A Unified Framework for Alternating Offline Model Training and Policy Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

In offline model-based reinforcement learning (offline MBRL), we learn a dynamic model from historically collected data, and then utilize the learned model and fixed dataset for policy learning, without further interacting with the environment. Offline MBRL algorithms can improve the efficiency and stability of policy learning over the model-free based algorithms. However, in most of the existing offline MBRL algorithms, the learning objectives for the dynamic models and the policies are isolated from each other. Such an *objective mismatch* issue may lead to inferior performance of the learned agents. In this paper, we address the issue by developing an iterative offline MBRL framework, where we maximize a lower bound of the true expected return, by alternating between dynamic model training and policy learning. With the proposed unified model-policy learning framework, we achieve competitive performance on a wide range of continuous control offline reinforcement learning datasets.

1 Introduction

Offline reinforcement learning (offline RL) [1, 2], where the agents are trained from static and pre-collected datasets, avoids direct interactions with the real-world underlying environment during the learning process. Unlike traditional online RL, whose success largely depends on simulator-based trial-and-error [*e.g.*, 3, 4], one can train policies for real-world applications in offline RL, where it is infeasible or even risky to collect online experimental data, such as robotics, advertisement, or dialog systems [*e.g.*, 5–7]. Though promising, it remains challenging to train agents under the offline setting, due to the discrepancy between the distribution of offline dataset and the state-action distribution induced by the current learning policy. With such discrepancy, directly transferring standard online off-policy RL methods [*e.g.*, 8–10] to the offline setting tends to be problematic [11, 12], especially when the offline data cannot sufficiently cover the state-action space [13]. To tackle this discrepancy issue in offline RL, recent works [*e.g.*, 14–16] propose to approximate the policy induced state-action distribution by leveraging a learned dynamic model to draw imaginary rollouts. These additional synthetic rollouts help mitigate the distribution discrepancy and stabilize the policy-learning algorithms under the offline setting.

Most of the prior offline model-based RL (MBRL) methods [*e.g.*, 15, 17–20], however, first *pretrain* a one-step forward dynamic model via maximum likelihood estimation (MLE) on the offline dataset, and then use the learned model to train the policy, without *further improving the dynamic model* during the policy learning process. As a result, the objective function used for model training (*e.g.*, MLE) and the objective of model utilization are *unrelated* with each other. Specifically, the model is trained to be "simply a mimic of the world," but is used to improve the performance of the learned policy [21–23]. Though such a training paradigm is historically rooted [24, 25], this issue of *objective mismatch* in the model training and model utilization has been identified as problematic in recent

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

works [22, 23]. In offline MBRL, this issue is exacerbated, since the learned model can hardly be globally accurate, due to the limited amount of offline data and the complexity of the control tasks.

Motivated by the objective-mismatch issue, we develop an iterative offline MBRL method, alternating between training the dynamic model and the policy to maximize a lower bound of the true expected return. This lower bound, leading to a weighted MLE objective for dynamic-model training, is relaxed to a tractable regularized objective for policy learning. To train the dynamic model with the new objective, we need to estimate the marginal importance weights (MIW) between the offline-data distribution and stationary state-action distribution of the current policy [26, 27]. This estimation tends to be unstable by standard approaches [*e.g.*, 28, 29], which require saddle-point optimization. Instead, we propose a simple yet stable iterative method for MIW estimation, which can be directly incorporated into our alternating training framework. With these considerations, our method performs competitively on a wide range of continuous-control offline RL datasets in the D4RL benchmark [30]. These empirical results and ablation study show the efficacy of our proposed algorithmic designs.

2 Background

Markov decision processes and offline RL. A Markov decision process (MDP) is denoted by $\mathcal{M} = (\mathbb{S}, \mathbb{A}, P, r, \gamma, \mu_0)$, where $\mathbb{S}$ is the state space, $\mathbb{A}$ the action space, $P\left(s' \,|\, s, a\right) : \mathbb{S} \times \mathbb{S} \times \mathbb{A} \to [0, 1]$ the environmental dynamic, $r(s, a) : \mathbb{S} \times \mathbb{A} \to [-r_{max}, r_{max}]$ the reward function, $\gamma \in (0, 1)$ the discount factor, and $\mu_0(s) : \mathbb{S} \to [0, 1]$ the initial state-distribution.

For any policy $\pi(a \,|\, s)$, we denote its state-action distribution at timestep $t \geq 0$ as $d^P_{\pi,t}(s, a) \triangleq \Pr\left(s_t = s, a_t = a \,|\, s_0 \sim \mu_0, a_t \sim \pi, s_{t+1} \sim P, \forall\, t \geq 0\right)$. The (discounted) stationary state-action distribution of π is denoted as $d^P_{\pi,\gamma}(s, a) \triangleq (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t d^P_{\pi,t}(s, a)$.

