Model-Bellman Inconsistency for Model-based Offline Reinforcement Learning

Yihao Sun [*1] **Jiaji Zhang** [*1 2] **Chengxing Jia** [1 2] **Haoxin Lin** [1 2] **Junyin Ye** [1 2] **Yang Yu** [1 2 3]

Abstract

For offline reinforcement learning (RL), model-based methods are expected to be data-efficient as they incorporate dynamics models to generate more data. However, due to inevitable model errors, straightforwardly learning a policy in the model typically fails in the offline setting. Previous studies have incorporated conservatism to prevent out-of-distribution exploration. For example, MOPO penalizes rewards through uncertainty measures from predicting the next states, which we have discovered are loose bounds of the ideal uncertainty, i.e., the Bellman error. In this work, we propose **MO**del-**B**ellman **I**nconsistency penalized off**L**in**E** Policy Optimization (MOBILE), a novel uncertainty-driven offline RL algorithm. MOBILE conducts uncertainty quantification through the inconsistency of Bellman estimations under an ensemble of learned dynamics models, which can be a better approximator to the true Bellman error, and penalizes the Bellman estimation based on this uncertainty. Empirically we have verified that our proposed uncertainty quantification can be significantly closer to the true Bellman error than the compared methods. Consequently, MOBILE outperforms prior offline RL approaches on most tasks of D4RL and NeoRL benchmarks.

1. Introduction

Offline reinforcement learning (RL) (Lange et al., 2012; Levine et al., 2020), which learns a policy from a previously collected static dataset, is a promising way to enable a safe training paradigm compared to conventional RL, which needs a lot of online trial-and-error explorations. However, direct applications of online off-policy algorithms perform poorly in the offline setting (Fujimoto et al., 2019; Kumar et al., 2019). This is mainly attributed to the distribution shift between the learned policy and the behavior policy throughout training, which makes the policy evaluation on out-of-distribution (OOD) samples quite inaccurate and induces terrible performance.

Thus, a major principle for offline RL is to introduce conservatism to prevent the learned policy from executing OOD actions. Model-free offline RL algorithms train a policy from only the offline data and achieve conservatism by compelling the learned policy to be close to the behavior policy (Kumar et al., 2019; Fujimoto & Gu, 2021), or by penalizing the learned value functions from being over-optimistic upon OOD actions (Kumar et al., 2020; Bai et al., 2022).

Compared to the model-free approaches, model-based offline RL approaches (Yu et al., 2020; Kidambi et al., 2020; Yu et al., 2021; Chen et al., 2021) are inherently data-efficient as they incorporate dynamics models to generate more data. With the help of supplementary synthetic data, model-based algorithms can potentially generalize better to states not present in the dataset. Similarly, model-based offline RL also needs to incorporate conservatism due to inevitable model errors (Xu et al., 2020; Janner et al., 2019; Luo et al., 2019). To this end, some work (Yu et al., 2020; Kidambi et al., 2020; Lu et al., 2022) proposes to quantify the uncertainty of learned dynamics models, and explicitly apply it to penalize reward. For example, MOPO (Yu et al., 2020) penalizes rewards through the aleatoric uncertainty from predicting the next states. However, referring to recent theoretical investigation on offline RL (Jin et al., 2021), we have discovered such uncertainty quantification is a loose bound of the ideal uncertainty, i.e., the Bellman error. Therefore, the performance of these model-based approaches falls short of the model-free counterparts.

In this work, we propose MOBILE (**MO**del-**B**ellman **I**nconsistency Penalized Off**L**in**E** Policy Optimization), an uncertainty-driven model-based algorithm for offline RL. To better incorporate conservatism into the model utilization, we devise a novel uncertainty quantifier termed *Model-Bellman Inconsistency* to penalize the value estimation in the learned dynamics. Concretely, this quantifier estimates the Bellman errors induced by the learned dynamics through

[*]Equal contribution [1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, China [2]Polixir Technologies, Nanjing, Jiangsu, China [3]Peng Cheng Laboratory, Shenzhen, 518055, China. Correspondence to: Yang Yu <yuy@nju.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

the inconsistency of Bellman estimations under different learned models. With penalization by such a quantifier, the policy is encouraged to yield consistent and reliable value estimation in the learned dynamics. Crucially, unlike other approaches such as MOPO, MOBILE leverages both the dynamics and the value to construct penalization and tends to attain a better estimation of the Bellman error. Empirically we have verified that our proposed uncertainty quantification can be significantly closer to the true Bellman error than the compared methods. Consequently, MOBILE outperforms prior offline RL approaches and achieves state-of-the-art performance on 20 out of 27 benchmark datasets. The code is available at https://github.com/yihaosun1124/mobile.

2. Preliminaries

2.1. MDPs and Offline RL

We consider an episodic MDP specified by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, r, T, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, H is the length of episodes, $r(s,a)$ is the reward function, $T(s'|s,a)$ is the transition function, and $\gamma \in (0,1)$ is the discount factor. The goal of RL is to learn a policy π that maximizes the expected cumulative reward $\mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t r_t\right]$. Such a policy can be derived from Q-learning, which learns a state-action value function that satisfies the following Bellman operator,

$$\mathcal{T}Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim T(s'|s,a)} \left[\max_{a'} Q(s',a')\right], \quad (1)$$

where $Q(s,a)$ represents the expected cumulative discounted reward when starting from state s and action a. In Deep Reinforcement Learning (DRL), the Q-value is parameterized by a neural network and trained by minimizing the TD error, namely $\mathbb{E}_{(s,a,r,s')}[(Q_\phi - \mathcal{T}Q_\phi)^2]$. The transitions for training are collected by iteratively interacting with the environment in online RL.

Nevertheless, in the context of offline RL, the transitions are sampled from a static dataset $\mathcal{D} = \{(s,a,r,s')\}$. Directly applying Q-learning in offline RL tends to have extrapolation error since (s',a') has barely occurred in $\mathcal{D}$, where a' is the action taken by the agent under state s'. Such errors will be accumulated due to the bootstrap training.

2.2. Model-based Offline RL Algorithms

Model-based offline RL attempts to find the optimal policy with the help of a learned dynamics model. Given a dataset $\mathcal{D}$, a dynamics model $\widehat{T}$ is typically trained using maximum likelihood estimation as: $\min_{\widehat{T}} \mathbb{E}_{(s,a,s') \sim \mathcal{D}}[-\log \widehat{T}(s'|s,a)]$. We assume the reward function r is known throughout the paper, although r can also be considered as part of the model if unknown. With the learned model, we can construct an estimated MDP $\widehat{\mathcal{M}}$. Thereafter, any planning or RL algorithm can be used to recover optimal policy in the estimated MDP.

Inevitably, the model will be inaccurate for some state-action pairs due to the partial coverage of the state-action space by the dataset. Therefore, naive policy optimization on a learned model in the offline setting can be vulnerable to model exploitation. An intuitive idea to mitigate this issue is to use the model conservatively. MOPO (Yu et al., 2020) and MOReL (Kidambi et al., 2020) optimize a lower bound of policy performance constructed by an uncertainty estimation of the learned model. With the penalization of such an uncertainty estimation, we hope the policy to avoid utilizing unreliable predictions of the model.

In line with some existing work, we choose model-based policy optimization (MBPO) (Janner et al., 2019) to learn the optimal policy for $\widehat{\mathcal{M}}$. MBPO utilizes a standard actor-critic RL algorithm but uses an augmented dataset $\mathcal{D} \cup \mathcal{D}_{\text{model}}$ to train the policy, where $\mathcal{D}_{\text{model}}$ is synthetic data generated by performing h-step rollouts in $\widehat{\mathcal{M}}$ starting from states in $\mathcal{D}$. During training policy, mini-batches of data are drawn from $\mathcal{D} \cup \mathcal{D}_{\text{model}}$, where each datapoint is sampled from the real data D with the probability f, and from $\mathcal{D}_{\text{model}}$ with probability $1-f$.

3. Model-Bellman Inconsistency Penalized Offline Policy Optimization

The key to devising an effective model-based offline reinforcement learning algorithm is reasonably leveraging the model. In principle, an agent should exploit "safe regions" where the model is accurate and avoid "dangerous regions" where the model is inaccurate. Meanwhile, it is crucial to balance conservatism and generalization. Specifically, finding a better policy requires taking the risk of accessing "dangerous" regions, while accessing "safe regions" too conservatively hinders finding a better policy.

To approach the optimal balance, we propose *Model-Bellman Inconsistency* for uncertainty quantification and then use such quantification as penalization (Section 3.1). This uncertainty quantification encourages the algorithm to learn a policy with reliable and consistent value estimations under ensemble dynamics models. Then, we provide a theoretical analysis to show that our proposed uncertainty quantification is reasonable and better than the previous approach (Section 3.2). Finally, we present our overall algorithm (Section 3.3).

3.1. Pessimistic Value Estimation via Model-Bellman Inconsistency

In MOBILE, we learn an ensemble of N dynamics models $\{\widehat{T}_\theta^i\}_{i=1}^N$. Based on the learned dynamics model, the Q-

function can be updated by fitting the following target

$$\widehat{\mathcal{T}}^{\pi} Q_{\psi}(s,a) := r(s,a) + \gamma \mathbb{E}_{\substack{s' \sim \widehat{T}_{\theta} \\ a' \sim \pi}} \left[Q_{\psi^-}(s',a') \right], \quad (2)$$

where $\widehat{T}_{\theta} = \frac{1}{N} \sum_{i=1}^{N} \widehat{T}_{\theta}^{i}$, ψ is the parameter of Q-network and ψ^- is the parameter of a separate target-network for stabilizing training (Mnih et al., 2015). Recall that we have assumed that r is known, so we do not parameterize it in Eq. (2). Here, we denote the empirical Bellman operator by $\widehat{\mathcal{T}}^{\pi}$, which estimates the TD target under the true dynamics. Note that we also refer to the estimated result of this empirical Bellman operator as "Bellman estimation" in this paper.

