You Only Evaluate Once:

a Simple Baseline Algorithm for Offline RL

Wonjoon Goo and Scott Niekum
Department of Computer Science
University of Texas at Austin
{wonjoon,sniekum}@cs.utexas.edu

Abstract: The goal of offline reinforcement learning (RL) is to find an optimal policy given prerecorded trajectories. Many current approaches customize existing off-policy RL algorithms, especially actor-critic algorithms in which policy evaluation and improvement are iterated. However, the convergence of such approaches is not guaranteed due to the use of complex non-linear function approximation and an intertwined optimization process. By contrast, we propose a simple baseline algorithm for offline RL that only performs the policy evaluation step once so that the algorithm does not require complex stabilization schemes. Since the proposed algorithm is not likely to converge to an optimal policy, it is an appropriate baseline for actor-critic algorithms that ought to be outperformed if there is indeed value in iterative optimization in the offline setting. Surprisingly, we empirically find that the proposed algorithm exhibits competitive and sometimes even state-of-the-art performance in a subset of the D4RL offline RL benchmark. This result suggests that future work is needed to fully exploit the potential advantages of iterative optimization in order to justify the reduced stability of such methods.

Keywords: offline reinforcement learning, conservative policy evaluation

1 Introduction

The standard reinforcement learning setting involves an active component during learning: an agent continuously gathers experience as it learns. This is a very general learning framework that resembles the way animals learn, but the interactive component often hurts the applicability of RL since the agent interaction can be expensive or unsafe. To address this challenge, the offline reinforcement learning paradigm has been proposed, which aims to learn a policy purely from pre-generated data [1]. Considering that many recent breakthroughs in machine learning can be attributed to large-scale data, this new paradigm is very promising. However, the offline setup causes significant theoretic and algorithmic difficulties that need to be resolved to fulfill this promise.

Specifically, actor-critic based off-policy RL algorithms, which iterate policy evaluation and improvement, suffer from the overestimation problem caused by function approximation error and bootstrapping [2, 3] in the offline RL setup, even though the algorithms are equipped with algorithmic techniques [4, 2, 5] that can stabilize learning and mitigate the so-called Deadly Triad [6]. This is because over-estimated values cannot be readjusted in offline RL, unlike the ordinary RL setup where incorrectly optimistic actions get executed and corrected.

The Deadly Triad states that when off-policy learning, function approximation, and bootstrapping are used together, the danger of instability and divergence arises [6]. Therefore, the actor-critic algorithms that leverage all three components are vulnerable. Many algorithms try to stabilize learning by ensuring the queries for bootstrapping to be within the data manifold of the given offline dataset [7, 8, 9, 10, 11, 12] since over-estimation is prominent when the value function is queried for out-of-distribution inputs. However, strictly speaking, the Deadly Triad occurs due to iteration of the actor-critic algorithm; the first critic update is simply policy evaluation of the action-value function of the behavior policy β using on-policy data. While the policy implied by the first value function \(Q^\beta\) is likely to be suboptimal that would be surpassed by iterative algorithms, it has not been fully examined that a greedy policy with regard to \(Q^\beta\) cannot work well in the offline RL setting.

5th Conference on Robot Learning (CoRL 2021), London, UK.
To this end, we propose a baseline algorithm, named YOE,O, that evaluates the value of a behavior policy once and extracts a greedy policy under the learned value approximation, and by doing so, we examine the hidden presumption of the actor-critic based offline RL algorithms that the iterative process is essential and beneficial. Our algorithm leverages pessimism under uncertainty [13, 14] within the distributional RL framework [15, 16, 17]. Surprisingly, we find that the greedy policy extracted from the value function of the behavior policy with our algorithm exhibits competitive and sometimes even state-of-the-art results in D4RL benchmarks [18]. Our results indicate not only the effectiveness of the proposed algorithm adopting pessimistic regularization, but also implies potential vulnerabilities of the iterative optimization process of actor-critic algorithms in the offline setting, especially when a complex function approximator like a deep neural network is used. We expect that the proposed strong baseline will foster future offline RL research by allowing researchers to measure the actual advantage coming from the iterative process, which is a main concern in offline RL.

2 Related Work

Evaluating and improving a policy with the data generated from a different policy (off-policy RL) has been widely investigated, and several papers have shown theoretical convergence properties of prediction and control algorithms in the off-policy setting [19, 20, 21]. However, theoretical frameworks are limited to linear function approximation, while non-linear function approximation is essential to handle large-scale MDPs. Yet, there have been efforts to build practical algorithms with non-linear function approximators, and they have shown considerable success in various domains [22, 23, 24, 25] tackling real-world RL problems.

