Proceedings of Machine Learning Research vol 242:1–14, 2024

SpOiLer: Offline Reinforcement Learning using Scaled Penalties

Padmanaba Srinivasan PS3416@IMPERIAL.AC.UK and
William Knottenbelt WJK@IMPERIAL.AC.UK
Department of Computing, Imperial College London, London, United Kingdom

Abstract

Offline Reinforcement Learning (RL) is a variant of off-policy learning where an optimal policy must be learned from a static dataset containing trajectories collected by an unknown behavior policy. In the offline setting, standard off-policy algorithms will overestimate values of out-of-distribution actions, and a policy trained naïvely in this way will perform poorly in the environment due to distribution shift between the implied and real environment; this is especially likely when modeling complex and multimodal data distributions. We propose **S**caled-**p**enalty **O**ff**li**ne **Le**a**r**ning (SpOiLer), an offline reinforcement learning algorithm that reduces the value of out-of-distribution actions relative to observed actions. The resultant pessimistic value function is a lower bound of the true value function and manipulates the policy towards selecting actions present in the dataset. Our method is a simple augmentation to the standard Bellman backup operator and implementation requires around 15 additional lines of code over soft actor–critic. We provide theoretical insights into how SpOiLer operates under the hood and show empirically that SpOiLer achieves remarkable performance against prior methods on a range of tasks.

Keywords: off-policy reinforcement learning, offline reinforcement learning

1. Introduction

Reinforcement Learning (RL) is a field of machine learning that trains a policy to complete a task via iterative improvement (Sutton and Barto, 2018; Arulkumaran et al., 2017). Typically, the RL setting assumes *online* access to an environment enabling exploration and the ability to gather experience directly and improve using the current policy (on-policy) or a mixture of previous policies (off-policy). However, online interaction can be impractical in many scenarios where it carries substantial expense or risk (Garcıa and Fernández, 2015; Lange et al., 2012). In such situations, a dataset of interactions produced by one or more, potentially suboptimal, behavioral policies (e.g. human demonstrations) may exist and can be used for learning instead(Lange et al., 2012); algorithms that can learn using a static dataset are termed *batch* or *offline* reinforcement learning algorithms.

In online settings, a policy can explore the entire action space, relying on feedback from the environment to evaluate the action and correct overestimated action-values. However, in offline RL, only a subset of actions are explored and the lack of evaluation of out-of-distribution (OOD) actions can result in extrapolation error (Fujimoto et al., 2019) and subsequent overestimation of the value (Kumar et al., 2019, 2020). Therefore, offline RL algorithms must employ mechanisms to conservatively estimate the values of OOD actions or constrain the support of actions (Arulkumaran et al., 2017).

Policy regularization constitutes a large portion of the work in offline RL in which a learned policy is constrained to produce actions similar to those in the dataset (Wu et al., 2019). This involves either ensuring that the policy can only select actions with sufficient support under the dataset state-occupancy distribution (Fujimoto et al., 2019) or applying additional distribution matching objectives on the policy (Jaques et al., 2019; Kumar et al., 2019). Such approaches are usually straightforward to implement, but do not directly address how the critic treats OOD values. As a result, the critic can still easily overestimate the values of OOD actions that subsequently dominate the policy constraint (Kostrikov et al., 2021a). Alternative policy regularization approaches perform weighted regression (Peng et al., 2019; Nair et al., 2020; Siegel et al., 2020; Wang et al., 2020, 2018).

Critic regularization methods are another offline RL paradigm which implicitly force the actor to select in-distribution actions. The critic is encouraged to be pessimistic about OOD actions (Kostrikov et al., 2021b) via additional objectives used to explicitly train the critic as an energy-based model (Kumar et al., 2020; Kostrikov et al., 2021a) or by penalizing OOD actions (Gulcehre et al., 2021; Li et al., 2022). The latter approach applies fixed penalties to OOD action-values that do not take into account how different an OOD action is from the observed one.

Contributions In this work, we propose SpOiLer, an alternative approach to critic regularization. We modify the standard soft Bellman backup operator to include the scaled log density of the behavior policy in Section 3.1. This results in a *proportional penalty* that becomes more pessimistic further from the dataset-action support that effectively curtails value overestimation without excessive pessimism. In Section 3.2 we show that our augmentation is equivalent to implicitly minimizing the Kullback–Leibler (KL) divergence between the learned policy and behavior policy and we interpret this to yield additional insight into the inner workings of SpOiLer. In Section 4 we evaluate SpOiLer on a range of offline tasks and datasets and compare it with previous approaches to demonstrate its high performance on a range of tasks. We also perform additional experiments to analyze empirical performance.