We prefer a policy with reliable value estimation under the learned models. In other words, we expect the Bellman estimation of a policy under the learned models to be close to that under the true dynamics. To this end, a natural way is to penalize the Bellman estimation according to the estimation error defined as

$$\begin{aligned} \epsilon &= \left| \widehat{\mathcal{T}}^{\pi} Q_{\psi}(s,a) - \mathcal{T}^{\pi} Q_{\psi}(s,a) \right| \\ &= \gamma \left| \mathbb{E}_{\substack{s' \sim \widehat{T}_{\theta} \\ a' \sim \pi}} \left[Q_{\psi^-}(s',a') \right] - \mathbb{E}_{\substack{s' \sim T^* \\ a' \sim \pi}} \left[Q_{\psi^-}(s',a') \right] \right|, \end{aligned} \quad (3)$$

where T^* is the transition function of the unknown true dynamics.

Remark 3.1. In this paper, we also use Bellman error to refer to the estimation error described in Eq. (3). Bellman error is traditionally defined as the discrepancy between the expected TD target, $\mathbb{E}_{s' \sim T^*, a' \sim \pi}[r(s,a) + \gamma Q_{\psi^-}(s',a')]$, and the current value estimation, $Q_{\psi}(s,a)$, for a given state-action pair (s,a). Despite the potential misuse of this terminology, it is justifiable in this context since we assume the reward function is known and the value function's fitting error is negligible with enough data generated by the learned dynamics model.

As we will show in Section 3.2, the Bellman error is the theoretically optimal penalization. However, such penalization is intractable since the true transition function is unknown. Our key idea is to build a proper uncertainty quantification to estimate the Bellman error ϵ. Therefore, we introduce the following uncertainty quantification based on the ensemble models,

$$\begin{aligned} \mathcal{U}(s,a) &:= \mathrm{Std} \left(\widehat{\mathcal{T}}_i^{\pi} Q_{\psi}(s,a) \right) \\ &= \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\widehat{\mathcal{T}}_i^{\pi} Q_{\psi}(s,a) - \widehat{\mathcal{T}}^{\pi} Q_{\psi}(s,a) \right)^2}, \end{aligned} \quad (4)$$

where $\widehat{\mathcal{T}}_i^{\pi} Q_{\psi}(s,a)$ is the Bellman estimation under the i-th dynamics model, i.e., $\widehat{\mathcal{T}}_i^{\pi} Q_{\psi}(s,a) = r(s,a) + \gamma \mathbb{E}_{\substack{s' \sim \widehat{T}_{\theta}^{i} \\ a' \sim \pi}} \left[Q_{\psi^-}(s',a') \right]$. We term this quantification as *Model-Bellman Inconsistency* since it quantifies the inconsistency of Bellman estimations under the learned ensemble dynamics models. Intuitively, Bellman estimations usually have small errors in areas with rich data and tend to yield low inconsistency under the learned ensemble models, while high estimation errors often appear in areas with scarce data and then tend to yield high inconsistency under these ensemble models. Now, we can penalize the Bellman estimation via the Model-Bellman Inconsistency to obtain a pessimistic value estimation,

$$\widehat{\mathcal{T}}^{\mathrm{MOBIP}} Q_{\psi}(s,a) := \widehat{\mathcal{T}}^{\pi} Q_{\psi}(s,a) - \beta \mathcal{U}(s,a), \quad (5)$$

where β is a tuning coefficient. We refer to this operator as *MOdel-Bellman Inconsistency Penalized (MOBIP) operator*.

The basic idea behind this novel operator is that if the learned policy outputs actions that have high Model-Bellman Inconsistency, we have reason to suspect that the models will induce inaccurate value estimations on these actions. Hence we give them large penalization to prevent the policy from taking these dangerous actions.

3.2. Theoretical Connections to PEVI

In this section, we show that our proposed pessimistic target is reasonable based on the recent theoretical investigation on offline RL (Jin et al., 2021). Meanwhile, our analysis also explains why pessimism implemented by Model-Bellman Inconsistency is superior to the pessimism implementation in MOPO.

We begin with the *pessimistic value iteration* algorithm (PEVI) (Jin et al., 2021), which simply penalizes the Bellman estimation with a penalty function. We refer to Appendix A for a detailed introduction to PEVI (Algorithm 2). Note that such a penalty function plays a key role in PEVI. Specifically, it has been proved that this function should be a ξ-uncertainty quantifier to ensure the effectiveness of the offline algorithm (Jin et al., 2021).

Definition 3.2. (ξ-Uncertainty Quantifier (Jin et al., 2021)). The set of penalization $\{\Gamma_h\}_{h \in [H]}$ forms a ξ-uncertainty quantifier if it holds with probability at least 1-ξ that

$$\left| \widehat{\mathcal{T}} \widehat{V}_{h+1}(s,a) - \mathcal{T} \widehat{V}_{h+1}(s,a) \right| \leq \Gamma_h(s,a) \quad (6)$$

for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, where $\widehat{V}_{h+1}$ is an estimated value function at step $h+1$ and $\widehat{\mathcal{T}}$ is an empirical Bellman operator estimating the true Bellman operator $\mathcal{T}$.

The ξ-uncertainty quantifier allows us to further characterize the suboptimality of the derived policy from PEVI by the following theorem.

Theorem 3.3. *(Suboptimality of PEVI (Jin et al., 2021)). Suppose $\{\Gamma_h\}_{h=1}^{H}$ in PEVI is a ξ-uncertainty quantifier.*

Then the derived policy $\widehat{\pi}$ satisfies

$$\left|V^{\pi^*}(s_1)-V^{\widehat{\pi}}(s_1)\right| \leq 2\sum_{h=1}^{H} \mathbb{E}_{\pi^*}\left[\Gamma_h\left(s_h,a_h\right)|\, s_1=s\right], \quad (7)$$

with probability at least $1-\xi$ for all $s \in \mathcal{S}$. Here $\mathbb{E}_{\pi^*}$ is with respect to the trajectory induced by the optimal policy π^* in the underlying MDP given the fixed function Γ_h.

It is worth noting that the optimality gap is dominated by the Bellman error and the uncertainty quantification Γ_h. On the one hand, we expect the Γ_h to be as small as possible to establish a tighter upper bound of the optimality gap. On the other hand, the Bellman error determines the lower bound of Γ_h, which makes it not arbitrarily small. This motivates us to reduce the Bellman error and construct a tighter uncertainty quantification Γ_h.

Compared to the model-free paradigm, the model-based paradigm can effectively reduce the Bellman error in the offline case, as demonstrated in Yu et al., 2020. Therefore, we mainly consider how to build a tighter quantification for the Bellman error.

Recall that in the model-based case, the Bellman error is defined as Eq. (3), which is caused by the inconsistency between the learned model $\widehat{T}_\theta$ and the true model T^*. In order to estimate the Bellman error, we consider the posterior distribution of the Bellman estimation $\widehat{\mathcal{T}}\widehat{V}_{h+1}(s,a)$, which is determined by the posterior distribution of the dynamics model $\widehat{T}_\theta$ given the offline dataset $\mathcal{D}$, and construct an epistemic uncertainty based on the distribution over $\widehat{\mathcal{T}}\widehat{V}_{h+1}(s,a)$. Formally, we make the following assumption.

Assumption 3.4. Consider an estimation of the standard deviation of the posterior over $\widehat{\mathcal{T}}\widehat{V}_{h+1}(s,a)$, which is constructed by an ensemble of N dynamics models $\{\widehat{T}_\theta^i\}_{i=1}^N$ trained on the dataset $\mathcal{D}$:

$$\begin{aligned}&\text{Std}\left(\widehat{\mathcal{T}}_i\widehat{V}_{h+1}(s,a)\right)\\ &\quad=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\widehat{\mathcal{T}}_i\widehat{V}_{h+1}(s,a)-\widehat{\mathcal{T}}\widehat{V}_{h+1}(s,a)\right)^2},\end{aligned} \quad (8)$$

where $\widehat{\mathcal{T}}_i\widehat{V}_{h+1}(s,a) = r(s,a)+\gamma\mathbb{E}_{s'\sim\widehat{T}_\theta^i}[\widehat{V}_{h+1}(s')]$ and $\widehat{\mathcal{T}}\widehat{V}_{h+1}(s,a) = \frac{1}{N}\sum_{i=1}^N \widehat{\mathcal{T}}_i\widehat{V}_{h+1}(s,a)$. Assume that this epistemic uncertainty is an admissible error estimator for $\widehat{\mathcal{T}}$ under an appropriately selected tuning parameter β_h, it follows that

$$\left|\widehat{\mathcal{T}}\widehat{V}_{h+1}(s,a)-\mathcal{T}\widehat{V}_{h+1}(s,a)\right| \leq \beta_h \text{Std}\left(\widehat{\mathcal{T}}_i\widehat{V}_{h+1}(s,a)\right) \quad (9)$$

for all $s\in\mathcal{S}, a\in\mathcal{A}$.

While this assumption lacks theoretical guarantees, using aleatoric or epistemic uncertainty to estimate the approximation error has been applied in many works (Yu et al., 2020; Bai et al., 2022). We have provided much evidence in Section 4.3 that this uncertainty quantification is sufficiently accurate to estimate the Bellman error. From this assumption, we have the following theorem.

Theorem 3.5. *Under Assumption 3.4, our proposed Model-Bellman Inconsistency*

$$\beta_h\mathcal{U}(s,a) = \beta_h \textit{Std}\left(\widehat{\mathcal{T}}_i\widehat{V}_{h+1}(s,a)\right)$$

forms a valid ξ-uncertainty quantifier.

To further understand why our proposed uncertainty quantification is superior to the model uncertainty used in MOPO, we revisit MOPO under the perspective of PEVI. MOPO adopts the max aleatoric error $\max_{i=1,\ldots,N}||\Sigma_\theta^i(s,a)||_F$ as penalization, where $\{\Sigma_\theta^i(s,a)\}_{i=1}^N$ are the variance heads of the ensemble models. The following theorem suggests that such an uncertainty quantification is also a valid ξ-uncertainty quantifier.