Denote $Q^P_\pi(s, a) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,|\, s_0 = s, a_0 = a\right]$ as the action-value function of policy π under the dynamic P. The goal of RL is to find a policy π maximizing the expected return

$$J(\pi, P) \triangleq (1 - \gamma)\mathbb{E}_{s\sim\mu_0, a\sim\pi(\cdot\,|\,s)}\left[Q^P_\pi(s, a)\right] = \mathbb{E}_{(s,a)\sim d^P_{\pi,\gamma}}\left[r(s, a)\right]. \tag{1}$$

In offline RL, the policy π and critic Q^P_π are typically approximated by parametric functions $\pi_{\boldsymbol{\phi}}$ and $Q_{\boldsymbol{\theta}}$, respectively, with parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. The critic $Q_{\boldsymbol{\theta}}$ is trained by Bellman backup:

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_{\text{env}}}\left[\left(Q_{\boldsymbol{\theta}}(s, a) - \left(r(s, a) + \gamma \mathbb{E}_{a'\sim\pi_{\boldsymbol{\phi}}(\cdot\,|\,s')}\left[Q_{\boldsymbol{\theta}'}(s', a')\right]\right)\right)^2\right], \tag{2}$$

where $Q_{\boldsymbol{\theta}'}$ is the target network [11, 12]. The actor $\pi_{\boldsymbol{\phi}}$ is trained in the policy improvement step:

$$\arg\max_{\boldsymbol{\phi}} \mathbb{E}_{s\sim\mathcal{D}_{\text{env}},\, a\sim\pi_{\phi}(\cdot\,|\,s)}\left[Q_{\boldsymbol{\theta}}\left(s, a\right)\right], \tag{3}$$

where $\mathcal{D}_{\text{env}}$ denotes the offline dataset drawn from $d^P_{\pi_b,\gamma}$ [2, 31], with π_b being the behavior policy.

Offline model-based RL. In offline model-based RL algorithms, the true environmental dynamic P^* is typically approximated by a parametric function $\widehat{P}\left(s' \,|\, s, a\right)$ in some function class $\mathcal{P}$. With the offline dataset $\mathcal{D}_{\text{env}}$, $\widehat{P}$ is trained via the MLE [14, 15, 17] as

$$\arg\max_{\widehat{P}\in\mathcal{P}} \mathbb{E}_{(s,a,s')\sim\mathcal{D}_{\text{env}}}\left[\log \widehat{P}\left(s' \,|\, s, a\right)\right]. \tag{4}$$

Similarly, the reward function can be approximated by a parametric model $\widehat{r}$ if assumed unknown. With $\widehat{P}$ and $\widehat{r}$, the true MDP $\mathcal{M}$ can be approximated by $\widehat{\mathcal{M}} = (\mathbb{S}, \mathbb{A}, \widehat{P}, \widehat{r}, \gamma, \mu_0)$. We further define $d^{P^*}_{\pi,\gamma}(s, a)$ as the stationary state-action distribution induced by π on P^* (or MDP $\mathcal{M}$), and $d^{\widehat{P}}_{\pi,\gamma}(s, a)$ as that on the learned dynamic $\widehat{P}$ (or MDP $\widehat{\mathcal{M}}$). We approximate $d^{P^*}_{\pi_{\boldsymbol{\phi}},\gamma}$ by simulating $\pi_{\boldsymbol{\phi}}$ on $\widehat{\mathcal{M}}$ for a short horizon h starting from state $s \in \mathcal{D}_{\text{env}}$, as in prior work [*e.g.*, 15, 17, 18, 20]. The resulting transitions are stored in a replay buffer $\mathcal{D}_{\text{model}}$, constructed similar to the off-policy RL [32, 8]. To better approximate $d^{P^*}_{\pi_{\boldsymbol{\phi}},\gamma}$, sampling from $\mathcal{D}_{\text{env}}$ in Eqs. (2) and (3) is commonly replaced by sampling from the augmented dataset $\mathcal{D} = f\mathcal{D}_{\text{env}} + (1 - f)\mathcal{D}_{\text{model}}$, $f \in [0, 1]$, denoting sampling from $\mathcal{D}_{\text{env}}$ and $\mathcal{D}_{\text{model}}$ with probabilities f and $1 - f$, respectively. We follow Yu et al. [17] to use $f = 0.5$.

3 Main method

Our goal is to derive the objectives for both dynamic model training and policy learning from a principled perspective. A natural idea is to build a tractable lower bound for $J(\pi, P^*)$, the expected return of the policy π under the true dynamic P^*, and then alternate between training the policy π and dynamic model $\widehat{P}$ to maximize the lower bound. Indeed, we can construct a lower bound as

$$J(\pi, P^*) \geq J(\pi, \widehat{P}) - \left| J(\pi, P^*) - J(\pi, \widehat{P}) \right|, \tag{5}$$

where $J(\pi, \widehat{P})$ is the expected return of policy π under the learned model $\widehat{P}$. From the right hand side (RHS) of Eq. (5), if the policy evaluation error $|J(\pi, P^*) - J(\pi, \widehat{P})|$ is small, $J(\pi, \widehat{P})$ will be a good proxy for the true expected return $J(\pi, P^*)$. We can empirically estimate $J(\pi, \widehat{P})$ via $\widehat{P}$ and π.

