8	56.2	1.7	57.5	1.1
	Medium	14.5	14.7	43.3	2.1	39.7	2.3
	Hard	9.3	64.4	24.2	1.9	19.6	1.8
Large	Vanilla	38.4	13.5	50.2	1.4	49.6	1.7
	Easy	28.1	12.8	40.5	1.5	40.4	1.7
	Medium	6.3	30.6	36.7	1.3	35.3	1.8
	Hard	0	300.5	34.2	2.6	31.6	1.6

To do so, we make custom datasets by discarding some transitions in the AntMaze datasets. For each transition, the closer it is to the target location, the higher probability it will be discarded from the dataset. This simulates the scenarios (i.e., robotic manipulation) where the dataset is fewer and has limited state coverage near the target location because the (stochastic) data generation policies maybe not be successful and are more determined when they get closer to the target location (Kumar et al., 2022). We use a hyperparameter to control the discarding ratio and build three new tasks: `Easy`, `Medium` and `Hard`, with dataset becomes smaller. For details please refer to Appendix D. We compare SQL with CQL as they use the same inherent value regularization but SQL uses in-sample learning while CQL uses out-of-sample learning,

We demonstrate the final normalized return (NR) during evaluation and the mean squared Bellman error (BE) during training in Table 2. It is shown that CQL has a significant performance drop when the difficulty of tasks increases, and the Bellman error also exponentially grows up, indicating that the value extrapolation error becomes large in small data regimes. SQL and EQL remain a stable yet good performance under all difficulties, the Bellman error of SQL is much smaller than that of CQL. This justifies the benefits of in-sample learning, i.e., it avoids erroneous value estimation by using only dataset samples while still allowing in-sample generalization to obtain a good performance.

6 Conclusions and Future Work

In this paper, we propose a general Implicit Value Regularization framework, which builds the bridge between behavior regularized and in-sample learning methods in offline RL. Based on this framework, we propose two practical algorithms, which use the same value regularization in existing works, but in a complete in-sample manner. We verify the effectiveness of our algorithms on both the D4RL benchmark and customed noisy and small data regimes by comparing it with different baselines. One future work is to scale our proposed framework to online RL or offline imitaiton learning (Li et al., 2023a). Another future work is, instead of only constraining action distribution, constraining the state-action distribution between d^π and d^D as considered in Nachum et al. (2019).

Acknowledgement

This work is supported by funding from Intel Corporation. The authors would like to thank the anonymous reviewers for their feedback on the manuscripts.

References

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *Proc. of ICML*, pp. 104–114, 2020.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Proc. of NeurIPS*, 2021.

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *Proc. of ICML*, pp. 151–160, 2019.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Proc. of NeurIPS*, 2021.

Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhi-Hong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *Proc. of ICLR*, 2021.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

David Brandfonbrener, William F Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *Proc. of NeurIPS*, 2021.

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. *Proc. of NeurIPS*, 2021.

Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith W. Ross. BAIL: best-action imitation learning for batch deep reinforcement learning. In *Proc. of NeurIPS*, 2020.

Yinlam Chow, Ofir Nachum, and Mohammad Ghavamzadeh. Path consistency learning in tsallis entropy regularized mdps. In *Proc. of ICML*, pp. 978–987, 2018.

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

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *ArXiv preprint*, 2021.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proc. of ICML*, pp. 1582–1591, 2018.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. In *Proc. of ICML*, pp. 2160–2169, 2019.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. In *Proc. of NeurIPS*, 2020.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proc. of ICML*, pp. 1352–1361, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. of ICML*, pp. 1856–1865, 2018.

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*, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of ICLR*, 2015.

Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *Proc. of ICML*, pp. 5774–5783, 2021a.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *ArXiv preprint*, 2021b.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Proc. of NeurIPS*, pp. 11761–11771, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Proc. of NeurIPS*, 2020.

Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. When Should We Prefer Offline Reinforcement Learning Over Behavioral Cloning? *ArXiv preprint*, 2022.

Kyungjae Lee, Sungyub Kim, Sungbin Lim, Sungjoon Choi, and Songhwai Oh. Tsallis reinforcement learning: A unified framework for maximum entropy reinforcement learning. *ArXiv preprint*, 2019.

Jianxiong Li, Xiao Hu, Haoran Xu, Jingjing Liu, Xianyuan Zhan, Qing-Shan Jia, and Ya-Qin Zhang. Mind the gap: Offline policy optimization for imperfect rewards. In *Proc. of ICLR*, 2023a.

Jianxiong Li, Xianyuan Zhan, Haoran Xu, Xiangyu Zhu, Jingjing Liu, and Ya-Qin Zhang. When data geometry meets deep function: Generalizing offline reinforcement learning. In *Proc. of ICLR*, 2023b.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *Proc. of ICLR*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Proc. of NeurIPS*, pp. 2775–2785, 2017.

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

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *ArXiv preprint*, 2020.

Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *ArXiv preprint*, 2017.