Theorem 3.6. *Assume $\mathcal{U}^{MOPO}(s,a)=\max_i||\Sigma_\theta^i(s,a)||_F$ is an admissible error estimator, specifically,*

$$D_{TV}(\widehat{T}_\theta(s,a),T^*(s,a)) \leq \mathcal{U}^{MOPO}(s,a).$$

Then $\gamma c\cdot\mathcal{U}^{MOPO}(s,a)$ forms a valid ξ-uncertainty quantifier where $c=\mathcal{O}(\frac{1}{1-\gamma})$.

Proof.

$$\begin{aligned}&\left|\widehat{\mathcal{T}}^\pi\widehat{V}_{h+1}(s,a)-\mathcal{T}^\pi\widehat{V}_{h+1}(s,a)\right|\\ &=\gamma\left|\mathbb{E}_{s'\sim\widehat{T}_\theta}[\widehat{V}_{h+1}(s')]-\mathbb{E}_{s'\sim T^*}[\widehat{V}_{h+1}(s')]\right|\\ &\leq\frac{\gamma r_{\max}}{1-\gamma}D_{\text{TV}}(\widehat{T}_\theta(s,a),T^*(s,a))\\ &\leq\frac{\gamma r_{\max}}{1-\gamma}\mathcal{U}^{\text{MOPO}}(s,a).\end{aligned}$$

Here the first inequality can be derived from the integral probability metric (IPM) (Müller, 1997) associated with a class $\mathcal{F}=\{f:\|f\|_\infty\leq 1\}$. We refer to Yu et al., 2020 for a detailed explanation. □

Theorem 3.6 shows that the uncertainty quantification in MOPO is theoretically feasible since it is an upper bound of the Bellman error. However, this quantification tends to be too loose to be effective in practice. It first scales the Bellman error to the model error and then estimates the model error. Therefore, it suffers from a loose bound $\mathcal{O}(\frac{1}{1-\gamma})$. In contrast, our proposed Model-Bellman Inconsistency regards the Bellman error as a whole instead of breaking it down and estimates it directly. Then our estimation has a tighter bound $\mathcal{O}(1)$ w.r.t. the horizon, implying that it can obtain a superior policy.

Algorithm 1 MOBILE

Require: Dataset $\mathcal{D}$, learned dyanmics models $\{\widehat{T}_\theta^i\}_{i=1}^N$, initialized policy and critics π_ϕ and $\{Q_{\psi_1}, Q_{\psi_2}\}$.
Train the probabilistic dynamics model $\widehat{T}_\theta\left(s', r \mid s, a\right) = \mathcal{N}\left(\mu_\theta(s,a), \Sigma_\theta(s,a)\right)$ on $\mathcal{D}$.
Initialize the replay buffer $\mathcal{D}_{\text{model}} \leftarrow \varnothing$.
for $i = 1$ **to** N_{iter} **do**
  Generate synthetic h-step rollouts by $\widehat{T}_\theta$. Add transition data to $\mathcal{D}_{\text{model}}$.
  Sample a mini-batch $B = \{s, a, r, s'\}$ from $\mathcal{D} \cup \mathcal{D}_{\text{model}}$.
  Compute targets for B according to Eq. (11) and Eq. (12).
  Update critics ψ_1, ψ_2 with gradient descent via minimizing Eq. (10).
  Update actor ϕ with gradient ascent via Eq. (14).
end for

3.3. Algorithm

We are now ready to present our overall approach in Algorithm 1, which is built upon an off-the-shelf model-based off-policy online RL algorithm, *model-based policy optimization* (MBPO) (Janner et al., 2019). The first step of MOBILE is to pretrain the environment dynamics models. Following MBPO, we learn an ensemble of N dynamics models $\{\widehat{T}_\theta^i = \mathcal{N}(\mu_\theta^i, \Sigma_\theta^i)\}_{i=1}^N$, each of which is a neural network that outputs a Gaussian distribution over the next state and reward and is trained independently via maximum likelihood.

Prior to each agent update, we generate synthetic h-step rollouts starting from states in $\mathcal{D}$ by simulating rollouts in the learned dynamics models $\widehat{T}_\theta$, and then add these transitions to the synthetic dataset $\mathcal{D}_{\text{model}}$. For agent training, we incorporate the MOBIP operator with the off-the-shelf *soft actor-critic* (SAC) algorithm (Haarnoja et al., 2018), as presented in Eq. (10).

$$\mathcal{L}_{\text{critic}} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}\cup\mathcal{D}_{\text{model}}}[(Q_{\psi_k} - y)^2], \tag{10}$$

where the target value for $(s, a, r, s') \in \mathcal{D}$ is

$$y = r + \gamma\left[\min_{k=1,2} Q_{\psi_k^-}(s', a') - \alpha \log \pi_\phi(a'|s')\right], \tag{11}$$

and the target for $(s, a, r, s') \in \mathcal{D}_{\text{model}}$ is

$$y = r + \gamma\left[\min_{k=1,2} Q_{\psi_k^-}(s', a') - \alpha \log \pi_\phi(a'|s')\right] - \beta\mathcal{U}(s,a). \tag{12}$$

Note that we do not penalize targets of real samples since we argue that these samples are unbiased and will not induce Bellman error. Following the formulas of the Model-Bellman Inconsistency in Eq. (4), $\mathcal{U}(s,a)$ is computed by:

$$\begin{aligned}\mathcal{U}(s,a) &= \text{Std}\left(\widehat{\mathcal{T}}_i^\pi Q_\psi(s,a)\right) \\ &= \text{Std}\left(\gamma\mathbb{E}_{\substack{\{s'_j\}^M\sim\widehat{T}_\theta^i \\ \{a'_j\}^M\sim\pi_\phi}}\left[\min_{k=1,2} Q_{\psi_k^-}(s'_j, a'_j)\right]\right).\end{aligned} \tag{13}$$

Here we omit the calculation of rewards due to the following two reasons: one is to be more in line with our theoretical analysis, and the other is that the std of rewards under different models is relatively small compared to that of value, which can be ignored. The policy is then optimized by solving the following optimization problem.

$$\pi_\phi := \max_\phi \mathbb{E}_{\substack{s\sim\mathcal{D}\cup\mathcal{D}_{\text{model}} \\ a\sim\pi_\phi}}\left[\min_{k=1,2} Q_{\psi_k}(s,a) - \alpha\log\pi_\phi(a \mid s)\right]. \tag{14}$$

4. Experiments

In this section, we focus on the following questions: 1) How does MOBILE compare to previous methods in standard offline RL benchmarks? 2) Is Model-Bellman Inconsistency effective in terms of uncertainty quantification? 3) Does Model-Bellman Inconsistency better estimate the Bellman error than uncertainty quantifiers used in other model-based offline RL algorithms?

We explore these questions using the standard D4RL offline RL benchmark (Fu et al., 2020), which includes Gym and Adroit domains, as well as the near-real-world NeoRL (Qin et al., 2022) benchmark.

4.1. Benchmark Results

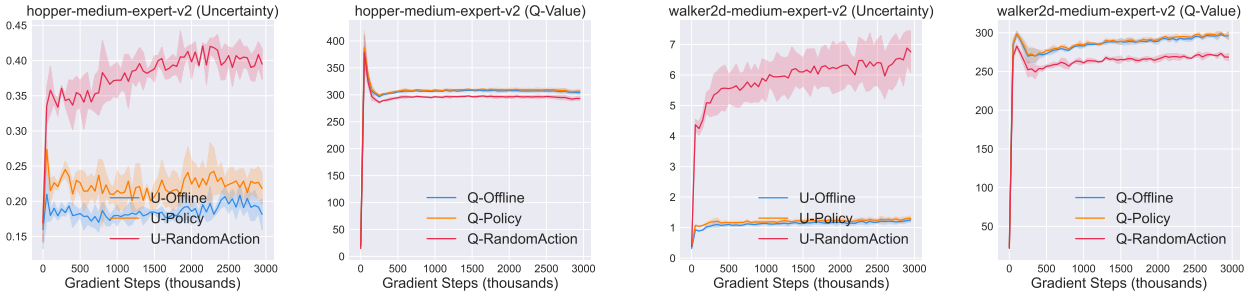
4.1.1. D4RL

We compare MOBILE with several offline RL algorithms, including model-free methods: behavioral cloning (BC) that simply mimics the data collecting policy, CQL (Kumar et al., 2020) that penalizes Q-values on OOD samples equally, TD3+BC (Fujimoto & Gu, 2021) that adopts a BC constraint when optimizing policy, EDAC (An et al., 2021) that quantifies the uncertainty of the Q-values via neural network ensemble; and model-based methods: MOPO (Yu et al., 2020) that uses the uncertainty of the transition prediction as penalty function, COMBO (Yu et al., 2021) that applies the penalty function of CQL within the model-based regime, TT (Janner et al., 2021) that uses a Transformer to model distributions over trajectories and incorporates beam search to plan, RAMBO (Rigter et al., 2022) that trains the policy and the dynamics model adversarially.

For the Gym domain, we evaluate these approaches on a total of twelve datasets involving three environments (hopper, walker2d, halfcheetah) and four dataset types (random,

Table 1. Normalized average returns on D4RL Gym tasks, averaged over 4 random seeds.

