Appendix A Proof of Theorem

In this section, we provide the proofs of the theorems in this paper.

Remark. We provide the proofs under a tabular setting. Most continuous space can be approximately discretized to a tabular form, although the cordiality of the tabular form may be large. We define P^{π} as the tabular transition probabilities under the policy π.

A.1 Proof of Theorem 4.1

In Eqn. 5, The Q-function update can be computed in a tabular setting, by setting the derivative of the augmented objective in Eqn. 5 with respect to Q to zero,

$$\varepsilon\left(d^{\phi}\pi - d^{\pi_\beta}\pi\right) + \alpha\left(d^{\pi_\beta}\pi - d^{\pi_\beta}\pi_\beta\right) + d^{\pi_\beta}\pi_\beta\left(Q - \mathcal{B}\hat{Q}^k\right) = 0$$

Therefore, we can obtain $\hat{Q}^{k+1}$ in terms of $\hat{Q}^k$ by rearranging the terms,

$$\hat{Q}^{k+1}(\mathbf{s},\mathbf{a}) = \hat{\mathcal{B}}^{\pi}\hat{Q}^k(\mathbf{s},\mathbf{a}) - \varepsilon\frac{\left(d^{\phi}(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s})\right)\pi(\mathbf{a}|\mathbf{s})}{d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})} - \alpha\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right], \tag{8}$$

for all $\mathbf{s}\in\mathcal{S}$, $\mathbf{a}\in\mathcal{A}$ and $k\in[0,+\infty)$. For the state-action pair $(\mathbf{s},\mathbf{a})$ such that $d^{\phi}(\mathbf{s}) < d^{\pi_\beta}(\mathbf{s})$ and $\pi(\mathbf{a}|\mathbf{s}) < \pi_\beta(\mathbf{a}|\mathbf{s})$, the last two terms of Eqn. 8, $-\varepsilon\frac{\left(d^{\phi}(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s})\right)\pi(\mathbf{a}|\mathbf{s})}{d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})} - \alpha\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right]$ is positive, so that we cannot simply lower bound the true Q-function $Q^{k+1}(\mathbf{s},\mathbf{a})$ by the estimated one $\hat{Q}^{k+1}(\mathbf{s},\mathbf{a})$ point-wise. However, we can prove that the value function, which is the expectation of the Q-function, can be lower bounded. Taking the expectation of both sides of Eqn. 8 under the distribution $\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})$, we have

$$\hat{V}^{k+1}(\mathbf{s}) = \mathcal{B}^{\pi}\hat{V}^k(\mathbf{s}) - \varepsilon\mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\frac{\left(d^{\phi}(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s})\right)\pi(\mathbf{a}|\mathbf{s})}{d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})}\right] - \alpha\mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right]. \tag{9}$$

The first goal is to prove that $\hat{V}^{k+1} \le \mathcal{B}^{\pi}\hat{V}^k$ which implies that each iteration introduces some underestimation, and $\hat{V}^k$ could eventually converge to a fixed point. Therefore, we need to prove the last two terms on the right hand side of Eqn. 9 is negative. We denote Δ to be the opposite of the last two terms on the right hand side of Eqn. 9, then

$$\begin{aligned}\Delta &= \varepsilon\mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\frac{\left(d^{\phi}(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s})\right)\pi(\mathbf{a}|\mathbf{s})}{d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})}\right] + \alpha\mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right] \\ &= \varepsilon\frac{d^{\phi}(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s})}{d^{\pi_\beta}(\mathbf{s})}\sum_{\mathbf{a}}\frac{\pi^2(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} + \alpha\sum_{\mathbf{a}}\pi(\mathbf{a}|\mathbf{s})\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right].\end{aligned} \tag{10}$$

From the proof in [1], we know that the second term in Eqn. 10 is non-negative when $\alpha > 0$, that is

$$D_{\text{CQL}}(\mathbf{s}) = \sum_{\mathbf{a}}\pi(\mathbf{a}|\mathbf{s})\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right] \ge 0. \tag{11}$$

Hence, when $d^{\phi}(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s}) \ge 0$, it is obvious that $\Delta \ge 0$ for all $\varepsilon > 0$. When $d^{\phi}(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s}) < 0$, if ε satisfies

$$\varepsilon \le \alpha D_{\text{CQL}}(\mathbf{s})\left(\frac{\left|d^{\phi}(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s})\right|}{d^{\pi_\beta}(\mathbf{s})}\sum_{\mathbf{a}}\frac{\pi^2(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}\right)^{-1}, \tag{12}$$

then we have $\Delta \ge 0$.