2. Related Work

A reinforcement learning task is a decision-making problem in an environment represented as a Markov decision process (MDP) $\{\mathcal{S}, \mathcal{A}, p_0(s), p(s'|s,a), r(s,a), \gamma\}$, where $\mathcal{S}$ is the state-space, $\mathcal{A}$ is the action-space, $p_0(s)$ is the initial state distribution, $p(s'|s,a)$ is the state transition distribution, $r(s,a)$ is the reward function and $\gamma \in [0,1)$ is the discount factor. The objective in RL is to find a policy $\pi \in \Pi$ that maximizes the expected cumulative discounted return:

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

The optimal state-value (or Q) function $Q^*(s,a)$ measures the expected return when taking action a in state s, assuming that subsequent actions are from π^*. The optimal policy is obtained through a greedy objective that selects the action such that $\pi^*(s) = \arg\max_{\pi} \; Q^*(s,a) \forall s \in \mathcal{S}$. Our work focuses on continuous action spaces with $\mathcal{A} \in \mathbb{R}^d$ for some positive integer d.

The optimal Q function is learned using the Q-learning algorithm (Watkins and Dayan, 1992; Hasselt, 2010) by repeated applications of the Bellman optimality operator defined as $\mathcal{T}Q(s,a) = r(s,a) + \gamma \max_{\pi} Q(s',a')$. In continuous action spaces, the Q function can be learned iteratively using a suitable function approximator $Q_\theta(s,a)$. The parameters θ of the model are learned by minimizing the mean-squared Bellman error on samples from a buffer that contains past trajectories, and a target Q function (Mnih et al., 2015). When solving an RL task in a continuous action-space, the analytic maximum is intractable so a separate actor network π_ϕ is trained to maximize the value function (Silver et al., 2014; Haarnoja et al., 2018; Lillicrap et al., 2015; Fujimoto et al., 2018).

Updating the Q function uses Temporal Difference (TD) learning with a target function $Q_{\bar{\theta}}$ and minimizes:

$$J_Q(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}[Q_\theta(s,a) - (r + \gamma Q_{\bar{\theta}}(s', \pi_\phi(s)))]^2. \tag{1}$$

The policy is updated by minimizing the negative estimated reward:

$$J_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}[-Q_\theta(s, \pi_\phi(s))]. \tag{2}$$

2.1. Offline Reinforcement Learning

In an offline setting, the agent cannot generate new experiences and instead learns from interactions collected in a dataset $\mathcal{D}$ consisting of tuples $\{s, a, r, s'\}$ produced by behavior policy rollouts, π_β, in the environment.

A key limitation facing offline RL is that the dataset does not provide complete coverage of all possible state-action pairs. As only a subset of the full state-action space, $\mathcal{S} \times \mathcal{A}$, is covered by the dataset, standard off-policy reinforcement learning algorithms (Haarnoja et al., 2018; Lillicrap et al., 2015; Fujimoto et al., 2018) will overestimate OOD action-values (Fujimoto et al., 2019; Kumar et al., 2019) and the resulting learned policy may choose actions that fall outside the dataset support. If the agent takes OOD actions to maximize its reward during training, applying the same action to a real environment can lead to catastrophic error due to the distributional shift between the (reward) environment implied by the Q function and the real environment.

Overcoming issues caused by extrapolation in offline RL involves: 1) policy regularization that encourages the learned policy to select actions such that deviation from observed state-action pairs is minimized; or 2) critic regularization that attempts to indirectly prevent selection of OOD actions by decreasing the Q values for such actions. We discuss both kinds of regularization in turn.

Policy Regularization This subset of offline RL directly constrains the actor to select actions present in the dataset. The simplest form adds a supervised component to the actor objective either as a standalone component (Fujimoto and Gu, 2021) or as reward-weighted behavioral cloning (BC) (Nair et al., 2020; Kostrikov et al., 2021b; Peng et al., 2019).

Constraints can be applied more firmly by (architecturally) limiting the action-space to lie within a small neighborhood by training the policy as a deterministic model followed by a perturbation model (Fujimoto et al., 2019; Kumar et al., 2019). These approaches train the deterministic model as a variational autoencoder (Kingma and Welling, 2013; Fujimoto et al., 2019) that minimizes KL divergence, or by minimizing the maximum mean discrepancy (MMD) (Gretton et al., 2012; Kumar et al., 2019). The deterministic model is thus trained as a density model and is fitted to the offline dataset. Actions from this model are passed to the perturbation model, which slightly adjusts the action to maximize the Q value. This approach typically requires extensive sampling from both the actor and critic models and may be too restrictive (Ghasemipour et al., 2021).

Wu et al. (2019) and Jaques et al. (2019) use a simpler critic with regularization applied as a probability divergence minimizing task, such as the KL divergence.