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

Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *Proc. of AAAI*, 2010.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Proc. of NeurIPS*, 1989.

Takuma Seno and Michita Imai. d3rlpy: An offline deep reinforcement learning library. *ArXiv preprint*, 2021.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 2017.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*. MIT press Cambridge, 1998.

Shengpu Tang and Jenna Wiens. Model selection for offline reinforcement learning: Practical considerations for healthcare settings. In *Proc. of ML4H*, 2021.

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 *Proc. of NeurIPS*, 2020.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *ArXiv preprint*, 2019.

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 *Proc. of ICML*, pp. 11319–11328, 2021.

Haoran Xu, Xianyuan Zhan, Jianxiong Li, and Honglei Yin. Offline reinforcement learning with soft behavior regularization. *ArXiv preprint*, 2021.

Haoran Xu, Li Jiang, Jianxiong Li, and Xianyuan Zhan. A policy-guided imitation approach for offline reinforcement learning. In *Proc. of NeurIPS*, 2022a.

Haoran Xu, Xianyuan Zhan, Honglei Yin, and Huiling Qin. Discriminator-weighted offline imitation learning from suboptimal demonstrations. In *Proc. of ICML*, pp. 24725–24742, 2022b.

Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proc. of AAAI*, pp. 8753–8760, 2022c.

Xianyuan Zhan, Haoran Xu, Yue Zhang, Xiangyu Zhu, Honglei Yin, and Yu Zheng. Deepthermal: Combustion optimization for thermal power generating units using offline reinforcement learning. In *Proc. of AAAI*, pp. 4680–4688, 2022.

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

A A Statistical View of Why SQL and EQL Work

Inspired by the analysis in IQL, we give another view of why SQL and EQL could learn the optimal policy. Consider estimating a parameter m_α for a random variable X using samples from a dataset $\mathcal{D}$, we show that m_α could fit the extrema of X by using the learning objective of V-function in SQL:

$$\arg\min_{m_\alpha} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{1}\left(1+\frac{x-m_\alpha}{2\alpha}>0\right)\left(1+\frac{x-m_\alpha}{2\alpha}\right)^2+\frac{m_\alpha}{\alpha}\right],$$

or using the learning objective of V-function in EQL:

$$\arg\min_{m_\alpha} \mathbb{E}_{x\sim\mathcal{D}}\left[\exp\left(\frac{x-m_\alpha}{\alpha}\right)+\frac{m_\alpha}{\alpha}\right]$$

In Figure 2 and Figure 3, we give an example of estimating the state conditional extrema of a two-dimensional random variable, as shown, $\alpha \to 0$ approximates the maximum operator over in-support values of y given x. This phenomenon can be justified in our IVR framework as the value function becomes more optimal with less value regularization. However, less value regularization also brings more distributional shift, so we need a proper α to trade-off optimality against distributional shift.

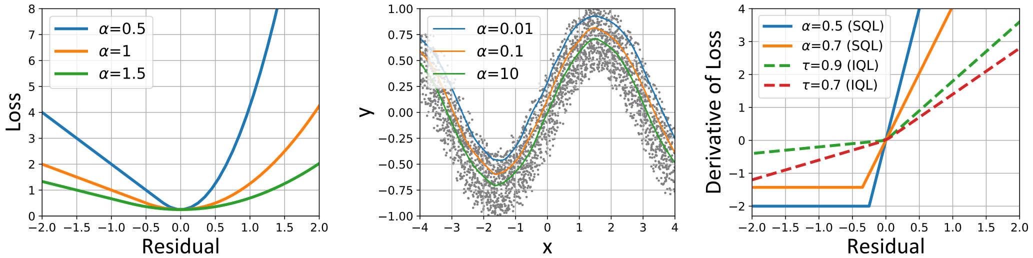

Figure 2: **Left:** The loss with respect to the residual $(Q-V)$ in the learning objective of V in SQL with different α. **Center:** An example of estimating state conditional extrema of a two-dimensional random variable (generated by adding random noise to samples from $y=\sin(x)$). Each x corresponds to a distribution over y. The loss fits the extrema more with α becoming smaller. **Right:** The comparison of the derivative of loss of SQL and IQL. In SQL, the derivative keeps unchanged when the residual is below a threshold.

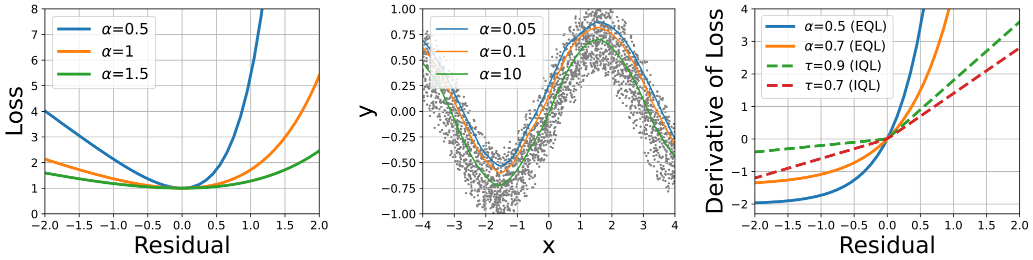

Figure 3: **Left:** The loss with respect to the residual $(Q-V)$ in the learning objective of V in EQL with different α. **Center:** An example of estimating state conditional extrema of a two-dimensional random variable (generated by adding random noise to samples from $y=\sin(x)$). Each x corresponds to a distribution over y. The loss fits the extrema more with α becoming smaller. **Right:** The comparison of the derivative of loss of EQL and IQL. In EQL, the derivative softly decreases and keeps (nearly) unchanged when the residual is below a threshold.