In summary, we have $\Delta \ge 0$ when Eqn. 12 holds. Since the exact Bellman update operator $\mathcal{B}^{\pi}$ is a contraction [4], we have

$$||\mathcal{B}^{\pi}\hat{V}^{k+1} - \mathcal{B}^{\pi}\hat{V}^k|| = ||(\mathcal{B}^{\pi}\hat{V}^{k+1} - \Delta) - (\mathcal{B}^{\pi}\hat{V}^k - \Delta)|| \le \gamma||\hat{V}^{k+1} - \hat{V}^k||$$

which implies that each value-function update $\hat{V}^{k+1} = \mathcal{B}^\pi \hat{V}^k - \Delta$ is a contraction. According to the contraction mapping theorem, the recursive update in Eqn. 9 will always lead value function to converge to a fixed point $\hat{V}^\pi$. Now that $V^\pi = \mathcal{B}^\pi V^\pi$ for the true value functions, by subtracting them from both side of Eqn. 9 and substitute $\hat{V}^{k+1}$ and $\hat{V}^k$ with the fixed point $\hat{V}^\pi$, we have

$$\hat{V}^\pi = V^\pi - \underbrace{(I - \gamma P^\pi)^{-1}}_{\text{Positive semi-definite}} \left[\underbrace{\alpha \mathbb{E}_\pi \left[\frac{\pi}{\pi_\beta} - \mathbf{1} \right] + \varepsilon \mathbb{E}_\pi \left[\frac{(d^\phi - d^{\pi_\beta})\pi}{d^{\pi_\beta} \pi_\beta} \right]}_{\geq \mathbf{0}} \right], \tag{13}$$

when ε satisfies

$$\frac{\varepsilon}{\alpha} \leq \min_{\mathbf{s}} \left(\sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1 \right] \right) \left(\frac{\left| d^\phi(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s}) \right|}{d^{\pi_\beta}(\mathbf{s})} \sum_{\mathbf{a}} \frac{\pi^2(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} \right)^{-1}. \tag{14}$$

In Eqn. 13, we stretch all notations to be vectors, which means $\hat{V}^\pi \in \Re^{|\mathcal{S}|}$, $V^\pi \in \Re^{|\mathcal{S}|}$, and $\alpha \mathbb{E}_\pi \left[\frac{\pi}{\pi_\beta} - \mathbf{1} \right] + \varepsilon \mathbb{E}_\pi \left[\frac{(d^\phi - d^{\pi_\beta})\pi}{d^{\pi_\beta} \pi_\beta} \right] \in \Re^{|\mathcal{S}|}$ are all vectors containing values for all states. Here, $\mathbf{1}$ denotes the vector in which the entries are all 1. The expectations and the operations inside $\alpha \mathbb{E}_\pi \left[\frac{\pi}{\pi_\beta} - \mathbf{1} \right] + \varepsilon \mathbb{E}_\pi \left[\frac{(d^\phi - d^{\pi_\beta})\pi}{d^{\pi_\beta} \pi_\beta} \right]$ are all computed in a point-wise manner.

Therefore, we can conclude from Eqn. 13 that the estimated value function $\hat{V}^\pi(\mathbf{s})$ is a lower bound of the true value-function $V^\pi(\mathbf{s})$ without considering any sampling error. Thus, we finish the proof of Thm. 4.1.

A.2 Value Lower Bound in Existence of Sampling Errors

We now take sampling error into account. First, we introduce a lemma from [1]:

Lemma A.1 *If with high probability δ the reward function $r(\mathbf{s}, \mathbf{a})$ and the transition function $T(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ can be estimated with bounded error, then the sampling error of the empirical Bellman operator is also bounded:*

$$\left| \hat{\mathcal{B}}^\pi V(\mathbf{s}) - \mathcal{B}^\pi V(\mathbf{s}) \right| \leq \frac{C_{r,T,\delta} R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}(\mathbf{s}, \mathbf{a})|}}, \tag{15}$$

where $C_{r,T,\delta}$ is a constant related to r, T, and δ, and $R_{\max}$ is the maximum possible reward in the environment.

Note that the bound of the error $\left| \hat{\mathcal{B}}^\pi V(\mathbf{s}) - \mathcal{B}^\pi V(\mathbf{s}) \right|$ in Lemma A.1 only holds for states and actions in the training datasets, i.e. $\mathbf{s}, \mathbf{a} \in \mathcal{D}$. We have no reward or transition pair collected at unseen states or actions outside the training dataset, so it is impossible to bound the error outside the training dataset when consider the sampling error introduced by the reward function and the transition function. Therefore, we can lower bound the true value function by the learned value function at states and actions in the training datasets as in the following corollary.