In theory, off-policy algorithms can be used in the offline setup without any modifications, but the algorithms often catastrophically fail when applied in the batch setting [7]. This is due to the accumulation of extrapolation error from bootstrapping and the policy improvement step (i.e. max operation) [7, 8]. In the non-batch setting, new experiences gathered via interaction can prevent this degenerate case, but it is impossible in the batch setting where interaction is prohibited. Pertaining to this problem, much research has been proposed, especially in the context of actor-critic algorithms in which policy evaluation and improvement are iterated. One class of solutions constrains the policy improvement step so that the optimized policy matches the behavior policy, in distribution [9] or in support [8]. In more recent work, a behavior policy for distribution matching is replaced with a prior policy, which is trained along with policy evaluation via weighted behavior cloning [12]. In [11], the policy improvement step is omitted while using the prior policy as a target policy for the evaluation. While the prior works try to overcome the Deadly Triad, we sidestep the problem and examine the hidden assumption of the actor-critic algorithm that the iterative process is essential and beneficial.

The most closely related prior works are behavior cloning-based methods [26, 10] which mimic a subset of good state-action pairs from the pre-generated trajectories. Since these methods approximate a value function for a behavior policy without an iterative policy improvement step, the algorithm does not diverge. The main difference with our proposed approach is that we train an action-value function with pessimistic regularization while the prior works evaluate a state- or action-value function to filter state-action pairs based on advantage calculated with Monte-Carlo return [26] or TD(k) return [10] and train a policy via behavior cloning with the filtered data.

For principled regularization of the action-value function, we adopt pessimism under uncertainty, which can address the overestimation problem when the given offline dataset is not informative enough to estimate the value of every action given a state [14]. This pessimism can be applied by directly penalizing $Q$ values for a particular state-action distribution [13], or with model-approximation [27] that leverages uncertainty prediction techniques developed for supervised learning, such as Lakshminarayanan et al. [28] or van Amersfoort et al. [29]. While the previous approaches apply pessimism unconditionally for every state [13] or use transition-dynamics based proxies [27] for measuring uncertainty, we propose a theoretically justifiable regularization method for estimating $Q^\beta$ that is based on distributional RL.

3 Preliminaries

The common mathematical framework for reinforcement learning is a Markov Decision Process (MDP), which is defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, d_0, r, \gamma)$ defined by a set of states $s \in \mathcal{S}$, a
set of actions $a \in \mathcal{A}$, conditional transition dynamics $T(s'|s,a)$, an initial state distribution $d_0$, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and a discount factor $\gamma \in (0,1]$. In this framework, the goal of reinforcement learning is to find an optimal policy $\pi(a|s)$ that maximizes an expected sum of discounted reward (return). Formally, the objective is defined as:

$$J(\pi) = \mathbb{E}_{\tau \sim p^\pi} \left[ \sum_{t=0}^{H} \gamma^t r(s_t, a_t) \right],$$

(1)

where $\tau$ is a sequence of states and actions $(s_0, a_0, \ldots, s_H, a_H)$ of length $H$, and $p^\pi$ is a trajectory distribution of a policy, which can be represented as:

$$p^\pi(\tau) = d_0(s_0) \prod_{t=0}^{H} \pi(a_t|s_t)T(s_{t+1}|s_t, a_t).$$

(2)

One way to find an optimal policy is to estimate an action-value function $Q^\pi$, which represents the expected return over possible trajectories following a policy $\pi$ starting from a given state and action: $Q^\pi(s, a) = \mathbb{E}_{\tau \sim p^\pi(s, a)} \left[ \sum_{t'=0}^{H} \gamma^{t'} r(s_{t'}, a_{t'}) \right]$. $Q^\pi$ function implies a greedy policy $\pi^+(a|s) = \delta (a = \arg \max_b Q^\pi(s, b))$, which is better than or equal to its original evaluation target policy $\pi$. Therefore, when we perform policy evaluation ($Q$ estimation) and policy updates iteratively, we can move toward the optimal policy $\pi^+$ and the optimal $Q$ function $Q^{\pi^+}$. Policy evaluation can be done with a Monte-Carlo method, but bootstrapping is commonly used, which utilizes a recursive equation that must be satisfied at convergence:

$$Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{a' \sim T(s'|s,a), s' \sim \pi(a'|s')} Q^\pi(s', a').$$

(3)

When an MDP is discrete and $Q$ can be represented by a tabular representation (i.e. when $|S| \times |\mathcal{A}|$ is small), it is known that policy evaluation converges to a correct solution in the limit of the number of transition tuples [6]. However, when an MDP has a large state or action space, $Q$ has to be represented with a function approximator, such as a deep neural network. In addition, when the action space is continuous, directly extracting a better policy from $Q$ becomes infeasible due to the $\arg \max$ operator. These restrictions are addressed in actor-critic algorithms [4, 2, 5] which explicitly alternate policy evaluation and policy improvement with a batch of (online) transition samples $D$ and a parameterized value function $Q_\theta$ and a policy $\pi_\phi$:

$$\theta^{k+1} \leftarrow \arg \min_{\theta} \mathbb{E}_{s, a, s' \sim D} \left[ d(\tilde{Q}_\theta(s, a), r(s, a) + \gamma \mathbb{E}_{a' \sim \pi_\phi(a'|s')} \tilde{Q}_\theta(s', a')) \right] \quad \text{(policy evaluation)},$$

(4)

$$\phi^{k+1} \leftarrow \arg \max_{\phi} \mathbb{E}_{s \sim D, a \sim \pi_\phi(a|s)} (\tilde{Q}_{\theta^{k+1}}(s, a)) \quad \text{(policy improvement)},$$

(5)

where $k$ is an update step, and $d$ is a distance measure such as squared $l_2$ or Huber loss.

