OFFLINE REINFORCEMENT LEARNING WITH IN-SAMPLE Q-LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline reinforcement learning requires reconciling two conflicting aims: learning a policy that improves over the behavior policy that collected the dataset, while at the same time minimizing the deviation from the behavior policy so as to avoid errors due to distributional shift. This tradeoff is critical, because most current offline reinforcement learning methods need to query the value of unseen actions during training to improve the policy, and therefore need to either constrain these actions to be in-distribution, or else regularize their values. We propose a new offline RL method that never needs to evaluate actions outside of the dataset, but still enables the learned policy to improve substantially over the best behavior in the data through generalization. The main insight in our work is that, instead of evaluating unseen actions from the latest policy, we can approximate the policy improvement step implicitly by treating the state value function as a random variable, with randomness determined by the action (while still integrating over the dynamics to avoid excessive optimism), and then taking a state conditional upper expectile of this random variable to estimate the value of the best actions in that state. This leverages the generalization capacity of the function approximator to estimate the value of the best available action at a given state without ever directly querying a Q-function with this unseen action. Our algorithm alternates between fitting this upper expectile value function and backing it up into a Q-function, without any explicit policy. Then, we extract the policy via advantage-weighted behavioral cloning, which also avoids querying out-of-sample actions. We dub our method in-sample Q-learning (IQL). IQL is easy to implement, computationally efficient, and only requires fitting an additional critic with an asymmetric L2 loss. IQL demonstrates the state-of-the-art performance on D4RL, a standard benchmark for offline reinforcement learning. We also demonstrate that IQL achieves strong performance fine-tuning using online interaction after offline initialization.

1 INTRODUCTION

Offline reinforcement learning (RL) addresses the problem of learning effective policies entirely from previously collected data, without online interaction (Fujimoto et al., 2019; Lange et al., 2012). This is very appealing in a range of real-world domains, from robotics to logistics and operations research, where real-world exploration with untrained policies is costly or dangerous, but prior data is available. However, this also carries with it major challenges: improving the policy beyond the level of the behavior policy that collected the data requires estimating values for actions other than those that were seen in the dataset, and this, in turn, requires trading off policy improvement against distributional shift, since the values of actions that are too different from those in the data are unlikely to be estimated accurately. Prior methods generally address this by either constraining the policy to limit how far it deviates from the behavior policy (Fujimoto et al., 2019; Wu et al., 2019; Fujimoto & Gu, 2021; Kumar et al., 2019; Nair et al., 2020; Wang et al., 2020), or by regularizing the learned value functions to assign low values to out-of-distribution actions (Kumar et al., 2020; Kostrikov et al., 2021). Nevertheless, this imposes a trade-off between how much the policy improves and how vulnerable it is to misestimation due to distributional shift. Can we devise an offline RL method that avoids this issue by never needing to directly query or estimate values for actions that were not seen in the data?

In this work, we ask whether it is possible to learn an optimal policy without *ever* querying the values of any unseen actions. The key idea in our method is to approximate an upper expectile of the distribution over Q-values with respect to the distribution of dataset actions for each state. We alternate between fitting this value function with expectile regression, and then using it to compute Bellman backups for training the Q-function. Once this Q-function has converged, we extract the corresponding policy using advantage-weighted behavioral cloning. Thus, our approach never queries values for unseen actions, either to compute the Bellman backups or to extract the policy. Therefore, it does not require either explicit constraints or explicit regularization of out-of-distribution actions during value function training, though our policy extraction step does implicitly enforce a constraint, as discussed in prior work on advantage-weighted regression (Nair et al., 2020; Wang et al., 2020; Peters & Schaal, 2007; Peng et al., 2019).

Our main contribution is IQL, a new offline RL algorithm that avoids ever querying values of unseen actions while still being able to perform multi-step dynamic programming updates. Our method is easy to implement by making a small change to the loss function in a simple SARSA-like TD update and is computationally very efficient. Furthermore, our approach demonstrates the state-of-the-art performance on D4RL, a popular benchmark for offline reinforcement learning. In particular, our approach significantly improves over the prior state-of-the-art on challenging Ant Maze tasks that require to "stitch" several sub-optimal trajectories. Moreover, we demonstrate that our approach is suitable for finetuning. Finally, after initialization from offline RL, IQL is capable of improving policy performance utilizing additional interactions.

