Conservative Q-Learning for Offline Reinforcement Learning

Aviral Kumar[1], **Aurick Zhou**[1], **George Tucker**[2], **Sergey Levine**[1,2]
[1]UC Berkeley, [2]Google Research, Brain Team
`aviralk@berkeley.edu`

Abstract

Effectively leveraging large, previously collected datasets in reinforcement learning (RL) is a key challenge for large-scale real-world applications. Offline RL algorithms promise to learn effective policies from previously-collected, static datasets without further interaction. However, in practice, offline RL presents a major challenge, and standard off-policy RL methods can fail due to overestimation of values induced by the distributional shift between the dataset and the learned policy, especially when training on complex and multi-modal data distributions. In this paper, we propose *conservative Q-learning (CQL)*, which aims to address these limitations by learning a conservative Q-function such that the expected value of a policy under this Q-function lower-bounds its true value. We theoretically show that CQL produces a lower bound on the value of the current policy and that it can be incorporated into a principled policy improvement procedure. In practice, CQL augments the standard Bellman error objective with a simple Q-value regularizer which is straightforward to implement on top of existing deep Q-learning and actor-critic implementations. On both discrete and continuous control domains, we show that CQL substantially outperforms existing offline RL methods, often learning policies that attain 2-5 times higher final return, especially when learning from complex and multi-modal data distributions.

1 Introduction

Recent advances in reinforcement learning (RL), especially when combined with expressive deep network function approximators, have produced promising results in domains ranging from robotics [26] to strategy games [3] and recommendation systems [31]. However, applying RL to real-world problems consistently poses practical challenges: in contrast to the kinds of data-driven methods that have been successful in supervised learning [21, 9], RL is classically regarded as an active learning process, where each training run requires active interaction with the environment. Interaction with the real world can be costly and dangerous, and the quantities of data that can be gathered online are substantially lower than the offline datasets that are used in supervised learning [8], which only need to be collected once. Offline RL, also known as batch RL, offers an appealing alternative [10, 14, 27, 2, 25, 45, 30]. Offline RL algorithms learn from large, previously collected datasets, without interaction. This in principle can make it possible to leverage large datasets, but in practice fully offline RL methods pose major technical difficulties, stemming from the distributional shift between the policy that collected the data and the learned policy. This has made current results fall short of the full promise of such methods.

Directly utilizing existing value-based off-policy RL algorithms in an offline setting generally results in poor performance, due to issues with bootstrapping from out-of-distribution actions [27, 14] and overfitting [12, 27, 2]. This typically manifests as erroneously optimistic value function estimates. If we can instead learn a *conservative* estimate of the value function, which provides a lower bound on the true values, this overestimation problem could be addressed. In fact, because policy evaluation

Preprint. Under review.

and improvement typically only use the value of the policy, we can learn a less conservative lower bound Q-function, such that only the expected value of Q-function under the policy is lower-bounded, as opposed to a point-wise lower bound. We propose a novel method for learning such conservative Q-functions via a simple modification to standard value-based RL algorithms. The key idea behind our method is to minimize values under an appropriately chosen distribution over state-action tuples, and then further tighten this bound by also incorporating a *maximization* term over the data distribution.

Our primary contribution is an algorithmic framework, which we call conservative Q-learning (CQL), for learning conservative, lower-bound estimates of the value function, by regularizing the Q-values during training. Our theoretical analysis of CQL shows that *only* the expected value of this Q-function under the policy lower-bounds the true policy value, preventing extra under-estimation that can arise with point-wise lower-bounded Q-functions, that have typically been explored in the opposite context in exploration literature [38, 24]. We also empirically demonstrate the robustness of our approach to Q-function estimation error. Our practical algorithm uses these conservative estimates for policy evaluation and offline RL. CQL can be implemented with less than **20** lines of code on top of a number of standard, online RL algorithms [18, 7], simply by adding the CQL regularization terms to the Q-function update. In our experiments, we demonstrate the efficacy of CQL for offline RL, in domains with complex dataset compositions, where prior methods are typically known to perform poorly [11] and domains with high-dimensional visual inputs [4, 2]. CQL outperforms prior methods by as much as **2-5x** on many benchmark tasks, and is the only method that can outperform simple behavioral cloning on a number of realistic datasets collected from human interaction.

2 Preliminaries

The goal in reinforcement learning is to learn a policy that maximizes the expected cumulative discounted reward in a Markov decision process (MDP), which is defined by a tuple $(\mathcal{S}, \mathcal{A}, T, r, \gamma)$. $\mathcal{S}, \mathcal{A}$ represent state and action spaces, $T(s'|s,a)$ and $r(s,a)$ represent the dynamics and reward function, and $\gamma \in (0,1)$ represents the discount factor. $\pi_\beta(\mathbf{a}|\mathbf{s})$ represents the action-conditional in the dataset, $\mathcal{D}$, and $d^{\pi_\beta}(\mathbf{s})$ is the discounted marginal state-distribution of $\pi_\beta(\mathbf{a}|\mathbf{s})$. The dataset $\mathcal{D}$ is sampled from $d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})$. On all states $\mathbf{s}_0 \in \mathcal{D}$, let $\hat{\pi}_\beta(\mathbf{a}_0|\mathbf{s}_0) := \frac{\sum_{\mathbf{s},\mathbf{a}\in\mathcal{D}} \mathbf{1}[\mathbf{s}=\mathbf{s}_0, \mathbf{a}=\mathbf{s}_0]}{\sum_{\mathbf{s}\in\mathcal{D}} \mathbf{1}[\mathbf{s}=\mathbf{s}_0]}$ denote the empirical behavior policy, at that state.

Off-policy RL algorithms based on dynamic programming maintain a parametric Q-function $Q_\theta(s,a)$ and, optionally, a parametric policy, $\pi_\phi(a|s)$. Q-learning methods train the Q-function by iteratively applying the Bellman optimality operator $\mathcal{B}^*Q(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}'\sim P(\mathbf{s}'|\mathbf{s},\mathbf{a})}[\max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}')]$, and use exact or an approximate maximization scheme, such as CEM [26] to recover the greedy policy. In an actor-critic algorithm, a separate policy is trained to maximize the Q-value. Actor-critic methods alternate between computing Q^π via (partial) policy evaluation, by iterating the Bellman operator, $\mathcal{B}^\pi Q = r + \gamma P^\pi Q$, where P^π is the transition matrix coupled with the policy: $P^\pi Q(\mathbf{s},\mathbf{a}) = \mathbb{E}_{\mathbf{s}'\sim T(\mathbf{s}'|\mathbf{s},\mathbf{a}), \mathbf{a}'\sim\pi(\mathbf{a}'|\mathbf{s}')}\left[Q(\mathbf{s}',\mathbf{a}')\right]$, and improving the policy $\pi(\mathbf{a}|\mathbf{s})$ by updating it towards actions that maximize the expected Q-value. Since $\mathcal{D}$ typically does not contain all possible transitions $(\mathbf{s},\mathbf{a},\mathbf{s}')$, the policy evaluation step actually uses an empirical Bellman operator that only backs up a single sample. We denote this operator $\hat{\mathcal{B}}^\pi$. Given a dataset $\mathcal{D} = \{(\mathbf{s},\mathbf{a},\mathbf{s}')\}$ of tuples from trajectories collected under a behavior policy π_β:

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q} \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left((r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}'\sim\hat{\pi}^k(\mathbf{a}'|\mathbf{s}')}[\hat{Q}^k(\mathbf{s}',\mathbf{a}')]) - Q(\mathbf{s},\mathbf{a})\right)^2\right] \text{ (policy evaluation)}$$

$$\hat{\pi}^{k+1} \leftarrow \arg\min_{\pi} \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi^k(\mathbf{a}|\mathbf{s})}\left[\hat{Q}^{k+1}(\mathbf{s},\mathbf{a})\right] \text{ (policy improvement)}$$

Offline RL algorithms based on this basic recipe suffer from action distribution shift [27, 47, 25, 30] during training, because the target values for Bellman backups in policy evaluation use actions sampled from the learned policy, π^k, but the Q-function is trained only on actions sampled from the behavior policy that produced the dataset $\mathcal{D}$, π_β. We also remark that Q-function training in offline RL does not suffer from state distribution shift, because the Q-function is never queried on out-of-distribution states. Since π is trained to maximize Q-values, it may be biased towards out-of-distribution (OOD) actions with erroneously high Q-values. In standard RL, such errors can be corrected by attempting an action in the environment and observing its actual value. However, the inability to interact with the environment makes it challenging to deal with Q-values for OOD actions in offline RL. Typical offline RL methods [27, 25, 47, 45] mitigate this problem by constraining the learned policy [30] away from OOD actions.

3 The Conservative Q-Learning (CQL) Framework

In this section, we develop a conservative Q-learning (CQL) algorithm, such that the expected value of a policy under the learned Q-function lower-bounds its true value. A lower bound on the Q-value prevents the over-estimation that is common in offline RL settings due to OOD actions and function approximation error [30, 27]. We use the term CQL to refer broadly to both Q-learning methods and actor-critic methods, though the latter also use an explicit policy. We start by focusing on the policy evaluation step in CQL, which can be used by itself as an off-policy evaluation procedure, or integrated into a complete offline RL algorithm, as we will discuss in Section 3.2.

3.1 Conservative Off-Policy Evaluation

We aim to estimate the value $V^\pi(\mathbf{s})$ of a target policy π given access to a dataset, $\mathcal{D}$, generated by following a behavior policy $\pi_\beta(\mathbf{a}|\mathbf{s})$. Because we are interested in preventing overestimation of the policy value, we learn a *conservative*, lower-bound Q-function by additionally minimizing Q-values alongside a standard Bellman error objective. Our choice of penalty is to minimize the expected Q-value under a particular distribution of state-action pairs, $\mu(\mathbf{s}, \mathbf{a})$. Since standard Q-function training does not query the Q-function value at unobserved states, but queries the Q-function at unseen actions, we restrict μ to match the state-marginal in the dataset, such that $\mu(\mathbf{s}, \mathbf{a}) = d^{\pi_\beta}(\mathbf{s})\mu(\mathbf{a}|\mathbf{s})$. This gives rise to the iterative update for training the Q-function, as a function of a tradeoff factor α:

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q} \alpha\, \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right] + \frac{1}{2}\,\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^\pi \hat{Q}^k(\mathbf{s},\mathbf{a})\right)^2\right], \quad (1)$$

In Theorem 3.1, we show that the resulting Q-function, $\hat{Q}^\pi := \lim_{k\to\infty}\hat{Q}^k$, lower-bounds Q^π at all $(\mathbf{s}, \mathbf{a})$. However, we can substantially tighten this bound if we are *only* interested in estimating $V^\pi(\mathbf{s})$. If we only require that the expected value of the $\hat{Q}^\pi$ under $\pi(\mathbf{a}|\mathbf{s})$ lower-bound V^π, we can improve the bound by introducing an additional Q-value *maximization* term under the data distribution, $\pi_\beta(\mathbf{a}|\mathbf{s})$, resulting in the iterative update (changes from Equation 1 in red):

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q}\; \alpha\cdot\left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right] - \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]\right)$$
$$+\frac{1}{2}\,\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^\pi \hat{Q}^k(\mathbf{s},\mathbf{a})\right)^2\right]. \quad (2)$$

In Theorem 3.2, we show that, while the resulting Q-value $\hat{Q}^\pi$ may not be a point-wise lower-bound, we have $\mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})}[\hat{Q}^\pi(\mathbf{s},\mathbf{a})] \leq V^\pi(\mathbf{s})$ when $\mu(\mathbf{a}|\mathbf{s}) = \pi(\mathbf{a}|\mathbf{s})$. Intuitively, since Equation 2 maximizes Q-values under the behavior policy $\hat{\pi}_\beta$, Q-values for actions that are likely under $\hat{\pi}_\beta$ might be overestimated, and hence $\hat{Q}^\pi$ may not lower-bound Q^π pointwise. While in principle the maximization term can utilize other distributions besides $\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})$, we prove in Appendix D.2 that the resulting value is not guaranteed to be a lower bound for other distribution besides $\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})$.

Theoretical analysis. We first note that Equations 1 and 2 use the empirical Bellman operator, $\hat{\mathcal{B}}^\pi$, instead of the actual Bellman operator, $\mathcal{B}^\pi$. Following [39, 24, 37], we use concentration properties of $\hat{\mathcal{B}}^\pi$ to control this error. Formally, for all $\mathbf{s}, \mathbf{a} \in \mathcal{D}$, with probability $\geq 1-\delta$, $|\hat{\mathcal{B}}^\pi - \mathcal{B}^\pi|(\mathbf{s},\mathbf{a}) \leq \frac{C_{r,T,\delta}}{\sqrt{|\mathcal{D}(\mathbf{s},\mathbf{a})|}}$, where $C_{r,T,\delta}$ is a constant dependent on the concentration properties (variance) of $r(\mathbf{s},\mathbf{a})$ and $T(\mathbf{s}'|\mathbf{s},\mathbf{a})$, and $\delta \in (0,1)$ (see Appendix D.3 for more details). Now, we show that the conservative Q-function learned by iterating Equation 1 lower-bounds the true Q-function. Proofs can be found in Appendix C.

Theorem 3.1. *For any* $\mu(\mathbf{a}|\mathbf{s})$ *with* $\text{supp}\,\mu \subset \text{supp}\,\hat{\pi}_\beta$*, with probability* $\geq 1-\delta$*,* $\hat{Q}^\pi$ *(the Q-function obtained by iterating Equation 1) satisifies:*

$$\forall \mathbf{s},\mathbf{a},\;\; \hat{Q}^\pi(s,a) \leq Q^\pi(\mathbf{s},\mathbf{a}) - \alpha\left(I-\gamma P^\pi\right)^{-1}\left[\frac{\mu(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}\right](\mathbf{s},\mathbf{a}) + \left(I-\gamma P^\pi\right)^{-1}\frac{C_{r,T,\delta}R_{\max}}{(1-\gamma)}.$$

Thus, if $\alpha > \frac{C_{r,T}R_{\max}}{1-\gamma}\cdot\max_{\mathbf{s},\mathbf{a}\in\mathcal{D}}\frac{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}{\mu(\mathbf{a}|\mathbf{s})}$*, then* $\hat{Q}^\pi(\mathbf{s},\mathbf{a}) \leq Q^\pi(\mathbf{s},\mathbf{a}), \forall \mathbf{s},\mathbf{a}$*. When* $\hat{\mathcal{B}}^\pi = \mathcal{B}^\pi$*, any* $\alpha > 0$ *guarantees* $\hat{Q}^\pi(\mathbf{s},\mathbf{a}) \leq Q^\pi(\mathbf{s},\mathbf{a}), \forall \mathbf{s},\mathbf{a}$*.*

Next, we show that Equation 2 lower-bounds the expected value under the policy π, when $\mu = \pi$. We also show that Equation 2 does not lower-bound the Q-value estimates pointwise.

Theorem 3.2 (Equation 2 results in a tighter lower bound)**.** *The value of the policy under the Q-function from Equation 2, $\hat{V}^\pi(\mathbf{s}) = \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})}[\hat{Q}^\pi(\mathbf{s},\mathbf{a})]$, lower-bounds the true value of the policy obtained via exact policy evaluation, $V^\pi(\mathbf{s}) = \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})}[Q^\pi(\mathbf{s},\mathbf{a})]$, when $\mu = \pi$, according to:*

$$\forall \mathbf{s},\ \ \hat{V}^\pi(\mathbf{s}) \leq V^\pi(\mathbf{s}) - \alpha \left(I - \gamma P^\pi\right)^{-1} \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})}\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})} - 1\right](\mathbf{s}) + (I - \gamma P^\pi)^{-1}\frac{C_{r,T,\delta} R_{\max}}{(1-\gamma)}.$$

Thus, if $\alpha > \frac{C_{r,T} R_{\max}}{1-\gamma} \cdot \max_{\mathbf{s}\in\mathcal{D}} \left[\sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s})(\frac{\pi(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s}))} - 1)\right]^{-1},\ \forall \mathbf{s} \in \mathcal{D},\ \hat{V}^\pi(\mathbf{s}) \leq V^\pi(\mathbf{s})$, with probability $\geq 1-\delta$. When $\hat{\mathcal{B}}^\pi = \mathcal{B}^\pi$, then any $\alpha > 0$ guarantees $\hat{V}^\pi(\mathbf{s}) \leq V^\pi(\mathbf{s}), \forall \mathbf{s} \in \mathcal{D}$.