4 You Only Evaluate Once

Behind actor-critic based offline RL algorithms, there is a common presumption that the iterative process is essential in achieving better performance than that of the behavior policy, even though it could sacrifice the reliability of the algorithm due to the over-estimation problem. This is because we want to find a policy that behaves differently from the data-generating policy by making counterfactual queries (policy improvement) and answering (policy evaluation) them iteratively [1]. However, it has not been established that a simpler and safer baseline cannot work well—a policy that selects the best action with regard to the action-value function of the behavior policy. Without rigorously examining this hypothesis, the true worth of iterative algorithms and counterfactual queries cannot be fully understood.

In this paper, we challenge the iterative offline algorithms by proposing a stable offline algorithm that recovers the best action of behavioral policy $\beta(s)$ that is used to generate an offline dataset $D$. Formally, our goal is to find the greedy behavioral policy $\beta^*(s)$ that selects the best action with respect to $Q^\beta$, only considering the action candidates of $\beta$:

$$\beta^*(s) := \arg \max_{a \in B(s)} Q^\beta(s, a) \quad \text{where} \quad B(s) = \text{supp } \beta(s) := \{ a \in \mathcal{A} : \beta(a|s) > 0 \}.$$
We propose a two-step approach: we first perform state value distribution $Q$ when we have a valid pessimistic approximation of $B$ with $\psi$ with $\max_{s}$. To address this problem, we express the two conditions of the theorem in terms of the state value $\max_{s}$ is queried probability. In IQN, the parameter $\beta$ models the value distribution in the form of an inverse cumulative distribution function parameterized by enforcing $\beta$ for off-policy input $(s, \hat{a})$ solely depends on the generalization ability of $\hat{Q}$. Figure 1 demonstrates one failure case in estimating $Q^\beta$; when $\hat{Q}^\beta$ is trained without any regularization, it behaves more like a state value function $V^\beta(s)$ ignoring the action input. Therefore, any erroneous action generated by estimated $B(s)$ will be treated as the best action.

Instead, we propose to learn the greedy behavioral policy $\beta^*$ without directly estimating $B(s)$. It can be achieved when $\hat{Q}^\beta$ is properly regularized so that the greedy policy implied by the regularized $\hat{Q}^\beta$ is $\beta^*$. We argue that pessimistic regularization satisfies such property:

**Theorem** ($\hat{Q}^\beta$ implies $\beta^*$). Let $\hat{Q}^\beta$ be a valid pessimistic approximation of $Q^\beta$: (1) $\hat{Q}^\beta(s, \hat{a}) < \max_{a \in B(s)} Q^\beta(s, a) \forall s \in D$, $\hat{a} \notin B(s)$ and (2) $\hat{Q}^\beta(s, a) = Q^\beta(s, a) \forall (s, a) \in D$. Then, a greedy policy over $\hat{Q}^\beta$ is $\beta^*$.

**Proof.**
$$\arg \max_{a \in A} \hat{Q}^\beta(s, a) = \arg \max_{a \in B(s)} \hat{Q}^\beta(s, a) = \arg \max_{a \in B(s)} Q^\beta(s, a) = \beta^*(s).$$

Pessimistic regularization is essential even for learning $Q^\beta$ since there are always out-of-distribution state-action pairs $(s, \hat{a})$ given a fixed dataset $D$, and the values of these inputs depend on the generalization characteristic of the function approximator used to learn $Q^\beta$. Therefore, we have to regularize $Q^\beta$ properly, and we adapt pessimism under the uncertainty principle since we cannot assume any value for OOD input unless we have a prior regarding an MDP. Yet, in contrast to pessimistic regularization for $Q^\beta$, which requires a calibrated epistemic uncertainty measurement [14] or suffers from over-regularization due to excessive pessimism [13], $\hat{Q}^\beta$ can be properly regularized by enforcing $\hat{Q}^\beta(s, \hat{a})$ to be smaller than the maximum value of $\hat{Q}^\beta(s, a)$ for $a \in B(s)$.

4.1 Practical Implementation

When we have a valid pessimistic approximation of $Q^\beta$, $\beta^*$ can be inferred without knowing $B(s)$ since the greedy action over the approximated value function $\hat{Q}^\beta$ is $\beta^*$. However, the first constraint of the valid pessimistic approximation still depends on $B(s)$, and it makes implementing the pessimistic regularization in its original form difficult.