Corollary A.1 *When the sampling error defined in Lemma A.1, for any state and any action in the training dataset, $\mathbf{s}, \mathbf{a} \in \mathcal{D}$, the learned value function via Eqn. 5 is a lower bound of the true one, i.e., $\hat{V}^\pi(\mathbf{s}) \leq V^\pi(\mathbf{s})$, if the trade-off factor ε and α satisfy the constraints*

$$\begin{aligned} \varepsilon &\leq \alpha \min_{\mathbf{s} \in \mathcal{D}} \left(\sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1 \right] \right) \left(\frac{\left| d^\phi(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s}) \right|}{d^{\pi_\beta}(\mathbf{s})} \sum_{\mathbf{a}} \frac{\pi^2(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} \right)^{-1} \\ \alpha &\geq \max_{\mathbf{s}, \mathbf{a} \in \mathcal{D}} \frac{C_{r,T,\delta} R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}(\mathbf{s}, \mathbf{a})|}} \cdot \max_{\mathbf{s}, \mathbf{a} \in \mathcal{D}} \left(\sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1 \right] \right)^{-1}. \end{aligned} \tag{16}$$

We now show the proof of Corollary A.1. From Eqn. 13, we can directly bound the estimated value function for any $\mathbf{s}, \mathbf{a} \in \mathcal{D}$ by

$$\hat{V}^{\pi} = V^{\pi} - (I - \gamma P^{\pi})^{-1} \left[\alpha \mathbb{E}_{\pi} \left[\frac{\pi}{\pi_{\beta}} - \mathbf{1} \right] + \varepsilon \mathbb{E}_{\pi} \left[\frac{(d^{\phi} - d^{\pi_{\beta}})\pi}{d^{\pi_{\beta}} \pi_{\beta}} \right] - \frac{C_{r,T,\delta} R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}(\mathbf{s}, \mathbf{a})|}} \right], \tag{17}$$

when ε and α satisfy the constraints in Eqn. 16. Note that in Eqn. 17, we use vector notations similar to those in Eqn. 13.

Therefore, when we consider sampling error introduced by the reward function and the transition pair, the learned value function by PessORL still lower bounds the true one for any states and actions in the training dataset. Thus, we finish the proof of Corollary A.1.

A.3 Proof of Theorem 4.2

We begin the proof from Eqn. 9. We first take the expectation of both side of Eqn. 9 under the distribution d^{ϕ}, then

$$\begin{aligned} \mathbb{E}_{d^{\phi}(\mathbf{s})} \left[\hat{V}^{k+1}(\mathbf{s}) \right] &= \mathbb{E}_{d^{\phi}(\mathbf{s})} \left[\mathcal{B}^{\pi} \hat{V}^{k}(\mathbf{s}) \right] \\ &- \varepsilon \sum_{\mathbf{s}} d^{\phi}(\mathbf{s}) \frac{d^{\phi}(\mathbf{s}) - d^{\pi_{\beta}}(\mathbf{s})}{d^{\pi_{\beta}}(\mathbf{s})} \sum_{\mathbf{a}} \frac{\pi^2(\mathbf{a}|\mathbf{s})}{\pi_{\beta}(\mathbf{a}|\mathbf{s})} - \alpha \sum_{\mathbf{s}} d^{\phi}(\mathbf{s}) \sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_{\beta}(\mathbf{a}|\mathbf{s})} - 1 \right]. \end{aligned} \tag{18}$$

Similarly, we take the expectation of both side of Eqn. 9 under the distribution $d^{\pi_{\beta}}$, then we have

$$\begin{aligned} \mathbb{E}_{d^{\pi_{\beta}}(\mathbf{s})} \left[\hat{V}^{k+1}(\mathbf{s}) \right] &= \mathbb{E}_{d^{\pi_{\beta}}(\mathbf{s})} \left[\mathcal{B}^{\pi} \hat{V}^{k}(\mathbf{s}) \right] \\ &- \varepsilon \sum_{\mathbf{s}} d^{\pi_{\beta}}(\mathbf{s}) \frac{d^{\phi}(\mathbf{s}) - d^{\pi_{\beta}}(\mathbf{s})}{d^{\pi_{\beta}}(\mathbf{s})} \sum_{\mathbf{a}} \frac{\pi^2(\mathbf{a}|\mathbf{s})}{\pi_{\beta}(\mathbf{a}|\mathbf{s})} - \alpha \sum_{\mathbf{s}} d^{\pi_{\beta}}(\mathbf{s}) \sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_{\beta}(\mathbf{a}|\mathbf{s})} - 1 \right]. \end{aligned} \tag{19}$$