Task Name	BC	CQL	TD3+BC	EDAC	MOPO	MOPO*	COMBO	TT	RAMBO	MOBILE (Ours)
halfcheetah-random	2.2	31.3	11.0	28.4	35.4	38.5	38.8	6.1	**39.5**	**39.3±3.0**
hopper-random	3.7	5.3	8.5	25.3	11.7	31.7	17.9	6.9	25.4	**31.9±0.6**
walker2d-random	1.3	5.4	1.6	16.6	13.6	7.4	7.0	5.9	0.0	**17.9±6.6**
halfcheetah-medium	43.2	46.9	48.3	65.9	42.3	73.0	54.2	46.9	**77.9**	74.6±1.2
hopper-medium	54.1	61.9	59.3	101.6	28.0	62.8	97.2	67.4	87.0	**106.6±0.6**
walker2d-medium	70.9	79.5	83.7	**92.5**	17.8	84.1	81.9	81.3	84.9	87.7±1.1
halfcheetah-medium-replay	37.6	45.3	44.6	61.3	53.1	**72.1**	55.1	44.1	68.7	**71.7±1.2**
hopper-medium-replay	16.6	86.3	60.9	101.0	67.5	**103.5**	89.5	99.4	99.5	**103.9±1.0**
walker2d-medium-replay	20.3	76.8	81.8	87.1	39.0	85.6	56.0	82.6	**89.2**	**89.9±1.5**
halfcheetah-medium-expert	44.0	95.0	90.7	106.3	63.3	90.8	90.0	95.0	95.4	**108.2±2.5**
hopper-medium-expert	53.9	96.9	98.0	110.7	23.7	81.6	111.1	110.0	88.2	**112.6±0.2**
walker2d-medium-expert	90.1	109.1	110.1	**114.7**	44.6	112.9	103.3	101.9	56.7	**115.2±0.7**
Average	36.5	61.6	58.2	76.0	36.7	70.3	66.8	62.3	67.7	**80.0**



(a) hopper-medium-expert-v2 (b) walker2d-medium-expert-v2

Figure 1. Illustration of the uncertainty and Q-value for different state-action sets in the training process.

medium, medium-replay, medium-expert) per environment. The datasets we use are of the "v2" version.

Table 1 reports the results in the Gym domain, including the normalized score for each dataset and the average performance over all datasets, which is obtained within the final online evaluation at the end of the training process. We find that MOBILE outperforms all baselines in most of the tasks and achieves the highest average score among all methods. Note that the results of MOPO in Table 1 come from two sources: 1) results presented in the original paper (marked with MOPO); 2) results obtained with our implementation (marked with MOPO*).[1] We refer to Appendix C.3 for more details. Thus, the significant advantages shown by MOBILE against MOPO/MOPO* provide empirical evidence that the Model-Bellman Inconsistency induces better policies than those induced solely by the uncertainty measures from predicting the next states.

Besides, we evaluate these methods in the Adroit domain. The Adroit tasks are more complex and challenging than the Gym tasks in terms of both the dataset composition and high dimensionality. We refer to the results in Appendix D.1.

4.1.2. NeoRL

NeoRL (Qin et al., 2022) is a benchmark that aims to simulate real-world scenarios by collecting datasets with a more conservative policy, which is more in line with real-world data-collection scenarios. The narrow and limited data makes it challenging for offline RL algorithms. Our study focuses on nine datasets, which involve three environments (HalfCheetah-v3, Hopper-v3, Walker2d-v3) and three dataset types (L, M, H) that represent low, medium, and high-quality datasets. It is worth noting that NeoRL provides different numbers of trajectories for training data (100, 1000, 10000) for each task, and we selected 1000 trajectories uniformly for our experiments.

In our evaluation, we compared the performance of our MOBILE algorithm with several baselines (refer to Section 4.1.1), excluding COMBO, TT, and RAMBO. The reason for excluding these algorithms is that there are no available results in their original papers or the NeoRL paper, and determining appropriate hyperparameters for them would be excessively time-consuming. We present the nor-

[1] MOPO and MOPO* differ in two ways: 1) MOPO*'s results are obtained on the "v2" datasets while MOPO's results are obtained on the "v0" datasets. 2) MOPO*'s hyperparameters are re-tuned.

Table 2. Normalized average returns on NeoRL tasks, averaged over 4 random seeds.

Task Name	BC	CQL	TD3+BC	EDAC	MOPO	MOBILE (Ours)
HalfCheetah-L	29.1	38.2	30.0	31.3	40.1	**54.7±3.0**
Hopper-L	15.1	16.0	15.8	**18.3**	6.2	**17.4±3.9**
Walker2d-L	28.5	**44.7**	43.0	40.2	11.6	37.6±2.0
HalfCheetah-M	49.0	54.6	52.3	54.9	62.3	**77.8±1.4**
Hopper-M	51.3	64.5	**70.3**	44.9	1.0	51.1±13.3
Walker2d-M	48.7	57.3	58.5	57.6	39.9	**62.2±1.6**
HalfCheetah-H	71.3	77.4	75.3	81.4	65.9	**83.0±4.6**
Hopper-H	43.1	76.6	75.3	52.5	11.5	**87.8±26.0**
Walker2d-H	72.6	**75.3**	69.6	**75.5**	18.0	**74.9±3.4**
Average	45.4	56.1	54.5	50.7	28.5	**60.7**

malized scores in Table 2.

Our MOBILE algorithm consistently achieves superior or competitive performance across the majority of tasks, as demonstrated by our results. Notably, when comparing the performance evaluated on the D4RL gym tasks, specifically focusing on the improvement achieved on BC, it is evident that most of the baselines experience a decline in performance. In contrast, MOBILE maintains its high level of performance on both the D4RL and NeoRL benchmarks. This remarkable success in the challenging NeoRL benchmark serves as compelling evidence for the potential of our algorithm in real-world scenarios.

4.2. Uncertainty Quantification

Following Bai et al., 2022, we record the Q-values and the uncertainties of different state-action sets in the training process to verify the effectiveness of MOBILE in terms of uncertainty quantification. The sets involve the same states from the offline dataset with different types of actions. The actions we consider include 1) actions from the offline dataset, whose Q-values and uncertainties are labeled as Q-Offline and U-Offline; 2) actions given by the policy learned via MOBILE, whose Q-values and uncertainties are labeled as Q-Policy and U-Policy; and 3) actions uniformly sampled from the action space, whose Q-values and uncertainties are labeled as Q-RandomAction and U-RandomAction.

Figure 1 shows the Q-values and uncertainty in the hopper and walker2d tasks with the "medium-expert" dataset. We can find that MOBILE assigns the largest uncertainty to the random actions and the smallest uncertainty to the actions from the offline dataset, which indicates that MOBILE can correctly quantify the uncertainties of the OOD and in-distribution actions. Secondly, the uncertainties of the actions given by the policy are slightly higher than those of the in-distribution actions, and the Q-values of the policy do not deviate much from the in-distribution actions, which shows that the learned policy avoids choosing actions with high uncertainty with the constraint of uncertainty penalization.

4.3. Bellman Error Estimation

To empirically prove that Model-Bellman Inconsistency better estimates the Bellman error than other uncertainty quantifiers solely considering the dynamics model, we record the uncertainty quantification of state-action pairs generated by the learned dynamics model and compute the corresponding Bellman error with the help of the real environment. Then we calculate the correlation coefficient as a measurement of the quality of the estimation. Other uncertainty quantifiers we consider involve (i) max-aleatoric quantifier, the uncertainty quantifier used in Yu et al., 2020, which corresponds to the maximum aleatoric error; (ii) max-pairwise-diff quantifier, the uncertainty quantifier used in Kidambi et al., 2020, which corresponds to the pairwise maximum difference of the ensemble predictions; (iii) ensemble-std quantifier, the uncertainty quantifier suggested in Lu et al., 2022, which combines the epistemic and aleatoric model uncertainty. We refer to more details about these uncertainty quantifiers in Appendix C.3.

Figure 2 (left) shows the correlation coefficients between the uncertainty quantification of the four quantifiers and the Bellman error. We can find that the Model-Bellman Inconsistency is significantly more relevant with the Bellman error, which means when incorporated with a proper coefficient β, MOBILE can better estimate the true Bellman error than the other three methods. Besides, in order to eliminate the impact of the different magnitude of the four uncertainty quantifiers, we adjust the coefficients of the max-aleatoric, max-pairwise-diff and ensemble-std quantifiers to make these methods yield the same magnitude of uncertainties, and record the uncertainties and corresponding induced policy performance throughout the training process, which are shown by Figure 2 (center) and Figure 2 (right), respectively. We observe that with the same magnitude of uncertainty, MOBILE still significantly outperforms the other three methods, which helps verify that the high per-

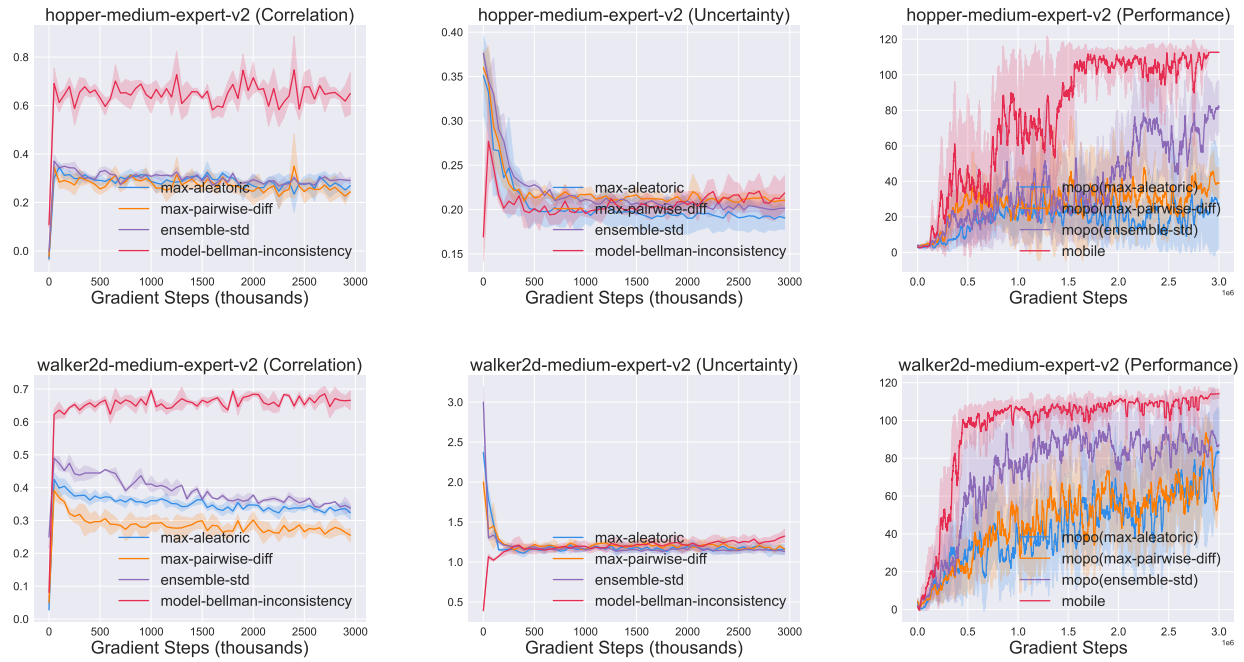


Figure 2. Illustration of the correlation coefficient between different quantifiers and the true Bellman error (left), the uncertainty quantification given by different quantifiers (center), and the corresponding performance of the induced policies (right).

formance of MOBILE benefits from the high correlation between the Model-Bellman Inconsistency and Bellman error rather than the different magnitude of uncertainties.