Policy regularization approaches can be intuitive; however, they apply an additional artificial constraint on the actor. The underlying problem of OOD state-action overestimation remains as the Q function receives no corrective learning signal regarding OOD actions.

Critic Regularization In an offline setting where limited or no exploration is possible, the Q function can overestimate values for OOD actions, resulting in poor real-world performance.

Consequently, regularizing the critic to be pessimistic about OOD actions can improve performance without having to place explicit constraints on the actor. Nachum et al. (2019) and Kumar et al. (2020) incorporate divergence-based regularization into the critic objective; Nachum et al. (2019) introduce penalty terms to reduce the Q values for actions sampled from the policy and Kumar et al. (2020) sample Q values and perform approximate numerical integration to train the Q function as an energy-based model. Kostrikov et al. (2021a) follow a similar idea using the Fisher divergence and demonstrate equivalence to Kumar et al. (2020).

Kostrikov et al. (2021b) train the value function to learn an expectile over returns and can be tuned to learn a value function that lies between the conservatism of SARSA Sutton and Barto (2018) and the greediness of Q-learning while remaining fully in-sample with respect to actions. Rezaeifar et al. (2022) takes a two-pronged perspective on regularization, firstly, by minimizing deviation from the dataset actions using a variational autoencoder-based actor model and secondly, by applying a penalty term to the critic objective for OOD actions. This facilitates learning by approximating a lower bound of Q^*. The actor is trained to maximize advantage with a reward-weighted imitation objective.

Wu et al. (2021) build on BEAR (Kumar et al., 2019) to penalize the variance between OOD estimates of Q functions, estimated using MC dropout (Gal and Ghahramani, 2016). Li et al. (2022) extend CQL (Kumar et al., 2020) by considering the difference in values between successive states, computing the mean and variance and using these to identify OOD actions and penalizing via a meta-function.

3. SpOiLer

We now describe our approach to critic regularization in the offline RL setting. Our algorithm uses a value penalization method that offers advantages over previous approaches, namely (1) computational efficiency even with sampling from the critic and (2) performing **implicit** KL regularization with strong theoretical guarantees (Vieillard et al., 2020b,a). Furthermore, Kumar et al. (2020) show that value penalization methods such as SpOiLer guarantee that the learned value function $\hat{Q}^\pi$ is a lower bound on the true value function Q^π and that conservative updates lead to safe policy improvement $J(\pi_{\text{CQL}}) \geq J(\pi_\beta) - \mathcal{O}(\frac{1}{(1-\gamma)^2})$ (Kumar et al. (2020) Theorem 3.6).

3.1. Conceptual Derivation

In offline RL, the dataset consists of several state-action tuples (s, a) for which a Q value can be learned by temporal difference learning. For the actor to avoid OOD actions, given the state s, all OOD actions a_d must have a lower action-value than the observed action a:

$$Q(s,a) > Q(s,a_d), \quad a_d \in \mathcal{A}, a_d \neq a \tag{3}$$
$$\text{Therefore} \quad Q(s,a) + \delta = Q(s,a_d), \quad \delta < 0 \tag{4}$$

where δ is an offset that ensures the condition in Equation 3 is met.

The problem, however, is that when evaluating OOD actions, the Q function will overestimate values leading to errors accumulating when bootstrapping estimates (Sutton and Barto, 2018). Kumar et al. (2020) and Kostrikov et al. (2021a) address this by adding an additional critic objective; the former relies on expensive sampling and numerical integration and the latter needs a gradient penalty to be computed; ultimately both implement an additional score-based objective independent of Equation 1.

If an optimal policy π^* is known, then the log density of the behavior policy (hereafter, referred to as the log-policy) approaches $-\infty$ at OOD actions. In offline RL, the action support is determined by the samples in the dataset; hence, constraining using the behavioral policy will yield a policy that performs at least as well as the behavioral policy. The behavior policy can be estimated (using a BC-trained policy) and so the log-policy is a powerful tool we can use to penalize actions via a simple augmentation to the TD update rule. Augmenting with the log-policy can act as a proportional penalty for OOD actions such that their value is less than that of in-dataset actions.

We apply our approach to Soft Actor Critic (SAC) (Haarnoja et al., 2018) which optimizes a stochastic policy in an off-policy way. SAC maximizes entropy using entropy regularization to encourage exploration and uses clipped double Q learning (Hasselt, 2010) for added stability. Our approach can be expressed as a modified Bellman backup operator:

$$\mathcal{T}^S Q(s,a) \triangleq y_t + \alpha\tau \log \pi_\beta(a|s) = r(s,a) + \alpha\tau \log \pi_\beta(a|s) + \mathbb{E}_{s'}[V(s')] \tag{5}$$

where the log-policy is highlighted with $0 \leq \alpha \leq 1$, $V(s) = \mathbb{E}_{a\sim\pi}[Q(s,a) - \log \pi(a|s)]$ is the soft state-value function (Haarnoja et al., 2018) and $y_t = r(s,a) + \mathbb{E}_{s'}[V(s')]$ is the unmodified TD target. π_β is the behavioral policy estimate learned using BC.