If we subtract Eqn. 19 from Eqn. 18, we get

$$\begin{aligned} \mathbb{E}_{d^{\phi}} \left[\hat{V}^{k+1}(\mathbf{s}) \right] - \mathbb{E}_{d^{\pi_{\beta}}} \left[\hat{V}^{k+1}(\mathbf{s}) \right] &= d^{\phi\top} \mathcal{B}^{\pi} \hat{V}^{k}(\mathbf{s}) - d^{\pi_{\beta}\top} \mathcal{B}^{\pi} \hat{V}^{k}(\mathbf{s}) \\ - \varepsilon \sum_{\mathbf{s}} \frac{(d^{\pi}(\mathbf{s}) - d^{\pi_{\beta}}(\mathbf{s}))^2}{d^{\pi_{\beta}}(\mathbf{s})} \sum_{\mathbf{a}} \frac{\pi^2(\mathbf{a}|\mathbf{s})}{\pi_{\beta}(\mathbf{a}|\mathbf{s})} - \alpha \sum_{\mathbf{s}} (d^{\phi}(\mathbf{s}) - d^{\pi_{\beta}}(\mathbf{s})) \sum_{\mathbf{a}} \frac{(\pi(\mathbf{a}|\mathbf{s}) - \pi_{\beta}(\mathbf{a}|\mathbf{s}))^2}{\pi_{\beta}(\mathbf{a}|\mathbf{s})} \end{aligned} \tag{20}$$

Therefore, we have $\mathbb{E}_{d^{\phi}} \left[\hat{V}^{k+1}(\mathbf{s}) \right] \leq \mathbb{E}_{d^{\pi_{\beta}}} \left[\hat{V}^{k+1}(\mathbf{s}) \right]$, if ε satisfies

$$\begin{aligned} \varepsilon \geq \left[d^{\phi\top} \mathcal{B}^{\pi} \hat{V}^{k}(\mathbf{s}) - d^{\pi_{\beta}\top} \mathcal{B}^{\pi} \hat{V}^{k}(\mathbf{s}) - \alpha \sum_{\mathbf{s}} (d^{\phi}(\mathbf{s}) - d^{\pi_{\beta}}(\mathbf{s})) \sum_{\mathbf{a}} \frac{(\pi(\mathbf{a}|\mathbf{s}) - \pi_{\beta}(\mathbf{a}|\mathbf{s}))^2}{\pi_{\beta}(\mathbf{a}|\mathbf{s})} \right] \\ \times \left[\sum_{\mathbf{s}} \frac{(d^{\pi}(\mathbf{s}) - d^{\pi_{\beta}}(\mathbf{s}))^2}{d^{\pi_{\beta}}(\mathbf{s})} \sum_{\mathbf{a}} \frac{\pi^2(\mathbf{a}|\mathbf{s})}{\pi_{\beta}(\mathbf{a}|\mathbf{s})} \right]^{-1}. \end{aligned} \tag{21}$$

A.4 Existence of Feasible Trade-off Factor

Note that both Eqn. 21 and Eqn. 16 put constraints on the trade-off factor ε. We show that we can choose an appropriate value of α to ensure that a feasible ε that satisfies both constraints exist.

Formally, we denote

$$
\begin{aligned}
X &= d^{\phi\top}\mathcal{B}^{\pi}\hat{V}^{k}(\mathbf{s}) - d^{\pi_\beta\top}\mathcal{B}^{\pi}\hat{V}^{k}(\mathbf{s}),\\
Y &= \sum_{\mathbf{s}}(d^{\phi}(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s}))\sum_{\mathbf{a}}\frac{(\pi(\mathbf{a}|\mathbf{s}) - \pi_\beta(\mathbf{a}|\mathbf{s}))^2}{\pi_\beta(\mathbf{a}|\mathbf{s})},\\
Z &= \sum_{\mathbf{s}}\frac{(d^{\pi}(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s}))^2}{d^{\pi_\beta}(\mathbf{s})}\sum_{\mathbf{a}}\frac{\pi^2(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})},\\
W &= \min_{\mathbf{s}\in\mathcal{D}}\left(\sum_{\mathbf{a}}\pi(\mathbf{a}|\mathbf{s})\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right]\right)\left(\frac{\left|d^{\phi}(\mathbf{s}) - d^{\pi_\beta}(\mathbf{s})\right|}{d^{\pi_\beta}(\mathbf{s})}\sum_{\mathbf{a}}\frac{\pi^2(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})}\right)^{-1}\\
U &= \max_{\mathbf{s},\mathbf{a}\in\mathcal{D}}\frac{C_{r,T,\delta}R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}(\mathbf{s},\mathbf{a})|}}\cdot\max_{\mathbf{s},\mathbf{a}\in\mathcal{D}}\left(\sum_{\mathbf{a}}\pi(\mathbf{a}|\mathbf{s})\left[\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1\right]\right)^{-1}
\end{aligned}
\tag{22}
$$