To address this problem, we express the two conditions of the theorem in terms of the state value distribution over $Y^\beta(s)$; $Y^\beta(s)$ is a random variable representing the return of the policy $\beta$ starting from a state $s$ where its expectation is the state value $V^\beta(s) = E[Y^\beta(s)]$ [15, 16, 17]. With the random variable $Y^\beta(s)$, the two constraints can be rewritten as follows: (1) $\hat{Q}^\beta(s, \cdot) \leq \max_{a \in B(s)} Q^\beta(s, a) = \max_w Y^\beta(s; w)$, and (2) $\hat{Q}^\beta(s, a) = r + \gamma E_{s' \sim \tau(s, a)} [r + \gamma E_{s'' \sim \tau(s', a)} [Y^\beta(s'') = r + \gamma E_{s'' \sim \tau(s', a)} [E_w [Y^\beta(s''; w)]]$ where $w$ is an outcome of the sample space of $Y$. This converted expression is defined without a need for $B(s)$, and therefore, we can now implement the pessimistic regularization with $Y^\beta(s)$ instead of $B(s)$. Note that the constraint is defined with the maximum and the expected value of $Y^\beta$ only for the state $s \in D$. This allows us to use any on-policy evaluation methods for $Y^\beta$ as long as the resulting approximation is precise for on-policy states.

We propose a two-step approach: we first perform state value distribution learning using $(s, r, s')$ transition tuple in $D$ (behavioral policy evaluation) and then train $\hat{Q}^\beta$ in a supervised manner that satisfies the given constraints. Specifically, we represent a state-value distribution of $Y^\beta$ using implicit quantile network (IQN) [17], which is a recently proposed distributional RL algorithm that models the value distribution in the form of an inverse cumulative distribution function parameterized with $\psi$: (with a slight abuse of notation) $\hat{Y}_\psi(s; \tau) = \inf \{ v \in [\tau] : \tau \leq \text{Prob}(Y(s) \leq v) \}$ where $\tau$ is queried probability. In IQN, the parameter $\psi$ can be trained by a distributional TD learning. For details about the distributional TD learning, please refer to Will et al. [17].
With the approximated state-value distribution $\hat{Y}_\psi$ of the behavior policy $\beta$, we train an action-value function $\hat{Q}_\theta$ with a supervised loss for on-policy state-action pair $(s, a)$ and a pessimistic regularization $R$ for off-policy pair $(s, \tilde{a})$:

$$L = \mathbb{E}_{(s,a,r,s') \sim D} \left[ \frac{1}{2} (\hat{Q}_\theta(s, a) - (r + \gamma \mathbb{E}[\hat{Y}_\psi(s')]])^2 + \lambda R(s; \theta) \right]$$

(7)

where $\lambda$ is a hyperparameter that controls the strength of pessimism. While we can approximate the expectation of $\hat{Y}_\psi$ with sampling, we use the median value of the distribution $\hat{Y}_\psi(s'; 0.5)$ for computational efficiency.

The main goal of $R$ is to implement the pessimism on $\hat{Q}_\theta$ and guarantee $\hat{Q}_\theta$ to be a valid pessimistic approximation of $Q^\beta$ that satisfies $\hat{Q}_\theta(s, \cdot) \leq \max Y^\beta(s)$. While every possible action $a \in A$ needs to be checked, it is computationally infeasible when the action space is large or infinite. Hence, we sample actions from a static distribution $\mu$ (passive pessimistic regularization) and a distribution $\pi_\phi$ that is actively changing over training iterations (active pessimistic regularization). Then, we penalize $\hat{Q}_\theta$ if the estimations on the action samples violate the pessimistic constraint. We use a different hyperparameter $\tau$ for each of action samples. We sample $n_a$ number of actions from each distribution and use the log-sum-exp trick to change the hard constraint into a soft constraint:

$$R(s; \theta) = \log \left[ \exp(\hat{Y}_\psi(s; \tau_1)) + \sum_{\tilde{a} \sim \pi_\phi(s)} \exp \hat{Q}_\theta(s, \tilde{a}) \right] + \log \left[ \exp(\hat{Y}_\psi(s; \tau_2)) + \sum_{b \sim \mu(a)} \exp \hat{Q}_\theta(s, b) \right].$$

(8)

We use a random uniform policy for $\mu$ while the $\pi_\phi$ is trained against trained $Q$ (or a set of $Q$ when an ensemble technique is used) as same as an ordinary actor in the actor-critic algorithm. However, it is different from the actor in that $\pi_\phi$ works as an active regularizer that adversarially finds the wrong generalization part of $Q^\beta$. For $\tau_1$ and $\tau_2$, we use 0.9 and 0.1 respectively. While the $\tau_1$ is decided following the definition of valid pessimism with a safe margin of 0.1, we use a small $\tau_2$ since we expect that a random policy will perform poorly. We train multiple $\hat{Q}_\theta$ models and aggregate them by taking a min over inferred $\hat{Q}_\theta$ values to get a robust value estimate. We train multiple models with different initial parameters and the same data following Osband et al. [30].

