Continuous Doubly Constrained Batch Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Reliant on too many experiments to learn good actions, current Reinforcement Learning (RL) algorithms have limited applicability in real-world settings, which can be too expensive to allow exploration. We propose an algorithm for batch RL, where effective policies are learned using only a fixed offline dataset instead of online interactions with the environment. The limited data in batch RL produces inherent uncertainty in value estimates of states/actions that were insufficiently represented in the training data. This leads to particularly severe extrapolation when our candidate policies diverge from one that generated the data. We propose to mitigate this issue via two straightforward penalties: a policy-constraint to reduce this divergence and a value-constraint that discourages overly optimistic estimates. Over a comprehensive set of 32 continuous-action batch RL benchmarks, our approach compares favorably to state-of-the-art methods, regardless of how the offline data were collected.

1 Introduction

Deep RL algorithms have demonstrated impressive performance in simulable digital environments like video games [40, 54, 55]. In these settings, the agent can execute different policies and observe their performance. Barring a few examples [36], advancements have not translated quite as well to real-world environments, where it is typically infeasible to experience millions of environmental interactions [12]. Moreover, in presence of an acceptable heuristic, it is inappropriate to deploy an agent that learns from scratch hoping that it may eventually outperform the heuristic after sufficient experimentation.

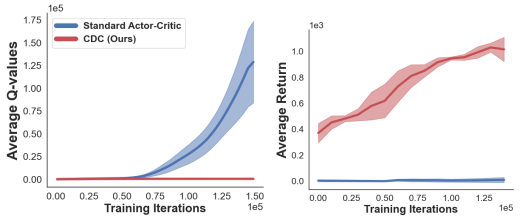


Figure 1: **Batch RL with CDC vs. No CDC**. Left: Standard actor-critic overestimates Q-values whereas CDC estimates are well controlled. Right: Wild overestimation leads to worse-performing policies whereas CDC performs well.

The setting of *batch* or *offline* RL instead offers a more pertinent framework to learn performant policies for real-world applications [33, 57]. Batch RL is widely applicable because this setting does not require that: a proposed policy be tested through real environment interactions, or that data be collected under a particular policy. Instead, the agent only has access to a fixed dataset $\mathcal{D}$ collected through actions taken according to some unknown *behavior* policy π_b. The main challenge in this setting is that data may only span a small subset of the possible state-action pairs. Worst yet, the agent cannot observe the effects of novel out-of-distribution (OOD) state-action combinations that, by definition, are not present in $\mathcal{D}$.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

A key challenge stems from the inherent uncertainty when learning from limited data [27, 35]. Failure to account for this can lead to wild extrapolation [17, 28] and over/under-estimation bias in value estimates [21, 22, 31, 58]. This is a systemic problem that is exacerbated for out-of-distribution (OOD) state-actions where data is scarce. Standard temporal difference updates to Q-values rely on the Bellman optimality operator which implies upwardly-extrapolated estimates tend to dominate these updates. As Q-values are updated with overestimated targets, they become upwardly biased even for state-actions well-represented in $\mathcal{D}$. In turn, this can further increase the upper limit of the extrapolation errors at OOD state-actions, which forms a vicious cycle of extrapolation-inflated overestimation (*extra-overestimation* for short) shown in Figure 1. This extra-overestimation is much more severe than the usual overestimation bias encountered in online RL [21, 58]. As such, we critically need to constrain value estimates whenever they lead to situations that look potentially 'too good to be true', in particular when they occur where a policy might exploit them.

Likewise, naive exploration can lead to policies that diverge significantly from π_b. This, in turn, leads to even greater estimation error since we have very little data in this un(der)-explored space. Note that this is not a reason for particular concern in online RL: after all, once we are done exploring a region of the space that turns out to be less promising than we thought, we simply update the value function and stop visiting or visit rarely. Not so in batch RL where we cannot adjust our policy based on observing its actual effects in the environment. These issues are exacerbated for applications with a large number of possible states and actions, such as the continuous settings considered in this work. Since there is no opportunity to try out a proposed policy in batch RL, learning must remain appropriately conservative for the policy to have reasonable effects when it is later actually deployed. Standard regularization techniques are leveraged in supervised learning to address such ill-specified estimation problems, and have been employed in the RL setting as well [13, 50, 61].

This paper adapts standard off-policy actor-critic RL to the batch setting by adding a simple pair of regularizers. In particular, our main contribution is to introduce two novel batch-RL regularizers: The first regularizer combats the extra-overestimation bias in regions that are out-of-distribution. The second regularizer is designed to hedge against the adverse effects of policy updates that severly diverge from $\pi_b(a|s)$. The resultant method, *Continuous Doubly Constrained Batch RL* (CDC) exhibits state-of-the-art performance across 32 continuous control tasks from the D4RL benchmark [15] demonstrating the usefulness of our regularizers for batch RL.

2 Background

Consider an infinite-horizon Markov Decision Process (MDP) [48], $(S, A, T, r, \mu_0, \gamma)$. Here S is the state space, $A \subset \mathbb{R}^d$ is a (continuous) action space, $T : S \times A \times S \to \mathbb{R}_+$ encodes transition probabilities of the MDP, μ_0 denotes the initial state distribution, $r(s, a)$ is the instantaneous reward obtained by taking action $a \in A$ in state $s \in S$, and $\gamma \in [0, 1]$ is a *discount factor* for future rewards.