B How Does Sparsity Benefit Value Learning in SQL?

In this section, we add more experiments about the sparsity characteristic of SQL. We use a toy example in the tabular setting to demonstrate how sparsity benefits value learning in SQL. We study the relationship of normalized score and sparsity ratio with α in the continuous action setting to clearly show that sparsity plays an important role in the performance of SQL.

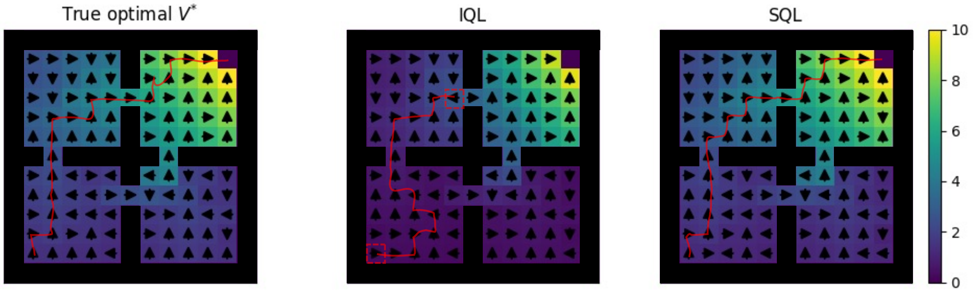

Figure 4: Evaluation of IQL and SQL on the Four Rooms environment. SQL learns a more optimal value function and produces a better policy than IQL when the dataset is heavily corrupted by suboptimal actions.

Table 3: The relationship of the normalized score (left) and non-sparsity ratio (right) with α in MuJoCo datasets.

α	0.5		1		2		5		10	
metrics	score	ratio	score	ratio	score	ratio	score	ratio	score	ratio
halfcheetah-m	48.3	0.72	48.7	0.91	48.1	0.96	47.4	0.97	47.1	1.00
hopper-m	62.5	0.84	74.5	0.90	74.1	0.98	68.5	0.99	62.4	0.99
walker2d-m	22.3	0.03	65.3	0.93	84.2	0.98	83.7	0.99	84.1	0.99
halfcheetah-m-r	43.2	0.65	44.2	0.84	44.8	0.89	44.9	0.97	44.8	0.99
hopper-m-r	43.5	0.53	95.5	0.74	100.7	0.78	63.3	0.94	74.2	0.99
walker2d-m-r	5.9	0.51	38.2	0.70	82.2	0.89	80.3	0.97	81.3	1.00
halfcheetah-m-e	40.2	0.38	39.3	0.33	35.8	0.27	94.2	0.99	94.8	0.99
hopper-m-e	18.6	0.07	106.3	0.89	106.6	0.94	111.9	0.96	111.5	0.99
walker2d-m-e	9.2	0.01	110.2	0.95	111.3	0.96	109.5	0.99	111.2	0.99
mujoco-mean	30.4	0.41	69.2	0.80	76.4	0.85	78.1	0.97	79.0	0.99

Table 4: The relationship of the normalized score (left) and non-sparsity ratio (right) with α in AntMaze datasets.

α	0.2		0.3		0.5		0.7		0.9		1.0		2.0	
metrics	score	ratio	score	ratio	score	ratio	score	ratio	score	ratio	score	ratio	score	ratio
antmaze-m-d	40.3	0.01	62.6	0.39	75.1	0.35	68.6	0.70	61.6	0.82	63.4	0.77	17.5	0.92
antmaze-m-p	0.0	0.03	70.2	0.32	80.2	0.62	67.5	0.81	69.3	0.83	67.3	0.75	2.0	0.88
antmaze-l-d	35.3	0.0	55.2	0.56	50.2	0.70	39.5	0.75	18.3	0.91	20.5	0.88	4.1	0.98
antmaze-l-p	0	0.48	38.3	0.33	52.3	0.51	18.2	0.70	20.3	0.89	11.5	0.80	2.1	0.95
antmaze-mean	18.9	0.13	56.6	0.40	64.5	0.79	48.5	0.74	42.4	0.86	40.7	0.80	7.5	0.94

B.1 Sparsity in the Tabular Setting

In Appendix A, we show the comparison of loss's derivative of SQL and IQL, we found that SQL keeps the derivative unchanged when the residual is below a threshold while IQL doesn't, the V-function in IQL will be over-underestimated by those bad actions whose Q-value is small.