The SAC TD target, that is, the target produced by the soft Bellman backup operator, is given as:

$$Q(s,a) = r(s,a) + \gamma \sum_{a'\in\mathcal{A}} \pi(a'|s')(Q(s',a') - \tau \log \pi(a'|s')) \tag{6}$$

where τ is a temperature parameter that scales entropy. When presented with OOD actions, we want to satisfy the inequality in Equation 3 to ensure that no OOD action achieves a higher action-value than an observed action. We rewrite Equation 5 the TD target as follows:

$$Q(s,a) = r(s,a) + \alpha\tau \log \pi_\beta(a|s) + \gamma \sum_{a'\in\mathcal{A}} \pi(a'|s')(Q(s',a') - \tau \log \pi(a'|s')). \tag{7}$$

When $\alpha = 0$ we recover the standard soft Bellman operator in Equation 6. The log-policy is adjusted to be less than 0 for OOD actions which, with iteration, will drive down their action values.

3.2. Why does SpOiLer work?

We now demonstrate *why* SpOiLer works. We begin with the definition:

$$Q'(s,a) \triangleq Q(s,a) - \alpha\tau \log \pi_\beta(a|s). \tag{8}$$

Equation 7 can be written as:

$$Q(s,a) - \alpha\tau \log \pi_\beta(a|s) = r(s,a) + \gamma \sum_{a'\in\mathcal{A}} \pi(a'|s')(Q(s',a') - \tau \log \pi(a'|s')) \tag{9}$$

We can rearrange Equation 9 to show that adding the log-policy implicitly factors in the KL divergence and entropy.

Theorem 1 *Equation 9 can be rewritten using the substitution of Equation 8 as:*

$$Q'(s,a) = r(s,a) - \gamma(\alpha\tau D_{KL}(\pi||\pi_\beta) + (1-\alpha)\tau\mathcal{H}(\pi)) + \gamma \sum_{a'\in\mathcal{A}} \pi(s',a')Q'(s',a')). \tag{10}$$

Proof We provide an outline:

$$\begin{aligned} Q'(s,a) &= r(s,a) + \gamma \sum_{a'\in\mathcal{A}} \pi(a'|s')(Q(s',a') - \tau\log\pi(a'|s') - \alpha\tau\pi_\beta(a'|s') + \alpha\tau\pi_\beta(a'|s')) \\ &= r(s,a) + \gamma \sum_{a'\in\mathcal{A}} \pi(a'|s')(Q'(s',a') - \tau\log\pi(a'|s') + \alpha\tau\pi_\beta(a'|s')) \\ &= r(s,a) - \gamma\tau(\alpha D_{KL}(\pi||\pi_\beta) + (1-\alpha)\mathcal{H}(\pi)) + \gamma \sum_{a'\in\mathcal{A}} \pi(a'|s')Q'(s',a'). \end{aligned}$$

■

Similarly for the policy, the actor aims to maximize the reward and entropy:

$$\pi^* = \underset{\pi, a\sim\pi}{\arg\max} \quad Q(s,a) + \tau\mathcal{H}(\pi). \tag{11}$$

Applying the definition from Equation 8 and rearranging, we can show this implicitly operates on both KL divergence and entropy.

Theorem 2 *Equation 11 can be rewritten as:*

$$\pi^* = \underset{\pi, a\sim\pi}{\arg\max} \quad Q'(s,a) - \alpha\tau D_{KL}(\pi(a|s)||\pi_\beta(a|s)) + (1-\alpha)\tau\mathcal{H}(\pi) \tag{12}$$

Proof We begin with the policy objective in Equation 11 and use the substitution in 8:

$$\begin{aligned} \pi^* &= \underset{\pi, a\sim\pi}{\arg\max}\, Q(s,a) - \tau \sum_{a\in\mathcal{A}} \pi(a|s)\log\pi(a|s) \\ &= \underset{\pi, a\sim\pi}{\arg\max}\, Q(s,a) - \sum_{a\in\mathcal{A}} \pi(a|s)(\alpha\tau\log\pi_\beta(a|s) - \alpha\tau\log\pi_\beta(a|s) + \tau\log\pi(a|s)) \\ &= \underset{\pi, a\sim\pi}{\arg\max}\, Q'(s,a) - \sum_{a\in\mathcal{A}} \pi(a|s)(\alpha\tau\log\pi(a|s) + (1-\alpha)\tau\log\pi(a|s) - \alpha\tau\log\pi_\beta(a|s)) \\ &= \underset{\pi, a\sim\pi}{\arg\max}\, Q'(s,a) - \alpha\tau D_{KL}(\pi(a|s)||\pi_\beta(a|s)) + (1-\alpha)\tau\mathcal{H}(\pi) \end{aligned}$$

■

From the critic perspective in Equation 10 it is clear that adding the log-policy is equivalent to scaling the SAC target by the KL divergence and from Equation 12, we see that the actor will both maximize the expected cumulative reward under the KL penalty weighted by a factor of $\alpha\tau$.