As an approximation of $\beta^*$, we use two policies defined on top of $\hat{Q}_\theta$. One is the last $\pi_\phi$ we have at the end of training since $\pi_\phi$ is directly trained to find the greedy policy over $\hat{Q}_\theta$. If $\hat{Q}_\theta$ is properly regularized (i.e. $\hat{Q}_\theta$ is a valid pessimistic approximation), $\pi_\phi$ will converge to $\beta^*$. Therefore, the policy will be implicitly constrained to use actions limited to the support of the behavior policy $\beta$. The other policy is a nonparametric policy $\hat{\pi}$ whose action space is explicitly constrained to the actions shown in the dataset $D$: $\hat{\pi}(s) = \arg \max_{a \sim D} Q^\theta(s, a)$. $\hat{\pi}$ can perform better than $\pi_\phi$ when the pessimistic constraint is hard to achieve due to restriction of a dataset or characteristics of an MDP, such as the dataset size $|D|$ or the large action space $|A|$, that allows room for adversarial attacks; when there is much room for adversarial attacks, $\pi_\phi$ will continuously try to regularize $\hat{Q}_\theta$ by suggesting diverse adversarial actions while it does not converge to the desired policy $\beta^*$. In such a case, directly utilizing $\pi_\phi$ would result in poor performance, but the action set restricted policy $\hat{\pi}$ can perform well by testing limited actions that are more likely to belong to $B(s)$. The overall algorithm is shown in Algorithm 1, and the implementation details are provided in Appendix.

Since the training of $\hat{Q}_\theta$ largely depends on the trained state value distribution $\hat{Y}_\psi$, the correctness of $\hat{Y}_\psi$ is essential in learning greedy behavioral policy $\beta^*$. Fortunately, since we are performing policy evaluation with on-policy data, we avoid the Deadly Triad, and it is more likely to converge to a correct $Y^\beta$ than other methods that perform off-policy learning. Furthermore, the correctness of $\hat{Y}_\psi$ and the regularized $\hat{Q}_\theta$ can be roughly tested by comparing the estimated value with the Monte-Carlo
We confirm the necessity of pessimistic regularization even when we do on-policy policy evaluation; we conduct an ablation study to show the effectiveness of the regularization and the contribution of each term in Eq. 8 (denoted as \( \hat{Q}^\beta \) trained only via TD loss based on \((s, a, r, s', a')\) tuple without any regularization (denoted as \( Q^\beta \{R \} \)) and (2) \( \hat{Q}^\beta \) trained without the second term in Eq. 8 (denoted as \( Q^\beta \{\mu \} \)). Additionally, we test the effect of the ensemble by changing the number of trained models. The experiment settings and the results are summarized in Table 2.

We aim to study the following question: how much performance gain do potentially risky policy type datasets in the kitchen task of which the size of the dataset is relatively small or the action space is large; since \( \hat{Q}^\beta \) trained only via TD loss based on \((s, a, r, s', a')\) tuple without any regularization (denoted as \( Q^\beta \{R \} \)) and (2) \( \hat{Q}^\beta \) trained without the second term in Eq. 8 (denoted as \( Q^\beta \{\mu \} \)). Additionally, we test the effect of the ensemble by changing the number of trained models. The experiment settings and the results are summarized in Table 2.

We use MuJoCo locomotion tasks \([32]\), Adroit hand-manipulation tasks \([34]\), and Franka kitchen \([35]\) tasks. For MuJoCo locomotion tasks, we use three environments (hopper, walker2d, and halfcheetah) in four different settings (random, medium-replay, medium, and medium-expert). For Adroit hand-manipulation tasks, we use four environments (pen, door, relocate, and hammer) in two settings (human and cloned). For Franka kitchen tasks, we use three different settings (complete, partial, and mixed). We train YOEO for 1 million stochastic gradient steps for both \( Y_\phi \) and \( \hat{Q}_\theta \), then we report the average normalized performance score over 100 trajectories. The results are displayed in Table 1.

Despite the fact that YOEO aims to learn a greedy behavioral policy \( \beta^*(s) \) with respect to \( Q^\beta \) rather than try to learn \( \pi^* \), it shows highly competitive results: it achieves better performance than the current state-of-the-art model-free offline RL algorithms in five datasets (hopper-random, walker2d-medium-replay, hopper-medium-expert, and hammer-cloned, kitchen-compete), and competitive performance across all other configurations surpassing most of the other offline RL algorithms. The results indicate that the previous offline RL algorithms fail to fully exploit the potential benefit of iterative evaluation and update.