The analysis presented above assumes that no function approximation is used in the Q-function, meaning that each iterate can be represented exactly. We can further generalize the result in Theorem 3.2 to the case of both linear function approximators and non-linear neural network function approximators, where the latter builds on the neural tangent kernel (NTK) framework [23]. Due to space constraints, we present these results in Theorem D.1 and Theorem D.2 in Appendix D.1.

In summary, we showed that the basic CQL evaluation in Equation 1 learns a Q-function that lower-bounds the true Q-function Q^π, and the evaluation in Equation 2 provides a *tighter* lower bound on the expected Q-value of the policy π. For suitable α, both bounds hold under sampling error and function approximation. Next, we will extend on this result into a complete RL algorithm.

3.2 Conservative Q-Learning for Offline RL

We now present a general approach for offline policy learning, which we refer to as conservative Q-learning (CQL). As discussed in Section 3.1, we can obtain Q-values that lower-bound the value of a policy π by solving Equation 2 with $\mu = \pi$. How should we utilize this for policy optimization? We could alternate between performing full off-policy evaluation for each policy iterate, $\hat{\pi}^k$, and one step of policy improvement. However, this can be computationally expensive. Alternatively, since the policy $\hat{\pi}^k$ is typically derived from the Q-function, we could instead choose $\mu(\mathbf{a}|\mathbf{s})$ to approximate the policy that would maximize the current Q-function iterate, thus giving rise to an online algorithm.

We can formally capture such online algorithms by defining a family of optimization problems over $\mu(\mathbf{a}|\mathbf{s})$, presented below, with modifications from Equation 2 marked in red. An instance of this family is denoted by CQL($\mathcal{R}$) and is characterized by a particular choice of regularizer $\mathcal{R}(\mu)$:

$$\min_{Q} \max_{\mu}\ \alpha \left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right] - \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]\right)$$
$$+\frac{1}{2}\,\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^{\pi_k}\hat{Q}^k(\mathbf{s},\mathbf{a})\right)^2\right] + \mathcal{R}(\mu) \quad (\text{CQL}(\mathcal{R}))\,. \quad (3)$$

Variants of CQL. To demonstrate the generality of the CQL family of optimization problems, we discuss two specific instances within this family that are of special interest, and we evaluate them empirically in Section 6. If we choose $\mathcal{R}(\mu)$ to be the KL-divergence against a prior distribution, $\rho(\mathbf{a}|\mathbf{s})$, i.e., $\mathcal{R}(\mu) = -D_{\mathrm{KL}}(\mu,\rho)$, then we get $\mu(\mathbf{a}|\mathbf{s}) \propto \rho(\mathbf{a}|\mathbf{s}) \cdot \exp(Q(\mathbf{s},\mathbf{a}))$ (for a derivation, see Appendix A). Frist, if $\rho = \text{Unif}(\mathbf{a})$, then the first term in Equation 3 corresponds to a soft-maximum of the Q-values at any state s and gives rise to the following variant of Equation 3, called CQL($\mathcal{H}$):

$$\min_{Q}\ \alpha\mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\left[\log\sum_{\mathbf{a}}\exp(Q(\mathbf{s},\mathbf{a})) - \mathbb{E}_{\mathbf{a}\sim\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]\right] + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q - \hat{\mathcal{B}}^{\pi_k}\hat{Q}^k\right)^2\right]. \quad (4)$$

Second, if $\rho(\mathbf{a}|\mathbf{s})$ is chosen to be the previous policy $\hat{\pi}^{k-1}$, the first term in Equation 4 is replaced by an exponential weighted average of Q-values of actions from the chosen $\hat{\pi}^{k-1}(\mathbf{a}|\mathbf{s})$. Empirically, we find that this variant can be more stable with high-dimensional action spaces (e.g., Table 2) where it is challenging to estimate $\log\sum_{\mathbf{a}}\exp$ via sampling due to high variance. In Appendix A, we discuss an additional variant of CQL, drawing connections to distributionally robust optimization [36]. We will discuss a practical instantiation of a CQL deep RL algorithm in Section 4. CQL can be instantiated as either a Q-learning algorithm (with $\mathcal{B}^*$ instead of $\mathcal{B}^\pi$ in Equations 3, 4) or as an actor-critic algorithm.

Theoretical analysis of CQL. Next, we will theoretically analyze CQL to show that the policy updates derived in this way are indeed "conservative", in the sense that each successive policy iterate is optimized against a lower bound on its value. For clarity, we state the results in the absence of finite-sample error, in this section, but sampling error can be incorporated in the same way as Theorems 3.1

and 3.2, and we discuss this in Appendix C. Theorem 3.3 shows that any variant of the CQL family learns Q-value estimates that lower-bound the actual Q-function under the action-distribution defined by the policy, π^k, under mild regularity conditions (slow updates on the policy).

Theorem 3.3 (CQL learns lower-bounded Q-values)**.** *Let $\pi_{\hat{Q}^k}(\mathbf{a}|\mathbf{s}) \propto \exp(\hat{Q}^k(\mathbf{s}, \mathbf{a}))$ and assume that $D_{\mathrm{TV}}(\hat{\pi}^{k+1}, \pi_{\hat{Q}^k}) \leq \varepsilon$ (i.e., $\hat{\pi}^{k+1}$ changes slowly w.r.t to $\hat{Q}^k$). Then, the policy value under $\hat{Q}^k$, lower-bounds the expected actual policy value, that is, $\hat{V}^{k+1}(\mathbf{s}) = \mathbb{E}_{\pi^{k+1}(\mathbf{a}|\mathbf{s})}[\hat{Q}^{k+1}(\mathbf{s}, \mathbf{a})] \leq \mathbb{E}_{\pi^{k+1}(\mathbf{a}|\mathbf{s})}[Q^{k+1}(\mathbf{s}, \mathbf{a})] = V^{k+1}(\mathbf{s})$ if*

$$\mathbb{E}_{\pi_{\hat{Q}^k}(\mathbf{a}|\mathbf{s})}\left[\frac{\pi_{\hat{Q}^k}(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})} - 1\right] \geq \max_{\mathbf{a} \text{ s.t. } \hat{\pi}_\beta(\mathbf{a}|\mathbf{s})>0} \left(\frac{\pi_{\hat{Q}^k}(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}\right) \cdot \varepsilon.$$

The LHS of this inequality is equal to amount of conservatism induced in the value, $\hat{V}^{k+1}$ in iteration $k+1$ of the CQL update, if the learned policy were equal to soft-optimal policy for $\hat{Q}^k$, i.e. $\hat{\pi}^{k+1} = \pi_{\hat{Q}^k}$. However, as the actual policy, $\hat{\pi}^{k+1}$, may be different, RHS is the maximal amount of potential overestimation due to this difference. To get a lower bound, we require the amount of underestimation to be higher, which is obtained if ε is small, i.e. the policy changes slowly.

Our final result shows that CQL Q-function update is "gap-expanding", by which we mean that the difference in Q-values at in-distribution actions and over-optimistically erroneous out-of-distribution actions is higher than the corresponding difference under the actual Q-function. This implies that the policy $\pi^k(\mathbf{a}|\mathbf{s}) \propto \exp(\hat{Q}^k(\mathbf{s}, \mathbf{a}))$, is constrained to be closer to the dataset distribution, $\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})$, thus the CQL update implicitly prevents the detrimental effects of OOD action and distribution shift, which has been a major concern in offline RL settings [27, 30, 14].

Theorem 3.4 (CQL is gap-expanding)**.** *At any iteration k, CQL expands the difference in expected Q-values under the behavior policy $\pi_\beta(\mathbf{a}|\mathbf{s})$ and μ_k, such that for large enough values of α_k, we have that $\forall \mathbf{s}$, $\mathbb{E}_{\pi_\beta(\mathbf{a}|\mathbf{s})}[\hat{Q}^k(\mathbf{s}, \mathbf{a})] - \mathbb{E}_{\mu_k(\mathbf{a}|\mathbf{s})}[\hat{Q}^k(\mathbf{s}, \mathbf{a})] > \mathbb{E}_{\pi_\beta(\mathbf{a}|\mathbf{s})}[Q^k(\mathbf{s}, \mathbf{a})] - \mathbb{E}_{\mu_k(\mathbf{a}|\mathbf{s})}[Q^k(\mathbf{s}, \mathbf{a})]$.*

When function approximation or sampling error makes OOD actions have higher learned Q-values, CQL backups are expected to be more robust, in that the policy is updated using Q-values that prefer in-distribution actions. As we will empirically show in Appendix B, prior offline RL methods that do not explicitly constrain or regularize the Q-function may not enjoy such robustness properties.

To summarize, we showed that the CQL RL algorithm learns lower-bound Q-values with large enough α, meaning that the final policy attains *at least* the estimated value. We also showed that the Q-function is *gap-expanding*, meaning that it should only ever *over-estimate* the gap between in-distribution and out-of-distribution actions, preventing OOD actions.

4 Practical Algorithm and Implementation Details

We now describe two practical offline deep reinforcement learning methods based on CQL: an actor-critic variant and a Q-learning variant. Pseudocode is shown in Algorithm 1, with differences from conventional actor-critic algorithms (e.g., SAC [18]) and deep Q-learning algorithms (e.g., DQN [34]) in red. Our algorithm uses the CQL($\mathcal{H}$) (or CQL($\mathcal{R}$) in general) objective from the CQL framework for training the Q-function Q_θ, which is parameterized by a neural network with parameters θ. For the actor-critic algorithm, a policy π_ϕ is trained as well. Our algorithm modifies the objective for the Q-function (swaps out Bellman error with CQL($\mathcal{H}$)) or CQL(ρ) in a standard actor-critic or Q-learning setting, as shown in Line 3. As discussed in Section 3.2, due to the explicit penalty on the Q-function, CQL methods do not use a policy constraint, unlike prior offline RL methods [27, 47, 45, 30]. Hence, we do not require fitting an additional behavior policy estimator, simplifying our method.

Algorithm 1 Conservative Q-Learning (both variants)

```
1: Initialize Q-function, $Q_\theta$, and optionally a policy, $\pi_\phi$.
2: for step $t$ in {1, ..., N} do
3:     Train the Q-function using $G_Q$ gradient steps on objective
       from Equation 4
       $\theta_t := \theta_{t-1} - \eta_Q \nabla_\theta \text{CQL}(\mathcal{R})(\theta)$
       (Use $\mathcal{B}^*$ for Q-learning, $\mathcal{B}^{\pi_{\phi_t}}$ for actor-critic)
4:     (only with actor-critic) Improve policy $\pi_\phi$ via $G_\pi$ gradient
       steps on $\phi$ with SAC-style entropy regularization:
       $\phi_t := \phi_{t-1} + \eta_\pi \mathbb{E}_{\mathbf{s}\sim\mathcal{D}, \mathbf{a}\sim\pi_\phi(\cdot|\mathbf{s})}[Q_\theta(\mathbf{s}, \mathbf{a}) - \log \pi_\phi(\mathbf{a}|\mathbf{s})]$
5: end for
```

Implementation details. Our algorithm requires an addition of only **20** lines of code on top of standard implementations of soft actor-critic (SAC) [18] for continuous control experiments and on top of QR-DQN [7] for the discrete control experiments. The tradeoff factor, α is automatically tuned via Lagrangian dual gradient descent for continuous control, and is fixed at constant values described in Appendix F for discrete control. We use default hyperparameters from SAC, except that the learning rate for the policy is chosen to be 3e-5 (vs 3e-4 or 1e-4 for the Q-function), as dictated by Theorem 3.3. Elaborate details are provided in Appendix F.

5 Related Work

We now briefly discuss prior work in offline RL and off-policy evaluation, comparing and contrasting these works with our approach. More technical discussion of related work is provided in Appendix E.

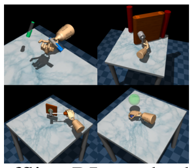
Off-policy evaluation (OPE). Several different paradigms have been used to perform off-policy evaluation. Earlier works [42, 41, 43] used per-action importance sampling on Monte-Carlo returns to obtain an OPE return estimator. Recent approaches [32, 16, 35, 48] use marginalized importance sampling by directly estimating the state-distribution importance ratios via some form of dynamic programming [30] and typically exhibit less variance than per-action importance sampling at the cost of bias. Because these methods use dynamic programming, they can suffer from OOD actions [30, 16, 19, 35]. In contrast, the regularizer in CQL explicitly addresses the impact of OOD actions due to its gap-expanding behavior, and obtains conservative value estimates.

Offline RL. As discussed in Section 2, offline Q-learning methods suffer from issues pertaining to OOD actions. Prior works have attempted to solve this problem by constraining the learned policy to be "close" to the behavior policy, for example as measured by KL-divergence [25, 47, 40, 45], Wasserstein distance [47], or MMD [27], and then only using actions sampled from this constrained policy in the Bellman backup or applying a value penalty. Most of these methods require a separately estimated model to the behavior policy, $\pi_\beta(\mathbf{a}|\mathbf{s})$ [14, 27, 47, 25, 45], and are thus limited by their ability to accurately estimate the unknown behavior policy, which might be especially complex in settings where the data is collected from multiple sources [30]. In contrast, CQL does not require estimating the behavior policy. Prior work has explored some forms of Q-function penalties [22, 46], but only in the standard online RL setting augmented with demonstrations. Alternate prior approaches to offline RL estimate some sort of uncertainty to determine the trustworthiness of a Q-value prediction [27, 2, 30], typically using uncertainty estimation techniques from exploration in online RL [39, 24, 38, 6]. These methods have not been generally performant in offline RL [14, 27, 30] due to the high-fidelity requirements of uncertainty estimates in offline RL [30]. The gap-expanding property of CQL backups, shown in Theorem 3.4, is related to how gap-increasing Bellman backup operators [5, 33] are more robust to estimation error in online RL.

6 Experimental Evaluation

We compare CQL to prior offline RL methods on a range of domains and dataset compositions, including continuous and discrete action spaces, state observations of varying dimensionality, and high-dimensional image inputs. We first evaluate actor-critic CQL, using CQL($\mathcal{H}$) from Algorithm 1, on continuous control datasets from the D4RL benchmark [11]. We compare to: prior offline RL methods that use a policy constraint – BEAR [27] and BRAC [47]; SAC [18], an off-policy actor-critic method that we adapt to offline setting; and behavioral cloning (BC).

Gym domains. Results for the gym domains are shown in Table 1. The results for BEAR, BRAC, SAC, and BC are based on numbers reported by Fu et al. [11]. On the datasets generated from a single policy, marked as "-random", "-expert" and "-medium", CQL roughly matches or exceeds the best prior methods, but by a small margin. However, on datasets that combine multiple policies ("-mixed", "-medium-expert" and "-random-expert"), that are more likely to be common in practical datasets, CQL outperforms prior methods by large margins, sometimes as much as **2-3x**.



Adroit tasks. The more complex Adroit [44] tasks (shown on the right) in D4RL require controlling a 24-DoF robotic hand, using limited data from human demonstrations. These tasks are substantially more difficult than the gym tasks in terms of both the dataset composition and high dimensionality. Prior offline RL methods generally struggle to learn meaningful behaviors on these tasks, and the strongest baseline is BC. As shown in Table 2, CQL variants are the only methods that improve over BC, attaining scores that are **2-9x** those of the next best offline RL method. CQL(ρ) with $\rho = \hat{\pi}^{k-1}$ (the previous policy) outperforms CQL($\mathcal{H}$) on a number of these tasks, due to the higher action dimensionality resulting in higher variance for the CQL($\mathcal{H}$) importance weights. Both variants outperform prior methods.