Given a stochastic policy $\pi(a|s)$, the sum of discounted rewards generated by taking a series of actions $a_t \sim \pi(\cdot|s_t)$ corresponds to the *return* $R_t^\pi = \sum_{i=t}^{\infty} \gamma^{i-t} r(s_i, a_i)$ achieved under policy π. The *action-value function* (Q-value for short) corresponding to π, $Q^\pi(s, a)$, is defined as the expected return expected for starting at state s, taking a, and acting according to π thereafter, $Q^\pi(s, a) = \mathbb{E}_{s_t \sim T, a_t \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid (s_0, a_0) = (s, a)\right]$. $Q^\pi(s, a)$ obeys the Bellman equation [6]:

$$Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim T(\cdot|s,a), a' \sim \pi(\cdot|s')} \left[Q^\pi(s', a')\right] \tag{1}$$

Unlike in online RL, no interactions with the environment is allowed here, so the agent does not have the luxury of exploration. $\mathcal{D}$ is previously collected via actions taken according to some unknown *behavior policy* $\pi_b(a|s)$. In this work, we assume $\mathcal{D}$ consists of 1-step transition: $\{(s_i, a_i, r_i, s_i')\}_{i=1}^n$ where no further sample collection is permitted. In particular, we do not even require that complete episode trajectories have been logged [59, 62]. This is valuable, for instance, whenever data privacy and sharing restrictions prevent the use of the latter [33]. It is also useful when combining data from sources where the interaction is still in progress, e.g. from ongoing user interactions.

We aim to learn an *optimal* policy π^* that maximizes the expected return, denoting the corresponding Q-values for this policy as $Q^* = Q^{\pi^*}$. Q^* is the fixed point of the Bellman *optimality* operator [6]: $\mathcal{T}Q^*(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim T(\cdot|s,a)} \left[\max_{a'} Q^*(s', a')\right]$. One way to learn π^* is via actor-critic methods [26], with policy π_ϕ and Q-value Q_θ, parametrized by ϕ and θ respectively.

Learning good policies becomes far more difficult in batch RL as it depends on the quality/quantity of available data. Moreover, for continuous control the set of possible actions is infinite, making it

nontrivial to find the optimal action even for online RL. One option is to approximate the maximization above by only considering finitely many actions sampled from some π. This leads to the Expected Max-Q (EMaQ) operator of Ghasemipour et al. [18]:

$$\bar{\mathcal{T}}Q(s,a) := r(s,a) + \gamma \mathbb{E}_{s' \sim T(\cdot|s,a)} \left[\max_{\{a'_k\}} Q(s', a'_k) \right]. \tag{2}$$

Here $a'_k \sim \pi_\phi(\cdot|s')$ for $k = 1, ..., N$, i.e. the candidate actions are drawn IID from the current (stochastic) policy rather than over all possible actions. When drawing only a single sample from π_ϕ, this reduces to the standard Bellman operator (in expectation). Conversely, when $N \to \infty$ and π_ϕ has support over A, this turns into the Bellman optimality operator. We learn Q by minimizing the standard 1-step temporal difference (TD) error. That is, we update at iteration t

$$\theta_t \leftarrow \operatorname*{argmin}_\theta \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\left(Q_\theta(s,a) - \overline{\mathcal{T}} Q_{\theta_{t-1}}(s,a) \right)^2 \right] \tag{3}$$

Throughout, the notation $\mathbb{E}_{(s,a)\sim\mathcal{D}}$ denotes an *empirical* expectation over dataset $\mathcal{D}$, whereas expectations with respect to π are taken over the true underlying distribution corresponding to policy π. Next, we update the policy by increasing the likelihood of actions with higher Q-values:

$$\phi_t \leftarrow \operatorname*{argmax}_\phi \mathbb{E}_{s\sim\mathcal{D}, \hat{a}\sim\pi_\phi(\cdot|s)} \left[Q_{\theta_t}(s, \hat{a}) \right] \tag{4}$$

using off-policy gradient-based updates [53]. Depending on the context, we omit t from Q_{θ_t} and π_{ϕ_t}.

2.1 Extrapolation-Inflated Overestimation

When our Q-values are estimated via function approximation[1] (with parameters θ), the Q-update can be erroneous and noisy [58]. Let $Q_{\theta_t}(s,a)$ denote the estimates of true underlying $Q^*(s,a)$ values at iteration t of a batch RL algorithm that iterates steps (3) and (4), with π_{ϕ_t} denoting the policy that maximizes Q_{θ_t}. For a proper learning method, we might hope that the *estimation error*, ER $:= Q_{\theta_t}(s,a) - Q^*(s,a)$, has expected value $= 0$ and variance $\sigma > 0$ for particular states/actions (the expectation here is over the sampling variability in the dataset $\mathcal{D}$ and stochastic updates in our batch RL algorithm). However even in this desirable scenario, Jensen's inequality nonetheless implies there will be *overestimation error* OE $:= \mathbb{E}[\max_a Q_{\theta_t}(s,a)] - \max_a Q^*(a,s) \geq 0$ for the actions currently favored by π_{ϕ_t}. Here the expectation is over the randomness of the underlying dataset $\mathcal{D}$ and the learning algorithm. OE will be strictly positive when the estimation errors are weakly correlated and will grow with the ER-variance σ [31, 58]. Under the Bellman optimality or EMAQ operator, these inflated estimates are used as target values in the next Q-update in (3), which thus produces a $Q_{\theta_{t+1}}(s,a)$ estimate that suffers from *overestimation bias*, meaning it is expected to exceed the true Q value even if this was not the case for initial estimate $Q_{\theta_t}(s,a)$ [14, 16, 21, 22, 29, 32, 58].

In continuous batch RL, ER may have far greater variance (larger σ) for OOD states/actions poorly represented in the dataset $\mathcal{D}$, as our function approximator Q_{θ_t} may wildly extrapolate in these data-scarce regions [24, 33]. This in turn implies the updated policy π_{ϕ_t} will likely differ significantly from π_b and favor some action $\hat{a} = \operatorname{argmax}_a Q_{\theta_t}(s,a)$ that is OOD [17]. The estimated value of this OOD action subsequently becomes the target in the Q-update (3), and its OE will now be more severe due to the larger σ [8]. Even though we only apply these Q-updates to non-OOD $(s,a) \in \mathcal{D}$ whose ER may be initially smaller, the severely overestimated target values can induce increased overestimation bias in $Q_{\theta_{t+1}}(s,a)$ for $(s,a) \in \mathcal{D}$. In a vicious cycle, the increase in $Q_{\theta_{t+1}}(s,a)$ for $(s,a) \in \mathcal{D}$ can cause extrapolated $Q_{\theta_{t+1}}$ estimates to also grow for OOD actions (as there is no data to ground these OOD estimates), such that overestimation at $s, a \in \mathcal{D}$ is further amplified through additional temporal difference updates. After many iterative updates, this *extra-overestimation* can eventually lead to the disturbing explosion of value estimates seen in Figure 1.