The performance of \( \hat{\pi} \) shown in the last column of Table 1 also supports our hypothesis on \( \pi_\phi \). While \( \hat{\pi} \) shows similar results in other datasets, \( \hat{\pi} \) shows a better result than \( \pi_\phi \) in -human type dataset in Adroit tasks and -partial and -mixed type datasets in the kitchen task of which the size of the dataset is relatively small or the action space is large; since \( \hat{Q}_\theta \) needs to be regularized for diverse set of actions, \( \pi_\phi \) would not converge to \( \beta^* \) whose actions are limited to the support of \( \beta \).

5 Experiments

We conduct an ablation study to show the effect of the regularizer and the contribution of each term in Eq. 8. Specifically, we consider two ablations: (1) \( \hat{Q}^\beta \) trained only via TD loss based on \((s, a, r, s', a')\) tuple without any regularization (denoted as \( Q^\beta \{R \} \)) and (2) \( \hat{Q}^\beta \) trained without the second term in Eq. 8 (denoted as \( Q^\beta \{\mu \} \)). Additionally, we test the effect of the ensemble by changing the number of trained models. The experiment settings and the results are summarized in Table 2.

We confirm the necessity of pessimistic regularization even when we do on-policy policy evaluation; the qualitative results shown in Figure 1 reveal that \( \hat{Q}_\theta \{R \} \) can estimate the ground-truth value for on-policy input \((s, a)\), but it fails to estimate the value for out-of-distribution input \((s, b)\), especially return. This is an extra debugging feature that is only available in YOEO, and we can leverage this to set the hyperparameters.

It is noteworthy that we are leveraging aleatoric uncertainty to implement pessimism. When epistemic uncertainty of action-value function \( Q^* (s, a) \) can be estimated with a fixed dataset, using it to penalize the value function is a theoretically justifiable implementation of pessimism \([14]\). However, measuring epistemic uncertainty is still an open problem when a deep neural network is used even in the simpler supervised learning setting that does not include bootstrapping as RL. For that reason, pessimism is commonly implemented with a proximal objective, such as learning a lower bound of the true \( Q^* \) \([13]\) or using an uncertainty proxy \([27, 31]\), and therefore, these approaches often require sensitive hyperparameter tuning to find the right level of pessimism \([1]\). In contrast, the objective of YOEO allows us to use aleatoric uncertainty that can be directly estimated. Also, the two-step training prevents the error from propagating through bootstrapping, so our method is less susceptible to divergence at the cost of optimality. This makes our algorithm a suitable baseline for offline RL that achieves stability at the cost of optimality.
Table 1: Performance of YOEO and prior methods on a subset of D4RL benchmarks. Each number represents the performance relative to a random policy as 0 and an expert policy as 100. All the numbers except ours are borrowed from [18] and [13]. The numbers of our results are averaged over 3 different random seeds.