for simplicity. From Eqn. 16 and 21, we have

$$
\begin{gathered}
(X - \alpha Y)Z^{-1} \le \varepsilon \le \alpha W\\
\alpha \ge U
\end{gathered}
$$

Hence, there exists a feasible ε when $(X - \alpha Y)Z^{-1} \le \alpha W$ and $\alpha \ge U$. Thus, when

$$
\alpha \ge \min\left(X(WZ + Y)^{-1}, U\right), \tag{23}
$$

we can choose a feasible ε from the interval $\left[(X - \alpha Y)\,Z^{-1}, \alpha W\right]$.

Appendix B Implementation Details of the Algorithm

The implementation of our algorithm is based on the original implementation of CQL: https://github.com/aviralkumar2907/CQL. We first train a bag of dynamics models $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\}$, and then train the Q-network and policy network. When we train the Q-network and policy network, the uncertainty estimation model $u_\pi(\mathbf{s})$ is induced by the pre-trained dynamics models. We found in the experiments that a fixed trade-off factor ε would cause the learned value function to be too conservative and hence the learned policy to fail, so we choose to use a varying ε which is adjusted by dual gradient-descent. Following the implementation of CQL, we introduce a "budget" parameter τ to automatically control ε as below:

$$\min_{Q} \max_{\varepsilon \geq 0} \left[\varepsilon \left(\log \sum_{\mathbf{s}} \zeta(\mathbf{s}) \exp\left(V^{\hat{\pi}^k}(\mathbf{s}) \right) - \mathbb{E}_{\mathbf{s} \sim d^{\pi_\beta}(\mathbf{s})}[V^{\hat{\pi}^k}(\mathbf{s})] \right) - \tau \right] + \mathcal{E}(Q, \hat{\mathcal{B}}^\pi \hat{Q}^k_\theta) + \mathcal{C}(Q). \quad (24)$$

From Eqn. 24, we can see that if the discrepancy between values is less than τ, then ε will be set to zero. When the discrepancy is larger than the threshold τ, ε will be increased to a large value to penalize harder on the value function. This automatic mechanism ensures a reasonable choice of ε, and reduce the tedious procedure to tune the hyperparameter ε. In the Gym MuJoCo domain, we choose $\tau = 0.0$ for the Hopper environment, and $\tau = 10.0$ for the Walker2d, Halfcheetah and Ant environment. In the Adroit domain, we choose $\tau = 20.0$ for all four environments.

For the dynamics models learning part, we follow the convention in model-based reinforcement learning domain, and adopt a four layers MLP with a size of 400 each. We choose to train five bootstrap models and collect them in a bag to estimate uncertainty later in the policy evaluation and improvement steps. Each dynamic model is a Gaussian model which outputs the mean and the logstd of the next state deviation. When we train the models, we iterate through all training data for 10 epochs with a learning rate of $1e-4$ and a batch size of 256.

For the Q-function learning part of the algorithm, we inherit the twin Q-function trick and soft target updates from the original SAC implementation on D4RL tasks. The Q-functions are optimized by Adam with a batch size of 256, and the learning rate is chosen to be $3e-4$ across the environments. We design the Q-functions to have three layers with a size of 256 each. When we evaluate the logsumexp term in Eqn. 24, we need to sample states from the state space. However, the state space is unbounded on the gym domain and Adroit domain, so we set the sample range to be $[\mu - 10 * \sigma, \mu + 10 * \sigma]$, where μ and σ are the mean and the standard deviation of the current batch sampled from the training dataset.