Several strategies address overestimation [14, 16, 21, 22, 29, 32]. Fujimoto et al. [17] proposed a straightforward convex combination of the extremes of an estimated distribution over plausible Q values. Given a set of estimates Q_{θ_j} for $j = 1, ...M$, they combine both the maximum and the minimum value for a given (s,a) pair:

$$\overline{Q}_\theta(s,a) = \nu \min_j Q_{\theta_j}(s,a) + (1-\nu) \max_j Q_{\theta_j}(s,a) \tag{5}$$

[1] While overestimation bias has been mainly studied in regard to function approximation error [14, 16, 22, 32, 58], Hasselt [21] shows that overestimation can also arise in tabular MDPs due to noise in the environment.

Here $\nu \in (0, 1)$ determines how conservative we wish to be, and the $\min/\max$ are taken across M Q-networks that only differ in their weight-initialization but are otherwise (independently) estimated. For larger $\nu > 0.5$, $\overline{Q}$ may be viewed as a *lower confidence bound* for Q^* where the epistemic uncertainty in Q estimates is captured via an ensemble of deep Q-networks [10].

3 Methods

Our previous discussion of extra-overestimation suggests two key sources of potential error in batch RL. Firstly, a policy learned by our algorithm might be too different from the behavior policy, which can lead to risky actions whose effects are impossible to glean from the limited data. To address this, we propose to add an *exploration-penalty* in policy updates that reduces the divergence between our learned policy π_ϕ and the policy π_b that generated the data. Secondly, we must restrict overestimation in Q-values, albeit only where it matters, that is, only when this leads to a policy exploiting overly optimistic estimates. As such, we only need to penalize suspiciously large Q-values for actions potentially selected by our candidate policy π_ϕ (e.g. if their estimated Q-value greatly exceeds the Q-value of actually observed actions).

3.1 Q-Value Regularization

While sequential interaction with the environment is a strong requirement that limits the practical applicability of online RL (and leads to other issues like exploration vs. exploitation), it has one critical benefit: although unreliable extrapolation of Q-estimates beyond the previous observations happens during training, it is naturally corrected through further interaction with the environment. OOD state-actions with wildly overestimated values are in fact likely to be explored in subsequent updates, and their values then corrected after observing their actual effect.

In contrast, extra-overestimation is a far more severe issue in batch RL, where we must be confident in the reliability of our learned policy before it is deployed. The issue can lead to completely useless Q-estimates. The policies corresponding to these wildly extrapolated Q-functions will perform poorly, pursuing risky actions whose true effects cannot be known based on the limited data in $\mathcal{D}$. Although it helps lessen this issue to some extent, the lower confidence bound $\overline{Q}$ in (5) cannot alone resolve extra-overestimation, unless we use extremely pessimistic Q-estimates that no longer faithfully value different actions (see Figure 2a, Table S2, and Section 4 of [35]).

To mitigate the key issue of extra-overestimation in $Q_\theta(s, a)$, we consider three particular aspects:

- An overall shift in Q-value is less important. A change from, say $Q_\theta(s, a)$ to $Q_\theta(s, a) + c(s)$ changes nothing about which action we might want to pick. As such, we only penalize the *relative* shift between Q-values.
- An overestimation of $Q_\theta(s, \hat{a})$ which still satisfies $Q_\theta(s, \hat{a}) \ll Q_\theta(s, a)$ for well-established $a, s \in \mathcal{D}$ will not change behavior and does not require penalization.
- Lastly, overestimation only matters if our policy is capable of discovering and exploiting it.

We use these three aspects to design a penalty for Q-value updates to be more pessimistic [8, 24].

$$\Delta(s, a) := \left[\max_{\hat{a} \in \{a_1, \dots a_N\} \sim \pi_\phi(.|s)} Q_\theta(s, \hat{a}) - Q_\theta(s, a) \right]_+^2 \tag{6}$$

where $s, a \in D$. We can see that the first requirement is easily satisfied, since we only compare differences $Q_\theta(s, \hat{a}) - Q_\theta(s, a)$ for different actions, given the same state s. The second aspect is addressed by taking the maximum between 0 and $Q_\theta(s, \hat{a}) - Q_\theta(s, a)$. As such, we do not penalize optimism when it does not rise to the level where it would effect a change in behavior. Lastly, taking the maximum over actions drawn from the π rather than from the maximum over all possible actions ensures that we only penalize when the overestimation would have observable consequences. As such, we limit ourselves to a rather narrow set of cases. As a result, we add this penalty to the Q-update:

$$\theta_t \leftarrow \operatorname*{argmin}_\theta \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\left(Q_\theta(s, a) - \overline{\mathcal{T}} Q_{\theta_{t-1}}(s, a) \right)^2 + \eta \cdot \Delta(s, a) \right] \tag{7}$$

Anatomy of the extra-overestimation penalty Δ. Our proposed Δ penalty in (6) mitigates extra-overestimation bias by hindering the learned Q-value from wildly extrapolating large values for OOD

state-actions. Estimated values of actions previously never seen in (known) state $s \in \mathcal{D}$ are instead encouraged to not significantly exceed the values of the actions a whose effects we have seen at s. Note that the temporal difference update and the extra-overestimation penalty Δ in (7) are both framed on a common scale as a squared difference between two Q-functions.