2 Related work

A significant portion of recently proposed offline RL methods are based on either constrained or regularized approximate dynamic programming (e.g., Q-learning or actor-critic methods), with the constraint or regularizer serving to limit deviation from the behavior policy. We will refer to these methods as "multi-step dynamic programming" algorithms, since they perform true dynamic programming for multiple iterations, and therefore can in principle recover the optimal policy if provided with high-coverage data. The constraints can be implemented via an explicit density model (Wu et al., 2019; Fujimoto et al., 2019; Kumar et al., 2019), implicit divergence constraints (Nair et al., 2020; Wang et al., 2020; Peters & Schaal, 2007; Peng et al., 2019), or by adding a supervised learning term to the policy improvement objective (Fujimoto & Gu, 2021). Several works have also proposed to directly regularize the Q-function to produce low values for out-of-distribution actions (Kostrikov et al., 2021; Kumar et al., 2020). Our method is also a multi-step dynamic programming algorithm. However, in contrast to prior works, our method completely avoids directly querying the learned Q-function with unseen actions during training, removing the need for any constraint during this stage, though the subsequent policy extraction, which is based on advantage-weighted regression (Peng et al., 2019; Nair et al., 2020), does apply an implicit constraint. However, this policy does not actually influence value function training.

In contrast to multi-step dynamic programming methods, several recent works have proposed methods that rely either on a single step of policy iteration, fitting the value function or Q-function of the behavior policy and then extracting the corresponding greedy policy (Peng et al., 2019; Brandfonbrener et al., 2021), or else avoid value functions completely and utilize behavioral cloning-style objectives (Chen et al., 2021). We collectively refer to these as "single-step" approaches. These methods avoid needing to query unseen actions as well, since they either use no value function at all, or learn the value function of the behavior policy. Although these methods are simple to implement and effective on the MuJoCo locomotion tasks in D4RL, we show that such single-step methods perform very poorly on more complex datasets in D4RL, which require combining parts of suboptimal trajectories ("stitching"). Prior multi-step dynamic programming methods perform much better in such settings, as does our method. We discuss this distinction in more detail in Section 5.1. Our method also shares the simplicity and computational efficiency of single-step approaches, providing an appealing combination of the strengths of both types of methods.

Our method is based on estimating the characteristics of a random variable. Several recent works involve approximating statistical quantities of the value function distribution. In particular, quantile regression (Koenker & Hallock, 2001) has been previously used in reinforcement learning to estimate the quantile function of a state-action value function (Dabney et al., 2018b;a; Kuznetsov et al.,

2020). Although our method is related, in that we perform expectile regression, we are not modeling the uncertainty over the values of the state-action value function for a fixed state and action. Rather, we estimate expectiles of a state-action value function with respect to the dataset action distribution. This is a very different statistic: our aim is not to determine how the Q-value can vary with different future outcomes, but how the Q-value can vary with different actions *while averaging together future outcomes due to stochastic dynamics*. That is, the source of randomness is the action. While prior work on distributional RL can also be used for offline RL, it would suffer from the same action extrapolation issues as other methods, and would require similar constraints or regularization, while our method does not.

3 PRELIMINARIES

The RL problem is formulated in the context of a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, p_0(s), p(s'|s,a), r(s,a), \gamma)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $p_0(s)$ is a distribution of initial states, $p(s'|s,a)$ is an environment dynamics, $r(s,a)$ is a reward function, and γ is an MDP discount. The agent interacts with the MDP according to a policy $\pi(a|s)$. The goal is to obtain a policy that maximizes the cumulative discounted returns:

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

Off-policy RL methods based on approximate dynamic programming typically utilize a state-action value function (Q-function), referred to as $Q(s,a)$, which correspond to the discounted returns obtained by starting from the state s and action a, and then following the policy π. We make no assumptions regarding the state and action spaces, and the proposed algorithms can be applied to discrete and continuous action spaces without significant modification.