For the policy learning part of the algorithm, we do not need an explicit policy network to model the policy, but just an implicit argmax policy in discrete settings such as Pointmass environment. In continuous settings on the Gym and the Adroit domain, we adopt a Tanh-Gaussian policy structure by the rlkit repository: https://github.com/vitchyr/rlkit. Since the action spaces in these domains are all bounded by -1 and 1, the Tanh-Gaussian policy can capture this constraint naturally. The policy network is designed to have three layers with a size of 256 each. The policy network is also optimized by Adam with a batch size of 256, with a learning rate of $3e-4$ for the Pointmass and the Gym domain, and $3e-5$ for the Adroit domain.

Appendix C Ablation Studies

C.1 What if $d^\phi(\mathbf{s})$ is induced by a uniform distribution?

In Sec. 5, we introduced a practical method to construct $d^\phi(\mathbf{s})$. Alternatively, we may simply set $\zeta(\mathbf{s})$ as a uniform distribution, which also satisfies our requirement and leads to $d^\phi(\mathbf{s}) \propto \exp\left(V^{\hat{\pi}^k}(\mathbf{s})\right)$. A uniform distribution could be more convenient if it is time consuming to construct $d^\phi(\mathbf{s})$ from data. Formally, if we choose $d^\phi(\mathbf{s}) \propto \exp\left(V^{\hat{\pi}^k}(\mathbf{s})\right)$ which is induced by a uniform distribution, then the practical PessORL policy evaluation step becomes:

$$\min_Q \max_{\varepsilon \geq 0} \left[\varepsilon \left(\log \sum_{\mathbf{s}} \exp\left(V^{\hat{\pi}^k}(\mathbf{s})\right) - \mathbb{E}_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s})}[V^{\hat{\pi}^k}(\mathbf{s})] \right) - \tau \right] + \mathcal{E}(Q, \hat{\mathcal{B}}^\pi \hat{Q}^k_\theta) + \mathcal{C}(Q). \quad (25)$$

We name this variant of PessORL as PessORL-uniform. We evaluate the variant PessORL-uniform on two D4RL MuJoCo environments, and compare its performance to the original PessORL algorithm, CQL, BEAR, and BC based on the average returns over three random seeds. In Fig. 3, we show the learning curves and the value gap Δ_k in hopper-medium-v0 and walker2d-medium-v0. The value gap $\Delta_k = \max_{\mathbf{s}\in\mathcal{S}}[V(\mathbf{s})] - \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}[V(\mathbf{s})]$ follows the same definition in Sec. 6.2, which evaluates whether a method can assign high values at the states in the training data and low values at the out-of-distribution states.

In Fig. 3(a) and (c), we can see that the average return of PessORL-uniform is lower than those of CQL and PessORL. Fig. 3(b) and (d) show that the value function learned by PessORL-uniform is either not pessimistic enough or too pessimistic, causing the average return to be lower than the other two rivals. We believe the reason is that a uniform distribution in PessORL-uniform does not discriminate between OOD states and in-distribution states, so the objective function in Eqn. 25 works differently from the original one in PessORL. Eqn. 25 increases the values at in-distribution states, instead of penalizing the values at OOD states as before. This mechanism cannot guarantee an accurate penalization on most OOD states, so we observe different gaps Δ_k in different environments.

In conclusion, constructing $d^\phi(\mathbf{s})$ via a uniform distribution leads to worse performance, but it could be a cheap alternative in terms of computational cost.

C.2 What if an uncertainty term is directly subtracted from the learned Q-function?

In Sec. 1 and Sec. 3.2, we discussed two main categories of method to obtain pessimistic value functions. Besides adding a regularization term in policy evaluation step as in the proposed PessORL, we can also improve the policy by a pessimistic estimate of Q-function. This pessimistic Q-function can be obtained by directly subtracting an uncertainty term from the learned value function, i.e., $\hat{Q}^c_\theta(\mathbf{s}, \mathbf{a}) = \hat{Q}^k_\theta(\mathbf{s}, \mathbf{a}) - \beta \text{Unc}(\mathbf{s}, \mathbf{a})$, where we use the superscript c to denote a conservative Q-function, and β is a trade-off factor that makes the scale of the Q-function and the uncertainty term comparable. In this section, we first introduce two variants of the proposed PessORL algorithm, named PessORL-unc and PessORL-OPIQ, in which we used the aforementioned pessimistic Q-function to improve the policy. Then we compare the variants PessORL-unc and PessORL-OPIQ to