How Δ affects θ becomes evident through its derivative:

$$\boldsymbol{\nabla}_\theta \Delta(s,a) = \begin{cases} \Big(\boldsymbol{\nabla}_\theta Q_\theta(s,\hat{a}) - \boldsymbol{\nabla}_\theta Q_\theta(s,a)\Big)\,\boldsymbol{\varepsilon} & \text{if } \boldsymbol{\varepsilon} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Here $\hat{a} := \arg\max_{\{\hat{a}_k\}_{k=1}^N} Q_\theta(s,\hat{a}_k)$ again taken over N actions sampled from our current policy π, and $\boldsymbol{\varepsilon} := Q_\theta(s,\hat{a}) - Q_\theta(s,a)$. Δ only affects certain temporal-differences where Q-values of (possibly OOD) state-actions have higher values than the $(s,a) \in \mathcal{D}$. In this case, Δ not only reduces $Q_\theta(s,\hat{a})$ by an amount proportional to $\boldsymbol{\varepsilon}$, but this penalty also increases the value of the previously-observed action $Q_\theta(s,a)$ to the same degree. Δ thus results in a value network that favors previously observed actions. We will generally want to choose a large conservative value of η in applications where we know either: that the behavior policy was of high-quality (since its chosen actions should then be highly valued), or that only a tiny fraction of the possible state-action space is represented in $\mathcal{D}$, perhaps due to small sample-size or a restricted behavior policy (since there may be severe extrapolation error).

3.2 Policy Regularization

In batch RL, the available offline data $\mathcal{D}$ can have varying quality depending on the behavior policy π_b used to collect the data. Since trying out actions is not possible in batch settings, our policy network is instead updated to favor not only actions with the highest estimated Q-value but also the actions observed in $\mathcal{D}$ (whose effects we can be more certain of). Thus we introduce an *exploration penalty* to regularize the policy update step: $\phi \leftarrow \operatorname{argmax}_\phi \mathbb{E}_{s\sim\mathcal{D},\hat{a}\sim\pi_\phi(\cdot|s)}\Big[Q_\theta(s,\hat{a})\Big] - \lambda \cdot \mathbb{D}(\pi_b, \pi_\phi)$.

In principle, various f-divergences [11] or Integral Probability Metrics [42] could employed in $\mathbb{D}(\cdot,\cdot)$. In practice, we limit our choice to quantities that do not require estimating the behavior policy π_b. This leaves us with the reverse KL-divergence and IPMs in Hilbert Space [2]. If we further restrict ourselves to distances that do not require sampling from π_ϕ, then only the reverse KL-divergence remains. We thus estimate

$$\mathrm{KL}(\pi_b, \pi_\phi) = \mathbb{E}_{a\sim\pi_b(\cdot|s)}[\log \pi_b(a|s)] - \mathbb{E}_{a\sim\pi_b(\cdot|s)}[\log \pi_\phi(a|s)] \tag{9}$$

$$\propto -\mathbb{E}_{a\sim\pi_b(\cdot|s)}[\log \pi_\phi(a|s)] \approx -\frac{1}{m}\sum_{i=1}^{m} \log \pi_\phi(a_i|s) \tag{10}$$

whenever $a_i \sim \pi_b(\cdot|s)$. This is exactly what happens in batch RL where we have plenty of data drawn from the behavior policy, albeit no access to its explicit functional form. Note the first entropy term in (9) can be ignored when we aim to minimize the estimated KL in terms of π_ϕ (as will be done in our exploration penalty). Using (10), we can efficiently minimize an estimated reverse KL divergence without having to know/estimate π_b or sample from π_ϕ.

Lemma 1 $\operatorname*{argmax}_{\pi_\phi} \mathbb{E}_{s,a\sim\pi_\phi}[Q_\theta(s,a)] - \lambda \cdot \mathbb{E}_s\big[\mathbb{D}\big(\pi_\phi(\cdot|s), \pi_b(\cdot|s)\big)\big]$ *is given by*

$$\pi_\phi(s|a) = \frac{\pi_b(a|s)}{Z}\exp\left(\frac{Q_\theta(s,a)}{\lambda}\right) \quad \textit{if } \mathbb{D} \textit{ is the forward KL divergence} = \mathrm{KL}(\pi_\phi(\cdot|s), \pi_b(\cdot|s))$$

$$\pi_\phi(s|a) = \frac{\pi_b(a|s)}{Z - Q_\theta(s,a)/\lambda} \quad \textit{if } \mathbb{D} \textit{ is the reverse KL divergence} = \mathrm{KL}(\pi_b(\cdot|s), \pi_\phi(\cdot|s))$$

where $Z \in \mathbb{R}$ is a normalizing constant in each case. Lemma 1 shows that using either forward or reverse KL-divergence as an objective, we recover $\pi_\phi = \pi_b$ in the limit of $\lambda \to \infty$. This is to be expected. After all, in this case we use the distance in distributions (thus policies) as our only criterion, and we prefer reverse KL to avoid having to estimate π_b. CDC thus employs the following policy-update (where the reverse KL is expressed as a log-likelihood as in (10))

$$\phi \leftarrow \operatorname*{argmax}_\phi \mathbb{E}_{s\sim\mathcal{D},\hat{a}\sim\pi_\phi(\cdot|s)}\Big[Q_\theta(s,\hat{a})\Big] + \lambda \cdot \mathbb{E}_{(s,a)\sim\mathcal{D}}\big[\log \pi_\phi(a|s)\big] \tag{11}$$

The exploration penalty helps ensure our learned π_ϕ is not significantly worse than π_b, which is far from guaranteed in batch settings without ever testing an action. If the data were collected by a fairly random (subpar) behavior policy, then this penalty (in expectation) acts similarly to a maximum-entropy term. The addition of such terms to similar policy-objectives has been shown to boost performance in RL methods like *soft actor-critic* [20].

Note that our penalization of exploration stands in direct contrast to online RL methods that specifically incentivize exploration [5, 44]. In the batch RL, exploration is extremely dangerous as it will only take place during deployment when a policy is no longer being updated in response to the effect of its actions. Constraining policy-updates around an existing data-generating policy has also been demonstrated as a reliable way to at least obtain an improved policy in both batch [17, 62] and online [51] settings. Even moderate policy-improvement can often be extremely valuable (the optimal policy may be too much ask for with data of limited size or coverage of the possible state-actions). *Reliable* improvement is crucial in batch settings as we cannot first test out our new policy.

Remark 1 (Behavioral cloning occurs as $\lambda \to \infty$) *Regularized policy updates with strong regularization (large λ) is in the limit imitation learning. In fact, this is the well-known likelihood based behavioral cloning algorithm used by [47].*

If the original behavior policy π_b^* was optimal (e.g. demonstration by a human-expert), then behavioral cloning should be utilized for learning from $\mathcal{D}$ [43]. However in practice, data are often collected from a subpar policy that we wish to improve upon via batch RL rather than simple imitation learning.