<table>
<thead>
<tr>
<th>Type</th>
<th>Environment</th>
<th>BC</th>
<th>SAC-offline</th>
<th>BEAR</th>
<th>BRAC</th>
<th>AWR</th>
<th>BCQ</th>
<th>CQL (η)</th>
<th>YOEO (σ_φ)</th>
<th>YOEO (ξ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>HalfCheetah</td>
<td>2.1</td>
<td>30.5</td>
<td>25.1</td>
<td>31.2</td>
<td>2.5</td>
<td>2.2</td>
<td>35.4</td>
<td>4.5</td>
<td>5.7</td>
</tr>
<tr>
<td></td>
<td>Walker2D</td>
<td>9.8</td>
<td>11.3</td>
<td>11.4</td>
<td>12.2</td>
<td>10.2</td>
<td>10.6</td>
<td>10.8</td>
<td>12.2</td>
<td>12.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.6</td>
<td>4.1</td>
<td>7.3</td>
<td>1.9</td>
<td>1.5</td>
<td>4.9</td>
<td>7</td>
<td>4.1</td>
<td></td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>HalfCheetah</td>
<td>38.4</td>
<td>-2.4</td>
<td>38.6</td>
<td>47.7</td>
<td>40.3</td>
<td>38.2</td>
<td>46.2</td>
<td>36.2</td>
<td>30.9</td>
</tr>
<tr>
<td></td>
<td>Hopper</td>
<td>11.8</td>
<td>3.5</td>
<td>33.7</td>
<td>0.6</td>
<td>28.4</td>
<td>33.1</td>
<td>48.6</td>
<td>42.5</td>
<td>37.2</td>
</tr>
<tr>
<td></td>
<td>Walker2D</td>
<td>11.3</td>
<td>1.9</td>
<td>19.2</td>
<td>0.9</td>
<td>15.5</td>
<td>15</td>
<td>32.6</td>
<td>41.9</td>
<td>30.1</td>
</tr>
<tr>
<td>Medium</td>
<td>HalfCheetah</td>
<td>36.1</td>
<td>-4.3</td>
<td>41.7</td>
<td>46.3</td>
<td>37.4</td>
<td>40.7</td>
<td>44.4</td>
<td>45.1</td>
<td>43.6</td>
</tr>
<tr>
<td></td>
<td>Hopper</td>
<td>29</td>
<td>0.8</td>
<td>52.1</td>
<td>31.1</td>
<td>35.9</td>
<td>54.5</td>
<td>86.6</td>
<td>71.9</td>
<td>62.5</td>
</tr>
<tr>
<td></td>
<td>Walker2D</td>
<td>6.6</td>
<td>0.9</td>
<td>59.1</td>
<td>81.1</td>
<td>17.4</td>
<td>53.1</td>
<td>74.5</td>
<td>74.1</td>
<td>76.2</td>
</tr>
<tr>
<td>Medium-Expert</td>
<td>HalfCheetah</td>
<td>35.8</td>
<td>1.8</td>
<td>53.4</td>
<td>44.2</td>
<td>52.7</td>
<td>64.7</td>
<td>62.4</td>
<td>41.9</td>
<td>35.9</td>
</tr>
<tr>
<td></td>
<td>Hopper</td>
<td>111.9</td>
<td>1.6</td>
<td>96.3</td>
<td>0.8</td>
<td>27.1</td>
<td>110.9</td>
<td>111</td>
<td>112.1</td>
<td>112</td>
</tr>
<tr>
<td></td>
<td>Walker2D</td>
<td>6.4</td>
<td>-0.1</td>
<td>40.1</td>
<td>81.6</td>
<td>53.8</td>
<td>57.5</td>
<td>98.7</td>
<td>89.5</td>
<td>84.3</td>
</tr>
<tr>
<td>Medium</td>
<td>Hopper</td>
<td>56.9</td>
<td>23.5</td>
<td>26.5</td>
<td>1.6</td>
<td>28</td>
<td>34.8</td>
<td>39.2</td>
<td>32.6</td>
<td>30.5</td>
</tr>
<tr>
<td></td>
<td>Walker2D</td>
<td>0.5</td>
<td>-0.3</td>
<td>-0.3</td>
<td>0.4</td>
<td>0</td>
<td>9.9</td>
<td>-0.1</td>
<td>5.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>-0.3</td>
<td>0.3</td>
<td>0.2</td>
<td>0.2</td>
<td>-0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.5</td>
<td>0.5</td>
<td>0.3</td>
<td>0.3</td>
<td>1.2</td>
<td>4.4</td>
<td>2.6</td>
<td>5.4</td>
<td></td>
</tr>
<tr>
<td>human</td>
<td>pen</td>
<td>34.4</td>
<td>6.3</td>
<td>-1</td>
<td>8.1</td>
<td>12.3</td>
<td>68.9</td>
<td>37.5</td>
<td>17.3</td>
<td>43.6</td>
</tr>
<tr>
<td></td>
<td>door</td>
<td>0.5</td>
<td>3.9</td>
<td>-0.3</td>
<td>0.3</td>
<td>0.4</td>
<td>9.9</td>
<td>-0.1</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>relocate</td>
<td>0</td>
<td>0</td>
<td>-0.3</td>
<td>0.3</td>
<td>0.2</td>
<td>0.2</td>
<td>-0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>hammer</td>
<td>1.5</td>
<td>0.5</td>
<td>0.3</td>
<td>0.3</td>
<td>1.2</td>
<td>4.4</td>
<td>2.6</td>
<td>5.4</td>
<td></td>
</tr>
<tr>
<td>cloned</td>
<td>pen</td>
<td>56.9</td>
<td>23.5</td>
<td>26.5</td>
<td>1.6</td>
<td>28</td>
<td>34.8</td>
<td>39.2</td>
<td>32.6</td>
<td>30.5</td>
</tr>
<tr>
<td></td>
<td>door</td>
<td>-0.1</td>
<td>0</td>
<td>-0.1</td>
<td>0.1</td>
<td>0</td>
<td>0.4</td>
<td>0.1</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>relocate</td>
<td>-0.1</td>
<td>-0.2</td>
<td>-0.3</td>
<td>-0.3</td>
<td>-0.2</td>
<td>-0.3</td>
<td>-0.1</td>
<td>-0.1</td>
<td>-0.2</td>
</tr>
<tr>
<td></td>
<td>hammer</td>
<td>0.8</td>
<td>0.2</td>
<td>0.3</td>
<td>0.3</td>
<td>0.4</td>
<td>0.4</td>
<td>2.1</td>
<td>2.7</td>
<td>1.2</td>
</tr>
<tr>
<td>kitchen</td>
<td>complete</td>
<td>33.8</td>
<td>15.0</td>
<td>0</td>
<td>0</td>
<td>8.1</td>
<td>43.8</td>
<td>63.2</td>
<td>31.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>partial</td>
<td>33.8</td>
<td>0</td>
<td>13.1</td>
<td>0</td>
<td>15.4</td>
<td>18.9</td>
<td>49.8</td>
<td>17.2</td>
<td>46.8</td>
</tr>
<tr>
<td></td>
<td>mixed</td>
<td>47.5</td>
<td>2.5</td>
<td>47.2</td>
<td>0</td>
<td>10.6</td>
<td>8.1</td>
<td>51.0</td>
<td>5.9</td>
<td>40.4</td>
</tr>
</tbody>
</table>

Table 2: Ablation experiment results. The normalized performance over 3 random seeds is displayed.