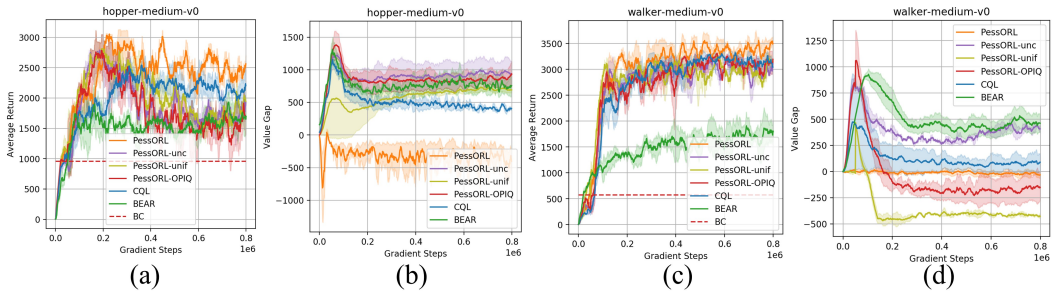


Figure 3: (a) and (c): The learning curves in hopper-medium-v0 and walker2d-medium-v0, respectively. (b) and (d): The value gap Δ_k in hopper-medium-v0 and walker2d-medium-v0, respectively.

the original PessORL, and discuss why we choose to obtain a pessimistic Q-function via adding a regularization term in policy evaluation step instead of directly subtracting an uncertainty term from the original learned value function.

We first introduce a variant of our algorithm, named PessORL-unc, in which the additional regularization term in the original PessORL policy evaluation step is removed. The policy evaluation step in PessORL-unc then becomes as follows:

$$\min_{Q} \mathcal{E}(Q, \hat{B}^{\pi}\hat{Q}^{k}_{\theta}) + \mathcal{C}(Q). \tag{26}$$

Actually, Eqn. 26 is the same as the policy evaluation step in CQL. In the ablation studies, we use this optimization problem to learn a Q-function first, and then we directly subtract an uncertainty term from the learned Q-function to get the final Conservative Q-function to be used in the policy improvement step:

$$\hat{\pi}_{k+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{\mathbf{s}\sim\mathcal{S},\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})} \left[\hat{Q}^{k+1}_{\theta}(\mathbf{s},\mathbf{a}) - \beta \text{Unc}(\mathbf{s},\mathbf{a})\right], \tag{27}$$

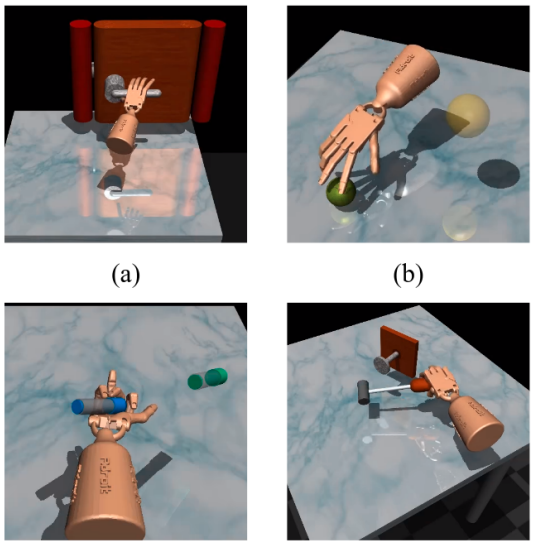
where the uncertainty term $\text{Unc}(\mathbf{s},\mathbf{a})$ is similar to the one in the original PessORL, but we no longer take the expectation over the action. Therefore, it becomes $\text{Unc}(\mathbf{s},\mathbf{a}) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{f}_{\phi_i}(\mathbf{s},\mathbf{a}) - \bar{f}_{\phi}(\mathbf{s},\mathbf{a})\right)^2$. This term evaluates the uncertainty of the dynamics model about a state-action pair. If the sampled state is not in the training dataset, the uncertainty about it will be larger than those in the training data. Therefore, the Q-function is constrained to be lower at out-of-distribution states in this way.

Second, we introduce the variant PessORL-OPIQ. It is inspired by the work in [5], which uses a pessimistic initialization and an optimistic component on the Q-function. Since in the offline settings we need a pessimistic component in the Q-function, we set the PessORL-OPIQ uncertainty term $\text{Unc}(\mathbf{s},\mathbf{a})$ to be the opposite of the optimistic component in OPIQ, i.e., $\text{Unc}(\mathbf{s},\mathbf{a}) = -\frac{C_{\text{action}}}{(N(\mathbf{s},\mathbf{a})+1)^M}$, where C_{action} and M are hyperparameters in OPIQ, and $N(\mathbf{s},\mathbf{a})$ is the pseudo count of the state-action pair $(\mathbf{s},\mathbf{a})$. Here, we use the continuous version of the OPIQ to obtain state-action counts. We adopt the pessimistic initialization from the OPIQ and update the Q function via similar policy evaluation step in Eqn. 26, but with an PessORL-OPIQ uncertainty term.