In SAC and related extensions to offline RL (Haarnoja et al., 2018; Kumar et al., 2020) the temperature parameter that scales entropy regularization (τ) is often automatically tuned via Lagrangian dual gradient descent. In SpOiLer, we hold τ constant to ensure the scaling of the log-policy does not change.

Proposition 3 (SpOiLer lower bounds the true value function) *We denote $Q^{SpOiLer}$ as the unique fixed point learned by repeatedly applying the operator $\mathcal{T}^S$. For any action $a \in \mathcal{A}$ we have $Q^{SpOiLer} \leq Q^*$ where Q^* is the Q function of the optimal policy in the batch.*

Proof For $a \in \text{Support}(\pi_\beta)$ it is easy to show that $Q^{\text{SpOiLer}} = Q^*(s, a)$. For $a \notin \text{Support}(\pi_\beta)$ we follow a similar procedure:

$$\begin{aligned}
Q^{\text{SpOiLer}} &= \mathcal{T}^S Q(s, a) && (13)\\
&= r(s, a) + \alpha\tau \log \pi_\beta(a|s) + \gamma \mathbb{E}_{s'}[\max_{a'} Q(s', a')] && (14)\\
&< r(s, a) + \gamma \mathbb{E}_{s'}[\max_{a'} Q(s', a')] && (15)\\
&= \mathcal{T} Q(s, a) = Q^*(s, a). && (16)
\end{aligned}$$

■

By carefully choosing the properties of the scaling function, it is possible to control the pessimism of SpOiLer.

Proposition 4 (SpOiLer is lower bounded by the value function of the behavior policy) *We denote $Q^{SpOiLer}$ to be the unique fixed point learned by repeatedly applying the operator $\mathcal{T}^S$. By controlling the scaling function such that $0 \leq -\alpha\tau \log \pi_\beta \leq \gamma \mathbb{E}_{s'}[\max_{a'} Q(s, a) - \mathbb{E}_{a'}[Q(s, a)]]$, for any action $a \in \mathcal{A}$ we have $Q^{SpOiLer} \geq Q^\mu$ where Q^μ is the Q function of the behavior policy.*

Proof For $a \notin \text{Support}(\pi_\beta)$ and α, τ such that $0 \leq -\alpha\tau \log \pi_\beta \leq \gamma \mathbb{E}_{s'}[\max_{a'} Q(s, a) - \mathbb{E}_{a'}[Q(s, a)]]$:

$$\begin{aligned}
Q^{\text{SpOiLer}} &= \mathcal{T}^S Q(s, a) && (17)\\
&= r(s, a) + \alpha\tau \log \pi_\beta(a|s) + \gamma \mathbb{E}_{s'}[\max_{a'} Q(s', a')] && (18)\\
&\geq r(s, a) + \gamma \mathbb{E}_{s'}[\mathbb{E}_{a'}[Q(s', a')]] && (19)\\
&= \mathcal{T}^\mu Q(s, a) = Q^\mu(s, a). && (20)
\end{aligned}$$

■

Propositions 3 and 4 together show that SpOiLer will perform at least as well as the behavior policy and that it approximates the optimal batch-constraint policy.

In the in-sample case the target y_t in Equation 5 is known since the dataset contains the next state s'. When sampling OOD actions using model-free algorithms, the next state is unknown and we resort to an approximation of y_t using an average over action-values estimated by the ensemble of K Q functions: $\hat{\mathcal{T}}_s Q(s, a_{\text{ood}}) = \alpha\tau \log \pi_\beta(a|s) + \frac{1}{K} \sum_{k=1}^{K} Q^k(s, a_{\text{ood}})$ where gradients are not propagated through the second term.

3.3. SpOiLer and M-DQN

Although on the surface, our approach performs a similar augmentation to Munchausen-DQN (M-DQN) (Vieillard et al., 2020b), the learning algorithm and implementation differ greatly. M-DQN is designed for discrete action-space tasks with the augmentation used to implicitly regularize between successive policies. SpOiLer uses augmentation to construct pseudo-targets for OOD actions that take into account how different an OOD action is from an observed one. Furthermore, SpOiLer builds on soft actor–critic to learn in a continuous action space and requires sampling of OOD actions to regularize the critic. M-DQN performance in an offline setting has been empirically evaluated (Liu et al., 2021) where it rarely outperforms BCQ, whereas SpOiLer consistently outperforms BCQ.