To justify this claim, we use the Four Rooms environment, where the agent starts from the bottom-left and needs to navigate through the four rooms to reach the goal in the up-right corner in as few steps as possible. There are four actions: $\mathcal{A} = \{up, down, right, left\}$. The reward is zero on each time step until the agent reaches the goal-state where it receives +10. The offline dataset is collected by a **random** behavior policy which takes each action with equal probability. We collect 30 trajectories and each trajectory is terminated after 20 steps if not succeed, γ is 0.9.

It can be seen from Fig. 4 that the value learning in IQL is corrupted by those suboptimal actions in this dataset. Those suboptimal actions prevent IQL from propagating correct learning signals from the goal location to the start location, resulting underestimated V-values and some mistaken Q-values. Particularly, incorrect Q-values at $(1, 1)$ and $(5, 9)$ make the agent fail to reach the goal. While in SQL, V-values and Q-values are more identical to the true optimal ones and the agent succeeds in reaching the goal location. This reveals that the sparsity term in SQL helps to alleviate the effect of bad dataset actions and learn a more optimal value function.

B.2 Sparsity in the Continuous Action Setting

The value of non-sparsity ratio (i.e., $\mathbb{E}_{(s,a)\sim\mathcal{D}}[\mathbb{1}(1+(Q(s,a)-V(s))/2\alpha > 0)]$) is controlled by the hyperparameter α. In the continuous action setting, we show the relationship of the normalized score and non-sparsity ratio with α in Table 3 and Table 4. It can be seen that typically a larger α gives less sparsity, sparsity plays an important role in the performance of SQL and we need to choose a proper sparsity ratio to achieve the best performance. The best sparsity ratio depends on the composition of the dataset, for example, the best sparsity ratios in MuJoCo datasets (around 0.1) are always larger than those in AntMaze datasets (around 0.4), this is because AntMaze datasets are kind of multi-task datasets (the start and goal location are different from the current ones), there is a large portion of useless transitions contained so it is reasonable to give those transitions zero weights by using more sparsity.

C Proofs

C.1 Proof of Theorem 1

In this section, we give the detailed proof for Theorem 1, which states the optimality condition of the behavior regularized MDP. The proof follows from the Karush-Kuhn-Tucker (KKT) conditions where the derivative of a Lagrangian objective function with respect to policy $\pi(a|s)$ is set zero. Hence, our main theory is necessary and sufficient.

Proof. The Lagrangian function of (4) is written as follows

$$\begin{aligned} L(\pi,\beta,u) = & \sum_s d^\pi(s) \sum_a \pi(a|s)\left(Q(s,a) - \alpha f\left(\frac{\pi(a|s)}{\mu(a|s)}\right)\right) \\ & - \sum_s d^\pi(s)\left[u(s)\left(\sum_a \pi(a|s) - 1\right) - \sum_a \beta(a|s)\pi(a|s)\right], \end{aligned}$$

where d^π is the stationary state distribution of the policy π, u and β are Lagrangian multipliers for the equality and inequality constraints respectively.

Let $h_f(x) = xf(x)$. Then the KKT condition of (4) are as follows, for all states and actions we have

$$0 \le \pi(a|s) \le 1 \text{ and } \sum_a \pi(a|s) = 1 \tag{18}$$

$$0 \le \beta(a|s) \tag{19}$$

$$\beta(a|s)\pi(a|s) = 0 \tag{20}$$

$$Q(s,a) - \alpha h'_f\left(\frac{\pi(a|s)}{\mu(a|s)}\right) - u(s) + \beta(a|s) = 0 \tag{21}$$

where (18) is the feasibility of the primal problem, (19) is the feasibility of the dual problem, (20) results from the complementary slackness and (21) is the stationarity condition. We eliminate $d^\pi(s)$ since we assume all policies induce an irreducible Markov chain.

From (21), we can resolve $\pi(a|s)$ as

$$\pi(a|s) = \mu(a|s) \cdot g_f\left(\frac{1}{\alpha}\left(Q(s,a) - u(s) + \beta(a|s)\right)\right)$$

Fix a state s. For any positive action, its corresponding Lagrangian multiplier $\beta(a|s)$ is zero due to the complementary slackness and $Q(s,a) > u(s) + \alpha h'_f(0)$ must hold. For any zero-probability action, its Lagrangian multiplier $\beta(a|s)$ will be set such that $\pi(a|s) = 0$. Note that $\beta(a|s) \ge 0$, thus $Q(s,a) \le u(s) + \alpha h'_f(0)$ must hold in this case. From these observations, $\pi(a|s)$ can be reformulated as

$$\pi(a|s) = \mu(a|s) \cdot \max\left\{g_f\left(\frac{1}{\alpha}\left(Q(s,a) - u(s)\right)\right), 0\right\} \tag{22}$$

By plugging (22) into (18), we can obtain an new equation

$$\mathbb{E}_{a\sim\mu}\left[\max\left\{g_f\left(\frac{1}{\alpha}\left(Q(s,a)-u(s)\right)\right),0\right\}\right]=1 \tag{23}$$

Note that (23) has and only has one solution denoted as u^* (because the LHS of (23) can be seen as a continuous and monotonic function of u), so u^* can be solved uniquely. We denote the corresponding policy π as π^*.