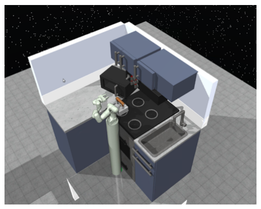
Task Name	SAC	BC	BEAR	BRAC-p	BRAC-v	CQL($\mathcal{H}$)
halfcheetah-random	2.1	30.5	25.5	23.5	28.1	**35.4**
hopper-random	9.8	**11.3**	9.5	**11.1**	**12.0**	**10.8**
walker2d-random	1.6	4.1	**6.7**	0.8	0.5	**7.0**
halfcheetah-medium	-4.3	36.1	38.6	**44.0**	**45.5**	**44.4**
walker2d-medium	0.9	6.6	33.2	72.7	**81.3**	79.2
hopper-medium	0.8	29.0	47.6	31.2	32.3	**58.0**
halfcheetah-expert	-1.9	**107.0**	**108.2**	3.8	-1.1	104.8
hopper-expert	0.7	**109.0**	**110.3**	6.6	3.7	**109.9**
walker2d-expert	-0.3	125.7	106.1	-0.2	-0.0	**153.9**
halfcheetah-medium-expert	1.8	35.8	51.7	43.8	45.3	**62.4**
walker2d-medium-expert	1.9	11.3	10.8	-0.3	0.9	**98.7**
hopper-medium-expert	1.6	**111.9**	4.0	1.1	0.8	**111.0**
halfcheetah-random-expert	53.0	1.3	24.6	30.2	2.2	**92.5**
walker2d-random-expert	0.8	0.7	1.9	0.2	2.7	**91.1**
hopper-random-expert	5.6	10.1	10.1	5.8	11.1	**110.5**
halfcheetah-mixed	-2.4	38.4	36.2	**45.6**	**45.9**	**46.2**
hopper-mixed	3.5	11.8	25.3	0.7	0.8	**48.6**
walker2d-mixed	1.9	11.3	10.8	-0.3	0.9	**26.7**

Table 1: Performance of CQL($\mathcal{H}$) and prior methods on gym domains from D4RL, on the normalized return metric, averaged over 4 seeds. Note that CQL performs similarly or better than the best prior method with simple datasets, and greatly outperforms prior methods with complex distributions ("–mixed", "–random-expert", "–medium-expert").

Domain	Task Name	BC	SAC	BEAR	BRAC-p	BRAC-v	CQL($\mathcal{H}$)	CQL(ρ)
AntMaze	antmaze-umaze	65.0	0.0	**73.0**	50.0	70.0	**74.0**	**73.5**
	antmaze-umaze-diverse	55.0	0.0	61.0	40.0	70.0	**84.0**	61.0
	antmaze-medium-play	0.0	0.0	0.0	0.0	0.0	**61.2**	4.6
	antmaze-medium-diverse	0.0	0.0	8.0	0.0	0.0	**53.7**	5.1
	antmaze-large-play	0.0	0.0	0.0	0.0	0.0	**15.8**	3.2
	antmaze-large-diverse	0.0	0.0	0.0	0.0	0.0	**14.9**	2.3
Adroit	pen-human	34.4	6.3	-1.0	8.1	0.6	37.5	**55.8**
	hammer-human	1.5	0.5	0.3	0.3	0.2	**4.4**	2.1
	door-human	0.5	3.9	-0.3	-0.3	-0.3	**9.9**	**9.1**
	relocate-human	0.0	0.0	-0.3	-0.3	-0.3	0.20	**0.35**
	pen-cloned	**56.9**	23.5	26.5	1.6	-2.5	39.2	40.3
	hammer-cloned	0.8	0.2	0.3	0.3	0.3	2.1	**5.7**
	door-cloned	-0.1	0.0	-0.1	-0.1	-0.1	0.4	**3.5**
	relocate-cloned	**-0.1**	-0.2	-0.3	-0.3	-0.3	**-0.1**	**-0.1**
Kitchen	kitchen-complete	33.8	15.0	0.0	0.0	0.0	**43.8**	31.3
	kitchen-partial	33.8	0.0	13.1	0.0	0.0	**49.8**	**50.1**
	kitchen-undirected	47.5	2.5	47.2	0.0	0.0	**51.0**	**52.4**

Table 2: Normalized scores of all methods on AntMaze, Adroit, and kitchen domains from D4RL, averaged across 4 seeds. On the harder mazes, CQL is the *only* method that attains non-zero returns, and is the only method to outperform simple behavioral cloning on Adroit tasks with human demonstrations. We observed that the CQL(ρ) variant, which avoids importance weights, trains more stably, with no sudden fluctuations in policy performance over the course of training, on the higher-dimensional Adroit tasks.

AntMaze. These D4RL tasks require composing parts of suboptimal trajectories to form more optimal policies for reaching goals on a MuJoco Ant robot. Prior methods make some progress on the simpler U-maze, but only CQL is able to make meaningful progress on the much harder medium and large mazes, outperforming prior methods by a very wide margin.



Kitchen tasks. Next, we evaluate CQL on the Franka kitchen domain [17] from D4RL [13]. The goal is to control a 9-DoF robot to manipulate multiple objects (microwave, kettle, etc.) *sequentially*, in a single episode to reach a desired configuration, with only sparse 0-1 completion reward for every object that attains the target configuration. These tasks are especially challenging, since they require composing parts of trajectories, precise long-horizon manipulation, and handling human-provided teleoperation data. As shown in Table 2, CQL outperforms prior methods in this setting, and is the only method that outperforms behavioral cloning, attaining over **40%** success rate on all tasks.

Offline RL on Atari games. Lastly, we evaluate a discrete-action Q-learning variant of CQL (Algorithm 1) on offline, image-based Atari games [4]. We compare CQL to REM [2] and QR-DQN [7] on the five Atari tasks (Pong, Breakout, Qbert, Seaquest and Asterix*) that are evaluated in detail by Agarwal et al. [2], using the dataset released by the authors.

Following the evaluation protocol of Agarwal et al. [2], we evaluated on two types of datasets, both of which were generated from the DQN-replay dataset, released by [2]: **(1)** a dataset consisting of the first 20% of the samples observed by an online DQN agent and **(2)** datasets consisting of only 1% and 10% of all samples observed by an online DQN agent (Figures 6 and 7 in [2]). In

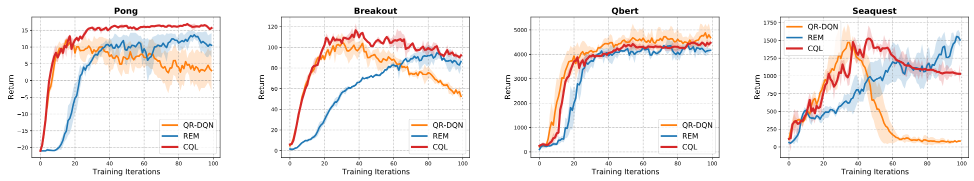


Figure 1: Performance of CQL, QR-DQN and REM as a function of training steps (x-axis) in setting **(1)** when provided with only the first 20% of the samples of an online DQN run. Note that CQL is able to learn stably on 3 out of 4 games, and its performance does not degrade as steeply as QR-DQN on Seaquest*.

setting **(1)**, shown in Figure 1, CQL generally achieves similar or better performance throughout as QR-DQN and REM. When only using only 1% or 10% of the data, in setting **(2)** (Table 3), CQL substantially outperforms REM and QR-DQN, especially in the harder 1% condition, achieving **36x** and **6x** times the return of the best prior method on Q*bert and Breakout, respectively.

Task Name	QR-DQN	REM	CQL($\mathcal{H}$)
Pong (1%)	-13.8	-6.9	**19.3**
Breakout	7.9	11.0	**61.1**
Q*bert	383.6	343.4	**14012.0**
Seaquest	672.9	499.8	**779.4**
Asterix*	166.3	386.5	**592.4**
Pong (10%)	15.1	8.9	**18.5**
Breakout	151.2	86.7	**269.3**
Q*bert	7091.3	8624.3	**13855.6**
Seaquest	2984.8	**3936.6**	3674.1
Asterix*	**189.2**	75.1	156.3

Table 3: CQL, REM and QR-DQN in setting **(1)** with 1% data (top), and 10% data (bottom). CQL drastically outperforms prior methods with 1% data, and usually attains better performance with 10% data.

Analysis of CQL. Finally, we perform empirical evaluation to verify that CQL indeed lower-bounds the value function, thus verifying Theorems 3.2, Appendix D.1 empirically. To this end, we estimate the average value of the learned policy predicted by CQL, $\mathbb{E}_{\mathbf{s}\sim\mathcal{D}}[\hat{V}^k(\mathbf{s})]$, and report the difference against the actual discounted return of the policy π^k in Table 4. We also estimate these values for baselines, including the minimum predicted Q-value under an ensemble [18, 15] of Q-functions with varying ensemble sizes, which is a standard technique to prevent overestimed Q-values [15, 18, 20] and BEAR [27], a policy constraint method. The results show that CQL learns a lower bound for all three tasks, whereas the baselines are prone to overestimation. We also evaluate a variant of CQL that uses Equation 1, and observe that the resulting values are lower (that is, under-estimate the true values) as compared to CQL($\mathcal{H}$). This provides empirical evidence that CQL($\mathcal{H}$) attains a tighter lower bound than the point-wise bound in Equation 1, as per Theorem 3.2.

Task Name	CQL($\mathcal{H}$)	CQL (Eqn. 1)	Ensemble(2)	Ens.(4)	Ens.(10)	Ens.(20)	BEAR
hopper-medium-expert	**-43.20**	-151.36	3.71e6	2.93e6	0.32e6	24.05e3	65.93
hopper-mixed	**-10.93**	-22.87	15.00e6	59.93e3	8.92e3	2.47e3	1399.46
hopper-medium	**-7.48**	-156.70	26.03e12	437.57e6	1.12e12	885e3	4.32

Table 4: Difference between policy values predicted by each algorithm and the true policy value for CQL, a variant of CQL that uses Equation 1, the minimum of an ensemble of varying sizes, and BEAR [27] on three D4RL datasets. CQL is the only method that lower-bounds the actual return (i.e., has negative differences), and CQL($\mathcal{H}$) is much less conservative than CQL (Eqn. 1).

We also present an empirical analysis to show that Theorem 3.4, that CQL is gap-expanding, holds in practice in Appendix B, and present an ablation study on various design choices used in CQL in Appendix G.

7 Discussion

We proposed conservative Q-learning (CQL), an algorithmic framework for offline RL that learns a lower bound on the policy value. Empirically, we demonstrate that CQL outperforms prior offline RL methods on a wide range of offline RL benchmark tasks, including complex control tasks and tasks with raw image observations. In many cases, the performance of CQL is substantially better than the best-performing prior methods, exceeding their final returns by 2-5x. The simplicity and efficacy of CQL make it a promising choice for a wide range of real-world offline RL problems. However, a number of challenges remain. While we prove that CQL learns lower bounds on the Q-function in the tabular, linear, and a subset of non-linear function approximation cases, a rigorous theoretical analysis of CQL with deep neural nets, is left for future work. Additionally, offline RL methods are liable to suffer from overfitting in the same way as standard supervised methods, so another important challenge for future work is to devise simple and effective early stopping methods, analogous to validation error in supervised learning.

*Neither CQL nor any prior method, including REM, attains non-trivial performance on Asterix. When using exactly the same dataset and implementation provided by Agarwal et al. [2], we were unable to reproduce the Asterix numbers reported in [2] for QR-DQN and REM.

Acknowledgements

We thank Mohammad Norouzi, Oleh Rybkin and anonymous reviewers from the Robotic AI and Learning Lab at UC Berkeley for their feedback on an earlier version of this paper. We thank Rishabh Agarwal for help with the Atari QR-DQN/REM codebase and for sharing baseline results. We thank Justin Fu for informative discussions. This research was funded by the DARPA Assured Autonomy program, and compute support from Google, Amazon, and NVIDIA.

References

[1] Joshua Achiam, Ethan Knight, and Pieter Abbeel. Towards characterizing divergence in deep q-learning. *arXiv preprint arXiv:1903.08894*, 2019.

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. 2019.

[3] DeepMind AlphaStar. Mastering the real-time strategy game starcraft ii. *URL: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii*.

[4] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[5] Marc G Bellemare, Georg Ostrovski, Arthur Guez, Philip S Thomas, and Rémi Munos. Increasing the action gap: New operators for reinforcement learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[6] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

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

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[11] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. In *arXiv*, 2020. URL `https://arxiv.org/pdf/2004.07219`.

[12] Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing bottlenecks in deep Q-learning algorithms. *arXiv preprint arXiv:1902.10250*, 2019.

[13] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for data-driven deep reinforcement learning. `https://github.com/rail-berkeley/d4rl/wiki/New-Franka-Kitchen-Tasks`, 2020. Github repository.

[14] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

[15] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, pages 1587–1596, 2018.

[16] Carles Gelada and Marc G Bellemare. Off-policy deep reinforcement learning by bootstrapping the covariate shift. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3647–3655, 2019.

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

[18] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning (ICML)*, 2017.

[19] Assaf Hallak and Shie Mannor. Consistent on-line off-policy evaluation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1372–1383. JMLR. org, 2017.

[20] Hado V Hasselt. Double q-learning. In *Advances in neural information processing systems*, pages 2613–2621, 2010.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[23] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems 31*. 2018.

[24] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

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

[26] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673, 2018.

[27] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019.

[28] Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. *arXiv preprint arXiv:2003.07305*, 2020.

[29] Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec):1107–1149, 2003.

[30] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[31] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[32] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, pages 5356–5366, 2018.

[33] Yingdong Lu, Mark S Squillante, and Chai Wah Wu. A general family of robust stochastic operators for reinforcement learning. *arXiv preprint arXiv:1805.08122*, 2018.

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

[35] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems*, pages 2315–2325, 2019.

[36] Hongseok Namkoong and John C Duchi. Variance-based regularization with convex objectives. In *Advances in neural information processing systems*, pages 2971–2980, 2017.

[37] Brendan O'Donoghue. Variational bayesian reinforcement learning with regret bounds. *arXiv preprint arXiv:1807.09647*, 2018.

[38] Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2701–2710. JMLR. org, 2017.

[39] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

[40] 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.

[41] Leonid Peshkin and Christian R Shelton. Learning from scarce experience. *arXiv preprint cs/0204043*, 2002.

[42] Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

[43] Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, pages 417–424, 2001.

[44] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems*, 2018.

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

[46] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

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

[48] Ruiyi Zhang, Bo Dai, Lihong Li, and Dale Schuurmans. Gendice: Generalized offline estimation of stationary values. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HkxlcnVFwB`.

Appendices

A Discussion of CQL Variants

We derive several variants of CQL in Section 3.2. Here, we discuss these variants on more detail and describe their specific properties. We first derive the variants: CQL($\mathcal{H}$), CQL(ρ), and then present another variant of CQL, which we call CQL(var). This third variant has strong connections to distributionally robust optimization [36].

CQL($\mathcal{H}$). In order to derive CQL($\mathcal{H}$), we substitute $\mathcal{R} = \mathcal{H}(\mu)$, and solve the optimization over μ in closed form for a given Q-function. For an optimization problem of the form:

$$\max_{\mu}\ \mathbb{E}_{\mathbf{x}\sim\mu(\mathbf{x})}[f(\mathbf{x})] + \mathcal{H}(\mu) \quad \text{s.t.} \quad \sum_{\mathbf{x}} \mu(\mathbf{x}) = 1,\ \mu(\mathbf{x}) \geq 0\ \forall \mathbf{x},$$

the optimal solution is equal to $\mu^*(\mathbf{x}) = \frac{1}{Z}\exp(f(\mathbf{x}))$, where Z is a normalizing factor. Plugging this into Equation 3, we exactly obtain Equation 4.