5. Related Work

Direct application of online off-policy RL algorithms fails in the offline RL setting, mainly due to value overestimation and policy distribution shift. Existing offline RL works differ in trading off conservatism against generalization.

Model-free offline RL. Existing model-free methods typically fall under two categories: *policy constraint* and *value regularization* methods. Policy constraint approaches restrict the learned policy close to the behavior policy to reduce distribution shift. For example, BEAR (Kumar et al., 2019) directly constrains the optimized policy by minimizing MMD distance to behavior policy. BCQ (Fujimoto et al., 2019) limits the action space to actions in the dataset with a learned generative model of the behavior policy. Alternatively, TD3+BC (Fujimoto & Gu, 2021) simply adds a behavioral cloning regularization term to the policy optimization objective and achieves excellent performance across various tasks. In comparison, value regularization methods obtain conservatism via regularization terms in the value optimization objective, which penalizes the value function from being over-optimistic upon OOD actions. CQL (Kumar et al., 2020) penalizes Q-values for all OOD samples equally, while EDAC (An et al., 2021) and PBRL (Bai et al., 2022) assign penalization depending on the uncertainty degree of the Q-value, which is quantified via neural network ensemble.

Model-based offline RL. We focus on Dyna-style model-based RL (Janner et al., 2019; Lin et al., 2022). Dyna-style model-based offline RL methods learn a dynamics model from the dataset and utilize the model to extend the dataset, which can largely improve the data efficiency. Meanwhile, as we can hardly learn a perfect model from the limited dataset, conservatism is still necessary to prevent the policy from generalizing to areas where the prediction of the dynamics model is erroneous. MOPO (Yu et al., 2020) and MOReL (Kidambi et al., 2020) implement conservatism by learning a pessimistic value function from rewards penalized with the uncertainty of the prediction given by the dynamics model. COMBO (Yu et al., 2021) applies CQL in Dyna-style and enforces small Q-values on OOD samples generated by the dynamics model. RAMBO (Rigter et al., 2022) implements conservatism via adversarially training the dynamics model to minimize the value function while keeping accurate on the transition prediction. CBOP (Jeong et al., 2022), proposed concurrently with our method, introduces adaptive weighting of h-step returns in the context of the model-based value expansion (MVE) technique (Feinberg et al., 2018). CBOP also adopts the variance of values under an ensemble of learned models to provide conservative value estimation. However, the difference is that CBOP is under the MVE framework while MOBILE is under the

Dyna-style framework.

Our approach is related to the uncertainty-driven offline RL algorithms, including PBRL (Bai et al., 2022), EDAC (An et al., 2021), and MOPO (Yu et al., 2020). We remark that all these methods can be regarded as practical implementations of a meta-algorithm, namely pessimistic value iteration (PEVI) (Jin et al., 2021). PEVI encourages an algorithm to achieve a smaller Bellman error and to devise a tighter estimation of the Bellman error. Compared to PBRL and EDAC, which are model-free methods, our proposed MOBILE incorporates learned dynamics models to augment the dataset and thus has a smaller Bellman error. MOPO also leverages learned dynamics models, but it suffers from a loose uncertainty quantification that only takes the dynamics model into account. Furthermore, Lu et al. discuss various design choices for reward penalization in MOPO's framework, and have shown that using the ensemble variance or standard deviation is more highly correlated with the true dynamics error and leads to improved empirical performance. However, this approach still suffers from a loose quantification as it does not incorporate the value function. In contrast, MOBILE uses the Model-Bellman Inconsistency as its uncertainty quantifier, which considers both the dynamics model and the value function. As a result, MOBILE achieves a better estimation of the Bellman error.

6. Conclusion

In this paper, we propose MOBILE, an uncertainty-driven model-based offline RL algorithm. To better incorporate conservatism into the model utilization, MOBILE conducts uncertainty quantification through the inconsistency of Bellman estimations under an ensemble of learned dynamics models, which inherently takes into account both the dynamics and the value function. Therefore it can better estimate the ideal uncertainty, i.e., the Bellman error. Through theoretical and empirical analysis, we find that our proposed uncertainty quantification enables a better estimation of the Bellman error compared to the model uncertainty as widely used in previous model-based offline RL work (Yu et al., 2020; Kidambi et al., 2020; Lu et al., 2022). Finally, on the standard D4RL and NeoRL benchmarks, MOBILE generally performs well across different datasets compared to prior offline RL approaches.

Acknowledgements

This work is supported by National Key Research and Development Program of China (2020AAA0107200), the National Science Foundation of China (61921006), and The Major Key Project of PCL (PCL2021A12).

References

An, G., Moon, S., Kim, J., and Song, H. O. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *Advances in Neural Information Processing Systems 34 (NeurIPS'21)*, virtual event, 2021.

Bai, C., Wang, L., Yang, Z., Deng, Z., Garg, A., Liu, P., and Wang, Z. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *The Tenth International Conference on Learning Representations (ICLR'22)*, virtual event, 2022.

Chen, X., Yu, Y., Li, Q., Luo, F., Qin, Z. T., Shang, W., and Ye, J. Offline model-based adaptable policy learning. In *Advances in Neural Information Processing Systems 34 (NeurIPS'21)*, virtual event, 2021.

Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., and Levine, S. Model-based value estimation for efficient model-free reinforcement learning. *CoRR*, abs/1803.00101, 2018. URL http://arxiv.org/abs/1803.00101.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020. URL https://arxiv.org/abs/2004.07219.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems 34 (NeurIPS'21)*, virtual event, 2021.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning (ICML'19)*, Long Beach, USA, 2019.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*, Stockholmsmässan, Sweden, 2018.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems 32 (NeurIPS'19)*, Vancouver, Canada, 2019.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems 34 (NeurIPS'21)*, virtual event, 2021.

Jeong, J., Wang, X., Gimelfarb, M., Kim, H., Abdulhai, B., and Sanner, S. Conservative bayesian model-based value expansion for offline policy optimization.

CoRR, abs/2210.03802, 2022. doi: 10.48550/arXiv.2210.03802. URL https://doi.org/10.48550/arXiv.2210.03802.

Jin, Y., Yang, Z., and Wang, Z. Is pessimism provably efficient for offline rl? In *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*, virtual event, 2021.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, virtual event, 2020.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems 32 (NeurIPS'19)*, Vancouver, Canada, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, virtual event, 2020.

Lange, S., Gabel, T., and Riedmiller, M. A. Batch reinforcement learning. In *Reinforcement Learning*, volume 12, pp. 45–73. 2012.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020. URL https://arxiv.org/abs/2005.01643.

Lin, H., Sun, Y., Zhang, J., and Yu, Y. Model-based reinforcement learning with multi-step plan value estimation. *CoRR*, abs/2209.05530, 2022. URL https://doi.org/10.48550/arXiv.2209.05530.

Lu, C., Ball, P. J., Parker-Holder, J., Osborne, M. A., and Roberts, S. J. Revisiting design choices in offline model based reinforcement learning. In *The Tenth International Conference on Learning Representations (ICLR'22)*, virtual event, 2022.

Luo, Y., Xu, H., Li, Y., Tian, Y., Darrell, T., and Ma, T. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *The Seventh International Conference on Learning Representations (ICLR'19)*, New Orleans, USA, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

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

Qin, R.-J., Zhang, X., Gao, S., Chen, X.-H., Li, Z., Zhang, W., and Yu, Y. NeoRL: A near real-world benchmark for offline reinforcement learning. In *Advances in Neural Information Processing Systems 35 (NeurIPS'22, Datasets and Benchmarks)*, New Orleans, USA, 2022.

Rigter, M., Lacerda, B., and Hawes, N. RAMBO-RL: robust adversarial model-based offline reinforcement learning. *CoRR*, abs/2204.12581, 2022. doi: 10.48550/arXiv.2204.12581. URL https://doi.org/10.48550/arXiv.2204.12581.

Xu, T., Li, Z., and Yu, Y. Error bounds of imitating policies and environments. In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, virtual event, 2020.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. MOPO: model-based offline policy optimization. In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, virtual event, 2020.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. COMBO: conservative offline model-based policy optimization. In *Advances in Neural Information Processing Systems 34 (NeurIPS'21)*, virtual event, 2021.

A. Introduction to Pessimistic Value Iteration

A.1. Background of Pessimistic Value Iteration

In this section, we introduce a meta-algorithm (Algorithm 2) for offline RL, namely pessimistic value iteration (PEVI) (Jin et al., 2021), which constructs an estimated Bellman operator $\widehat{\mathcal{T}}$ based on dataset $\mathcal{D}$ so that $\widehat{\mathcal{T}}\widehat{V}_{h+1} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ approximates $\mathcal{T}\widehat{V}_{h+1} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Here $\widehat{V}_{h+1} : \mathcal{S} \to \mathbb{R}$ is an estimated value function constructed by this meta-algorithm.