Next we aim to obtain the optimal state value V^*. It follows that

$$\begin{aligned}
&V^*(s)=\mathcal{T}_f^*V^*(s)\\
&=\sum_a\pi^*(a|s)\left(Q^*(s,a)-\alpha f\left(\frac{\pi^*(a|s)}{\mu(a|s)}\right)\right)\\
&=\sum_a\pi^*(a|s)\left(u^*(s)+\alpha\frac{\pi^*(a|s)}{\mu(a|s)}f'\left(\frac{\pi^*(a|s)}{\mu(a|s)}\right)\right)\\
&=u^*(s)+\alpha\sum_a\frac{\pi^*(a|s)^2}{\mu(a|s)}f'\left(\frac{\pi^*(a|s)}{\mu(a|s)}\right)\\
&=u^*(s)+\alpha\mathbb{E}_{a\sim\mu}\left[\left(\frac{\pi^*(a|s)}{\mu(a|s)}\right)^2f'\left(\frac{\pi^*(a|s)}{\mu(a|s)}\right)\right]
\end{aligned}$$

The first equality follows from the definition of the optimal state value. The second equality holds because π maximizes $\mathcal{T}_f^*V^*(s)$. The third equality results from plugging (21).

To summarize, we obtain the optimality condition of the behavior regularized MDP as follows

$$\begin{aligned}
Q^*(s,a)&=r(s,a)+\gamma\mathbb{E}_{s'|s,a}\left[V^*\left(s'\right)\right]\\
\pi^*(a|s)&=\mu(a|s)\cdot\max\left\{g_f\Big(\frac{Q^*(s,a)-u^*(s)}{\alpha}\Big),0\right\}\\
V^*(s)&=u^*(s)+\alpha\mathbb{E}_{a\sim\mu}\Big[\Big(\frac{\pi^*(a|s)}{\mu(a|s)}\Big)^2f'\Big(\frac{\pi^*(a|s)}{\mu(a|s)}\Big)\Big]
\end{aligned}$$

□

C.2 PROOF OF THEOREM 2

Proof. For any two state value functions V_1 and V_2 , let π_i be the policy that maximizes $\mathcal{T}_f^*V_i$, $i\in 1,2$. Then it follows that for any state s in $\mathcal{S}$,

$$\begin{aligned}
&\left(\mathcal{T}_f^*V_1\right)(s)-\left(\mathcal{T}_f^*V_2\right)(s)\\
&=\sum_a\pi_1(a|s)\left[r+\gamma\mathbb{E}_{s'}\left[V_1\left(s'\right)\right]-\alpha f\left(\frac{\pi_1(a|s)}{\mu(a|s)}\right)\right]-\max_\pi\sum_a\pi(a|s)\left[r+\gamma\mathbb{E}_{s'}\left[V_2\left(s'\right)\right]-\alpha f\left(\frac{\pi(a|s)}{\mu(a|s)}\right)\right]\\
&\leq\sum_a\pi_1(a|s)\left[r+\gamma\mathbb{E}_{s'}\left[V_1\left(s'\right)\right]-\alpha f\left(\frac{\pi_1(a|s)}{\mu(a|s)}\right)\right]-\sum_a\pi_1(a|s)\left[r+\gamma\mathbb{E}_{s'}\left[V_2\left(s'\right)\right]-\alpha f\left(\frac{\pi_1(a|s)}{\mu(a|s)}\right)\right]\\
&=\gamma\sum_a\pi_1(a|s)\mathbb{E}_{s'}\left[V_1\left(s'\right)-V_2\left(s'\right)\right]\leq\gamma\left\|V_1-V_2\right\|_\infty
\end{aligned}$$

By symmetry, it follows that for any state s in $\mathcal{S}$,

$$\left(\mathcal{T}_f^*V_1\right)(s)-\left(\mathcal{T}_f^*V_2\right)(s)\leq\gamma\left\|V_1-V_2\right\|_\infty$$

Therefore, it follows that

$$\left\|\mathcal{T}_f^*V_1-\mathcal{T}_f^*V_2\right\|_\infty\leq\gamma\left\|V_1-V_2\right\|_\infty$$

□

D Experimental Details

D4RL experimental details For MuJoCo locomotion and Kitchen tasks, we average mean returns over 10 evaluations every 5000 training steps, over 5 random seeds. For AntMaze tasks, we average over 100 evaluations every 0.1M training steps, over 5 random seeds. Followed by IQL, we standardize the rewards by dividing the difference in returns of the best and worst trajectories in MuJoCo and kitchen tasks, we subtract 1 to rewards in AntMaze tasks.