CQL(ρ). In order to derive CQL(ρ), we follow the above derivation, but our regularizer is a KL-divergence regularizer instead of entropy.

$$\max_{\mu}\ \mathbb{E}_{\mathbf{x}\sim\mu(\mathbf{x})}[f(\mathbf{x})] + D_{\mathrm{KL}}(\mu||\rho) \quad \text{s.t.} \quad \sum_{\mathbf{x}} \mu(\mathbf{x}) = 1,\ \mu(\mathbf{x}) \geq 0\ \forall \mathbf{x}.$$

The optimal solution is given by, $\mu^*(\mathbf{x}) = \frac{1}{Z}\rho(\mathbf{x})\exp(f(\mathbf{x}))$, where Z is a normalizing factor. Plugging this back into the CQL family (Equation 3), we obtain the following objective for training the Q-function (modulo some normalization terms):

$$\min_{Q}\ \alpha\mathbb{E}_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s})}\left[\mathbb{E}_{\mathbf{a}\sim\rho(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\frac{\exp(Q(\mathbf{s},\mathbf{a}))}{Z}\right] - \mathbb{E}_{\mathbf{a}\sim\pi_\beta(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]\right] + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q - \mathcal{B}^{\pi_k}\hat{Q}^k\right)^2\right]. \tag{5}$$

CQL(var). Finally, we derive a CQL variant that is inspired from the perspective of distributionally robust optimization (DRO) [36]. This version penalizes the variance in the Q-function across actions at all states $\mathbf{s}$, under some action-conditional distribution of our choice. In order to derive a canonical form of this variant, we invoke an identity from Namkoong and Duchi [36], which helps us simplify Equation 3. To start, we define the notion of "robust expectation": for any function $f(\mathbf{x})$, and any empirical distribution $\hat{P}(\mathbf{x})$ over a dataset $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ of N elements, the "robust" expectation defined by:

$$R_N(\hat{P}) := \max_{\mu(\mathbf{x})}\ \mathbb{E}_{\mathbf{x}\sim\mu(\mathbf{x})}[f(\mathbf{x})] \quad \text{s.t.} \quad D_f(\mu(\mathbf{x}), \hat{P}(\mathbf{x})) \leq \frac{\delta}{N},$$

can be approximated using the following upper-bound:

$$R_N(\hat{P}) \leq \mathbb{E}_{\mathbf{x}\sim\hat{P}(\mathbf{x})}[f(\mathbf{x})] + \sqrt{\frac{2\delta\ \mathrm{var}_{\hat{P}(\mathbf{x})}(f(\mathbf{x}))}{N}}, \tag{6}$$

where the gap between the two sides of the inequality decays inversely w.r.t. to the dataset size, $\mathcal{O}(1/N)$. By using Equation 6 to simplify Equation 3, we obtain an objective for training the Q-function that penalizes the variance of Q-function predictions under the distribution $\hat{P}$.

$$\min_{Q}\ \frac{1}{2}\,\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q - \mathcal{B}^{\pi_k}\hat{Q}^k\right)^2\right] + \alpha\mathbb{E}_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s})}\left[\sqrt{\frac{\mathrm{var}_{\hat{P}(\mathbf{a}|\mathbf{s})}\left(Q(\mathbf{s},\mathbf{a})\right)}{d^{\pi_\beta}(s)|\mathcal{D}|}}\right]$$
$$+\ \alpha\mathbb{E}_{s\sim d^{\pi_\beta}(\mathbf{s})}\left[\mathbb{E}_{\hat{P}(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\pi_\beta(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})]\right] \tag{7}$$

The only remaining decision is the choice of $\hat{P}$, which can be chosen to be the inverse of the empirical action distribution in the dataset, $\hat{P}(\mathbf{a}|\mathbf{s}) \propto \frac{1}{\hat{D}(\mathbf{a}|\mathbf{s})}$, or even uniform over actions, $\hat{P}(\mathbf{a}|\mathbf{s}) = \mathrm{Unif}(\mathbf{a})$, to obtain this variant of variance-regularized CQL.

B Discussion of Gap-Expanding Behavior of CQL Backups

In this section, we discuss in detail the consequences of the gap-expanding behavior of CQL backups over prior methods based on policy constraints that, as we show in this section, may not exhibit such gap-expanding behavior in practice. To recap, Theorem 3.4 shows that the CQL backup operator increases the difference between expected Q-value at in-distribution ($\mathbf{a} \sim \pi_\beta(\mathbf{a}|\mathbf{s})$) and out-of-distribution ($\mathbf{a}$ s.t. $\frac{\mu_k(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} << 1$) actions. We refer to this property as the gap-expanding property of the CQL update operator.

Function approximation may give rise to erroneous Q-values at OOD actions. We start by discussing the behavior of prior methods based on policy constraints [27, 14, 25, 47] in the presence of function approximation. To recap, because computing the target value requires $\mathbb{E}_\pi[\hat{Q}(\mathbf{s}, \mathbf{a})]$, constraining π to be close to π_β will avoid evaluating $\hat{Q}$ on OOD actions. These methods typically do not impose any further form of regularization on the learned Q-function. Even with policy constraints, because function approximation used to represent the Q-function, learned Q-values at two distinct state-action pairs are coupled together. As prior work has argued and shown [1, 12, 28], the "generalization" or the coupling effects of the function approximator may be heavily influenced by the properties of the data distribution [12, 28]. For instance, Fu et al. [12] empirically shows that when the dataset distribution is narrow (i.e. state-action marginal entropy, $\mathcal{H}(d^{\pi_\beta}(\mathbf{s}, \mathbf{a}))$, is low [12]), the coupling effects of the Q-function approximator can give rise to incorrect Q-values at different states, though this behavior is absent without function approximation, and is not as severe with high-entropy (e.g. Uniform) state-action marginal distributions.

In offline RL, we will shortly present empirical evidence on high-dimensional MuJoCo tasks showing that certain dataset distributions, $\mathcal{D}$, may cause the learned Q-value at an OOD action $\mathbf{a}$ at a state $\mathbf{s}$, to in fact take on high values than Q-values at in-distribution actions at intermediate iterations of learning. This problem persists even when a large number of samples (e.g. $1M$) are provided for training, and the agent cannot correct these errors due to no active data collection.

Since actor-critic methods, including those with policy constraints, use the learned Q-function to train the policy, in an iterative online policy evaluation and policy improvement cycle, as discussed in Section 2, the errneous Q-function may push the policy towards OOD actions, especially when no policy constraints are used. Of course, policy constraints should prevent the policy from choosing OOD actions, however, as we will show that in certain cases, policy constraint methods might also fail to prevent the effects on the policy due to incorrectly high Q-values at OOD actions.

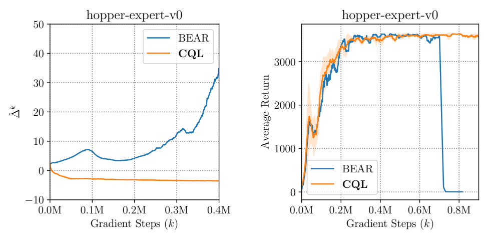
How can CQL address this problem? As we show in Theorem 3.4, the difference between expected Q-values at in-distribution actions and out-of-distribution actions is expanded by the CQL update. This property is a direct consequence of the specific nature of the CQL regularizer – that maximizes Q-values under the dataset distribution, and minimizes them otherwise. This difference depends upon the choice of α_k, which can directly be controlled, since it is a free parameter. Thus, by effectively controlling α_k, CQL can push down the learned Q-value at out-of-distribution actions as much is desired, correcting for the erroneous overestimation error in the process.

Empirical evidence on high-dimensional benchmarks with neural networks. We next empirically demonstrate the existence of of such Q-function estimation error on high-dimensional MuJoCo domains when deep neural network function approximators are used with stochastic optimization techniques. In order to measure this error, we plot the difference in expected Q-value under actions sampled from the behavior distribution, $\mathbf{a} \sim \pi_\beta(\mathbf{a}|\mathbf{s})$, and the maximum Q-value over actions sampled from a uniformly random policy, $\mathbf{a} \sim \text{Unif}(\mathbf{a}|\mathbf{s})$. That is, we plot the quantity

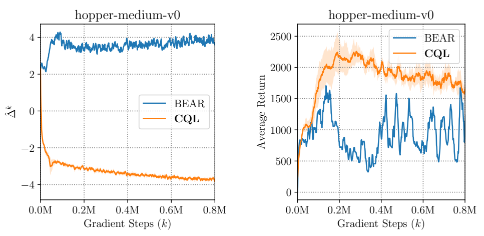
$$\hat{\Delta}^k = \mathbb{E}_{\mathbf{s}, \mathbf{a} \sim \mathcal{D}}\left[\max_{\mathbf{a}'_1, \cdots, \mathbf{a}'_N \sim \text{Unif}(\mathbf{a}')} [\hat{Q}^k(\mathbf{s}, \mathbf{a}')] - \hat{Q}^k(\mathbf{s}, \mathbf{a})\right] \tag{8}$$

over the iterations of training, indexed by k. This quantity, intuitively, represents an estimate of the "advantage" of an action $\mathbf{a}$, under the Q-function, with respect to the optimal action $\max_{\mathbf{a}'} \hat{Q}^k(\mathbf{s}, \mathbf{a}')$. Since, we cannot perform exact maximization over the learned Q-function in a continuous action space to compute Δ, we estimate it via sampling described in Equation 8.

We present these plots in Figure 2 on two datasets: hopper-expert and hopper-medium. The expert dataset is generated from a near-deterministic, expert policy, exhibits a narrow coverage of the state-action space, and limited to only a few directed trajectories. On this dataset, we find that $\hat{\Delta}^k$ is



(a) hopper-expert-v0



(b) hopper-medium-v0

Figure 2: Δ^k as a function of training iterations for hopper-expert and hopper-medium datasets. Note that CQL (left) generally has negative values of Δ, whereas BEAR (right) generally has positive Δ values, which also increase during training with increasing k values.

always positive for the policy constraint method (Figure 2(a)) and increases during training – note, the continuous rise in $\hat{\Delta}^k$ values, in the case of the policy-constraint method, shown in Figure 2(a). This means that even if the dataset is generated from an expert policy, and policy constraints correct target values for OOD actions, incorrect Q-function generalization may make an out-of-distribution action appear promising. For the more stochastic hopper-medium dataset, that consists of a more diverse set of trajectories, shown in Figure 2(b), we still observe that $\hat{\Delta}^k > 0$ for the policy-constraint method, however, the relative magnitude is smaller than hopper-expert.

In contrast, Q-functions learned by CQL, generally satisfy $\hat{\Delta}^k < 0$, as is seen and these values are clearly smaller than those for the policy-constraint method. This provides some empirical evidence for Theorem 3.4, in that, the maximum Q-value at a randomly chosen action from the uniform distribution the action space is smaller than the Q-value at in-distribution actions.

On the hopper-expert task, as we show in Figure 2(a) (right), we eventually observe an "unlearning" effect, in the policy-constraint method where the policy performance deteriorates after a extra iterations in training. This "unlearning" effect is similar to what has been observed when standard off-policy Q-learning algorithms without any policy constraint are used in the offline regime [27, 30], on the other hand this effect is absent in the case of CQL, even after equally many training steps. The performance in the more-stochastic hopper-medium dataset fluctuates, but does not deteriorate suddenly.

To summarize this discussion, we concretely observed the following points via empirical evidence:

- CQL backups are gap expanding in practice, as justified by the negative $\hat{\Delta}^k$ values in Figure 2.
- Policy constraint methods, that do not impose any regularization on the Q-function may observe highly positive $\hat{\Delta}^k$ values during training, especially with narrow data distributions, indicating that gap-expansion may be absent.
- When $\hat{\Delta}^k$ values continuously grow during training, the policy might eventually suffer from an unlearning effect [30], as shown in Figure 2(a).

C Theorem Proofs

In this section, we provide proofs of the theorems in Sections 3.1 and 3.2. We first redefine notation for clarity and then provide the proofs of the results in the main paper.

Notation. Let $k \in \mathbb{N}$ denote an iteration of policy evaluation (in Section 3.1) or Q-iteration (in Section 3.2). In an iteration k, the objective – Equation 2 or Equation 3 – is optimized using the previous iterate (i.e. $\hat{Q}^{k-1}$) as the target value in the backup. Q^k denotes the true, tabular Q-function iterate in the MDP, without any correction. In an iteration, say $k+1$, the current tabular Q-function iterate, Q^{k+1} is related to the previous tabular Q-function iterate Q^k as: $Q^{k+1} = \mathcal{B}^\pi Q^k$ (for policy evaluation) or $Q^{k+1} = \mathcal{B}^{\pi_k} Q^k$ (for policy learning). Let $\hat{Q}^k$ denote the k-th Q-function iterate obtained from CQL. Let $\hat{V}^k$ denote the value function, $\hat{V}^k := \mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}[\hat{Q}^k(\mathbf{s}, \mathbf{a})]$.

A note on the value of α. Before proving the theorems, we remark that while the statements of Theorems 3.2, 3.1 and D.1 (we discuss this in Appendix D) show that CQL produces lower bounds if

α is larger than some threshold, so as to overcome either sampling error (Theorems 3.2 and 3.1) or function approximation error (Theorem D.1). While the optimal α_k in some of these cases depends on the current Q-value, $\hat{Q}^k$, we can always choose a worst-case value of α_k by using the inequality $\hat{Q}^k \leq R_{\max}/(1-\gamma)$, still guaranteeing a lower bound.

We first prove Theorem 3.1, which shows that policy evaluation using a simplified version of CQL (Equation 1) results in a point-wise lower-bound on the Q-function.

Proof of Theorem 3.1. In order to start, we first note that the form of the resulting Q-function iterate, $\hat{Q}^k$, in the setting without function approximation. By setting the derivative of Equation 1 to 0, we obtain the following expression for $\hat{Q}^{k+1}$ in terms of $\hat{Q}^k$,

$$\forall\, \mathbf{s}, \mathbf{a} \in \mathcal{D}, k, \;\; \hat{Q}^{k+1}(\mathbf{s}, \mathbf{a}) = \hat{\mathcal{B}}^\pi \hat{Q}^k(\mathbf{s}, \mathbf{a}) - \alpha \frac{\mu(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}. \tag{9}$$

Now, since, $\mu(\mathbf{a}|\mathbf{s}) > 0, \alpha > 0, \hat{\pi}_\beta(\mathbf{a}|\mathbf{s}) > 0$, we observe that at each iteration we underestimate the next Q-value iterate, i.e. $\hat{Q}^{k+1} \leq \hat{\mathcal{B}}^\pi \hat{Q}^k$.

Accounting for sampling error. Note that so far we have only shown that the Q-values are upper-bounded by the the "empirical Bellman targets" given by, $\hat{\mathcal{B}}^\pi \hat{Q}^k$. In order to relate $\hat{Q}^k$ to the true Q-value iterate, Q^k, we need to relate the empirical Bellman operator, $\hat{\mathcal{B}}^\pi$ to the actual Bellman operator, $\mathcal{B}^\pi$. In Appendix D.3, we show that if the reward function $r(\mathbf{s}, \mathbf{a})$ and the transition function, $T(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ satisfy "concentration" properties, meaning that the difference between the observed reward sample, r $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}') \in \mathcal{D})$ and the actual reward function $r(\mathbf{s}, \mathbf{a})$ (and analogously for the transition matrix) is bounded with high probability, then overestimation due to the empirical Backup operator is bounded. Formally, with high probability (w.h.p.) $\geq 1-\delta$, $\delta \in (0, 1)$,

$$\forall Q, \mathbf{s}, \mathbf{a} \in \mathcal{D}, \;\; \left| \hat{\mathcal{B}}^\pi Q(\mathbf{s}, \mathbf{a}) - \mathcal{B}^\pi Q(\mathbf{s}, \mathbf{a}) \right| \leq \frac{C_{r,T,\delta} R_{\max}}{1-\gamma}.$$

Hence, the following can be obtained, w.h.p.:

$$\hat{Q}^{k+1}(\mathbf{s}, \mathbf{a}) = \mathcal{B}^\pi \hat{Q}^k(\mathbf{s}, \mathbf{a}) \leq \mathcal{B}^\pi \hat{Q}^k(\mathbf{s}, \mathbf{a}) - \alpha \frac{\mu(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})} + \frac{C_{r,T,\delta} R_{\max}}{1-\gamma}. \tag{10}$$

Now we need to reason about the fixed point of the update procedure in Equation 9. The fixed point of Equation 9 is given by:

$$\hat{Q}^\pi \leq \mathcal{B}^\pi \hat{Q}^\pi - \alpha \frac{\mu(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})} + \frac{C_{r,T,\delta} R_{\max}}{1-\gamma} \implies \hat{Q}^\pi \leq (I - \gamma P^\pi)^{-1} \left[R - \alpha \frac{\mu(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})} + \frac{C_{r,T,\delta} R_{\max}}{1-\gamma} \right]$$
$$\hat{Q}^\pi(\mathbf{s}, \mathbf{a}) \leq Q^\pi(\mathbf{s}, \mathbf{a}) - \alpha \left(I - \gamma P^\pi\right)^{-1} \left[\frac{\mu(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})} \right](\mathbf{s}, \mathbf{a}) + (I - \gamma P^\pi)^{-1} \frac{C_{r,T,\delta} R_{\max}}{1-\gamma},$$

thus proving the relationship in Theorem 3.1.