Offline reinforcement learning. In contrast to online (on-policy or off-policy) RL methods, offline RL uses previously collected data without any additional data collection. Like many recent offline RL methods, our work builds on approximate dynamic programming methods that minimize temporal difference error, according to the following loss:

$$L_{TD}(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[(r(s,a) + \gamma \max_{a'} Q_{\hat{\theta}}(s',a') - Q_\theta(s,a))^2], \tag{1}$$

where $\mathcal{D}$ is the dataset, $Q_\theta(s,a)$ is a parameterized Q-function, $Q_{\hat{\theta}}(s,a)$ is a target network (e.g., with soft parameters updates defined via Polyak averaging), and the policy is defined as $\pi(s) = \arg\max_a Q_\theta(s,a)$. Most recent offline RL methods modify either the value function loss (above) to regularize the value function in a way that keeps the resulting policy close to the data, or constrain the $\arg\max$ policy directly. This is important because out-of-distribution actions a' can produce erroneous values for $Q_{\hat{\theta}}(s',a')$ in the above objective, often leading to *overestimation* as the policy is defined to maximize the (estimated) Q-value.

4 IN-SAMPLE Q-LEARNING

In this work, we instead entirely avoid querying *out-of-sample* (unseen) actions in our TD loss. We start by considering fitted Q evaluation with a SARSA-style objective, which simply aims to learn the value of the dataset policy π_β (also called the behavior policy):

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

This objective never queries values for out-of-sample actions, in contrast to Eqn. (1). One specific property of this objective that is important for this work is that it uses mean squared error (MSE) that fits $Q_\theta(s,a)$ to predict the mean statistics of the TD targets. Therefore, for the optimal values of the parameters we get

$$Q_{\theta^*}(s,a) \approx r(s,a) + \gamma \mathbb{E}_{\substack{s'\sim p(\cdot|s,a) \\ a'\sim\pi_\beta(\cdot|s)}}[Q_{\hat{\theta}}(s',a')]. \tag{3}$$

Prior work (Brandfonbrener et al., 2021) has proposed directly using this objective to learn Q^{π_β}, and then train the policy π_ψ to maximize Q^{π_β}. This avoids any issues with out-of-distribution actions,

since the TD loss only uses dataset actions. However, while this procedure works well empirically on simple MuJoCo locomotion tasks in D4RL, we will show that it performs very poorly on more complex tasks that benefit from multi-step dynamic programming. In our method, which we derive next, we retain the benefits of using this SARSA-like objective, but modify it so that it allows us to perform multi-step dynamic programming and learn a near-optimal Q-function.

Our method will perform a Q-function update similar to Eqn. (2), but we will maintain a separate estimate of the value function $V(s')$ in the target value, which we will train to estimate the maximum Q-value over actions that are in the support of the data distribution. Crucially, we will show that it is possible to do this *without every querying the learned Q-function on out-of-sample actions* by utilizing expectile regression. Formally, the value function we aim to learn is given by:

$$V^*(s) = \max_{\substack{a \in \mathcal{A} \\ \text{s.t. } \pi_\beta(a|s) > 0}} Q(s, a).$$

This work aims to estimate this value function by evaluating the Q-function only for the state-action pairs sampled from the dataset. Since directly learning this state-conditional maximum by sampling is not feasible, we propose to estimate state-conditional quantiles of expectiles of $Q(s, a)$ evaluated on these samples. Then, taking specific state-conditional quantiles or expectiles approximates the maximization defined above. We can extend this procedure by using the approximated maximum to estimate target backups for fitting Q-function. As we demonstrate in Section 4.4, alternating between approximating the maximum and updating the Q-function we can approximate multi-step dynamical programming.

4.1 Expectile Regression

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

$$\arg\min_{m_\tau} \mathbb{E}_{x \sim X}[L_2^\tau(x - m_\tau)]$$

where

$$L_2^\tau(u) = |\tau - \mathbb{1}(u < 0)|u^2.$$

That is, for $\tau > 0.5$, this asymmetric loss function downweights the contributions of x values smaller than m_τ while giving more weights to larger values (see Fig. 1, left).

Expectile regression is closely related to quantile regression (Koenker & Hallock, 2001), which is a popular technique for estimating quantiles of a distribution widely used in reinforcement learning (Dabney et al., 2018b;a) [1]. In particular, for the quantile $\tau \in [0, 1]$, the quantile regression loss is defined as an asymmetric ℓ_1 loss:

$$L_1^\tau(u) = |\tau - \mathbb{1}(u < 0)| \cdot |u|.$$

We can also use this formulation to predict expectiles of a conditional distribution:

$$\arg\min_{m_\tau(x)} \mathbb{E}_{(x,y) \sim \mathcal{D}}[L_2^\tau(y - m_\tau(x))].$$

Fig. 1 (right) illustrates conditional expectile regression on a simple two-dimensional distribution. Note that we can optimize this objective with stochastic gradient descent. It provides unbiased gradients and is easy to implement with standard machine learning libraries.