Our implementation of 10%BC is as follows, we first filter the top 10 $\%$ trajectories in terms of the trajectory return, and then run behaviour cloning on those filtered data. We re-run IQL on all datasets and report the score of IQL by choosing the best score from τ in $[0.5, 0.6, 0.7, 0.8, 0.9, 0.99]$, using author-provided implementation[†] We re-run CQL on AntMaze datasets as we find the performance can be improved by carefully sweeping the hyperparameter `min-q-weight` in $[0.5, 1, 2, 5, 10]$, using the PyTorch-version implementation[‡]. Other baseline results are taken directly from their corresponding papers.

In SQL and EQL, we use 2-layer MLP with 256 hidden units, we use Adam optimizer (Kingma & Ba, 2015) with a learning rate of $2 \cdot 10^{-4}$ for all neural networks. Following Mnih et al. (2013); Lillicrap et al. (2016), we introduce a target critic network with soft update weight $5 \cdot 10^{-3}$. We implement our method in the framework of JAX. The only hyperparameter α used in SQL and EQL is listed in Table 5. The sensitivity of α in SQL can be found in Table 3 and Table 4. The sensitivity of τ in IQL can be found in Table 3. The runtime of different algorithms can be found in Table 7.

Table 5: α used for SQL and EQL

Env	α (SQL)	α (EQL)
halfcheetah-medium-v2	2.0	2.0
hopper-medium-v2	2.0	2.0
walker2d-medium-v2	2.0	2.0
halfcheetah-medium-replay-v2	2.0	2.0
hopper-medium-replay-v2	2.0	2.0
walker2d-medium-replay-v2	2.0	2.0
halfcheetah-medium-expert-v2	5.0	5.0
hopper-medium-expert-v2	5.0	5.0
walker2d-medium-expert-v2	5.0	5.0
antmaze-umaze-v2	0.5	0.5
antmaze-umaze-diverse-v2	5.0	5.0
antmaze-medium-play-v2	0.5	0.5
antmaze-medium-diverse-v2	0.5	0.5
antmaze-large-play-v2	0.5	0.5
antmaze-large-diverse-v2	0.5	0.5
kitchen-c	2.0	2.0
kitchen-p	2.0	2.0
kitchen-m	2.0	2.0

Table 6: The sensitivity of τ in IQL.

τ	0.5	0.6	0.7	0.8	0.9	0.99
hopper-m-r	57.1	71.3	95.2	74.4	59.5	2.8
hopper-m-e	99.7	101.1	94.5	22.5	13.5	30.4
walker2d-m-r	74.3	76.1	72.3	41.7	20.3	4.3
walker2d-m-e	109.94	106.7	109.6	109.3	78.2	50.3
antmaze-m-d	0	0	2.5	51.6	71.0	12.1
antmaze-m-p	0	0	8.0	51.2	72.1	11.5
antmaze-l-d	0	0	1.2	12.4	47.5	7.6
antmaze-l-p	0	0	1.3	10.4	39.6	5.3

RL Unplugged experimental details We use d3rlpy (Seno & Imai, 2021), a modularized offline RL library that contains several SOTA offline RL algorithms and provides an easy-to-use wrapper

[†] https://github.com/ikostrikov/implicit_q_learning

[‡] https://github.com/young-geng/CQL

Table 7: The runtime of different algorithms.

algorithms	BC	10%BC	BCQ	DT	One-step	TD3+BC	CQL	IQL	SQL	EQL
runtime	20m	20m	60m	950m	20m	25m	80m	20m	20m	20m

for the offline Atari datasets introduced in (Agarwal et al., 2020). There are three types of Atari datasets in d3rlpy: `mixed`: datasets collected at the first 1M training steps of an online DQN agent, `medium`: datasets collected at between 9M steps and 10M training steps of an online DQN agent, `expert`: datasets collected at the last 1M training steps of an online DQN agent. To make the task more challenging, we use only 10% or 5% of origin datasets. We choose three image-based Atari games: Breakout, Qbert and Seaquest.

We implement the discrete version of SQL (D-SQL) and IQL (D-IQL) based on d3rlpy. The implementation of discrete CQL (D-CQL) and discrete BCQ (D-BCQ) are directly taken from d3rlpy. We use consistent preprocessing and network structures to ensure a fair comparision.

For baselines, we report the score of D-IQL by choosing the best score from τ in $[0.5, 0.7, 0.9]$, we report the score of D-CQL by choosing the best score from `min-q-weight` in $[1, 2, 5]$, we report the score of D-BCQ by choosing the best score from τ in $[0.1, 0.3, 0.5]$. For D-SQL, we use $\alpha = 1.0$ for all datasets.