Algorithm 2 Pessimistic Value Iteration (PEVI): General MDP

Input: Dataset $\mathcal{D} = \left\{\left(s_h^\tau, a_h^\tau, r_h^\tau, s_{h+1}^\tau\right)\right\}_{\tau,h=1}^{K,H}$.
Initialization: Set $\widehat{V}_{H+1} \leftarrow 0$.
for step $h = H, H-1, ..., 1$ **do**
  Construct Bellman estimation $\widehat{\mathcal{T}}\widehat{V}_{h+1}(\cdot,\cdot)$ and uncertainty quantifier $\Gamma_h(\cdot,\cdot)$.
  Set $\bar{Q}_h(\cdot,\cdot) \leftarrow \widehat{\mathcal{T}}\widehat{V}_{h+1}(\cdot,\cdot) - \Gamma_h(\cdot,\cdot)$.
  Set $\bar{Q}_h(\cdot,\cdot) \leftarrow \min\{\bar{Q}_h(\cdot,\cdot), H-h+1\}^+$.
  $\widehat{\pi}_h(\cdot|\cdot) \leftarrow \arg\max_{\pi_h} \langle \widehat{Q}_h(\cdot,\cdot), \pi_h(\cdot \mid \cdot)\rangle_{\mathcal{A}}$.
  $\widehat{V}_h(\cdot) \leftarrow \langle \widehat{Q}_h(\cdot,\cdot), \widehat{\pi}_h(\cdot \mid \cdot)\rangle_{\mathcal{A}}$.
end for
Output: $\mathrm{Pess}(\mathcal{D}) = \{\widehat{\pi}_h\}_{h=1}^H$.

The penalty function Γ_h plays a key role in Algorithm 2, which guarantees the conservatism of the learned policy. Especially, the $\{\Gamma_h\}_{h=1}^H$ should be a ξ-uncertainty quantifier as follows.

Definition A.1. (ξ-Uncertainty Quantifier (Jin et al., 2021)). The set of penalization $\{\Gamma_h\}_{h\in[H]}$ forms a ξ-uncertainty quantifier if it holds with probability at least 1-ξ that

$$\left|\widehat{\mathcal{T}}\widehat{V}_{h+1}(s,a) - \mathcal{T}\widehat{V}_{h+1}(s,a)\right| \le \Gamma_h(s,a)$$

for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, where $\mathcal{T}$ is the true Bellman operator and $\widehat{\mathcal{T}}$ is the empirical Bellman operator that estimates $\mathcal{T}$.

The ξ-uncertainty quantifier allows us to further characterize the suboptimality of Algorithm 2 by the following theorem.

Theorem A.2. *(Suboptimality of PEVI (Jin et al., 2021)). Suppose $\{\Gamma_h\}_{h=1}^H$ in Algorithm 2 is a ξ-uncertainty quantifier. $\mathrm{Pess}(\mathcal{D})$ in Algorithm 2 satisfies*

$$\mathrm{SubOpt}(\mathrm{Pess}(\mathcal{D}); s) \le 2\sum_{h=1}^{H} \mathbb{E}_{\pi^*}\left[\Gamma_h\left(s_h, a_h\right) \mid s_1 = s\right], \tag{15}$$

with probability at least $1-\xi$ for all $s \in \mathcal{S}$. Here $\mathbb{E}_{\pi^*}$ is with respect to the trajectory induced by the optimal policy π^* in the underlying MDP given the fixed function Γ_h.

Proof. See (Jin et al., 2021) for detailed proof. □

Theorem A.2 suggests finding a ξ-uncertainty quantifier that is sufficiently small to establish an adequately tight upper bound of the suboptimality in Eq. (15).

A.2. Pratical Implementation of PEVI

We can specialize the meta-algorithm (Algorithm 2) by constructing $\widehat{\mathcal{T}}\widehat{V}_{h+1}$ and Γ_h. In model-free case (Bai et al., 2022; An et al., 2021), $\widehat{\mathcal{T}}\widehat{V}_{h+1}$ can be estimated by the state-action value function Q_ψ where the parameter ψ is solved by the Least-Squares Value Iteration (LSVI):

$$\psi = \arg\min_{\psi} \mathbb{E}_{(s_h,a_h,r_h,s_{h+1})\sim\mathcal{D}}\left[\left(Q_\psi(s_h,a_h) - r_h - \gamma\widehat{V}_{h+1}(s_{h+1})\right)^2\right]. \tag{16}$$

In addition, the uncertainty quantification Γ_h can be constructed by the epistemic uncertainty of the Q functions as follows,

$$\mathrm{Std}\left(Q^i_\psi(s,a)\right) = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(Q^i_\psi(s,a) - \bar{Q}_\psi(s,a)\right)^2}, \tag{17}$$

where it maintains N bootstrapped Q-functions $\{Q^i_\psi\}_{i=1}^N$ to quantify the epistemic uncertainty.

However, due to the limited samples in $\mathcal{D}$, the Bellman estimation by Eq. (16) hardly generalizes beyond the state and action support of the offline data. Instead, model-based RL methods make a natural choice for enabling generalization. Specifically, given a dynamics model $\widehat{T}_\theta$ trained on the dataset $\mathcal{D}$, we can directly estimate the true Bellman equation for any state-action pair as follows,

$$\widehat{\mathcal{T}}\widehat{V}_{h+1}(s,a) := r(s,a) + \gamma \mathbb{E}_{s'\sim\widehat{T}_\theta}[\widehat{V}_{h+1}(s')]. \tag{18}$$

Inspired by the uncertainty quantification in model-free case (Eq. (17)), we can use the epistemic uncertainty of $\widehat{\mathcal{T}}\widehat{V}_{h+1}$ as an uncertainty quantification,

$$\mathrm{Std}\left(\widehat{\mathcal{T}}_i\widehat{V}_{h+1}(s,a)\right) = \sqrt{\frac{1}{N}\left(\widehat{\mathcal{T}}_i\widehat{V}_{h+1}(s,a) - \widehat{\mathcal{T}}\widehat{V}_{h+1}(s,a)\right)^2}, \tag{19}$$

where we maintain N dynamics models $\{\widehat{T}^i_\theta\}_{i=1}^N$ and denote $\widehat{\mathcal{T}}_i$ by the Bellman estimation under the i-th dynamics model.

B. Implementation Details

B.1. Dynamics Model Training

In this work, we represent the model as a probabilistic neural network that outputs a Gaussian distribution over the next state and reward given the current state and action:

$$\widehat{T}_\theta\left(s_{t+1}, r_t \mid s_t, a_t\right) = \mathcal{N}\left(\mu_\theta\left(s_t,a_t\right), \Sigma_\theta\left(s_t,a_t\right)\right).$$

We train an ensemble of 7 such dynamics models following (Janner et al., 2019; Yu et al., 2020) and pick the best 5 models based on the validation prediction error on a held-out set that contains 1000 transitions in the offline dataset $\mathcal{D}$. Each model in the ensemble is represented as a 4-layer feedforward neural network with 200 hidden units. During model rollouts, we randomly pick one dynamics model from the best 5 models.

B.2. Policy Optimization

The policy optimization in our method is based on SAC, and most hyperparameters follow its standard implementations. For each update, we sample a batch size of 256 transitions where 5% of them is from the real dataset $\mathcal{D}$ and another 95% is from the synthetic dataset $\mathcal{D}_{\text{model}}$. We use the hyperparameter settings in Table 3 for all the Gym domain and NeoRL domain tasks. We use different settings of networks and the number of critics for the experiment in the Adroit domain. See D.1 for the setup of Adroit.

C. Experimental Details

C.1. Benchmarks

We conduct experiments on Gym tasks (v2 version) and Adroit tasks (v1 version), which are included in the D4RL (Fu et al., 2020) benchmark. In addition, we challenge NeoRL (Qin et al., 2022), a near real-world benchmark, to better evaluate offline RL algorithms on real-world tasks. As opposed to other benchmarks, NeoRL adopts a more conservative policy for data-collection, which is more in line with real-world scenarios.

We now introduce the sources of our reported performance on these benchmarks.

D4RL. (i) For MOPO (Yu et al., 2020) and CQL (Kumar et al., 2020), we use our own codebase github.com/yihaosun1124/OfflineRL-Kit and the official implementation github.com/aviralkumar2907/CQL to retrain

Table 3. Hyperparameters of Policy Optimization in MOBILE.

Hyperparameters	Value	Description
K	2	The number of critics.
Policy network	FC(256,256)	Fully Connected (FC) layers with ReLU activations.
Q-network	FC(256,256)	Fully Connected (FC) layers with ReLU activations.
τ	$5e-3$	Target network smoothing coefficient.
γ	0.99	Discount factor.
lr of actor	$1e-4$	Policy learning rate.
lr of critic	$3e-4$	Critic learning rate.
Optimizer	Adam	Optimizers of the actor and critics.
f	0.05	Ratio of the real samples.
Batch size	256	Batch size for each update.
N_{iter}	3M	Total gradient steps.

them on the "v2" datasets. (ii) For TD3+BC (Fujimoto & Gu, 2021), EDAC (An et al., 2021), TT (Janner et al., 2021), and RAMBO (Rigter et al., 2022), since their original papers report the performance of Gym on the "v2" datasets, we directly cite the reported scores in Table 1. (iii) COMBO (Yu et al., 2021) does not provide source codes. Therefore we include the results from the original paper.

NeoRL. We report the performance of BC, CQL, and MOPO based on the original paper of NeoRL and retrain TD3+BC and EDAC with their official implementations (refer to Appendix D.2).

Table 4. Hyperparameters of MOBILE.