[1] Our method could also be derived with quantiles, but since we are not interested in learning all of the expectiles/quantiles, unlike prior work (Dabney et al., 2018b;a), it is more convenient to estimate a single expectile because this involves a simple modification to the MSE loss that is already used in standard RL methods. Also, we found it to work somewhat better than quantile regression with its corresponding ℓ_1 loss.

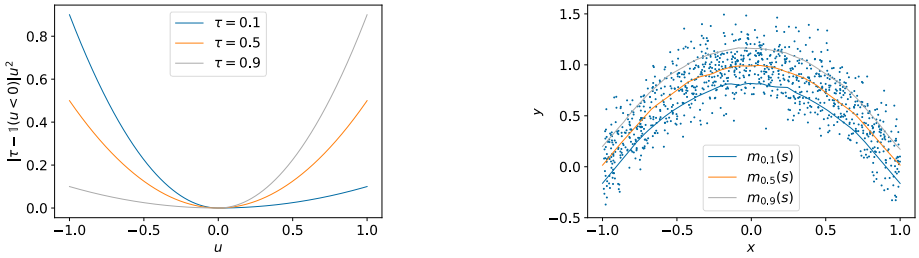


Figure 1: **Left:** The asymmetric squared loss used for expectile regression. $\tau = 0.5$ corresponds to the standard mean squared error loss, while $\tau = 0.9$ gives more weight to positives differences. **Right:** an example of estimating state conditional expectiles of a two-dimensional random variable. Each x corresponds to a distribution over y. We can approximate a maximum of this random variable with expectile regression: $\tau = 0.5$ correspond to the conditional mean statistics of the distribution, while $\tau \approx 1$ approximates the maximum operator over in-support values of y.

4.2 Learning the Value Function with Expectile Regression

Expectile regression provides us with a powerful framework to estimate statistics of a random variable beyond mean regression. We can use expectile regression to modify the policy evaluation objective in Eqn. (1) to update the state-action values with Q-targets that approximate the maximum of $r(s,a) + \gamma Q_{\hat{\theta}}(s', a')$ over a', instead of the mean value. In particular, we can perform expectile regression as follows:

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

However, this formulation has a significant drawback. Instead of estimating expectiles with respect to the actions in the support of the data, it also incorporates stochasticity that comes from the environment dynamics $s' \sim p(\cdot|s,a)$. Therefore, a large target value might not necessarily reflect the existence of a single action that achieves that value, but rather a "lucky" sample that happened to have transitioned into a good state. We resolve this problem by introducing a separate value function that approximates an expectile only with respect to the action distribution. Nonetheless, the formulation will still pick not only the state-action value uncertainty with respect to the action distribution but also model uncertainty. For this reason, we also use a version of clipped Double Q-learning (Fujimoto et al., 2018); we take a minimum of two Q-functions for V-function updates:

$$L_V(\psi) = \mathbb{E}_{(s,a) \sim \mathcal{D}}[L_2^\tau(\min_{i=1,2} Q_{\hat{\theta}_i}(s,a) - V_\psi(s))^2]. \tag{4}$$

We can then use this estimate to update the Q-functions with the standard MSE loss, which averages the uncertainty that comes from the environment dynamics, using the learned value function to estimate the target value:

$$L_Q(\theta_i) = \mathbb{E}_{(s,a,s') \sim \mathcal{D}}[(r(s,a) + \gamma V_\psi(s') - Q_{\theta_i}(s,a))^2] \text{ for } i = 1, 2. \tag{5}$$

Note that we do not use any explicit policy, and we sample actions only from the dataset for both objectives, similarly to SARSA-style policy evaluation.

4.3 Policy Extraction and Algorithm Summary

Our modified TD learning procedure learns an approximation to the optimal Q-function. However, it does not explicitly represent the corresponding policy and therefore requires a separate policy extraction step. In the spirit of preserving simplicity and efficiency, we aim for a simple method for policy extraction. As before, we want to avoid using out-of-samples actions in this stage. Therefore, we extract the policy using advantage weighted regression (Peters & Schaal, 2007; Nair et al., 2020; Wang et al., 2018; Peng et al., 2019):

$$L_\pi(\phi) = \mathbb{E}_{(s,a) \sim \mathcal{D}}[\exp(\beta(\min_{i=1,2} Q_{\hat{\theta}_i}(s,a) - V_\psi(s))) \log \pi_\phi(a|s)], \tag{6}$$

where $\beta \in [0, \infty)$ is an inverse temperature. For smaller hyperparameter values, the objective behaves similarly to behavioral cloning, while for larger values, it attempts to recover the maximum of the Q-function. This objective corresponds to weighted supervised learning, where more weight is given to the actions with larger Q-values.