Nosiy data regime experimental details In this experiment setting, we introduce the `noisy` dataset by mixing the `expert` and `random` dataset with different expert using MuJoCo locomotion datasets. The number of total transitions of the noisy dataset is $100,000$. We provide details in Table 8. We report the score of IQL by choosing the best score from τ in $[0.5, 0.6, 0.7, 0.8, 0.9]$.

Table 8: Noisy dataset of MuJoCo locomotion tasks with different expert ratios.

Env	Expert ratio	Total transitions	Expert transitions	Random transitions
Walker2d	1%	100,000	1,000	99,000
	5%	100,000	5,000	95,000
	10%	100,000	10,000	90,000
	20%	100,000	20,000	80,000
	30%	100,000	30,000	70,000
Halfcheetah	1%	100,000	1,000	99,000
	5%	100,000	5,000	95,000
	10%	100,000	10,000	90,000
	20%	100,000	20,000	80,000

Small data regime experimental details We generate the small dataset using the following psedocode 1, its hardness level can be found at Table 9. We report the score of CQL by choosing the best score from `min-q-weight` in $[0.5, 1, 2, 5, 10]$.

Listing 1: The sketch of generation procedure of small data regimes with different hard levels. Given an AntMaze environment and a hardness level, we discard some transitions by following the rule in the Coding List. Intuitively, the closer the transition is to the `GOAL`, the higher the probability that it will be discarded.

```
# LEVEL = {'easy', 'medium', 'hard'}
obs = dataset['observations']
length = dataset['observations'].shape[0]
POSITIONS = env.get_position(obs)
GOAL = env.get_goal()
MINIMAL_POSITION = env.get_minimal_position()
# get maximal Euclidean distance
MAX_EU_DIS = (GOAL - MINIMAL_POSITION)**2
DIS = ((POSITIONS - MINIMAL_POSITION)**2) / MAX_EU_DIS
save_idx = np.random.random(size=length) > (DIS * hardness['LEVEL'])
small_data = collections.defaultdict()
for key in dataset.keys():
   small_data[key] = dataset[key][save_idx]
```

Table 9: Details of small data regimes with different task difficulties.

Dataset (AntMaze)		Hardness	Total transitions	Reward signals
medium-play	Vanilla	NA	100, 000	10,000
	Easy	0	56,000	800
	Medium	0.07	48,000	150
	Hard	0.1	45,000	10
large-play	Vanilla	NA	100,000	12500
	Easy	0	72,000	5,000
	Medium	0.3	42,000	1,000
	Medium	0.35	37,000	500
	Hard	0.38	35,000	100

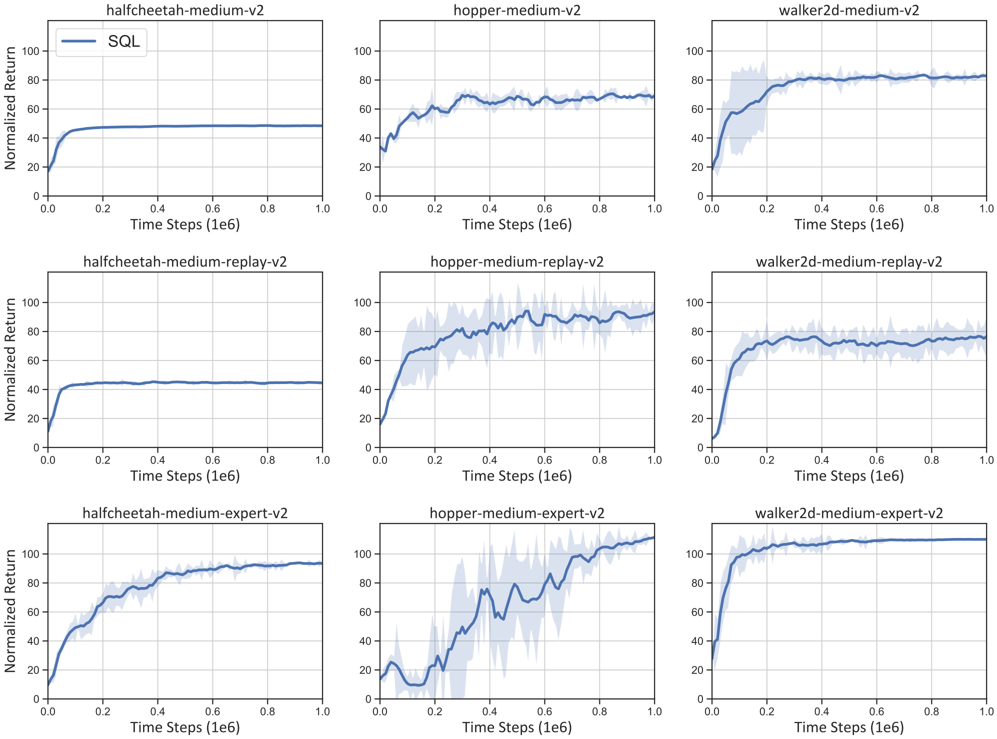

Figure 5: Learning curves of SQL on D4RL MuJoCo locomotion datasets.