3.4. Links to BRAC

From Equations 10 and 12, it is clear that setting $\alpha = 1$ yields the primal form of BRAC/BRAC+ (Wu et al., 2019; Zhang et al., 2021) using KL divergence. BRAC uses either value or policy regularization, the latter of which is reminiscent of KL control policies (Jaques et al., 2019). BRAC's value regularization method augments the target with the explicit KL divergence of the next action, which is distinctly different from SpOiLer which regularizes using the current action and behavior policy only and in combination with entropy regularization, yields an implied KL divergence with the next action.

We justify the constraint $0 \leq \alpha \leq 1$ as setting $\alpha = 0$ recovers SAC (losing KL regularization) and $\alpha = 1$ yields BRAC (losing the benefits of entropy regularization (Eysenbach and Levine, 2021)). From Equations 10 and 12, it is clear that with $\alpha > 1$ the actor will minimize entropy and KL divergence leading to an increasingly deterministic policy. On the other hand, $\alpha < 0$ will maximize entropy and KL divergence.

3.5. Implementation

We instantiate the empirical behavior policy π_β as a Gaussian policy and train using supervised learning for 500k steps prior to actor-critic training. Our assumptions at this stage are that the true behavior policy is unimodal and the the estimated behavior policy is a good approximation of the true policy.

Adding the scaled log-policy for continuous action spaces can cause numerical issues with high-dimensional actions as the log-policy is unbounded. Limiting the log-policy using a scaling function $g(\log \pi_\beta)$ can alleviate this. We instantiate $g(\cdot)$ as a function that firstly subtracts the log-policy of the dataset action which upper bounds the penalty at 0 and clips the lower bound to be no lower than -1 (Vieillard et al., 2020b). We find that $\alpha = 0.9$ and $\tau = 0.1$ perform well and use these parameters for all experiments. We draw 10 samples from the policy and penalize their corresponding values using SpOiLer's log-policy augmentation penalty.

4. Experiments

In this section, we conduct experiments to evaluate SpOiLer on several D4RL (Fu et al., 2020) tasks. Our evaluation focuses on continuous action space tasks from the MuJoCo, Maze, FrankaKitchen (Gupta et al., 2019) and Adroit (Rajeswaran et al., 2017) domains.

We compare against: BEAR (Kumar et al., 2019), BRAC (Wu et al., 2019), CQL (Kumar et al., 2020), BCQ (Fujimoto et al., 2019), IQL (Kostrikov et al., 2021b), TD3+BC (Fujimoto and Gu, 2021), TD3-CVAE (Rezaeifar et al., 2022), PessORL (Peng et al., 2019), SAW (Lyu et al., 2022), and behavioral cloning (BC). Results for BEAR, BRAC, BC and CQL are reported from Fu et al. (2020) and results for other algorithms are from their respective papers.

Table 1: Normalized scores for SpOiLer and other methods in the Gym (top) and Adroit (bottom) domains. For SpOiLer, we compute the average normalized score and standard deviation over 4 seeds. Top two scores are highlighted in bold.