3.3 CDC Algorithm

Furnished with the tools for Q-value and policy regularization proposed in previous sections, we introduce CDC in Algorithm 1. CDC utilizes an actor-critic framework [26] for continuous actions with stochastic policy π_ϕ and Q-value Q_θ, parameterized by ϕ and θ respectively. Our major additions to that Δ penalty that mitigates overestimation bias by reducing wild extrapolation in value estimates and the *exploration penalty* ($\log \pi_\phi$) that discourages the estimated policy from straying to OOD state-actions very different from those whose effects we have observed in $\mathcal{D}$.

Algorithm 1 Continuous Doubly Constrained Batch RL

1: Initialize policy π_ϕ and Qs: $\{Q_{\theta_j}\}_{j=1}^M$
2: Initialize Target Qs: $\{Q_{\theta'_j} : \theta'_j \leftarrow \theta_j\}_{j=1}^M$
3: **for** t in $\{1, \ldots, \mathrm{T}\}$ **do**
4: Sample $\mathcal{B} = \{(s, a, r, s')\} \sim \mathcal{D}$
5: For each $s, s' \in \mathcal{B}$: sample N actions $\{\hat{a}_k\}_{k=1}^N \sim \pi_\phi(\cdot|s)$, $\{a'_k\}_{k=1}^N \sim \pi_\phi(\cdot|s')$
6: Q_θ**- value update:**

$$y(s') := r + \gamma \max_{a'_k} \left[\overline{Q}_{\theta'}(s', a'_k)\right] \quad (\overline{Q} \text{ given by Eq 5})$$

$$\Delta_j(s,a) := \left(\left[\max_{\hat{a}_k} Q_{\theta_j}(s, \hat{a}_k) - Q_{\theta_j}(s,a)\right]_+\right)^2$$

$$\theta_j \leftarrow \operatorname*{argmin}_{\theta_j} \sum_{(s,a,s')\in\mathcal{B}} \left[\left(Q_{\theta_j}(s,a) - y(s')\right)^2 + \eta \cdot \Delta_j(s,a)\right] \text{ for } j = 1, ..., M$$

7: π_ϕ **- policy update:**

$$\phi \leftarrow \operatorname*{argmax}_{\phi} \sum_{(s,a)\in\mathcal{B}, \hat{a}\sim\pi_\phi(\cdot|s)} \left[\overline{Q}_\theta(s, \hat{a}) + \lambda \cdot \log \pi_\phi(a|s)\right]$$

8: **Update Target Networks:**

$$\theta'_j \leftarrow \tau\theta_j + (1-\tau)\theta'_j \;\; \forall j \in M$$

9: **end for**

Although the particular form of CDC presented in Algorithm 1 optimizes a stochastic policy with the off-policy updates of [53] and temporal difference value-updates using (3), we emphasize that the general idea behind CDC can be utilized with other forms of actor-critic updates such as those considered by [13, 16, 20]. In practice, CDC estimates expectations of quantities introduced throughout via mini-batch estimates derived from samples taken from $\mathcal{D}$, and each optimization is performed via a few stochastic gradient method iterates.

To account for epistemic uncertainty due to the limited data, the value update in Step 6 of Algorithm 1 uses $\overline{Q}_\theta$ from (5) in place of Q_θ. In CDC, we can simply utilize the same moderately conservative value of $\nu = 0.75$ used by [17], since we are not purely relying on the lower confidence bound $\overline{Q}_\theta$ to correct all overestimation. For this reason, CDC is able to achieve strong performance with a small ensemble of $M = 4$ Q-networks (used throughout this work), whereas [18] require larger ensembles of 16 Q-networks and an extremely conservative $\nu = 1$ in order to achieve good performance.

To correct extra-overestimation within each of the M individual Q-networks, Algorithm 1 actually applies a separate extra-overestimation penalty Δ_j specific to each Q-network. The steps of our proposed CDC method are detailed in Algorithm 1. In blue, we highlight the only modifications CDC makes to a standard off-policy actor-critic framework that has been suitably adapted for continuous batch RL via the aforementioned techniques like EMAQ [18] and lower-confidence bounds for Q-values [17]. Throughout, we use $\eta = 0$ & $\lambda = 0$ to refer to this baseline framework (without our

proposed penalties), and note that majority of modern batch RL methods like CQL [28], BCQ [17], BEAR [27], BRAC [62] are built upon similar frameworks.

Theorem 1 *For $\overline{Q}_\theta$ in (5), let $\mathcal{T}_{CDC} : \overline{Q}_{\theta_t} \to \overline{Q}_{\theta_{t+1}}$ denote the operator corresponding to the $\overline{Q}_\theta$-updates resulting from the tth iteration of Steps 6-7 of Algorithm 1. $\mathcal{T}_{CDC}$ is a L_∞ contraction under standard conditions that suffice for the ordinary Bellman operator to be contractive [3, 7, 9, 56].*

The proof and formal list of assumptions are in Appendix D.1. Together with Banach's theorem, the contraction property established in Theorem 1 above guarantees that our CDC updates converge to a fixed point under commonly-assumed conditions that suffice for standard RL algorithms to converge [30]. Due to issues of (nonconvex) function approximation, it is difficult to guarantee this in practice or empirical optimality of the resulting estimates [37, 39]. We do note that the addition of our two novel regularizers further enhances the contractive nature and stability of the CDC updates when $\eta, \lambda > 0$ by shrinking Q-values and policy action-probabilities toward the corresponding values estimated for the behavior policy (i.e. values computed for observations in $\mathcal{D}$). Our CDC penalties can thus not only lead to less wildly-extrapolated batch estimates, but also faster (and more stable) convergence of the learning process (as shown in Figure 1, where *Standard actor-critic* refers to Algorithm 1 where $\eta = \lambda = 0$).