Therefore, our final algorithm consists of two stages. First, we fit the value function and Q. In particular, we perform a number of gradient updates alternating between Eqn. (4) and (5). Second, we perform stochastic gradient descent on Eqn. (6). We summarize our final method in Algorithm 1. Note that the value function is learned entirely independently of the policy, and therefore these stages can be performed either sequentially or in parallel.

Algorithm 1 In-sample Q-learning

Initialize parameters ψ, θ_i, $\hat{\theta}_i$, ϕ.
TD learning (IQL):
for each gradient step **do**
  $\psi \leftarrow \psi - \lambda_V \nabla_\psi L_V(\psi)$
  for $i = 1, 2$ **do**
    $\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} L_Q(\theta_i)$
    $\hat{\theta}_i \leftarrow (1-\alpha)\hat{\theta}_i + \alpha\theta_i$
  end for
end for
Policy extraction (AWR):
for each gradient step **do**
  $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi L_\pi(\phi)$
end for

4.4 Analysis

First, we prove a simple lemma that we will then use to show how our approach can enable learning the optimal value function.

Lemma 1. *Let X be a random variable with a bounded support and supremum x^*. Then,*

$$\lim_{\tau \to 1} m_\tau = x^*$$

Proof. Expectiles of a random variable are bounded by the same x^*. Moreover, for all τ_1 and τ_2 such that $\tau_1 < \tau_2$, we get $m_{\tau_1} \leq m_{\tau_2}$. Therefore, the limit follows from the properties of bounded monotonically non-decreasing functions. □

In the following theorems, we show that under certain assumptions, our method indeed approximates the optimal state-action value Q^* and performs multi-step dynamical programming. We first prove a technical lemma relating different expectiles of the Q-function, and then derive our main result regarding the optimality of our method.

For the sake of simplicity, we introduce the following notation for our analysis. Let $\mathbb{E}^\tau_{x \sim X}[x]$ be a τth expectile of X (e.g., $\mathbb{E}^{0.5}$ corresponds to the standard expectation). Then, we define $V_\tau(s)$ and $Q_\tau(s,a)$, which correspond to optimal solutions of Eqn. 4 and 5 correspondingly, recursively as:

$$V_\tau(s) = \mathbb{E}^\tau_{a \sim \mu(\cdot|s)}[Q_\tau(s,a)],$$
$$Q_\tau(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim p(\cdot|s,a)}[V_\tau(s')].$$

Lemma 2. *For all s, τ_1 and τ_2 such that $\tau_1 < \tau_2$ we get*

$$V_{\tau_1}(s) \leq V_{\tau_2}(s).$$

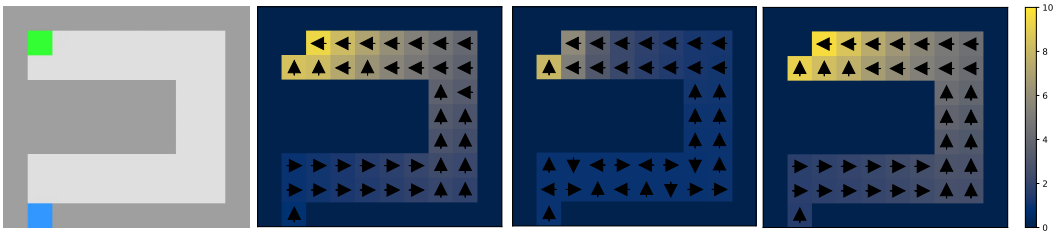
Proof. The proof mostly follows the policy improvement proof (Sutton & Barto, 2018). See appendix A for a proof. □

Corollary 2.1. *For any τ and s we have*

$$V_\tau(s) \leq \max_{\substack{a \in \mathcal{A} \\ s.t.\ \pi_\beta(a|s) > 0}} Q^*(s,a)$$

where $V_\tau(s)$ is defined as above and $Q^*(s,a)$ is an optimal state-action value function constrained to the dataset and defined as