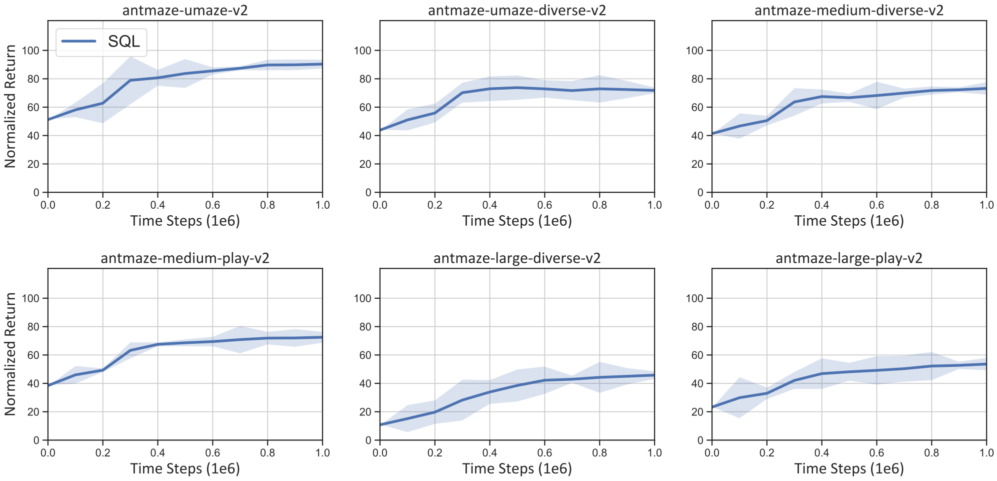

Figure 6: Learning curves of SQL on D4RL AntMaze datasets.

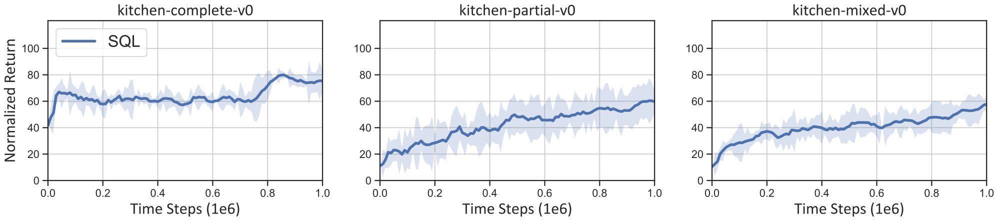

Figure 7: Learning curves of SQL on D4RL Kitchen datasets.

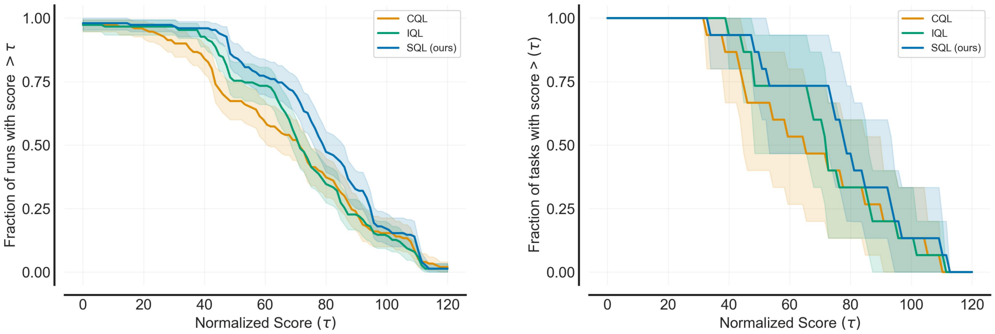

Figure 8: Performance profiles of CQL, IQL and SQL generated by the rliable library (Agarwal et al., 2021) on D4RL datasets based on score distributions (left), and average score distributions (right). Shaded regions show pointwise 95% confidence bands based on percentile bootstrap with stratified sampling.

Table 10: Performance in setting with 10% (top) and 5% (bottom) Atari dataset. SQL achieves the best performance in 10 out of 12 games.

Task	D-BCQ	D-IQL	D-CQL	D-SQL
breakout-medium-v0 (10%)	3.5	20.1	15.1	28.0
qbert-medium-v0	395.4	3717.5	4141.5	5213.4
seaquest-medium-v0	438.1	404.9	359.4	465.3
breakout-mixed-v0	8.1	10.9	9.3	13.3
qbert-mixed-v0	557.5	1000.3	890.2	1244.3
seaquest-mixed-v0	300.3	326.1	337.5	330.4
breakout-medium-v0 (5%)	1.6	13.9	13.3	16.5
qbert-medium-v0	301.6	2788.7	3147.3	2970.6
seaquest-medium-v0	301.9	294.5	272.5	361.3
breakout-mixed-v0	5.3	10.5	8.7	11.8
qbert-mixed-v0	576.4	931.1	925.3	965.0
seaquest-mixed-v0	275.5	292.2	321.4	336.7