Theorem 2 *Let $\pi_\phi \in \Pi$ be the policy learned by CDC, γ denote discount factor, and n denote the sample size of dataset $\mathcal{D}$ generated from π_b. Also let $J(\pi)$ represent the true expected return produced by deploying policy π in the environment. Under mild assumptions listed in Appendix D, there exist constants r^*, C_λ, V such that with high probability $\geq 1 - \delta$:*

$$J(\pi_\phi) \geq J(\pi_b) - \frac{r^*}{(1-\gamma)^2}\sqrt{C_\lambda + \sqrt{(V - \log\delta)/n}}$$

Appendix D.2 contains a proof and descriptions of the assumptions in this result. Theorem 2 assures us of the reliability of the policy π_ϕ produced by CDC, guaranteeing that with high probability π_ϕ will not have much worse outcomes than the behavior policy π_b, where the probability here depends on the size of the dataset $\mathcal{D}$ and our choice of policy regularization penalty λ (the constant C_λ is a decreasing function of λ). In batch settings, expecting to learn the optimal policy is futile from limited data. Even ensuring *any* improvement at all over an arbitrary π_b is ambitious when we cannot ever test any policies in the environment, and reliability of the learned π_ϕ is thus a major concern.

Theorem 3 *Let $\mathrm{OE}_{\mathrm{ag}} = \mathbb{E}[\max_a Q_\theta(s,a)] - \max_a Q^*(s,a)$ be the overestimation error in actions favored by an agent* ag*. Here Q_θ denotes the estimate of true Q-value learned by* ag*, which may either use **CDC** (with $\eta > 0$) or a **baseline** version of Algorithm 1 with $\eta = 0$ (with the same value of λ). Under the assumptions listed in Appendix D.3, there co-exist constants L_1 and L_2 such that*

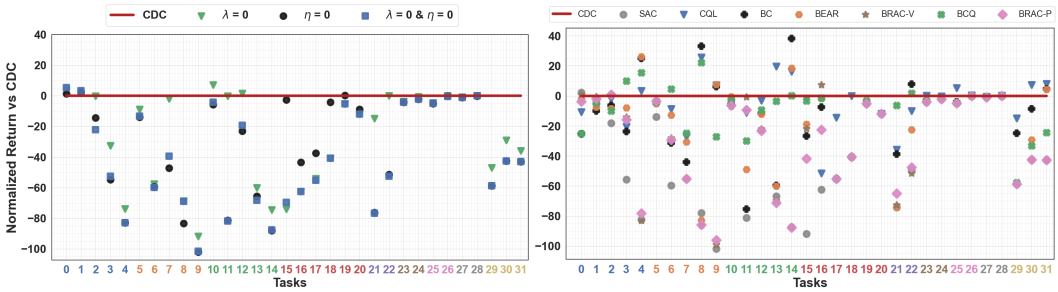
$$\mathrm{OE}_{\mathrm{CDC}} \leq L_1 - \eta L_2 \leq \mathrm{OE}_{\mathrm{baseline}}$$

This theorem (proved in Appendix D.3) underscores the influence of the η parameter in terms of containing the overestimation problem in offline Q-learning. Mitigating this overestimation, which can be done using non-zero η, can ultimately lead into better returns as we show in the experimental section. In particular, CDC achieves lower overestimation by deliberately underestimating Q-values for non-observed state-actions (but it limits the degree of downward bias as described in Remark 2). Buckman et al. [8], Jin et al. [24] prove that some degree of pessimism is unavoidable to ensure non-catastrophic deployment of batch RL in practice, where it is unlikely there will ever be sufficient data for the agent to accurately estimate the consequences of all possible actions in all states.

Remark 2 (Pessimism is limited in CDC) *Extreme pessimism leads to overly conservative policies with limited returns. The degree of pessimism in CDC remains limited (capped once $\Delta_j = 0$), unlike lower-confidence bounds which can become arbitrarily pessimistic and hence limited in their return.*

4 Related Work

Aiming for a practical framework to improve arbitrary existing policies, much research has studied batch RL [33, 35] and the issue of overestimation [21, 22, 58]. [25, 63] consider model-based approaches for batch RL, and [1] find ensembles partly address some of the issues that arise in batch



(a) (b)

Figure 2: **Difference in (normalized) return achieved by various algorithms vs CDC in 32 D4RL tasks**. X-axis colors indicate environments (see Table S1), and points below the line (▬) indicate worse performance than CDC. **Figure 2a** shows that fixing η or λ to zero (i.e. omitting our penalties) produces far worse returns than CDC (see also Table S2). This ablation study proves that major performance gains for CDC stem from our novel pair of regularizers, as the *only* difference between CDC and these ablated variants is either η or λ or both are set to zero in Algorithm 1 (all other details are exactly the same). **Figure 2b** compares CDC against existing batch RL algorithms, where CDC overall compares favorably to each other method in head-to-head comparisons (see also Table S1). Note these figures can be compared to each other as well.

settings. To remain suitably conservative, a popular class of approaches constrain the policy updates to remain in the vicinity of π_b via, e.g., distributional matching [17], support matching [27, 62], imposition of a behavior-based prior [52], or implicit constraints via selective policy-updates [46, 59]. Similar to imitation learning in online setting [23, 43, 47, 49], many of such methods need to explicitly estimate the behavior policy [17, 18, 27]. Although methods like [46, 59] do not have an explicit constraint on the policy update, they still can be categorized as a policy constrained-based approach as the policy update rule has been changed in a such a way that it selectively updates the policy utilizing information contained in the Q-values. Although these approaches show promising results, policy-constraint methods often work best for data collected from a high-quality (expert) behavior policy, and may struggle to significantly improve upon highly suboptimal π_b. Compared to the previous works, our CDC does not need to severely constrain candidate policies around π_b, which reduces achievable returns. Even with a strong policy constraint, the resulting policy is still affected by the learned Q-value, thus we still must correct Q-value issues. Instead of constraining policy updates, [28] advocate conservatively lower-bounding estimates of the value function. This allows for more flexibility to improve upon low-quality π_b. [38] considers a pessimistic and conservative approach to update Q-value by utilizing the marginalized state-action distribution of available data. Our proposed CDC algorithm is inspired by ideas from both the policy-constraint and value-constraint literature, demonstrating these address complementary issues of the batch RL problem and are both required in a performant solution.