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim p(\cdot|s,a)} \left[\max_{\substack{a' \in \mathcal{A} \\ s.t.\ \pi_\beta(a'|s') > 0}} Q^*(s',a') \right].$$



(a) Maze (b) Ground Truth (c) Onestep RL (d) IQL

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

Proof. The proof follows from the observation that convex combination is smaller than maximum. □

Theorem 3.

$$\lim_{\tau \to 1} V_\tau(s) = \max_{\substack{a \\ s.t. a \sim \mu(\cdot|s)}} Q^*(s, a).$$

Proof. Follows from combining lemma 1 and Corollary 2.1. □

Therefore, for a larger value of $\tau < 1$, we get a better approximation of the maximum. On the other hand, it also becomes a more challenging optimization problem. Thus, we treat τ as a hyper-parameter. Due to the property discussed in Theorem 3 we dub our method in-sample Q-learning (IQL).

5 Experimental Evaluation

We evaluate the performance of our method on a variety of offline RL tasks, ranging from didactic examples on gridworlds to high-dimensional benchmarks. We aim to evaluate how our method compares to prior work, and whether our approach can perform multi-step dynamic programming and estimate a near-optimal value function using only the actions in the dataset. We first evaluate our algorithm on a simple maze environment to illustrate the difference between our approach one-step policy improvement (Brandfonbrener et al., 2021). Then, we evaluate our approach against prior state-of-the-art methods on D4RL (Fu et al., 2020), a standard benchmark for offline reinforcement learning.

5.1 The Difference Between One-Step Policy Improvement and IQL

We first use a simple maze environment to illustrate the importance of multi-step dynamic programming for offline RL. The maze has a u-shape, a single start state, and a single goal state (see Fig. 2a). The agent receives a reward of 10 for entering the goal state and zero reward for all other transitions. With a probability of 0.25, the agent transitions to a random state, and otherwise to the commanded state. The dataset consists of 1 optimal trajectory and 99 trajectories with uniform random actions. Due to a short horizon of the problem, we use $\gamma = 0.9$.

Fig. 2 (b, c) illustrates the difference between Onepstep RL (Brandfonbrener et al., 2021; Wang et al., 2018) and IQL with $\tau = 0.95$. Although states closer to the high reward state will still have higher values, these values decay much faster as we move further away than they would for the optimal value function, and the resulting policy is highly suboptimal. At the same time, since our method performs iterative dynamic programming, it correctly propagates the signal, and the values are no longer dominated by noise.

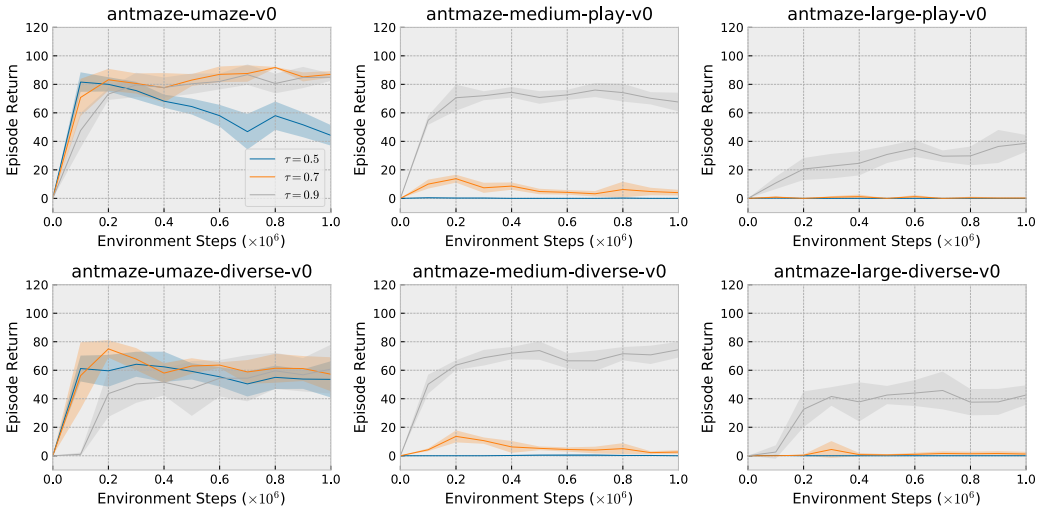


Figure 3: Estimating a larger expectile τ is crucial for antmaze tasks that require dynamical programming ('stitching'). IQL with $\tau = 0.5$ is not able to propagate the signal while a larger expectile $\tau = 0.9$ reaches the state-of-the-art performance on the tasks.