In order to guarantee a lower bound, α can be chosen to cancel any potential overestimation incurred by $\frac{C_{r,T,\delta} R_{\max}}{1-\gamma}$. Note that this choice works, since $(I - \gamma P^\pi)^{-1}$ is a matrix with all non-negative entries. The choice of α that guarantees a lower bound is then given by:

$$\alpha \cdot \min_{\mathbf{s}, \mathbf{a}} \left[\frac{\mu(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})} \right] \geq \frac{C_{r,T,\delta} R_{\max}}{1-\gamma}$$
$$\implies \alpha \geq \frac{C_{r,T,\delta} R_{\max}}{1-\gamma} \cdot \max_{\mathbf{s}, \mathbf{a}} \left[\frac{\mu(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})} \right]^{-1}.$$

Of course, we need $\mu(\mathbf{a}|\mathbf{s}) > 0$ whenever $\hat{\pi}_\beta(\mathbf{a}|\mathbf{s}) > 0$, for this to hold, and this is assumed in the theorem statament. Note that since, $\frac{C_{r,T,\delta} R_{\max}}{1-\gamma} = 0$, when $\hat{\mathcal{B}}^\pi = \mathcal{B}^\pi$, any $\alpha \geq 0$ guarantees a lower bound.

Next, we prove Theorem 3.3 that shows that the additional term that maximizes the expected Q-value under the dataset distribution, $\mathbb{D}(\mathbf{s}, \mathbf{a})$, (or $d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})$, in the absence of sampling error), results in a lower-bound on only the expected value of the policy at a state, and not a pointwise lower-bound on Q-values at all actions.

Proof of Theorem 3.2. We first prove this theorem in the absence of sampling error, and then incorporate sampling error at the end, using a technique similar to the previous proof. In the tabular setting, we can set the derivative of the modified objective in Equation 2, and compute the Q-function update induced in the exact, tabular setting (this assumes $\hat{\mathcal{B}}^\pi = \mathcal{B}^\pi$) and $\pi_\beta(\mathbf{a}|\mathbf{s}) = \hat{\pi}_\beta(\mathbf{a}|\mathbf{s})$).

$$\forall\, \mathbf{s}, \mathbf{a}, k \;\; \hat{Q}^{k+1}(\mathbf{s}, \mathbf{a}) = \mathcal{B}^\pi \hat{Q}^k(\mathbf{s}, \mathbf{a}) - \alpha \left[\frac{\mu(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right]. \tag{11}$$

Note that for state-action pairs, $(\mathbf{s}, \mathbf{a})$, such that, $\mu(\mathbf{a}|\mathbf{s}) < \pi_\beta(\mathbf{a}|\mathbf{s})$, we are infact adding a positive quantity, $1 - \frac{\mu(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}$, to the Q-function obtained, and this we cannot guarantee a point-wise lower bound, i.e. $\exists\; \mathbf{s}, \mathbf{a}, \text{ s.t. } \hat{Q}^{k+1}(\mathbf{s}, \mathbf{a}) \geq Q^{k+1}(\mathbf{s}, \mathbf{a})$. To formally prove this, we can construct a counter-example three-state, two-action MDP, and choose a specific behavior policy $\pi(\mathbf{a}|\mathbf{s})$, such that this is indeed the case.

The value of the policy, on the other hand, $\hat{V}^{k+1}$ is underestimated, since:

$$\hat{V}^{k+1}(\mathbf{s}) := \mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\hat{Q}^{k+1}(\mathbf{s}, \mathbf{a})\right] = \mathcal{B}^\pi \hat{V}^k(\mathbf{s}) - \alpha \mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\frac{\mu(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right]. \tag{12}$$

and we can show that $D_{\text{CQL}}(\mathbf{s}) := \sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \left[\frac{\mu(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right]$ is always positive, when $\pi(\mathbf{a}|\mathbf{s}) = \mu(\mathbf{a}|\mathbf{s})$. To note this, we present the following derivation:

$$\begin{aligned}
D_{\text{CQL}}(\mathbf{s}) :=& \sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \left[\frac{\mu(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right] \\
&= \sum_{\mathbf{a}} (\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s}) + \pi_\beta(\mathbf{a}|\mathbf{s})) \left[\frac{\mu(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right] \\
&= \sum_{\mathbf{a}} (\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})) \left[\frac{\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s}}\right] + \sum_{\mathbf{a}} \pi_\beta(\mathbf{a}|\mathbf{s}) \left[\frac{\mu(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right] \\
&= \sum_{\mathbf{a}} \underbrace{\left[\frac{(\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s}))^2}{\pi_\beta(\mathbf{a}|\mathbf{s})}\right]}_{\geq 0} + 0 \;\; \text{since, } \sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) = \sum_{\mathbf{a}} \pi_\beta(\mathbf{a}|\mathbf{s}) = 1.
\end{aligned}$$

Note that the marked term, is positive since both the numerator and denominator are positive, and this implies that $D_{\text{CQL}}(\mathbf{s}) \geq 0$. Also, note that $D_{\text{CQL}}(\mathbf{s}) = 0$, iff $\pi(\mathbf{a}|\mathbf{s}) = \pi_\beta(\mathbf{a}|\mathbf{s})$. This implies that each value iterate incurs some underestimation, $\hat{V}^{k+1}(\mathbf{s}) \leq \mathcal{B}^\pi \hat{V}^k(\mathbf{s})$.

Now, we can compute the fixed point of the recursion in Equation 12, and this gives us the following estimated policy value:

$$\hat{V}^\pi(\mathbf{s}) = V^\pi(\mathbf{s}) - \alpha \underbrace{\underbrace{(I - \gamma P^\pi)^{-1}}_{\text{non-negative entries}} \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})}\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right]}_{\geq 0},$$

thus showing that in the absence of sampling error, Theorem 3.2 gives a lower bound. It is straightforward to note that this expression is tighter than the expression for policy value in Proposition 3.2, since, we explicitly subtract 1 in the expression of Q-values (in the exact case) from the previous proof.

Incorporating sampling error. To extend this result to the setting with sampling error, similar to the previous result, the maximal overestimation at each iteration k, is bounded by $\frac{C_{r,T,\delta} R_{\max}}{1-\gamma}$. The resulting value-function satisfies (w.h.p.),

$$\hat{V}^\pi(\mathbf{s}) \leq V^\pi(\mathbf{s}) - \alpha (I - \gamma P^\pi)^{-1} \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})}\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})} - 1\right] + (I - \gamma P^\pi)^{-1} \frac{C_{r,T,\delta} R_{\max}}{1-\gamma},$$

thus proving the theorem statement. In this case, the choice of α, that prevents overestimation w.h.p. is given by:

$$\alpha \geq \frac{C_{r,T} R_{\max}}{1-\gamma} \cdot \max_{\mathbf{s}\in\mathcal{D}} \left[\sum_{\mathbf{a}\in\mathcal{D}} \pi(\mathbf{a}|\mathbf{s}) \left(\frac{\pi(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s}))} - 1\right)\right]^{-1}.$$

Next we provide a proof for Theorem 3.3.

Proof of Theorem 3.3. In order to prove this theorem, we compute the difference induced in the policy value, $\hat{V}^{k+1}$, derived from the Q-value iterate, $\hat{Q}^{k+1}$, with respect to the previous iterate $\mathcal{B}^\pi \hat{Q}^k$. If this difference is negative at each iteration, then the resulting Q-values are guaranteed to lower bound the true policy value.

$$\begin{aligned}
\mathbb{E}_{\hat{\pi}^{k+1}(\mathbf{a}|\mathbf{s})}[\hat{Q}^{k+1}(\mathbf{s},\mathbf{a})] &= \mathbb{E}_{\hat{\pi}^{k+1}(\mathbf{a}|\mathbf{s})}\left[\mathcal{B}^\pi \hat{Q}^k(\mathbf{s},\mathbf{a})\right] - \mathbb{E}_{\hat{\pi}^{k+1}(\mathbf{a}|\mathbf{s})}\left[\frac{\pi_{\hat{Q}^k}(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})} - 1\right] \\
&= \mathbb{E}_{\hat{\pi}^{k+1}(\mathbf{a}|\mathbf{s})}\left[\mathcal{B}^\pi \hat{Q}^k(\mathbf{s},\mathbf{a})\right] - \underbrace{\mathbb{E}_{\pi_{\hat{Q}^k}(\mathbf{a}|\mathbf{s})}\left[\frac{\pi_{\hat{Q}^k}(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})} - 1\right]}_{\text{underestimation, (a)}} \\
&\quad + \sum_{\mathbf{a}} \underbrace{\left(\pi_{\hat{Q}^k}(\mathbf{a}|\mathbf{s}) - \hat{\pi}^{k+1}(\mathbf{a}|\mathbf{s})\right)}_{\text{(b)},\ \leq \mathrm{D}_{\mathrm{TV}}(\pi_{\hat{Q}^k}, \hat{\pi}^{k+1})} \frac{\pi_{\hat{Q}^k}(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}
\end{aligned}$$

If (a) has a larger magnitude than (b), then the learned Q-value induces an underestimation in an iteration $k+1$, and hence, by a recursive argument, the learned Q-value underestimates the optimal Q-value. We note that by upper bounding term (b) by $\mathrm{D}_{\mathrm{TV}}(\pi_{\hat{Q}^k}, \hat{\pi}^{k+1}) \cdot \max_{\mathbf{a}} \frac{\pi_{\hat{Q}^k}(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}$, and writing out (a) > upper-bound on (b), we obtain the desired result.

Finally, we show that under specific choices of $\alpha_1, \cdots, \alpha_k$, the CQL backup is gap-expanding by providing a proof for Theorem 3.4

Proof of Theorem 3.4 (CQL is gap-expanding). For this theorem, we again first present the proof in the absence of sampling error, and then incorporate sampling error into the choice of α. We follow the strategy of observing the Q-value update in one iteration. Recall that the expression for the Q-value iterate at iteration k is given by:

$$\hat{Q}^{k+1}(\mathbf{s},\mathbf{a}) = \mathcal{B}^{\pi^k} \hat{Q}^k(\mathbf{s},\mathbf{a}) - \alpha_k \frac{\mu_k(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}.$$

Now, the value of the policy $\mu_k(\mathbf{a}|\mathbf{s})$ under $\hat{Q}^{k+1}$ is given by:

$$\mathbb{E}_{\mathbf{a}\sim\mu_k(\mathbf{a}|\mathbf{s})}[\hat{Q}^{k+1}(\mathbf{s},\mathbf{a})] = \mathbb{E}_{\mathbf{a}\sim\mu_k(\mathbf{a}|\mathbf{s})}[\mathcal{B}^{\pi^k} \hat{Q}^k(\mathbf{s},\mathbf{a})] - \alpha_k \underbrace{\mu_k^T \left(\frac{\mu_k(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}\right)}_{:=\hat{\Delta}^k, \geq 0,\ \text{by proof of Theorem 3.2.}}$$

Now, we also note that the expected amount of extra underestimation introduced at iteration k under action sampled from the behavior policy $\pi_\beta(\mathbf{a}|\mathbf{s})$ is 0, as,

$$\mathbb{E}_{\mathbf{a}\sim\pi_\beta(\mathbf{a}|\mathbf{s})}[\hat{Q}^{k+1}(\mathbf{s},\mathbf{a})] = \mathbb{E}_{\mathbf{a}\sim\pi_\beta(\mathbf{a}|\mathbf{s})}[\mathcal{B}^{\pi^k} \hat{Q}^k(\mathbf{s},\mathbf{a})] - \alpha_k \underbrace{{\pi_\beta}^T \left(\frac{\mu_k(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}\right)}_{=0}.$$

where the marked quantity is equal to 0 since it is equal since $\pi_\beta(\mathbf{a}|\mathbf{s})$ in the numerator cancels with the denominator, and the remaining quantity is a sum of difference between two density functions, $\sum_{\mathbf{a}} \mu_k(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})$, which is equal to 0. Thus, we have shown that,

$$\mathbb{E}_{\pi_\beta(\mathbf{a}|\mathbf{s})}[\hat{Q}^{k+1}(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\mu_k(\mathbf{a}|\mathbf{s})}[\hat{Q}^{k+1}(\mathbf{s},\mathbf{a})] = \mathbb{E}_{\pi_\beta(\mathbf{a}|\mathbf{s})}[\mathcal{B}^{\pi^k} \hat{Q}^k(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\mu_k(\mathbf{a}|\mathbf{s})}[\mathcal{B}^{\pi^k} \hat{Q}^k(\mathbf{s},\mathbf{a})] - \alpha_k \hat{\Delta}^k.$$

Now subtracting the difference, $\mathbb{E}_{\pi_\beta(\mathbf{a}|\mathbf{s})}[Q^{k+1}(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\mu_k(\mathbf{a}|\mathbf{s})}[Q^{k+1}(\mathbf{s},\mathbf{a})]$, computed under the tabular Q-function iterate, Q^{k+1}, from the previous equation, we obtain that

$$\begin{aligned}
\mathbb{E}_{\mathbf{a}\sim\pi_\beta(\mathbf{a}|\mathbf{s})}[\hat{Q}^{k+1}(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\pi_\beta(\mathbf{a}|\mathbf{s})}[Q^{k+1}(\mathbf{s},\mathbf{a})] &= \mathbb{E}_{\mu_k(\mathbf{a}|\mathbf{s})}[\hat{Q}^{k+1}(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\mu_k(\mathbf{a}|\mathbf{s})}[Q^{k+1}(\mathbf{s},\mathbf{a})] \\
&\quad + (\mu_k(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s}))^T \underbrace{\left[\mathcal{B}^{\pi^k}\left(\hat{Q}^k - Q^k\right)(\mathbf{s},\cdot)\right]}_{(a)} - \alpha_k \hat{\Delta}^k.
\end{aligned}$$

Now, by choosing α_k, such that any positive bias introduced by the quantity $(\mu_k(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s}))^T(a)$ is cancelled out, we obtain the following gap-expanding relationship:

$$\mathbb{E}_{\mathbf{a}\sim\pi_\beta(\mathbf{a}|\mathbf{s})}[\hat{Q}^{k+1}(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\pi_\beta(\mathbf{a}|\mathbf{s})}[Q^{k+1}(\mathbf{s},\mathbf{a})] > \mathbb{E}_{\mu_k(\mathbf{a}|\mathbf{s})}[\hat{Q}^{k+1}(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\mu_k(\mathbf{a}|\mathbf{s})}[Q^{k+1}(\mathbf{s},\mathbf{a})]$$

for, α_k satisfying,

$$\alpha_k > \max\left(\frac{(\pi_\beta(\mathbf{a}|\mathbf{s}) - \mu_k(\mathbf{a}|\mathbf{s}))^T\left[\mathcal{B}^{\pi^k}\left(\hat{Q}^k - Q^k\right)(\mathbf{s},\cdot)\right]}{\hat{\Delta}^k}, 0\right),$$

thus proving the desired result.

To avoid the dependency on the true Q-value iterate, Q^k, we can upper-bound Q^k by $\frac{R_{\max}}{1-\gamma}$, and upper-bound $(\pi_\beta(\mathbf{a}|\mathbf{s}) - \mu_k(\mathbf{a}|\mathbf{s}))^T\mathcal{B}^{\pi^k}Q^k(\mathbf{s},\cdot)$ by $D_{\text{TV}}(\pi_\beta, \mu_k)\cdot\frac{R_{\max}}{1-\gamma}$, and use this in the expression for α_k. While this bound may be loose, it still guarantees the gap-expanding property, and we indeed empirically show the existence of this property in practice in Appendix B.