5 Experiments

In this section, we evaluate our CDC algorithm against existing methods for batch RL on the 32 tasks in the D4RL benchmark [15]. We also investigate the utility of individual CDC penalties through ablation analyses, and demonstrate the broader applicability of our extra-overestimation penalty to off-policy evaluation in addition to batch RL. We follow exactly the same train/evaluation setups as existing works [15, 17, 27, 28]. See Appendices A, B, and C for complete experiment details.

Setup. We compare CDC against existing batch RL methods: BEAR [27], BRAC-V/P [62], BC [62], CQL [28], BCQ [17], and SAC [20]. This selection of methods covers a rich set of strong batch RL methods ranging from behavioral cloning to value-constrained-based pessimistic methods, with exception of SAC. SAC is an off-policy method that empirically performs quite well in online RL. SAC is included to study how online RL methods fare when straightforwardly applied in the batch setting (unsurprisingly, quite poorly). Note that CDC was simply run on every task using the same network and the original rewards/actions provided in the task, without any manual task-specific reward-normalization/action-smoothing. Moreover, all these baseline methods also utilize an ensemble of Q networks as in (5).

Results. Figure 2b and Table S1 illustrate that CDC performs better than the majority of the other considered batch RL methods on the D4RL tasks. CDC is total normalized return across all 32 tasks is 1397, whereas the next-best method (CQL) achieves 1245. In head-to-head comparisons against each other batch RL method, CDC generates statistically significantly greater overall returns (Table S1). Unsurprisingly, behavioral-cloning (BC) works well on tasks with data generated by an expert π_b, while the online RL method, SAC, fares poorly in many tasks. CDC remains reasonably competitive across all tasks, regardless of the environment or the quality of π_b (i.e. random vs. expert).

Next we perform a comprehensive set of ablation studies to gauge the contribution of our proposed penalties in CDC. Here we run additional variants of Algorithm 1 without our penalties (i.e. $\eta = \lambda = 0$) , with only our extra-overestimation penalty ($\lambda = 0$), and with only our exploration penalty ($\eta = 0$). Figure 2a and Tables S2 show that both penalties are critical for the strong performance of CDC, with the extra-overestimation penalty Δ being of greater importance than exploration (see also Figure 2a). Note that *all* our ablation variants still employ the lower confidence bound from (5), which alone clearly does not suffice to correct extra-overestimation.

5.1 Offline Policy Evaluation

The true practical applicability of batch RL remains however hampered without the ability to do proper algorithm/hyperparameter selection. Table S1 shows that no algorithm universally dominates all others across all environments or behavior-policies. In practice, it is difficult to know which technique will perform best, unless one can do proper offline policy evaluation (OPE) of different candidate policies before their actual online deployment [45].

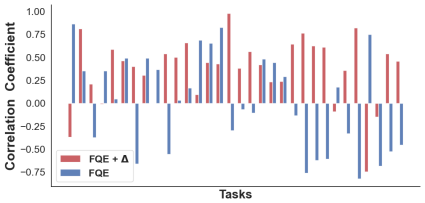


Figure 3: **How well OPE estimates correlate with actual return** achieved by 20 different policies for each D4RL task. Due to unmitigated overestimation, FQE estimates correlate *negatively* with true returns in 15 of 32 tasks (using Δ in FQE reduces this to 4).

OPE aims to estimate the performance of a given policy under the same setting considered here, with offline data collected by an unknown behavior policy [34, 45]. Beyond algorithm/hyperparameter comparison, OPE is often employed for critical policy-making decisions where environment interaction is no longer an option, e.g., sensitive healthcare applications [19]. One practical OPE method for data of the form in $\mathcal{D}$ is Fitted Q Evaluation (FQE) [34]. To score a given policy π, FQE iterates temporal difference updates of the form (3) using the standard Bellman operator from (1) in place of EMaQ. After learning an estimate $\hat{Q}^\pi$, FQE simply estimates the return of π via the expectation of $\hat{Q}^\pi(s, a)$ over the initial state distribution and actions sampled from π.

However, like batch RL, OPE also relies on limited data and thus can still suffer from severe Q-value estimation errors. To curb the corresponding overestimation bias, we can regularize the FQE temporal difference updates with our Δ penalty, in a similar manner to our previous (7). Figure 3 compares the performance of Δ-penalization of FQE (with $\eta = 1$ throughout) against the standard unregularized FQE. Here we use both OPE methods to score 20 different policies (learned via different settings) and gauge OPE-quality via the Pearson correlation between OPE estimated returns and the actual return (over our 20 policies). The higher correlation for FQE + Δ (0.37 on average) over FQE (0.01 on average) in the majority of tasks demonstrates how the inclusion of our Δ penalty can lead to more reliable OPE estimates. Our strategies for mitigating overestimation are thus not only useful for batch RL but also related tasks like off-policy evaluation.

6 Discussion

Here we propose a simple and effective algorithm for batch RL by introducing a simple pair of regularizers that abate the challenge of learning how to act from limited data. The first constrains the value update to mitigate extra-overestimation error, while the latter constrains the policy update to ensure candidate policies do not stray too far from the offline data. Comprehensive experiments on standard offline continuous-control benchmarks suggest that CDC compares favorably with state-of-the-art methods for batch RL, and our proposed penalties are also useful to improve offline policy evaluation. The broader impact of this work will hopefully be to improve batch RL performance in offline applications, but we caution that unobserved confounding remains another key challenge in real-world data that was not addressed in this work.

References

[1] R. Agarwal, D. Schuurmans, and M. Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, 2020.

[2] Y. Altun and A. Smola. Unifying divergence minimization and statistical inference via convex duality. In *International Conference on Computational Learning Theory*, pages 139–153. Springer, 2006.

[3] A. Antos, R. Munos, and C. Szepesvari. Fitted q-iteration in continuous action-space mdps. In *Advances in Neural Information Processing Systems*, 2007.

[4] P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[5] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, volume 29, pages 1471–1479, 2016.