Task	BC	BCQ	BRAC	BEAR	CQL	TD3+BC	TD3+CVAE	IQL	PessORL	SAW	SpOiLer
halfcheetah-m	36.1	40.7	43.8	41.7	44.4	42.8	43.2	47.4	**49.2**	47.5	**48.7 (0.3)**
walker2d-m	6.6	53.1	**81.1**	59.1	79.2	79.7	68.2	78.3	75.6	74.8	**79.8 (6.4)**
hopper-m	29.0	54.5	32.7	52.1	58.0	**99.5**	55.9	66.3	76.4	**95.4**	52.0 (9.3)
halfcheetah-m-r	38.4	38.2	**47.7**	38.6	46.2	43.3	45.3	44.2	-	43.9	**50.6 (0.5)**
walker2d-m-r	11.3	15.0	0.9	19.2	26.7	25.2	15.4	**73.9**	-	**58.0**	42.6 (7.8)
hopper-m-r	11.8	33.1	0.6	33.7	48.6	31.4	46.7	**94.7**	-	**97.3**	66.9 (11.5)
halfcheetah-m-e	35.8	64.7	44.2	53.4	62.4	**97.9**	86.1	86.7	24.3	**89.9**	64.4 (15.8)
walker2d-m-e	6.4	57.5	81.6	40.1	**111.0**	101.1	84.9	109.6	89.7	107.2	**112.5 (0.6)**
hopper-m-e	111.9	110.9	1.9	96.3	98.7	**112.2**	111.6	91.5	**112.8**	90.0	106.0 (8.0)
halfcheetah-e	**105.2**	-	3.8	-	104.8	**105.7**	-	95.0	71.3	95.4	101.9 (5.9)
walker2d-e	56.0	-	-0.2	-	**109.9**	105.7	-	109.4	109.4	103.5	**112.1 (0.7)**
hopper-e	**111.5**	-	6.6	-	103.9	**112.2**	-	109.9	110.7	102.6	106.4 (7.6)
pen-human	34.4	68.9	8.1	-1.0	37.5	-	59.2	**71.5**	63.1	-	**93.0 (16.2)**
hammer-human	1.5	0.5	0.3	0.3	**4.4**	-	0.2	1.4	4.2	-	**9.7 (6.6)**
door-human	0.5	-0.0	-0.3	-0.3	**9.9**	-	0.0	4.3	2.3	-	**9.2 (6.7)**
relocate-human	0.0	-0.1	-0.3	-0.3	**0.2**	-	0.0	0.1	**0.3**	-	**0.2 (0.2)**
pen-cloned	**56.9**	44.0	1.6	26.5	39.2	-	45.4	37.3	39.0	-	**66.4 (18.4)**
hammer-cloned	0.8	0.4	0.3	0.3	**2.1**	-	0.3	**2.1**	1.0	-	**3.4 (2.2)**
door-cloned	-0.1	0.0	-0.1	-0.1	0.4	-	0.0	1.6	**1.7**	-	**2.2 (0.5)**
relocate-cloned	**-0.1**	-0.3	-0.3	-0.3	**-0.1**	-	-0.3	-0.2	-0.3	-	**-0.1 (0.1)**

4.1. Results on Datasets

Gym In this domain, we evaluate SpOiLer on the locomotion environments in MuJoCo. Results are summarized in Table 1 (top) with SpOiLer exhibiting strong performance across all datasets. SpOiLer achieves the best, or near-best performance on the Walker2D and Halfcheetah tasks.

Adroit The adroit domain poses more challenging tasks where a simulated robotic hand with 24-DoF must complete object manipulation tasks. In Table 1 (bottom) SpOiLer achieves the best performance across all datasets, with large improvements over other methods in pen-human, hammer-human and pen-cloned.

FrankaKitchen This environment consists of a robot with 9 DoF that must perform tasks in a kitchen in a particular sequence to receive the full reward. In Table 2, SpOiLer outperforms all models in the -partial and -mixed datasets and consistently completes over 50% of sub-tasks.

Table 2: Normalized scores for SpOiLer and prior methods in FrankaKitchen. Top two scores are highlighted in bold.

Algorithm	-complete	-partial	-mixed
BC	33.8	33.8	47.5
BCQ	8.1	18.9	8.1
BRAC	0.0	0.0	0.0
BEAR	0.0	13.1	47.2
CQL	43.8	**49.8**	**51.0**
IQL	**62.5**	46.3	**51.0**
SpOiLer	**52.5**	**50.1**	**56.2**

Table 3: Normalized scores for SpOiLer and prior methods in the Maze2D environment. Top two scores are highlighted in bold.

Algorithm	-umaze	-medium	-large
BC	3.8	**30.3**	5.0
BCQ	**12.8**	8.3	6.2
BRAC	4.7	**33.8**	**40.6**
BEAR	3.4	29.0	4.6
CQL	5.7	5.0	**12.5**
SpOiLer	**15.6**	10.7	11.8

Maze2D The Maze environment involves moving a ball in a maze. Actor constraint methods tend to outperform critic-regularized methods. SpOiLer is able to make excellent progress on these tasks, achieving the best performance in -umaze and doubling CQL's performance in -medium.

4.2. Ablations

We present ablations showing the effect of τ and α in Figure 1. Increasing τ increases the weighting of the KL and entropy terms compared to the value objective and reduces performance. This also highlights why we used a fixed α in SpOiLer; an automatically tuned, changing τ will change the magnitude of penalty term during training leading to instability. Increasing α increases the relative weighting of KL to entropy and generally improves performance.