To incorporate sampling error, we can follow a similar strategy as previous proofs: the worst case overestimation due to sampling error is given by $\frac{C_{r,T,\delta}R_{\max}}{1-\gamma}$. In this case, we note that, w.h.p.,

$$\left|\hat{\mathcal{B}}^{\pi^k}\left(\hat{Q}^k - Q^k\right) - \mathcal{B}^{\pi^k}\left(\hat{Q}^k - Q^k\right)\right| \le \frac{2\cdot C_{r,T,\delta}R_{\max}}{1-\gamma}.$$

Hence, the presence of sampling error adds $D_{\text{TV}}(\hat{\pi_\beta}, \mu_k)\cdot\frac{2\cdot C_{r,T,\delta}R_{\max}}{1-\gamma}$ to the value of α_k, giving rise to the following, sufficient condition on α_k for the gap-expanding property:

$$\alpha_k > \max\left(\frac{(\pi_\beta(\mathbf{a}|\mathbf{s}) - \mu_k(\mathbf{a}|\mathbf{s}))^T\left[\mathcal{B}^{\pi^k}\left(\hat{Q}^k - Q^k\right)(\mathbf{s},\cdot)\right]}{\hat{\Delta}^k} + D_{\text{TV}}(\hat{\pi_\beta}, \mu_k)\cdot\frac{2\cdot C_{r,T,\delta}R_{\max}}{1-\gamma}, 0\right),$$

concluding the proof of this theorem.

D Additional Theoretical Analysis

In this section, we present a theoretical analysis of additional properties of CQL. For ease of presentation, we state and prove theorems in Appendices D.1 and D.2 in the absence of sampling error, but as discussed extensively in Appendix C, we can extend each of these results by adding extra terms induced due to sampling error.

D.1 CQL with Linear and Non-Linear Function Approximation

Theorem D.1. *Assume that the Q-function is represented as a linear function of given state-action feature vectors* $\mathbf{F}$, *i.e.,* $Q(s,a) = w^T\mathbf{F}(s,a)$. *Let* $D = diag\left(d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})\right)$ *denote the diagonal matrix with data density, and assume that* $\mathbf{F}^TD\mathbf{F}$ *is invertible. Then, the expected value of the policy under Q-value from Eqn 2 at iteration* $k+1$, $\mathbb{E}_{d^{\pi_\beta}(\mathbf{a})}[\hat{V}^{k+1}(\mathbf{s})] = \mathbb{E}_{d^{\pi_\beta}(\mathbf{s}),\pi(\mathbf{a}|\mathbf{s})}[\hat{Q}^{k+1}(\mathbf{s},\mathbf{a})]$, *lower-bounds the corresponding tabular value,* $\mathbb{E}_{d^{\pi_\beta}(\mathbf{s})}[V^{k+1}(\mathbf{s})] = \mathbb{E}_{d^{\pi_\beta}(\mathbf{s}),\pi(\mathbf{a}|\mathbf{s})}[Q^{k+1}(\mathbf{s},\mathbf{a})]$, *if*

$$\alpha_k \ge \max\left(\frac{D^T\left[\mathbf{F}\left(\mathbf{F}^TD\mathbf{F}\right)^{-1}\mathbf{F}^T - I\right]\left((\mathcal{B}^\pi\hat{Q}^k)(\mathbf{s},\mathbf{a})\right)}{D^T\left[\mathbf{F}\left(\mathbf{F}^TD\mathbf{F}\right)^{-1}\mathbf{F}^T\right]\left(D\left[\frac{\pi(\mathbf{a}|\mathbf{s})-\pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}\right]\right)}, 0\right).$$

The choice of α_k in Theorem D.1 intuitively amounts to compensating for overestimation in value induced if the true value function cannot be represented in the chosen linear function class (numerator), by the potential decrease in value due to the CQL regularizer (denominator). This implies that if the actual value function can be represented in the linear function class, such that the numerator can be made 0, then **any** $\alpha > 0$ is sufficient to obtain a lower bound. We now prove the theorem.

Proof. In order to extend the result of Theorem 3.2 to account for function approximation, we follow the similar recipe as before. We obtain the optimal solution to the optimization problem below in the family of linearly expressible Q-functions, i.e. $\mathcal{Q} := \{\mathbf{F}w | w \in \mathbb{R}^{dim(\mathbf{F})}\}$.

$$\min_{Q \in \mathcal{Q}} \; \alpha_k \cdot \Big(\mathbb{E}_{d^{\pi_\beta}(\mathbf{s}), \mu(\mathbf{a}|\mathbf{s})} \left[Q(\mathbf{s}, \mathbf{a}) \right] - \mathbb{E}_{d^{\pi_\beta}(\mathbf{s}), \pi_\beta(\mathbf{a}|\mathbf{s})} \left[Q(\mathbf{s}, \mathbf{a}) \right] \Big) + \frac{1}{2} \mathbb{E}_{\mathcal{D}} \left[\Big(Q(\mathbf{s}, \mathbf{a}) - \mathcal{B}^\pi \hat{Q}^k(\mathbf{s}, \mathbf{a}) \Big)^2 \right].$$

By substituting $Q(\mathbf{s}, \mathbf{a}) = w^T \mathbf{F}(\mathbf{s}, \mathbf{a})$, and setting the derivative with respect to w to be 0, we obtain,

$$\alpha \sum_{\mathbf{s}, \mathbf{a}} d^{\pi_\beta}(\mathbf{s}) \cdot \left(\mu(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s}) \right) \mathbf{F}(\mathbf{s}, \mathbf{a}) + \sum_{\mathbf{s}, \mathbf{a}} d^{\pi_\beta}(\mathbf{s}) \pi_\beta(\mathbf{a}|\mathbf{s}) \left(Q(\mathbf{s}, \mathbf{a}) - \mathcal{B}^\pi \hat{Q}^k(\mathbf{s}, \mathbf{a}) \right) \mathbf{F}(\mathbf{s}, \mathbf{a}) = 0.$$

By re-arranging terms, and converting it to vector notation, defining $D = \text{diag}(d^{\pi_\beta}(\mathbf{s}) \pi_\beta(\mathbf{s}))$, and referring to the parameter w at the k-iteration as w^k we obtain:

$$\left(\mathbf{F}^T D \mathbf{F} \right) w^{k+1} = \underbrace{\mathbf{F}^T D \left(\mathcal{B}^\pi \hat{Q}^k \right)}_{\text{LSTD iterate}} - \underbrace{\alpha_k \mathbf{F}^T \text{diag} \left[d^{\pi_\beta}(\mathbf{s}) \cdot \left(\mu(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s}) \right) \right]}_{\text{underestimation}}.$$

Now, our task is to show that the term labelled as "underestimation" is indeed negative in expectation under $\mu(\mathbf{a}|\mathbf{s})$ (This is analogous to our result in the tabular setting that shows underestimated values). In order to show this, we write out the expression for the value, under the linear weights w^{k+1} at state s, after substituting $\mu = \pi$,

$$\hat{V}^{k+1}(\mathbf{s}) := \pi(\mathbf{a}|\mathbf{s})^T \mathbf{F} w^{k+1} \tag{13}$$

$$= \pi(\mathbf{a}|\mathbf{s})^T \underbrace{\mathbf{F} \left(\mathbf{F}^T D \mathbf{F} \right)^{-1} \mathbf{F}^T D \left(\mathcal{B}^\pi \hat{Q}^k \right)}_{\text{value under LSTD-Q [29]}} - \alpha_k \pi(\mathbf{a}|\mathbf{s})^T \mathbf{F} \left(\mathbf{F}^T D \mathbf{F} \right)^{-1} \mathbf{F}^T D \left[\frac{\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} \right]. \tag{14}$$

Now, we need to reason about the penalty term. Defining, $P_{\mathbf{F}} := \mathbf{F} \left(\mathbf{F}^T D \mathbf{F} \right)^{-1} \mathbf{F}^T D$ as the projection matrix onto the subspace of features $\mathbf{F}$, we need to show that the product that appears as a penalty is positive: $\pi(\mathbf{a}|\mathbf{s})^T P_{\mathbf{F}} \left[\frac{\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} \right] \geq 0$. In order to show this, we compute minimum value of this product optimizing over π. If the minimum value is 0, then we are done.

Let's define $f(\pi) = \pi(\mathbf{a}|\mathbf{s})^T P_{\mathbf{F}} \left[\frac{\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} \right]$, our goal is to solve for $\min_\pi f(\pi)$. Setting the derivative of $f(\pi)$ with respect to π to be equal to 0, we obtain (including Lagrange multiplier η that guarantees $\sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) = 1$,

$$\left(P_{\mathbf{F}} + P_{\mathbf{F}}^T \right) \left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} \right] = P_{\mathbf{F}} \vec{1} + \eta \vec{1}.$$

By solving for η (using the condition that a density function sums to 1), we obtain that the minimum value of $f(\pi)$ occurs at a $\pi^*(\mathbf{a}|\mathbf{s})$, which satisfies the following condition,

$$\left(P_{\mathbf{F}} + P_{\mathbf{F}}^T \right) \left[\frac{\pi^*(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} \right] = \left(P_{\mathbf{F}} + P_{\mathbf{F}}^T \right) \vec{1}.$$

Using this relation to compute f, we obtain, $f(\pi^*) = 0$, indicating that the minimum value of 0 occurs when the projected density ratio matches under the matrix $(P_{\mathbf{F}} + P_{\mathbf{F}}^T)$ is equal to the projection of a vector of ones, $\vec{1}$. Thus,

$$\forall \; \pi(\mathbf{a}|\mathbf{s}), \; f(\pi) = \pi(\mathbf{a}|\mathbf{s})^T P_{\mathbf{F}} \left[\frac{\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} \right] \geq 0.$$

This means that: $\forall \; \mathbf{s}, \hat{V}^{k+1}(\mathbf{s}) \leq \hat{V}^{k+1}_{\text{LSTD-Q}}(\mathbf{s})$ given identical previous $\hat{Q}^k$ values. This result indicates, that if $\alpha_k \geq 0$, the resulting CQL value estimate with linear function approximation is guaranteed to lower-bound the value estimate obtained from a least-squares temporal difference Q-learning algorithm (which only minimizes Bellman error assuming a linear Q-function parameterization), such as LSTD-Q [29], since at each iteration, CQL induces a lower-bound with respect to the previous value iterate, whereas this underestimation is absent in LSTD-Q, and an inductive argument is applicable.

So far, we have only shown that the learned value iterate, $\hat{V}^{k+1}(\mathbf{s})$ lower-bounds the value iterate obtained from LSTD-Q, $\forall$ s, $\hat{V}^{k+1}(\mathbf{s}) \leq \hat{V}^{k+1}_{\text{LSTD-Q}}(\mathbf{s})$. But, our final aim is to prove a stronger result, that the learned value iterate, $\hat{V}^{k+1}$, lower bounds the exact tabular value function iterate, V^{k+1}, at each iteration. The reason why our current result does not guarantee this is because function approximation may induce overestimation error in the linear approximation of the Q-function.

In order to account for this change, we make a simple change: we choose α_k such that the resulting penalty nullifes the effect of any over-estimation caused due to the inability to fit the true value function iterate in the linear function class parameterized by $\mathbf{F}$. Formally, this means:

$$\begin{aligned}
\mathbb{E}_{d^{\pi_\beta}(\mathbf{s})}\left[\hat{V}^{k+1}(\mathbf{s})\right] &\leq \mathbb{E}_{d^{\pi_\beta}(\mathbf{s})}\left[\hat{V}^{k+1}_{\text{LSTD-Q}}(\mathbf{s})\right] - \alpha_k \mathbb{E}_{d^{\pi_\beta}(\mathbf{s})}[f(\pi(\mathbf{a}|\mathbf{s}))] \\
&\leq \mathbb{E}_{d^{\pi_\beta}(\mathbf{s})}\left[V^{k+1}(\mathbf{s})\right] - \underbrace{\mathbb{E}_{d^{\pi_\beta}(\mathbf{s})}\left[\hat{V}^{k+1}_{\text{LSTD-Q}}(\mathbf{s}) - V^{k+1}(\mathbf{s})\right] - \alpha_k \mathbb{E}_{d^{\pi_\beta}(\mathbf{s})}[f(\pi(\mathbf{a}|\mathbf{s}))]}_{\text{choose } \alpha_k \text{ to make this negative}} \\
&\leq \mathbb{E}_{d^{\pi_\beta}(\mathbf{s})}\left[V^{k+1}(\mathbf{s})\right]
\end{aligned}$$

And the choice of α_k in that case is given by:

$$\begin{aligned}
\alpha_k &\geq \max\left(\frac{\mathbb{E}_{d^{\pi_\beta}(\mathbf{s})}\left[\hat{V}^{k+1}_{\text{LSTD-Q}}(\mathbf{s}) - V^{k+1}(\mathbf{s})\right]}{\mathbb{E}_{d^{\pi_\beta}(\mathbf{s})}[f(\pi(\mathbf{a}|\mathbf{s}))]}, 0\right) \\
&\geq \max\left(\frac{D^T\left[\mathbf{F}\left(\mathbf{F}^T D \mathbf{F}\right)^{-1}\mathbf{F}^T - I\right]\left((\mathcal{B}^\pi \hat{Q}^k)(\mathbf{s}, \mathbf{a})\right)}{D^T\left[\mathbf{F}\left(\mathbf{F}^T D \mathbf{F}\right)^{-1}\mathbf{F}^T\right]\left(D\left[\frac{\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}\right]\right)}, 0\right).
\end{aligned}$$

Finally, we note that since this choice of α_k induces under-estimation in the next iterate, $\hat{V}^{k+1}$ with respect to the previous iterate, $\hat{V}^k$, for all $k \in \mathbb{N}$, by induction, we can claim that this choice of $\alpha_1, \cdots, \alpha_k$ is sufficient to make $\hat{V}^{k+1}$ lower-bound the tabular, exact value-function iterate. V^{k+1}, for all k, thus completing our proof. □

We can generalize Theorem D.1 to non-linear function approximation, such as neural networks, under the standard NTK framework [23], assuming that each iteration k is performed by a single step of gradient descent on Equation 2, rather than a complete minimization of this objective. As we show in Theorem D.2, CQL learns lower bounds in this case for an appropriate choice of α_k. We will also empirically show in Appendix G that CQL can learn effective conservative Q-functions with multilayer neural networks.