5.2 Comparisons on Offline RL Benchmarks

Next, we evaluate our approach on the D4RL benchmark in comparison to prior methods. In particular, we focus on two sets of tasks from the benchmark: the widely used MuJoCo locomotion tasks, and the Ant Maze tasks. The latter are particularly challenging for offline RL algorithms that do not perform multi-step dynamic programming, because on the harder mazes, the dataset has few or no trajectories that traverse a full path from the start to the goal. Instead, offline RL methods must stitch together different parts of suboptimal trajectories between different start-goal pairs in order to recover near-optimal behavior (Fu et al., 2020). We provide further details, including runtime and hyperparameters, in appendix B.

Effect of τ hyperparameter. We also demonstrate that it is crucial to compute a larger expectile on tasks that require "stitching" (see Fig. 3). For $\tau = 0.5$, which corresponds to standard one-step evaluation, our method fails to learn a good policy on more challenging antmaze-medium and antmaze-large tasks which require dynamical programming ('stitching' several trajectories).

5.3 Online Fine-tuning after Offline RL

While offline RL is an important problem, the solution obtained by offline RL can often be improved upon with a small amount of online interaction. For instance, the antmaze domain is one where no offline method achieves a near-maximum score, suggesting the need for active interaction and online improvement. IQL is suited for online fine-tuning after offline RL for two reasons. First, IQL has strong offline performance, as shown previously with state-of-the-art performance on D4RL datasets. This provides a strong initialization for fine-tuning. Second, IQL implements a weighted behavioral cloning policy improvement step, which

Dataset	AWAC	CQL	Ours
antmaze-umaze-v0	56.7 → 59.0	70 → **99**	86.7 → **96**
antmaze-umaze-diverse-v0	49.3 → 49	31 → **99**	75 → 84
antmaze-medium-play-v0	0 → 0	23 → 0	72 → **95**
antmaze-medium-diverse-v0	0.7 → 0.3	23 → 49	68.3 → **92**
antmaze-large-play-v0	0 → 0	1 → 0	25.5 → **46**
antmaze-large-diverse-v0	1 → 0	4.3 → 0	42.6 → **60.7**
antmaze total	107.7 → 108.3	152.3 → 247	370.1 → **473.7**

Table 2: Results for online finetuning experiments. Each entry shows the initial performance after offline RL, followed by performance after 1M steps of online RL. In all six antmaze environments, IQL is able to finetune to a significantly higher performance.

Table 1: Averaged normalized scores on MuJoCo locomotion and Ant Maze tasks. Our method outperforms prior methods on the challenging Ant Maze tasks, which require dynamic programming. We measure runtime for our reimplementations in JAX, which are faster than the original implementations, for all methods except Decision Transformers (DT). For DT, we take numbers reported by Fujimoto & Gu (2021).
*: Note that it is challenging to directly compare one-step and multi-step methods. Also, Brandfonbrener et al. (2021) reports results for a set of hyperparameters that is significantly different to other methods. We report runtime for a comparable set of hyperparameters (batch and network size). Note that it corresponds to marginally worse performance.