[6] R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[7] D. P. Bertsekas and S. Shreve. *Stochastic optimal control: the discrete-time case*. Athena Scientific, 2004.

[8] J. Buckman, C. Gelada, and M. G. Bellemare. The importance of pessimism in fixed-dataset policy optimization. In *International Conference on Learning Representations*, 2021.

[9] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC press, 2010.

[10] R. Y. Chen, S. Sidor, P. Abbeel, and J. Schulman. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 2017.

[11] I. Csiszár and P. Shields. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1(4):417–528, 2004. ISSN 1567-2190. doi: 10.1561/0100000004.

[12] G. Dulac-Arnold, D. Mankowitz, and T. Hester. Challenges of real-world reinforcement learning. In *ICML Reinforcement Learning for Real Life (RL4RealLife) Workshop*, 2019.

[13] R. Fakoor, P. Chaudhari, and A. J. Smola. P3O: policy-on policy-off policy optimization. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019*, page 371, 2019.

[14] R. Fakoor, P. Chaudhari, and A. J. Smola. Ddpg++: Striving for simplicity in continuous-control off-policy reinforcement learning. *arXiv:2006.15199*, 2020.

[15] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv:2004.07219*, 2020.

[16] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 2018.

[17] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062, 2019.

[18] S. K. S. Ghasemipour, D. Schuurmans, and S. S. Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. *arXiv:2007.11091*, 2021.

[19] O. Gottesman, J. Futoma, Y. Liu, S. Parbhoo, L. Celi, E. Brunskill, and F. Doshi-Velez. Interpretable off-policy evaluation in reinforcement learning by highlighting influential transitions. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3658–3667. PMLR, 2020.

[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv:1801.01290*, 2018.

[21] H. V. Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems 23*, pages 2613–2621, 2010.

[22] H. v. Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2094–2100. AAAI Press, 2016.

[23] T. Hester, M. Vecerík, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. P. Agapiou, J. Z. Leibo, and A. Gruslys. Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732, 2017.

[24] Y. Jin, Z. Yang, and Z. Wang. Is pessimism provably efficient for offline RL? *arXiv preprint arXiv:2012.15085*, 2020.

[25] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. Morel : Model-based offline reinforcement learning. *arXiv:2005.05951*, 2020.

[26] V. Konda and J. Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 1008–1014. MIT Press, 2000.

[27] A. Kumar, J. Fu, G. Tucker, and S. Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. *arXiv:1906.00949*, Nov. 2019.

[28] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative Q-Learning for Offline Reinforcement Learning. *arXiv:2006.04779*, June 2020.

[29] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5556–5566. PMLR, 13–18 Jul 2020.

[30] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

[31] Q. Lan, Y. Pan, A. Fyshe, and M. White. Maxmin q-learning: Controlling the estimation bias of q-learning. In *International Conference on Learning Representations*, 2019.

[32] Q. Lan, Y. Pan, A. Fyshe, and M. White. Maxmin q-learning: Controlling the estimation bias of q-learning. In *International Conference on Learning Representations*, 2020.

[33] S. Lange, T. Gabel, and M. Riedmiller. Batch reinforcement learning. In M. Wiering and M. van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, pages 45–73. Springer, 2012.

[34] H. Le, C. Voloshin, and Y. Yue. Batch policy learning under constraints. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3703–3712. PMLR, 2019.

[35] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv:2005.01643*, May 2020.

[36] Y. Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[37] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[38] Y. Liu, A. Swaminathan, A. Agarwal, and E. Brunskill. Provably good batch off-policy reinforcement learning without great exploration. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1264–1274. Curran Associates, Inc., 2020.

[39] G. Matheron, N. Perrin, and O. Sigaud. The problem with ddpg: understanding failures in deterministic environments with sparse rewards. *arXiv preprint arXiv:1911.11679*, 2019.

[40] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[41] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[42] A. Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997. ISSN 00018678.

[43] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7(1-2):1–179, 2018.

[44] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2721–2730. PMLR, 06–11 Aug 2017.

[45] T. L. Paine, C. Paduraru, A. Michi, C. Gulcehre, K. Zolna, A. Novikov, Z. Wang, and N. de Freitas. Hyperparameter selection for offline reinforcement learning. *arXiv preprint arXiv:2007.09055*, 2020.

[46] X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *CoRR*, abs/1910.00177, 2019.

[47] D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

[48] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. USA, 1st edition, 1994. ISBN 0471619779.

[49] S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In G. J. Gordon, D. B. Dunson, and M. Dudík, editors, *AISTATS*, volume 15 of *JMLR Proceedings*, pages 627–635. JMLR.org, 2011.

[50] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[51] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[52] N. Siegel, J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess, and M. Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020.

[53] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 387–395. PMLR, 2014.

[54] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[55] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419): 1140–1144, 2018. ISSN 0036-8075. doi: 10.1126/science.aar6404.

[56] C. Szepesvári. Efficient approximate planning in continuous space markovian decision problems. *AI Communications*, 14(3):163–176, 2001.

[57] P. Thomas, G. Theocharous, and M. Ghavamzadeh. High confidence policy improvement. In *International Conference on Machine Learning*, pages 2380–2388. PMLR, 2015.

[58] S. Thrun and A. Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School*, pages 255–263. Lawrence Erlbaum, 1993.

[59] Z. Wang, A. Novikov, K. Zolna, J. T. Springenberg, S. Reed, B. Shahriari, N. Siegel, J. Merel, C. Gulcehre, N. Heess, and N. de Freitas. Critic regularized regression. *arXiv:2006.15134*, 2020.

[60] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[61] R. J. Williams and J. Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

[62] Y. Wu, G. Tucker, and O. Nachum. Behavior Regularized Offline Reinforcement Learning. *arXiv:1911.11361*, 2019.

[63] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. *arXiv:2005.13239*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Briefly in the Discussion
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix D
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix D
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide full detail of experiments and instructions needed to reproduce, see Appendix A and B, and Algorithm 1 but code is not included.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B, A, and C
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] For relevant experiments, See Figure S1
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Table S4
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use D4RL [15] benchmark which is described in Appendix C
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]