Theorem D.2 (Extension to non-linear function approximation)**.** *Assume that the Q-function is represented by a general non-linear function approximator parameterized by θ, $Q_\theta(\mathbf{s}, \mathbf{a})$. let $D =$ diag$(d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s}))$ denote the matrix with the data density on the diagonal, and assume that $\nabla_\theta Q_\theta^T D \nabla_\theta Q_\theta$ is invertible. Then, the expected value of the policy under the Q-function obtained by taking a gradient step on Equation 2, at iteration $k+1$ lower-bounds the corresponding tabular function iterate if:*

Proof. Our proof strategy is to reduce the non-linear optimization problem into a linear one, with features $\mathbf{F}$ (in Theorem D.1) replaced with features given by the gradient of the current Q-function $\hat{Q}^k_\theta$ with respect to parameters θ, i.e. $\nabla_\theta \hat{Q}^k$. To see, this we start by writing down the expression for $\hat{Q}^{k+1}_\theta$ obtained via one step of gradient descent with step size η, on the objective in Equation 2.

$$\begin{aligned}
\theta^{k+1} = \theta^k &- \eta\alpha_k\left(\mathbb{E}_{d^{\pi_\beta}(\mathbf{s}), \mu(\mathbf{a}|\mathbf{s})}\left[\nabla_\theta \hat{Q}^k(\mathbf{s}, \mathbf{a})\right] - \mathbb{E}_{d^{\pi_\beta}(\mathbf{s}), \pi_\beta(\mathbf{a}|\mathbf{s})}\left[\nabla_\theta \hat{Q}^k(\mathbf{s}, \mathbf{a})\right]\right) \\
&- \eta \mathbb{E}_{d^{\pi_\beta}(\mathbf{s}), \pi_\beta(\mathbf{a}|\mathbf{s})}\left[\left(\hat{Q}^k - \mathcal{B}^\pi \hat{Q}^k\right)(\mathbf{s}, \mathbf{a}) \cdot \nabla_\theta \hat{Q}^k(\mathbf{s}, \mathbf{a})\right].
\end{aligned}$$

Using the above equation and making an approximation linearization assumption on the non-linear Q-function, for small learning rates $\eta << 1$, as has been commonly used by prior works on the neural tangent kernel (NTK) in deep learning theory [23] in order to explain neural network learning dynamics in the infinite-width limit, we can write out the expression for the next Q-function iterate, $\hat{Q}_\theta^{k+1}$ in terms of $\hat{Q}_\theta^k$ as [1, 23]:

$$\begin{aligned}\hat{Q}_\theta^{k+1}(\mathbf{s},\mathbf{a}) &\approx \hat{Q}_\theta^k(\mathbf{s},\mathbf{a}) + \left(\theta^{k+1}-\theta^k\right)^T \nabla_\theta \hat{Q}_\theta^k(\mathbf{s},\mathbf{a}) \quad \text{(under NTK assumptions)}\\ &= \hat{Q}_\theta^k(\mathbf{s},\mathbf{a}) - \eta\alpha_k \mathbb{E}_{d^{\pi_\beta}(\mathbf{s}'),\mu(\mathbf{a}'|\mathbf{s}')}\left[\nabla_\theta \hat{Q}^k(\mathbf{s}',\mathbf{a}')^T \nabla_\theta \hat{Q}^k(\mathbf{s},\mathbf{a})\right]\\ &+ \eta\alpha_k \mathbb{E}_{d^{\pi_\beta}(\mathbf{s}'),\pi_\beta(\mathbf{a}'|\mathbf{s}')}\left[\nabla_\theta \hat{Q}^k(\mathbf{s}',\mathbf{a}')^T \nabla_\theta \hat{Q}^k(\mathbf{s},\mathbf{a})\right]\\ &- \eta \mathbb{E}_{d^{\pi_\beta}(\mathbf{s}'),\pi_\beta(\mathbf{a}'|\mathbf{s}')}\left[\left(\hat{Q}^k - \mathcal{B}^\pi \hat{Q}^k\right)(\mathbf{s}',\mathbf{a}') \cdot \nabla_\theta \hat{Q}^k(\mathbf{s}',\mathbf{a}')^T \nabla_\theta \hat{Q}^k(\mathbf{s},\mathbf{a})\right].\end{aligned}$$

To simplify presentation, we convert into matrix notation, where we define the $|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}||\mathcal{A}|$ matrix, $\mathbf{M}^k = \left(\nabla_\theta \hat{Q}^k\right)^T \nabla_\theta \hat{Q}^k$, as the neural tangent kernel matrix of the Q-function at iteration k. Then, the vectorized $\hat{Q}^{k+1}$ (with $\mu = \pi$) is given by,

$$\hat{Q}^{k+1} = \hat{Q}^k - \eta\alpha_k \mathbf{M}^k D \left[\frac{\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}\right] + \eta \mathbf{M}^k D \left(\mathcal{B}^\pi \hat{Q}^k - \hat{Q}^k\right).$$

Finally, the value of the policy is given by:

$$\hat{V}^{k+1} := \underbrace{\pi(\mathbf{a}|\mathbf{s})^T \hat{Q}^k(\mathbf{s},\mathbf{a}) + \eta\pi(\mathbf{a}|\mathbf{s})\mathbf{M}^k D\left(\mathcal{B}^\pi\hat{Q}^k - \hat{Q}^k\right)}_{\text{(a) unpenalized value}} \underbrace{- \eta\alpha_k \pi(\mathbf{a}|\mathbf{s})^T \mathbf{M}^k D \left[\frac{\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}\right]}_{\text{(b) penalty}}. \tag{15}$$

Term marked (b) in the above equation is similar to the penalty term shown in Equation 14, and by performing a similar analysis, we can show that (b) ≥ 0. Again similar to how $\hat{V}^{k+1}_{\text{LSTD-Q}}$ appeared in Equation 14, we observe that here we obtain the value function corresponding to a regular gradient-step on the Bellman error objective. □

Again similar to before, term (a) can introduce overestimation relative to the tabular counterpart, starting at $\hat{Q}^k$: $Q^{k+1} = \hat{Q}^k - \eta\left(\mathcal{B}^\pi\hat{Q}^k - \hat{Q}^k\right)$, and we can choose α_k to compensate for this potential increase as in the proof of Theorem D.1. As the last step, we can recurse this argument to obtain our final result, for underestimation.

D.2 Choice of Distribution to Maximize Expected Q-Value in Equation 2

In Section 3.1, we introduced a term that maximizes Q-values under the dataset $d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})$ distribution when modifying Equation 1 to Equation 2. Theorem 3.2 indicates the "sufficiency" of maximizing Q-values under the dataset distribution – this guarantees a lower-bound on value. We now investigate the neccessity of this assumption: We ask the formal question: **For which other choices of $\nu(\mathbf{a}|\mathbf{s})$ for the maximization term, is the value of the policy under the learned Q-value, $\hat{Q}_\nu^{k+1}$ guaranteed to be a lower bound on the actual value of the policy?**

To recap and define notation, we restate the objective from Equation 1 below.

$$\hat{Q}^{k+1} \leftarrow \arg\min_Q \; \alpha\, \mathbb{E}_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s}),\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right] + \frac{1}{2}\, \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \mathcal{B}^\pi\hat{Q}^k(\mathbf{s},\mathbf{a})\right)^2\right]. \tag{16}$$

We define a general family of objectives from Equation 2, parameterized by a distribution ν which is chosen to maximize Q-values as shown below (CQL is a special case, with $\nu(\mathbf{a}|\mathbf{s}) = \pi_\beta(\mathbf{a}|\mathbf{s})$):

$$\begin{aligned}\hat{Q}_\nu^{k+1} \leftarrow \arg\min_Q \;& \alpha \cdot \left(\mathbb{E}_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s}),\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right] - \mathbb{E}_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s}),\mathbf{a}\sim\nu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]\right)\\ &+ \frac{1}{2}\, \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \mathcal{B}^\pi\hat{Q}^k(\mathbf{s},\mathbf{a})\right)^2\right]. \quad (\nu\text{-CQL})\end{aligned} \tag{17}$$

In order to answer our question, we prove the following result:

Theorem D.3 (Necessity of maximizing Q-values under $\pi_\beta(\mathbf{a}|\mathbf{s})$.)**.** *For any policy $\pi(\mathbf{a}|\mathbf{s})$, any $\alpha > 0$, and for all $k > 0$, the value of the policy., $\hat{V}_\nu^{k+1}$ under Q-function iterates from $\nu-$CQL, $\hat{Q}_\nu^{k+1}(\mathbf{s}, \mathbf{a})$ is guaranteed to be a lower bound on the exact value iterate, $\hat{V}^{k+1}$, only if $\nu(\mathbf{a}|\mathbf{s}) = \pi_\beta(\mathbf{a}|\mathbf{s})$.*

Proof. We start by noting the parametric form of the resulting tabular Q-value iterate:

$$\hat{Q}_\nu^{k+1}(\mathbf{s}, \mathbf{a}) = \mathcal{B}^\pi \hat{Q}_\nu^k(\mathbf{s}, \mathbf{a}) - \alpha_k \frac{\mu(\mathbf{a}|\mathbf{s}) - \nu(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}. \tag{18}$$

The value of the policy under this Q-value iterate, when distribution μ is chosen to be the target policy $\pi(\mathbf{a}|\mathbf{s})$ i.e. $\mu(\mathbf{a}|\mathbf{s}) = \pi(\mathbf{a}|\mathbf{s})$ is given by:

$$\hat{V}_\nu^{k+1}(\mathbf{s}) := \mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\hat{Q}_\nu^{k+1}(\mathbf{s}, \mathbf{a})\right] = \mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\mathcal{B}^\pi \hat{Q}_\nu^k(\mathbf{s}, \mathbf{a})\right] - \alpha_k\, \pi(\mathbf{a}|\mathbf{s})^T \left(\frac{\pi(\mathbf{a}|\mathbf{s}) - \nu(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}\right). \tag{19}$$

We are interested in conditions on $\nu(\mathbf{a}|\mathbf{s})$ such that the penalty term in the above equation is positive. It is clear that choosing $\nu(\mathbf{a}|\mathbf{s}) = \pi_\beta(\mathbf{a}|\mathbf{s})$ returns a policy that satisfies the requirement, as shown in the proof for Theorem 3.2. In order to obtain other choices of $\nu(\mathbf{a}|\mathbf{s})$ that guarantees a lower bound for all possible choices of $\pi(\mathbf{a}|\mathbf{s})$, we solve the following concave-convex maxmin optimization problem, that computes a $\nu(\mathbf{a}|\mathbf{s})$ for which a lower-bound is guaranteed for *all* choices of $\mu(\mathbf{a}|\mathbf{s})$:

$$\begin{aligned} \max_{\nu(\mathbf{a}|\mathbf{s})}\ \min_{\pi(\mathbf{a}|\mathbf{s})}\ & \sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \cdot \left(\frac{\pi(\mathbf{a}|\mathbf{s}) - \nu(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}\right) \\ \text{s.t.}\ & \sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) = 1,\ \sum_{\mathbf{a}} \nu(\mathbf{a}|\mathbf{s}) = 1,\ \nu(\mathbf{a}|\mathbf{s}) \geq 0,\ \pi(\mathbf{a}|\mathbf{s}) \geq 0. \end{aligned}$$

We first solve the inner minimization over $\pi(\mathbf{a}|\mathbf{s})$ for a fixed $\nu(\mathbf{a}|\mathbf{s})$, by writing out the Lagrangian and setting the gradient of the Lagrangian to be 0, we obtain:

$$\forall \mathbf{a},\ \ 2 \cdot \frac{\pi^*(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - \frac{\nu(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - \zeta(\mathbf{a}|\mathbf{s}) + \eta = 0,$$

where $\zeta(\mathbf{a}|\mathbf{s})$ is the Lagrange dual variable for the positivity constraints on $\pi(\mathbf{a}|\mathbf{s})$, and η is the Lagrange dual variable for the normalization constraint on π. If $\pi(\mathbf{a}|\mathbf{s})$ is full support (for example, when it is chosen to be a Boltzmann policy), KKT conditions imply that, $\zeta(\mathbf{a}|\mathbf{s})$ = 0, and computing η by summing up over actions, $\mathbf{a}$, the optimal choice of π for the inner minimization is given by:

$$\pi^*(\mathbf{a}|\mathbf{s}) = \frac{1}{2}\nu(\mathbf{a}|\mathbf{s}) + \frac{1}{2}\pi_\beta(\mathbf{a}|\mathbf{s}). \tag{20}$$

Now, plugging Equation 20 in the original optimization problem, we obtain the following optimization over only $\nu(\mathbf{a}|\mathbf{s})$:

$$\max_{\nu(\mathbf{a}|\mathbf{s})} \sum_{\mathbf{a}} \pi_\beta(\mathbf{a}|\mathbf{s}) \cdot \left(\frac{1}{2} - \frac{\nu(\mathbf{a}|\mathbf{s})}{2\pi_\beta(\mathbf{a}|\mathbf{s})}\right) \cdot \left(\frac{1}{2} + \frac{\nu(\mathbf{a}|\mathbf{s})}{2\pi_\beta(\mathbf{a}|\mathbf{s})}\right) \quad \text{s.t.}\ \sum_{\mathbf{a}} \nu(\mathbf{a}|\mathbf{s}) = 1,\ \nu(\mathbf{a}|\mathbf{s}) \geq 0. \tag{21}$$

Solving this optimization, we find that the optimal distribution, $\nu(\mathbf{a}|\mathbf{s})$ is equal to $\pi_\beta(\mathbf{a}|\mathbf{s})$. and the optimal value of penalty, which is also the objective for the problem above is equal to 0. Since we are maximizing over ν, this indicates for other choices of $\nu \neq \pi_\beta$, we can find a π so that the penalty is negative, and hence a lower-bound is not guaranteed. Therefore, we find that with a worst case choice of $\pi(\mathbf{a}|\mathbf{s})$, a lower bound can only be guaranteed only if $\nu(\mathbf{a}|\mathbf{s}) = \pi_\beta(\mathbf{a}|\mathbf{s})$. This justifies the necessity of $\pi_\beta(\mathbf{a}|\mathbf{s})$ for maximizing Q-values in Equation 2. The above analysis doesn't take into account the effect of function approximation or sampling error. We can, however, generalize this result to those settings, by following a similar strategy of appropriately choosing α_k, as previously utilized in Theorem D.1. □

D.3 CQL with Empirical Dataset Distributions

The results in Sections 3.1 and 3.2 account for sampling error due to the finite size of the dataset $\mathcal{D}$. In our practical implementation as well, we optimize a sample-based version of Equation 2, as shown below:

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q} \;\alpha \cdot \left(\sum_{\mathbf{s} \in \mathcal{D}} \mathbb{E}_{\mathbf{a} \sim \mu(\mathbf{a}|\mathbf{s})} \left[Q(\mathbf{s}, \mathbf{a}) \right] - \sum_{\mathbf{s} \in \mathcal{D}} \mathbb{E}_{\mathbf{a} \sim \pi_\beta(\mathbf{a}|\mathbf{s})} \left[Q(\mathbf{s}, \mathbf{a}) \right] \right) + \frac{1}{2|\mathcal{D}|} \sum_{\mathbf{s}, \mathbf{a}, \mathbf{s}' \in \mathcal{D}} \left[\left(Q(\mathbf{s}, \mathbf{a}) - \hat{\mathcal{B}}^\pi \hat{Q}^k(\mathbf{s}, \mathbf{a}) \right)^2 \right], \quad (22)$$

where $\hat{\mathcal{B}}^\pi$ denotes the "empirical" Bellman operator computed using samples in $\mathcal{D}$ as follows:

$$\forall \mathbf{s}, \mathbf{a} \in \mathcal{D}, \;\; \left(\hat{\mathcal{B}}^\pi \hat{Q}^k \right)(\mathbf{s}, \mathbf{a}) = r + \gamma \sum_{\mathbf{s}'} \hat{T}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) \mathbb{E}_{\mathbf{a}' \sim \pi(\mathbf{a}'|\mathbf{s}')} \left[\hat{Q}^k(\mathbf{s}', \mathbf{a}') \right], \quad (23)$$

where r is the empirical average reward obtained in the dataset when executing an action $\mathbf{a}$ at state $\mathbf{s}$, i.e. $r = \frac{1}{|\mathcal{D}(\mathbf{s}, \mathbf{a})|} \sum_{\mathbf{s}_i, \mathbf{a}_i in \mathcal{D}} \mathbf{1}_{\mathbf{s}_i = \mathbf{s}, \mathbf{a}_i = \mathbf{a}} \cdot r(\mathbf{s}, \mathbf{a})$, and $\hat{T}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ is the empirical transition matrix. Note that expectation under $\pi(\mathbf{a}|\mathbf{s})$ can be computed exactly, since it does not depend on the dataset. The empirical Bellman operator can take higher values as compared to the actual Bellman operator, $\mathcal{B}^\pi$, for instance, in an MDP with stochastic dynamics, where $\mathcal{D}$ may not contain transitions to all possible next-states $\mathbf{s}'$ that can be reached by executing action $\mathbf{a}$ at state $\mathbf{s}$, and only contains an optimistic transition.

We next show how the CQL lower bound result (Theorem 3.2) can be modified to guarantee a lower bound even in this presence of sampling error. To note this, following prior work [24, 38], we assume concentration properties of the reward function and the transition dynamics:

Assumption D.1. $\forall\, \mathbf{s}, \mathbf{a} \in \mathcal{D}$, *the following relationships hold with high probability,* $\geq 1 - \delta$

$$|r - r(\mathbf{s}, \mathbf{a})| \leq \frac{C_{r,\delta}}{\sqrt{|\mathcal{D}(\mathbf{s}, \mathbf{a})|}}, \quad ||\hat{T}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) - T(\mathbf{s}'|\mathbf{s}, \mathbf{a})||_1 \leq \frac{C_{T,\delta}}{\sqrt{|\mathcal{D}(\mathbf{s}, \mathbf{a})|}}.$$

Under this assumption, the difference between the empirical Bellman operator and the actual Bellman operator can be bounded:

$$\begin{aligned} \left| \left(\hat{\mathcal{B}}^\pi \hat{Q}^k \right) - \left(\mathcal{B}^\pi \hat{Q}^k \right) \right| &= \left| (r - r(\mathbf{s}, \mathbf{a})) + \gamma \sum_{\mathbf{s}'} \left(\hat{T}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) - T(\mathbf{s}'|\mathbf{s}, \mathbf{a}) \right) \mathbb{E}_{\pi(\mathbf{a}'|\mathbf{s}')} \left[\hat{Q}^k(\mathbf{s}', \mathbf{a}') \right] \right| \\ &\leq |r - r(\mathbf{s}, \mathbf{a})| + \gamma \left| \sum_{\mathbf{s}'} \left(\hat{T}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) - T(\mathbf{s}'|\mathbf{s}, \mathbf{a}) \right) \mathbb{E}_{\pi(\mathbf{a}'|\mathbf{s}')} \left[\hat{Q}^k(\mathbf{s}', \mathbf{a}') \right] \right| \\ &\leq \frac{C_r + \gamma C_T R_{\max}/(1-\gamma)}{\sqrt{|\mathcal{D}(\mathbf{s}, \mathbf{a})|}} \leq C_r + \gamma C_T R_{\max}/(1-\gamma) \leq C_{r,T,\delta} \frac{R_{\max}}{1-\gamma}. \end{aligned}$$

This gives us an expression to bound the potential overestimation that can occur due to sampling error, as a function of a constant, $C_{r,T,\delta}$.