Dataset	BC	10%BC	DT	AWAC	Onestep RL	TD3+BC	CQL	Ours
halfcheetah-medium	42.6	42.5	42.6	43.5	48.4	48.3	44.0	45.0
hopper-medium	52.9	56.9	67.6	57.0	59.6	59.3	58.5	57.4
walker2d-medium	75.3	75.0	74.0	72.4	81.8	83.7	72.5	79.1
halfcheetah-medium-replay	36.6	40.6	36.6	40.5	38.1	44.6	45.5	43.0
hopper-medium-replay	18.1	75.9	82.7	37.2	97.5	60.9	95.0	63.1
walker2d-medium-replay	26.0	62.5	66.6	27.0	49.5	81.8	77.2	66.6
halfcheetah-medium-expert	55.2	92.9	86.8	42.8	93.4	90.7	91.6	80.6
hopper-medium-expert	52.5	110.9	107.6	55.8	103.3	98.0	105.4	107.9
walker2d-medium-expert	107.5	109	108.1	74.5	113	110.1	108.8	108.8
locomotion total	466.7	666.2	672.6	450.7	684.6	677.4	698.5	651.5
antmaze-umaze-v0	54.6	62.8	59.2	56.7	64.3	78.6	74.0	85.0
antmaze-umaze-diverse-v0	45.6	50.2	53.0	49.3	60.7	71.4	84.0	60.8
antmaze-medium-play-v0	0.0	5.4	0.0	0.0	0.3	10.6	61.2	67.6
antmaze-medium-diverse-v0	0.0	9.8	0.0	0.7	0.0	3.0	53.7	74.4
antmaze-large-play-v0	0.0	0.0	0.0	0.0	0.0	0.2	15.8	38.6
antmaze-large-diverse-v0	0.0	6.0	0.0	1.0	0.0	0.0	14.9	42.6
antmaze total	100.2	134.2	112.2	107.7	125.3	163.8	303.6	369.0
total	566.9	800.4	784.8	558.4	809.9	841.2	1002.1	1020.5
runtime	10m	10m	960m	20m	$\approx$ 20m*	20m	80m	20m

has previously been shown to allow for better online policy improvement compared to other types of offline constraints such as CQL (Kumar et al., 2020; Nair et al., 2020). To evaluate the finetuning capability of various RL algorithms, we first run offline RL on a dataset, then run 1M steps of online RL and report the final performance.

The results of online finetuning for AWAC, CQL, and IQL on the antmaze datasets in D4RL are shown in Table 2. On all six tasks, IQL, which was already the best performing offline RL algorithm, is able to improve significantly with online finetuning. In contrast, the other two methods evaluated do not improve.

Dataset	AWAC	Ours
pen-binary-v0	45 $\rightarrow$ **70**	87 $\rightarrow$ 61
door-binary-v0	1 $\rightarrow$ 30	1 $\rightarrow$ 31
relocate-binary-v0	1 $\rightarrow$ 20	1 $\rightarrow$ **38**
total	47 $\rightarrow$ 125	39 $\rightarrow$ 130

Table 3: Results for online finetuning experiments on dextrous manipulation tasks from Nair et al. (2020), which are a mix of demonstration and behavioral cloning data. Each entry shows the initial performance after offline RL, then performance after additional online RL.

6 Conclusion

We presented In-sample Q-Learning, a general algorithm for offline RL that completely avoids any queries to out-of-sample actions during training while still enabling multi-step dynamic programming. To our knowledge, this is the first method that combines both of these features. This has a number of important benefits. First, our algorithm is computationally efficient: we can perform 1M updates on one GTX1080 GPU in less than 20 minutes. Second, it is simple to implement, requiring only minor modifications over a standard SARSA-like TD algorithm and performing policy extraction with a simple weighted behavioral cloning procedure resembling supervised learning. Third, the method is general, makes no environment-specific assumptions, and can be applied to continuous and discrete action spaces. Despite the simplicity and efficiency of this method, we show that it attains excellent performance across all of the tasks in the D4RL benchmark, matching the best prior methods on the MuJoCo locomotion tasks, where methods based on behavioral cloning or a single step of policy evaluation tend to produce the best prior results, and exceeding the state-of-the-art performance on the challenging ant maze environments, where such BC-like or single-step methods perform poorly, and dynamical programming methods such as CQL attain by far the best prior results. This illustrates that our method successfully combines the best properties of both classes of approaches and successfully performs multiple steps of dynamic programming despite never querying out-of-sample actions.

REFERENCES

David Brandfonbrener, William F Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *arXiv preprint arXiv:2106.08909*, 2021.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pp. 1096–1105. PMLR, 2018a.

Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018b.

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. *arXiv preprint arXiv:2106.06860*, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062. PMLR, 2019.

Roger Koenker and Kevin F Hallock. Quantile regression. *Journal of economic perspectives*, 15(4): 143–156, 2001.

Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783. PMLR, 2021.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, pp. 5556–5566. PMLR, 2020.

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

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

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

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

Qing Wang, Jiechao Xiong, Lei Han, Peng Sun, Han Liu, and Tong Zhang. Exponentially weighted imitation learning for batched historical data. In *NeurIPS*, pp. 6291–6300, 2018.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *arXiv preprint arXiv:2006.15134*, 2020.

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