Domain Name	Task Name	β	h
Gym	halfcheetah-random	0.5	5
	hopper-random	0.1	5
	walker2d-random	2.0	5
	halfcheetah-medium	0.5	5
	hopper-medium	1.5	5
	walker2d-medium	1.0	5
	halfcheetah-medium-replay	0.5	5
	hopper-medium-replay	0.1	5
	walker2d-medium-replay	0.5	1
	halfcheetah-medium-expert	1.0	5
	hopper-medium-expert	1.5	5
	walker2d-medium-expert	1.5	1
Adroit	pen-human	1.0	1
	door-human	3.0	3
	hammer-human	0.5	3
	pen-cloned	0.5	1
	door-cloned	0.5	3
	hammer-cloned	3.0	3
NeoRL	HalfCheetah-L	0.5	5
	Hopper-L	2.5	5
	Walker2d-L	2.5	1
	HalfCheetah-M	0.5	5
	Hopper-M	1.5	5
	Walker2d-M	2.5	1
	HalfCheetah-H	1.5	5
	Hopper-H	2.5	5
	Walker2d-H	2.5	1

C.2. Hyperparameters

We list the hyperparameters we have tuned as follows.

Penalty coefficient β. For Gym tasks, we tune β in the range of $\{0.1, 0.5, 1.0, 1.5\}$ (except for walker2d-random since we find that a larger coefficient works well in it). For Adroit tasks, we sweep the parameters on $\beta \in \{0.5, 1.0, 3.0\}$. For NeoRL

tasks, we search $\beta \in \{0.5, 1.5, 2.5\}$.

Rollout length h. We perform short-horizon branch rollouts in MOBILE, similar to MOPO. Specifically, we tune h in the range of $\{1, 5\}$ for Gym and NeoRL tasks. For Adroit tasks, we sweep the parameters on $h \in \{1, 3\}$.

The selected hyperparameters on each task are listed in Table 4.

C.3. Tuning for MOPO

We note that the original paper of MOPO conducts experiments on the old version of the D4RL benchmark. Therefore some of its original hyperparameters (e.g., penalty coefficient and rollout length) are no longer suitable for the tasks evaluated in our paper. In addition, we include different uncertainty quantifiers to implement MOPO, from recent works in offline MBRL:

Max Aleatoric (Yu et al., 2020): $\max_{i=1,\dots,N} ||\Sigma^i_\phi(s,a)||_F$, which corresponds to the maximum aleatoric error.

Max Pairwise Diff (Kidambi et al., 2020): $\max_{i,j} ||\mu^i_\phi(s,a) - \mu^j_\phi(s,a)||_2$, which corresponds to the pairwise maximum difference of the ensemble predictions.

Ensemble-std (Lu et al., 2022): $\Sigma^*(s,a) = \frac{1}{N}\sum_i^N((\Sigma^i_\phi(s,a))^2 + (\mu^i_\phi(s,a))^2) - (\mu^*(s,a))^2$ where $\mu^*(s,a) = \frac{1}{N}\sum_i^N \mu^i_\phi(s,a)$. This corresponds to a combination of epistemic and aleatoric model uncertainty.

For a fair comparison, we only tune the penalty coefficient and the rollout length for these variants of MOPO and keep the other hyperparameters the same as ours. For MOPO (max-aleatoric) and MOPO (max-pairwise-diff), we tune the penalty coefficient β in the range of $\{0.5, 2.5, 5.0\}$. For MOPO (ensemble-std) we sweep the parameters on $\beta \in \{2.0, 10.0, 20.0\}$. All these three variants search on the rollout length h within $\{1, 5\}$. The selected hyperparameters of different variants on each task are listed in Table 5.

Table 5. Hyperparameters of different variants of MOPO used in the D4RL datasets.

Task Name	**max-aleatoric**		**max-pairwise-diff**		**ensemble-std**	
	β	h	β	h	β	h
halfcheetah-random	0.5	5	0.5	5	2.0	5
hopper-random	5.0	5	2.5	5	10.0	5
walker2d-random	0.5	1	0.5	1	2.0	1
halfcheetah-medium	0.5	5	0.5	5	2.0	5
hopper-medium	5.0	5	2.5	5	10.0	5
walker2d-medium	0.5	5	2.5	5	2.0	5
halfcheetah-medium-replay	0.5	5	0.5	5	10.0	5
hopper-medium-replay	2.5	5	0.5	5	10.0	5
walker2d-medium-replay	2.5	1	2.5	1	10.0	1
halfcheetah-medium-expert	2.5	5	2.5	5	10.0	5
hopper-medium-expert	5.0	5	5.0	5	10.0	5
walker2d-medium-expert	2.5	1	5.0	1	20.0	1

We present the performance of these variants of MOPO in Table 6 and report the best result for each task in Table 1 (marked with MOPO*).

C.4. Computational Cost Comparison

To perform a computational cost comparison, we measure the runtime per epoch (1K gradient steps) and the number of parameters for each algorithm based on the hopper-medium-v2 task. All the experiments are run with a single GeForce GTX 3070 GPU and an AMD Ryzen 5900X CPU at 4.8GHz. We summarize the results in Table 7. As the results show, our approach has almost the same computational cost as MOPO, which is more efficient than the other two algorithms. CQL takes more runtime due to the additional computations for Q-value regularization, and EDAC has much more parameters due to a large number of ensemble Q networks.

Table 6. Normalized average returns on D4RL Gym tasks for different variants of MOPO, averaged over 4 random seeds.

Task Name	**max-aleatoric**	**max-pairwise-diff**	**ensemble-std**
halfcheetah-random	37.3	38.0	38.5
hopper-random	31.7	14.3	10.2
walker2d-random	4.1	7.4	3.0
halfcheetah-medium	72.4	71.1	73.0
hopper-medium	62.8	33.0	62.5
walker2d-medium	84.1	83.4	79.2
halfcheetah-medium-replay	72.1	67.0	68.1
hopper-medium-replay	92.8	100.4	103.5
walker2d-medium-replay	85.2	83.0	85.6
halfcheetah-medium-expert	83.6	83.7	90.8
hopper-medium-expert	74.9	73.8	81.6
walker2d-medium-expert	105.3	112.3	112.9
Average	67.2	64.0	67.4

Table 7. Comparison of computational costs.

	Runtime (s/epoch)	Number of parameters
CQL	12	0.7M
EDAC	10	13.7M
MOPO	7	2.2M
MOBILE	8	2.2M

D. Omitted Experiments

D.1. Experiments in Adroit Domain

Adroit is a more challenging task that involves controlling a 24-DoF simulated robotic hand that aims at hammering a nail, opening a door, twirling a pen, or picking moving a ball. We select two types of datasets: "human", containing 25 trajectories of human demonstrations, and "cloned", a 50-50 mixture between the demonstration data and the behavioral cloned policy on the demonstration.

Since the observation spaces and action spaces of these tasks are more complex than those of MuJoCo Gym tasks, we adopt deeper actor and critic networks which contain 3 hidden layers, i.e., FC(256, 256, 256) as well as a wider dynamics model, i.e., FC(400, 400, 400, 400), and set the learning rate of actor to $3e-5$. Similar to EDAC (An et al., 2021), we adopt max Q backup from CQL (Kumar et al., 2020) and normalize the rewards for training stability. Meanwhile, following the recommendation of EDAC, we choose to use early-stopping and train each algorithm for 200,000 steps.[2] In addition, we increase the number of critics from 2 to 10 for better performance. We present the normalized scores of different algorithms in Table 8.

In Table 8, the scores of BC, CQL, and EDAC are adopted from the results reported in EDAC. We retrain MOPO on our own codebase and keep its hyperparameters the same as ours except for the penalty coefficient β. For TD3+BC, we attempt different BC weights but can not get reasonable performance. We do not include COMBO, TT, and RAMBO since they do not conduct experiments in the Adroit domain in their original papers and finding proper hyperparameters for these algorithms is very time-consuming.

We find that MOBILE achieves the best performance on the "cloned" datasets and underperforms EDAC and CQL on the "human" datasets. We hypothesize that the small amount of samples in the "human" datasets (only 5000 transitions) makes it more difficult to learn a model that generalizes well. Fortunately, this setting is one in which model-free methods can perform well, suggesting that model-based and model-free approaches are able to perform well in complementary settings.

[2]In EDAC, they find that the performance of CQL and EDAC degrades after about 200,000 steps. Therefore they choose to use early-stopping and train each algorithm for 200,000 steps.

Table 8. Normalized average returns on D4RL Adroit tasks, averaged over 4 random seeds.

Task Name	BC	CQL	TD3+BC	EDAC	MOPO	MOBILE (Ours)
pen-human	25.8	35.2	-1.0	**52.1**	10.7	30.1±14.6
door-human	2.8	9.1	-0.2	**10.7**	-0.2	-0.2±0.1
hammer-human	**3.1**	0.6	0.2	0.8	0.3	0.4±0.2
pen-cloned	38.3	27.2	-2.1	68.2	54.6	**69.0±9.3**
door-cloned	0.0	3.5	0.0	9.6	15.3	**24.0±22.8**
hammer-cloned	0.7	1.4	-0.1	0.3	0.5	**1.5±0.4**

D.2. Hyperparameters for TD3+BC and EDAC in NeoRL

We retrain TD3+BC (Fujimoto & Gu, 2021) and EDAC (An et al., 2021) with their official implementations for the NeoRL benchmark (Qin et al., 2022). We present the selected hyperparameters in Table 9.

Table 9. Hyperparameters for TD3+BC and EDAC in the NeoRL datasets.