We evaluate the variants PessORL-unc and PessORL-OPIQ on two D4RL MuJoCo environments, and compare their performance to the original algorithm PessORL, CQL, BEAR, and BC based on the average returns on three random seeds. From Fig. 3(a) and (c), we can see that the variants PessORL-unc and PessORL-OPIQ both converge to similar returns which are lower than CQL and the original PessORL. Actually, the performance of these two variants are close to each other. We believe the reason is that they both require a super accurate uncertainty estimation and a stringent trade-off factor β. A rough uncertainty estimation may sometimes be harmful to the performance. As we discussed in Sec. 6.2, our uncertainty estimation method based on bootstrapped dynamics models still cannot guarantee an extremely precise estimation. Therefore, directly subtracting the uncertainty term from the learned Q-function to obtain a conservative Q-function may cause the algorithm to perform poorly. On the other side, the original PessORL mitigates the requirement of an accurate uncertainty estimation method because of the additional regularization term in Eqn. 7. We also noticed that PessORL-unc and PessORL-OPIQ still outperform the BEAR algorithm. The reason is that we implement PessORL-unc and PessORL-OPIQ on top of CQL, which is a strong offline RL method, so we can attribute most of their good performance to CQL.

From Fig. 3(b) and (d), we can see that PessORL-unc maintains a value gap that is among the highest. PessORL-OPIQ is not pessimistic enough as shown in Fig. 3(b), but is too pessimistic as shown in Fig. 3(d). This indicates that they cannot effectively control the gap and thus cannot assign lower values to out-of-distribution states. The original PessORL algorithm can successfully maintain a value gap that is close to zero, indicating that it shape the value function to be the desired shape.

Appendix D Generalization to Real Robots



(a) (b) (c) (d)

Figure 4: Screen shots of Adroit tasks. (a) Door; (b) Relocate; (c) Pen; (d) Hammer.

Although the Adroit environments are robotic manipulation simulations, they are considered to be complex enough so that the performance of an offline RL method on these tasks can be viewed as a strong evidence of how the performance will be on real robots. There are actually prior works [2] that use Adroit environments to demonstrate the ability of RL algorithms to adapt to complex tasks. The Adroit environments, as shown in Fig. 4, contains four challenging dexterous robot manipulation tasks, which include controlling a 24-DoF simulated manipulator to twirl a pen, open a door, hammer a nail, and relocate a ball. The simulator of Adroit environments provides carefully modeled kinematics, dynamics, and sensing details of the physical hardware to encourage physical realism. The observations contain joint angles, position and orientations of the hand, the object and the target. The actions include the desired position of hand joints. These are all considered to be highly realistic. Therefore, the performance of our proposed PessORL on the Adroit domain compared to other offline RL methods can indicate a good evidence of how it will behave when transferred to real robots.

Besides, our proposed method PessORL matches the simulation results of prior methods that have been demonstrated to work on real robots. Singh et al. [3] showed that CQL can chain behaviors from data and the learned policy can work on a real robot WidowX. From Fig. 1, Fig. 2, and Tab. 1, we can see that PessORL achieves similar or higher performances on all three domains. Therefore, we believe our proposed method should not be too hard to achieve good performance on real robots.

Furthermore, a big advantage of offline reinforcement learning methods is that unlike online RL methods, they are designed to learn policies from pure data and then can be deployed to real robots. Learning directly from previously collected data can dramatically reduce the safety risk in the learning process in safe-critical tasks, such as robotic manipulation and autonomous driving. Actually, it can be a major block for online reinforcement learning to be deployed on real-world scenarios, because online interaction can be impractical in many settings. We show the performance of our proposed PessORL algorithm learned from human demonstrations in Sec. 6. These experiments align with the motivation of developing offline reinforcement learning methods, and we believe they can provide a good evidence of the transfer-ability of our method to real-robots.

References

[1] A. Kumar, A. Zhou, G. Tucker, and S.Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[2] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[3] A. Singh, A.Yu, J. Yang, J. Zhang, A. Kumar, and S. Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500*, 2020.

[4] Smith, Trey, and Reid Simmons. Point-based POMDP algorithms: Improved analysis and implementation. *arXiv preprint arXiv:1207.1412*, 2012.

[5] Rashid, Tabish, et al. Optimistic exploration even with a pessimistic initialisation. *arXiv preprint arXiv:2002.12174*, 2020.