<table>
<thead>
<tr>
<th></th>
<th>( \hat{Q}_\theta \mid \bar{R} )</th>
<th>( \hat{Q}_\theta \mid \mu )</th>
<th>( \hat{Q}_\theta ) (Ens. 1)</th>
<th>( \hat{Q}_\theta ) (Ens. 3)</th>
<th>YOEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised w/ ( \hat{Y}_\psi )</td>
<td>( \times )</td>
<td>( \checkmark )</td>
<td>( \checkmark )</td>
<td>( \checkmark )</td>
<td>( \checkmark )</td>
</tr>
<tr>
<td>( \bar{R} ) w/ ( \mu )</td>
<td>( \times )</td>
<td>( \checkmark )</td>
<td>( \checkmark )</td>
<td>( \checkmark )</td>
<td>( \checkmark )</td>
</tr>
<tr>
<td>( \bar{R} ) w/ ( \pi )</td>
<td>( \times )</td>
<td>( \checkmark )</td>
<td>( \checkmark )</td>
<td>( \checkmark )</td>
<td>( \checkmark )</td>
</tr>
<tr>
<td># Ensembles</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>Hopper</td>
<td>30.6</td>
<td>49.1</td>
<td>21.3</td>
<td>36.5</td>
</tr>
<tr>
<td></td>
<td>Walker2D</td>
<td>3.7</td>
<td>21.2</td>
<td>18.9</td>
<td>28.2</td>
</tr>
<tr>
<td>Medium</td>
<td>Hopper</td>
<td>2.7</td>
<td>14.9</td>
<td>54.6</td>
<td>65.9</td>
</tr>
<tr>
<td></td>
<td>Walker2D</td>
<td>-0.2</td>
<td>29</td>
<td>66.3</td>
<td>75.9</td>
</tr>
<tr>
<td>Medium-Expert</td>
<td>Hopper</td>
<td>10.4</td>
<td>58.2</td>
<td>89.6</td>
<td>112</td>
</tr>
<tr>
<td></td>
<td>Walker2D</td>
<td>3</td>
<td>0.3</td>
<td>62.2</td>
<td>94.9</td>
</tr>
</tbody>
</table>

when the given dataset is homogeneous such as the -medium dataset whose behavior policy \( \beta \) is not a mixture of different policies. This is because \( \hat{Q}^\beta \) behaves more like a \( \hat{V}^\beta \), ignoring the action, since an effective action \( a \) is predictable based on \( s \). The performance degradation of \( \hat{Q}_\theta \mid \mu \) indicates that a random policy \( \mu \) is a good heuristic that can prevent the degeneration of the action-value function; the second regularization term using \( \mu \) can foster the discrimination ability of the value function by enforcing \( \hat{Q}_\theta \) to estimate a different value for \((s, a)\) and \((s, b)\), and this can increase the performance significantly especially when the given dataset is generated with a homogeneous policy.

We also observe the performance benefit of training more models and ensembling them, especially for -medium-replay type datasets. We hypothesize that YOEO’s pessimistic regularization method has high variance due to the stochasticity of the loss function, and the ensembling technique can improve learning by enabling robust prediction for off-policy data from which the stochasticity is derived.

5.2 Why do CQL and other methods fail?
We investigate a simple baseline algorithm for offline RL that only evaluates the value function of the behavior policy as opposed to approximate $Q^*$ with unstable iterative process. Since the proposed algorithm does not involve a policy optimization step and value re-evaluation based on the updated policy, the algorithm can be stable, but the resulting policy is more likely to be suboptimal. This makes the algorithm an appropriate baseline for actor-critic algorithms that ought to outperform this baseline if there is indeed value in iterative optimization in the offline setting. In the experiments, the proposed baseline surprisingly shows competitive results on the several D4RL benchmarks, surpassing the state-of-the-art results in some tasks. This implies the usefulness of conservativeness under uncertainty, which can prevent incorrect generalization behavior of a complex function approximator occurring due to lack of data, as well as the potential flaws of iterative optimization in actor-critic algorithms in the offline setting. Therefore, it is essential for future work to build a theoretical framework that sheds light on iterative optimization and generalization of offline actor-critic methods that use deep neural networks, if iterative optimization is to be fully taken advantage of.
Acknowledgments

This work has taken place in the Personal Autonomous Robotics Lab (PeARL) at The University of Texas at Austin. PeARL research is supported in part by the NSF (IIS-1724157, IIS-1638107, IIS-1749204, IIS-1925082), ONR (N00014-18-2243), AFOSR (FA9550-20-1-0077), and ARO (78372-CS).

References