E Extended Related Work and Connections to Prior Methods

In this section, we discuss related works to supplement Section 5. Specifically, we discuss the relationships between CQL and uncertainty estimation and policy-constraint methods.

Relationship to uncertainty estimation in offline RL. A number of prior approaches to offline RL estimate some sort of epistemic uncertainty to determine the trustworthiness of a Q-value prediction [27, 14, 2, 30]. The policy is then optimized with respect to lower-confidence estimates derived using the uncertainty metric. However, it has been empirically noted that uncertainty-based methods are not sufficient to prevent against OOD actions [14, 27] in and of themselves, are often augmented with policy constraints due to the inability to estimate tight and calibrated uncertainty sets. Such loose or uncalibrated uncertainty sets are still effective in providing exploratory behavior in standard, online RL [39, 38], where these methods were originally developed. However, offline RL places high demands on the fidelity of such sets [30], making it hard to directly use these methods.

How does CQL relate to prior uncertainty estimation methods? Typical uncertainty estimation methods rely on learning a pointwise upper bound on the Q-function that depends on epistemic uncertainty [24, 38] and these upper-confidence bound values are then used for exploration. In the context of offline RL, this means learning a pointwise lower-bound on the Q-function. We show in Section 3.1 that, with a naïve choice of regularizer (Equation 1), we can learn a uniform lower-bound on the Q-function, however, we then showed that we can improve this bound since the value of the policy is the primary quantity of interest that needs to be lower-bounded. This implies that CQL strengthens the popular practice of point-wise lower-bounds made by uncertainty estimation methods.

Can we make CQL dependent on uncertainty? We can slightly modify CQL to make it be account for epistemic uncertainty under certain statistical concentration assumptions. Typical uncertainty estimation methods in RL [39, 24] assume the applicability of concentration inequalities (for example, by making sub-Gaussian assumptions on the reward and dynamics), to obtain upper or lower-confidence bounds and the canonical amount of over- (under-) estimation is usually given by, $\mathcal{O}\left(\frac{1}{\sqrt{n(\mathbf{s},\mathbf{a})}}\right)$, where $n(\mathbf{s},\mathbf{a})$ is the number of times a state-action pair $(\mathbf{s},\mathbf{a})$ is observed in the dataset. We can incorporate such behavior in CQL by modifying Equation 3 to update Bellman error weighted by the cardinality of the dataset, $|\mathcal{D}|$, which gives rise to the following effective Q-function update in the tabular setting, without function approximation:

$$\hat{Q}^{k+1}(\mathbf{s},\mathbf{a}) = \mathcal{B}^\pi \hat{Q}^k(\mathbf{s},\mathbf{a}) - \alpha \frac{\mu(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s})}{n(\mathbf{s},\mathbf{a})} \rightarrow \mathcal{B}^\pi \hat{Q}^k(\mathbf{s},\mathbf{a}) \ \text{ as } \ n(\mathbf{s},\mathbf{a}) \rightarrow \infty.$$

In the limit of infinite data, i.e. $n(\mathbf{s},\mathbf{a}) \rightarrow \infty$, we find that the amount of underestimation tends to 0. When only a finite-sized dataset is provided, i.e. $n(\mathbf{s},\mathbf{a}) < N$, for some N, we observe that by making certain assumptions, previously used in prior work [24, 39] on the concentration properties of the reward value, $r(\mathbf{s},\mathbf{a})$ and the dynamics function, $T(\mathbf{s}'|\mathbf{s},\mathbf{a})$, such as follows:

$$||\hat{r}(\mathbf{s},\mathbf{a}) - r(\mathbf{s},\mathbf{a})|| \leq \frac{C_r}{\sqrt{n(\mathbf{s},\mathbf{a})}} \quad \text{and} \quad ||\hat{T}(\mathbf{s}'|\mathbf{s},\mathbf{a}) - T(\mathbf{s}'|\mathbf{s},\mathbf{a})|| \leq \frac{C_T}{\sqrt{n(\mathbf{s},\mathbf{a})}},$$

where C_r and C_T are constants, that depend on the concentration properties of the MDP, and by appropriately choosing α, i.e. $\alpha = \Omega(n(\mathbf{s},\mathbf{a}))$, such that the learned Q-function still lower-bounds the actual Q-function (by nullifying the possible overestimation that appears due to finite samples), we are still guaranteed a lower bound.

F Additional Experimental Setup and Implementation Details

In this section, we discuss some additional implementation details related to our method. As discussed in Section 4, CQL can be implemented as either a Q-learning or an actor-critic method. For our experiments on D4RL benchmarks [11], we implemented CQL on top of soft actor-critic (SAC) [18], and for experiments on discrete-action Atari tasks, we implemented CQL on top of QR-DQN [7]. We experimented with two ways of implementing CQL, first with a fixed α, where we chose $\alpha = 5.0$, and second with a varying α chosen via dual gradient-descent. The latter formulation automates the choice of α by introducing a "budget" parameter, τ, as shown below:

$$\min_Q \max_{\alpha \geq 0} \alpha \left(\mathbb{E}_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s})} \left[\log \sum_{\mathbf{a}} \exp(Q(\mathbf{s},\mathbf{a})) - \mathbb{E}_{\mathbf{a}\sim\pi_\beta(\mathbf{a}|\mathbf{s})} \left[Q(\mathbf{s},\mathbf{a})\right] \right] - \tau \right) + \frac{1}{2} \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}} \left[\left(Q - \mathcal{B}^{\pi_k} \hat{Q}^k \right)^2 \right]. \tag{24}$$

Equation 24 implies that if the expected difference in Q-values is less than the specified threshold τ, α will adjust to be close to 0, whereas if the difference in Q-values is higher than the specified threshold, τ, then α is likely to take on high values, and thus more aggressively penalize Q-values. We refer to this version as CQL-Lagrange, and we found that this version outperforms the version with a fixed α on the gym-MuJoCo D4RL benchmarks, and drastically outperforms the fixed α version on the more complex AntMazes.

Choice of α. Our experiments in Section 6 use the Lagrange version to automatically tune α during training. For our experiments, across D4RL Gym MuJoCo domains, we choose $\tau = 10.0$. For the other D4RL domains (Franka Kitchen and Adroit), we chose $\tau = 5.0$. However, for our Atari experiments, we used a fixed penalty, with $\alpha = 1.0$ chosen uniformly for Table 3 (with 10% data), $\alpha = 4.0$ chosen uniformly for Table 3 (with 1% data), and $\alpha = 0.5$ for Figure 1.

Computing $\log \sum_{\mathbf{a}} \exp(Q(\mathbf{s},\mathbf{a})$. CQL($\mathcal{H}$) uses log-sum-exp in the objective for training the Q-function (Equation 4). In discrete action domains, we compute the log-sum-exp exactly by invoking the standard `tf.reduce_logsumexp()` (or `torch.logsumexp()`) functions provided by autodiff libraries. In continuous action tasks, CQL($\mathcal{H}$) uses importance sampling to compute this quantity, where in practice, we sampled **10** action samples each at every state s from a uniform-at-random Unif($\mathbf{a}$) and the current policy, $\pi(\mathbf{a}|\mathbf{s})$ and used these alongside importance sampling to compute it as follows using $N = 10$ action samples:

$$\begin{aligned}
\log \sum_{\mathbf{a}} \exp(Q(\mathbf{s}, \mathbf{a})) &= \log \left(\frac{1}{2} \sum_{\mathbf{a}} \exp(Q(\mathbf{s}, \mathbf{a})) + \frac{1}{2} \sum_{\mathbf{a}} \exp(Q(\mathbf{s}, \mathbf{a})) \right) \\
&= \log \left(\frac{1}{2} \mathbb{E}_{\mathbf{a} \sim \text{Unif}(\mathbf{a})} \left[\frac{\exp(Q(\mathbf{s}, \mathbf{a}))}{\text{Unif}(\mathbf{a})} \right] + \frac{1}{2} \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} \left[\frac{\exp(Q(\mathbf{s}, \mathbf{a}))}{\pi(\mathbf{a}|\mathbf{s})} \right] \right) \\
&\approx \log \left(\frac{1}{2N} \sum_{\mathbf{a}_i \sim \text{Unif}(\mathbf{a})}^{N} \left[\frac{\exp(Q(\mathbf{s}, \mathbf{a}_i))}{\text{Unif}(\mathbf{a})} \right] + \frac{1}{2N} \sum_{\mathbf{a}_i \sim \pi(\mathbf{a}|\mathbf{s})}^{N} \left[\frac{\exp(Q(\mathbf{s}, \mathbf{a}_i))}{\pi(\mathbf{a}_i|\mathbf{s})} \right] \right).
\end{aligned}$$

Hyperparameters. For the D4RL tasks, we built CQL on top of the implementation of SAC provided at: `https://github.com/vitchyr/rlkit/`. Our implementation mimicked the RLkit SAC algorithm implementation, with the exception of a smaller policy learning rate, which was chosen to be 3e-5 or 1e-4 for continuous control tasks. Following the convention set by D4RL [11], we report the normalized, smooth average undiscounted return over 4 seeds for in our results in Section 6.

The other hyperparameters we evaluated on during our preliminary experiments, and might be helpful guidelines for using CQL are as follows:

- **Q-function learning rate.** We tried two learning rate values $[1e-4, 3e-4]$ for the Q-function. We didn't observe a significant difference in performance across these values. $3e-4$ which is the SAC default was chosen to be the default for CQL.
- **Policy learning rate.** We evaluated CQL with a policy learning rate in the range of $[3e-5, 1e-4, 3e-4$. We found $3e-5$ to almost uniformly attain good performance. While $1e-4$ seemed to be better on some of our experiments (such as hopper-medium-v0), but it performed badly with the real-human demonstration datasets, such as the Adroit tasks. We chose $3e-5$ as the default across all environments.
- **Lagrange threshold τ.** We ran our preliminary experiments with three values of threshold, $\tau = [2.0, 5.0, 10.0]$. However, we found that $\tau = 2.0$, led to a huge increase in the value of α (sometimes upto the order of millions), and as a result, highly underestimated Q-functions on all domains (sometimes upto the order of -1e6). On the datasets with human demonstrations – the Franka Kitchen and Adroit domains, we found that $\tau = 5.0$ obtained lower-bounds on Q-values, whereas $\tau = 10.0$ was unable to prevent overestimation in Q-values in a number of cases and Q-values diverged to highly positive values (> 1e+6). For the MuJoCo domains, we observed that $\tau = 10.0$ gave rise to a stable curve of Q-values, and hence this threshold was chosen for these experiments. Note that none of these hyperparameter selections required any notion of onine evaluation, since these choices were made based on the predicted Q-values on dataset state-actions pairs.
- **Number of gradient steps.** We evaluated our method on varying number of gradient steps. Since CQL uses a reduced policy learning rate (3e-5), we trained CQL methods for 1M gradient steps. Due to a lack of a proper valdiation error metric for offline Q-learning methods, deciding the number of gradient steps dynamically has been an open problem in offline RL. [30]. Different prior methods choose the number of steps differently. For our experiments, we used 1M gradient steps for all D4RL domains, and followed the convention from Agarwal et al. [2], to report returns after 5X gradient steps of training for the Atari results in Table 3.

Other hyperparameters, were kept identical to SAC on the D4RL tasks, including the twin Q-function trick, soft-target updates, etc. In the Atari domain, we based our implementation of CQL on top of the QR-DQN implementation provided by Agarwal et al. [2]. We did not tune any parameter from the QR-DQN implementation released with the official codebase with [2].

G Ablation Studies

In this section. we describe the experimental findings of some ablations for CQL. Specifically we aim to answer the following questions:

1. How does CQL($\mathcal{H}$) compare to CQL(ρ), with $\rho = \hat{\pi}^{k-1}$, the previous policy?
2. How does the CQL variant which uses Equation 1 compare to CQL($\mathcal{H}$) variant in Equation 4, that results in a tighter lower-bound theoretically?
3. How do the Lagrange (Equation 24) and non-Lagrange (Equation 4) formulations of CQL($\mathcal{H}$) empirically compare to each other?

We start with question **(1)**. On three MuJoCo tasks from D4RL, we evaluate the performance of both CQL($\mathcal{H}$) and CQL(ρ), as shown in Table 5. We observe that on these tasks, CQL($\mathcal{H}$) generally performs better than CQL(ρ). However, when a sampled estimate of log-sum-exp of the Q-values becomes inaccurate due to high variance importance weights, especially in large action spaces, such as in the Adroit environments, we observe in Table 2 that CQL(ρ) tends to perform better.

Task Name	**CQL($\mathcal{H}$)**	**CQL(ρ)**
halfcheetah-medium-expert	**7234.5**	3995.6
walker2d-mixed	**1227.2**	812.7
hopper-medium	**1866.1**	1166.1

Table 5: Average return obtained by CQL($\mathcal{H}$), and CQL(ρ) on three D4RL MuJoCo environments. Observe that on these environments, CQL($\mathcal{H}$) generally outperforms CQL(ρ).

Next, we evaluate the answer to question **(2)**. On three MuJoCo tasks from D4RL, as shown in Table 6, we evaluate the performance of CQL($\mathcal{H}$), with and without the dataset maximization term in Equation 2. We observe that omitting this term generally seems to decrease performance, especially in cases when the dataset distribution is generated from a single policy, for example, hopper-medium.

Task Name	**CQL($\mathcal{H}$)**	**CQL($\mathcal{H}$) (w/ Equation 1)**
hopper-medium-expert	3628.4	3610.3
hopper-mixed	**1563.2**	864.6
hopper-medium	**1866.1**	1028.4

Table 6: Average return obtained by CQL($\mathcal{H}$) and CQL($\mathcal{H}$) without the dataset average Q-value maximization term. The latter formulation corresponds to Equation 1, which is void of the dataset Q-value maximization term. We show in Theorem 3.2 that Equation 1 results in a weaker lower-bound. In this experiment, we also observe that this approach is generally outperformed by CQL($\mathcal{H}$).

Next, we answer question **(3)**. Table 7 shows the performance of CQL and CQL-Lagrange on some gym-MuJoCo and AntMaze tasks from D4RL. In all of these tasks, we observe that the Lagrange version (Equation 24, which automates the choice of α using dual gradient descent on the CQL regularizer, performs better than the non-Lagrange version (Equation 4). In some cases, for example, the AntMazes, the difference can be as high as 30% of the maximum achievable performance. On the gym MuJoCo tasks, we did not observe a large benefit of using the Lagrange version, however, there are some clear benefits, for instance in the setting when learning from hopper-mixed dataset.

Task Name	**CQL($\mathcal{H}$) (Lagrange),** $\tau = 10.0$	**CQL($\mathcal{H}$),** $\alpha = 5.0$
hopper-medium-expert	3628.4	3589.4
hopper-mixed	**1866.1**	1002.8
walker-mixed	1227.2	1055.7
walker2d-random-expert	4183.0	3934.5
hopper-medium	1866.1	1583.4
antmaze-medium-diverse	**0.53**	0.21
antmaze-large-play	**0.15**	0.02
antmaze-large-diverse	**0.14**	0.05

Table 7: Average return obtained by CQL($\mathcal{H}$) and CQL($\mathcal{H}$) with automatic tuning for α by using a Lagrange version. Observe that both versions are generally comparable, except in the AntMaze tasks, where an adaptive value of α greatly outperforms a single chosen value of α.