Task Name	TD3+BC	EDAC	
	λ	N	η
HalfCheetah-L	2.5	10	0.0
Hopper-L	2.5	50	0.0
Walker2d-L	2.5	10	1.0
HalfCheetah-M	2.5	10	1.0
Hopper-M	2.5	50	1.0
Walker2d-M	2.5	10	1.0
HalfCheetah-H	2.5	10	5.0
Hopper-H	2.5	50	1.0
Walker2d-H	2.5	10	5.0

D.3. MOBILE with Larger Ensemble

The supervised learning literature suggests that a higher number of models in the ensemble can improve the quality of the uncertainty estimate, which could be an effective improvement to our algorithm. We therefore try to increase the number of models in the ensemble from 7 to 30 and re-evaluate our algorithm on several tasks. The results are listed below:

Table 10. Normalized average returns on D4RL Gym tasks, averaged over 4 random seeds.

Task Name	MOBILE (7 ensemble)	MOBILE (30 ensemble)
walker2d-medium	87.7	92.4
halfcheetah-medium-replay	71.7	74.4
hopper-medium-replay	103.9	104.9
walker2d-medium-replay	89.9	94.1

D.4. Omitted Uncertainty Quantification in Section 4.2

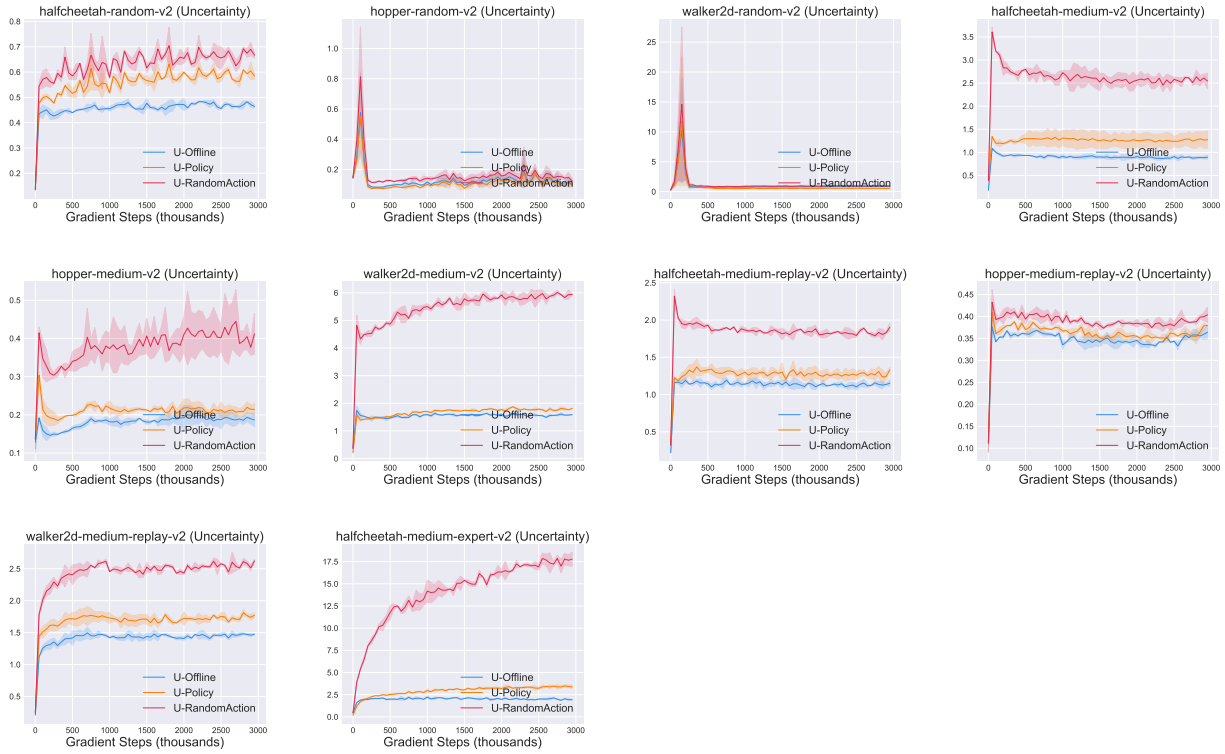


Figure 3. Illustration of the uncertainty for different state-action sets in the training process.

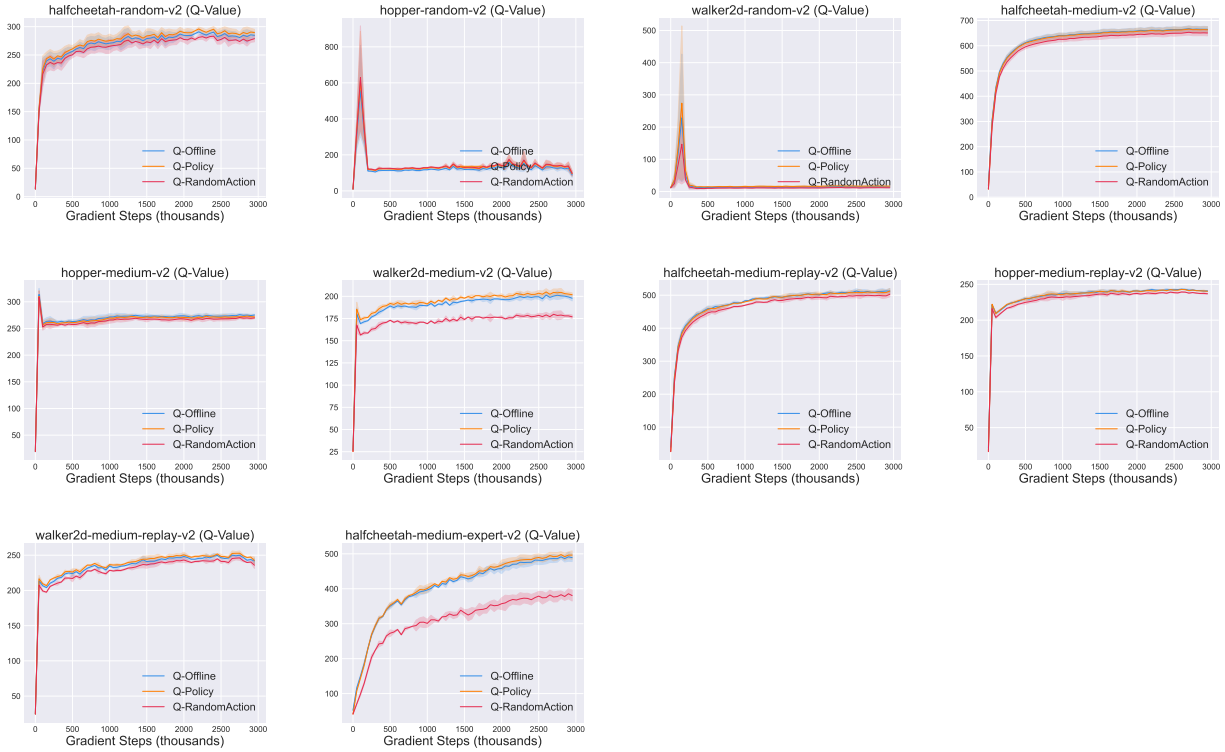


Figure 4. Illustration of the Q-value for different state-action sets in the training process.

D.5. Omitted Bellman Error Estimation in Section 4.3

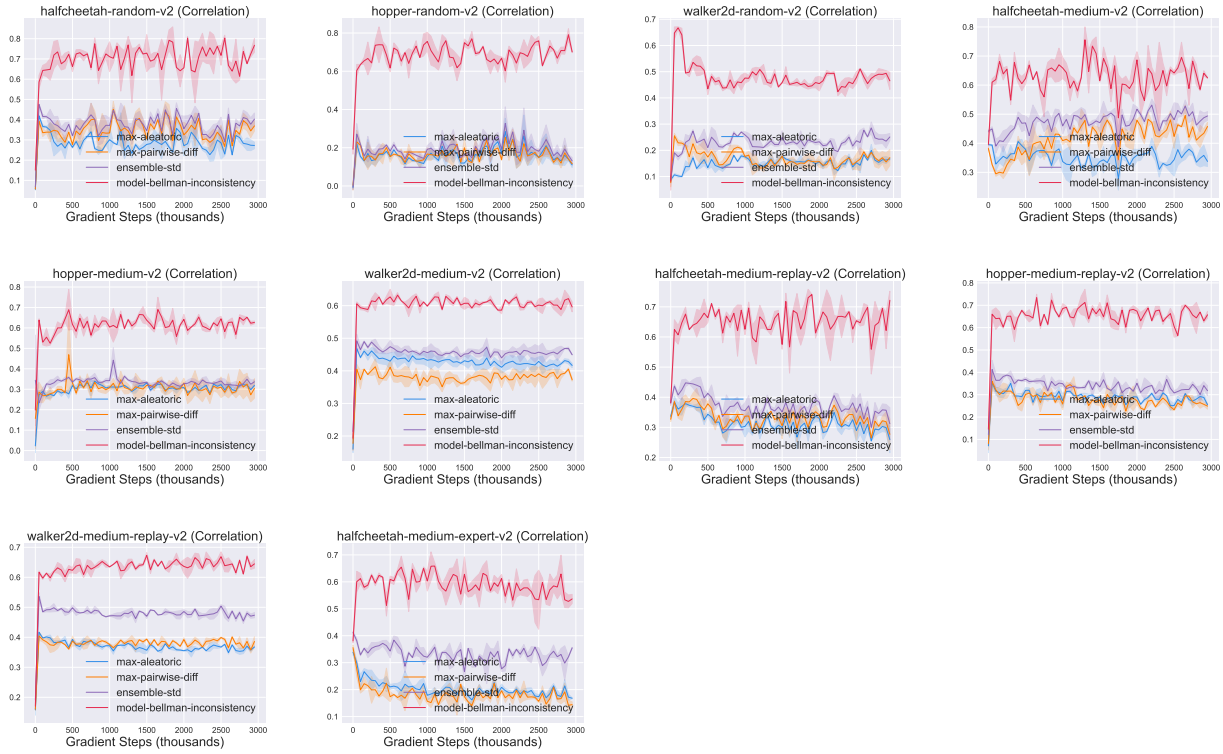


Figure 5. Illustration of the correlation coefficient between different quantifiers and the true Bellman error.