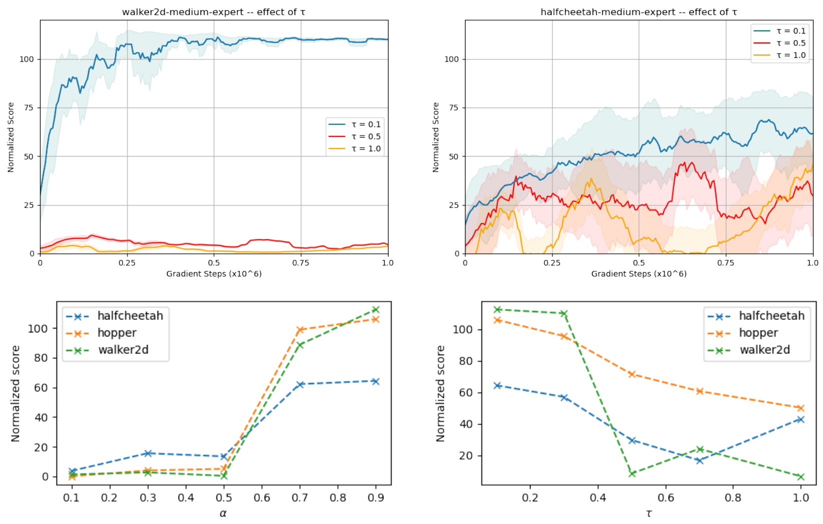

Figure 1: The effect of τ and α on learning. Top: learning curves on walker2d-m-e and halfcheetah-m-e for $\tau = \{0.1, 0.5, 1.0\}$. Bottom: ablations for values of α and τ with $\times$ marks at evaluated values. Bottom left: scores for of α with $\tau = 0.1$. Bottom right: scores for values of τ with $\alpha = 0.9$.

4.3. Limitations

Our method relies on sampling to reduce the overestimation of OOD values. This adds computational cost compared to standard SAC, leading to marginally longer training times than the actor-constraining IQL and TDC+BC. Compared to CQL, SpOiLer does not need to sample as extensively; it avoids being excessively pessimistic about OOD actions while training significantly faster. SpOiLer performs well on both mixed-policy and expert datasets. In the other domains with sparse rewards, the performance advantage of all offline RL methods is diminished and closer to BC.

5. Conclusion

This paper presented SpOiLer, an algorithm that performs offline reinforcement learning using critic regularization. Our algorithm is simple, requiring 10-15 lines of code on top of soft actor–critic to prevent overestimation of values for out-of-distribution actions. We have shown that our method implicitly minimizing KL divergence between the the current and behavior policies and that it lower bounds the true value function. We have evaluated SpOiLer on various offline benchmark tasks and show that our method is competitive with prior algorithms.

References

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

Benjamin Eysenbach and Sergey Levine. Maximum entropyRL (provably) solves some robust RL problems. *arXiv preprint arXiv:2103.06257*, 2021.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018. ISBN 2640-3498.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019. ISBN 2640-3498.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max Q-learnarning operator for simple yet effective offline and online R. In *International Conference on Machine Learning*, pages 3682–3691. PMLR, 2021. ISBN 2640-3498.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012. ISSN 1532-4435.

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

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. ISBN 2640-3498.

Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with Fisher divergence critic regularization. In *International Conference on Machine Learning*, pages 5774–5783. PMLR, 2021a. ISBN 2640-3498.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learnarning. *arXiv preprint arXiv:2110.06169*, 2021b.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy Q-learnarning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learnarning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. *Batch reinforcement learning*, pages 45–73. Springer, 2012.

Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Dealing with the unknown: Pessimistic offline reinforcement learning. In *Conference on Robot Learning*, pages 1455–1464. PMLR, 2022. ISBN 2640-3498.

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. *arXiv preprint arXiv:1509.02971*, 2015.

Hsin-Yu Liu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong. Offline reinforcement learning with Munchausen regularization. In *Offline Reinforcement Learning Workshop at Neural Information Processing Systems*, volume 2021, 2021.

Jiafei Lyu, Aicheng Gong, Le Wan, Zongqing Lu, and Xiu Li. State advantage weighting for offline RL. *arXiv preprint arXiv:2210.04251*, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. ISSN 1476-4687.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Shideh Rezaeifar, Robert Dadashi, Nino Vieillard, Léonard Hussenot, Olivier Bachem, Olivier Pietquin, and Matthieu Geist. Offline reinforcement learning as anti-exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8106–8114, 2022. ISBN 2374-3468.

Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. ISBN 0262352702.

Nino Vieillard, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Rémi Munos, and Matthieu Geist. Leverage the average: an analysis of KL regularization in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:12163–12174, 2020a.

Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4235–4246, 2020b.

Qing Wang, Jiechao Xiong, Lei Han, Han Liu, and Tong Zhang. Exponentially weighted imitation learning for batched historical data. *Advances in Neural Information Processing Systems*, 31, 2018.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, and Nicolas Heess. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020.

Christopher JCH Watkins and Peter Dayan. Q-learnarning. *Machine learning*, 8(3):279–292, 1992. ISSN 1573-0565.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2105.08140*, 2021.

Chi Zhang, Sanmukh Kuppannagari, and Prasanna Viktor. Brac+: Improved behavior regularized actor critic for offline reinforcement learning. In *Asian Conference on Machine Learning*, pages 204–219. PMLR, 2021. ISBN 2640-3498.