Further, if a tractable upper bound for $|J(\pi, P^*) - J(\pi, \widehat{P})|$ can be constructed, it can serve as a unified training objective for both dynamic model $\widehat{P}$ and policy π. We can then alternate between optimizing the dynamic model $\widehat{P}$ and policy π to maximize the lower bound of $J(\pi, P^*)$, *i.e.*, simultaneously minimizing the upper bound of the evaluation error $|J(\pi, P^*) - J(\pi, \widehat{P})|$. This gives us an *iterative, maximization-maximization* algorithm for model and policy learning.

The following theorem indicates a tractable upper bound for $|J(\pi, P^*) - J(\pi, \widehat{P})|$, which can be subsequently relaxed for model training and policy learning.

Theorem 1. *Let P^* be the true dynamic and $\widehat{P}$ be the approximated dynamic model. Suppose the reward function $|r(s,a)| \leq r_{\max}$, then we have*

$$\left| J(\pi, P^*) - J(\pi, \widehat{P}) \right| \leq \frac{\gamma \cdot r_{\max}}{\sqrt{2}(1-\gamma)} \cdot \sqrt{D_\pi(P^*, \widehat{P})},$$

$$\textit{with} \quad D_\pi(P^*, \widehat{P}) \triangleq \mathbb{E}_{(s,a)\sim d^{P^*}_{\pi_b,\gamma}} \left[\omega(s,a) \mathrm{KL}\left(P^*(s' \mid s,a)\pi_b(a' \mid s') \,||\, \widehat{P}(s' \mid s,a)\pi(a' \mid s') \right) \right],$$

where π_b is the behavior policy, $d^{P^*}_{\pi_b,\gamma}$ is the offline-data distribution, and $\omega(s,a) \triangleq \frac{d^{P^*}_{\pi,\gamma}(s,a)}{d^{P^*}_{\pi_b,\gamma}(s,a)}$ is the marginal importance weight (MIW) between the offline-data distribution and the stationary state-action distribution of policy π [26, 28].

Detailed proof of Theorem 1 can be found in Appendix B.2.

The KL term in $D_\pi(P^*, \widehat{P})$ indicates the following two principles for model and policy learning:

⋆ For the dynamic model, $s' \sim \widehat{P}(\cdot \mid s,a)$ should be close to the true next state $\tilde{s}' \sim P^*(\cdot \mid s,a)$, with (s,a) pairs drawn from the stationary state-action distribution of policy π. Since we can not directly draw samples from $d^{P^*}_{\pi,\gamma}$, we reweight the offline data with $\omega(s,a)$. This leads to a weighted KL minimization objective for the model training.

⋆ For the policy π, the KL term indicates a regularization term, that the tuple (s', a') from the joint conditional distribution $\widehat{P}(s' \mid s,a)\pi(a' \mid s')$ should be close to the tuple $(\tilde{s}', \tilde{a}')$ from $P^*(\tilde{s}' \mid s,a)\pi_b(\tilde{a}' \mid \tilde{s}')$. $(\tilde{s}', \tilde{a}')$ is just the sample from the offline dataset.

Based on the above observations, we can fixed π to train the dynamic model $\widehat{P}$ by minimizing $D_\pi(P^*, \widehat{P})$. Similarly, we can fix the dynamic model $\widehat{P}$ and learn the policy π to maximize the lower bound of $J(\pi, P^*)$. This alternating training scheme provides a unified approach for model and policy learning. In the following section, we discuss how to optimize the dynamic model $\widehat{P}$, the policy π , and the MIW ω under our alternating training farmework.

3.1 Dynamic model training

Expanding the KL term in $D_\pi(P^*, \widehat{P})$, we have

$$D_\pi(P^*, \widehat{P}) = \overbrace{\mathbb{E}_{(s,a,s',a')\sim d^{P^*}_{\pi_b,\gamma}} \left[\omega(s,a) \left(\log P^*(s' \mid s,a) + \log \pi_b(a' \mid s') - \log \pi\left(a' \mid s'\right) \right) \right]}^{\triangleq ①}$$
$$- \mathbb{E}_{(s,a,s')\sim d^{P^*}_{\pi_b,\gamma}} \left[\omega(s,a) \log \widehat{P}(s' \mid s,a) \right],$$

where the tuple (s, a, s', a') is simply two consecutive state-action pairs in the offline dataset. Further, if the policy π is fixed, the term 1 is a constant with respect to $\widehat{P}$. Thus, given ω, we can optimize $\widehat{P}$ by minimizing the following loss

$$\ell(\widehat{P}) \triangleq -\mathbb{E}_{(s,a,s')\sim d^{P^*}_{\pi_b,\gamma}} \left[\omega(s,a) \log \widehat{P}(s' \mid s, a)\right], \tag{6}$$

which is an MLE objective weighted by $\omega(s, a)$. We discuss how to estimate $\omega(s, a)$ in Section 3.3.

3.2 Policy learning

The lower bound for $J(\pi, P^*)$ implied by Theorem 1 is

$$J(\pi, \widehat{P}) - \tfrac{\gamma \cdot r_{\max}}{\sqrt{2}(1-\gamma)} \cdot \sqrt{D_\pi(P^*, \widehat{P})}\,, \tag{7}$$

where $J(\pi, \widehat{P})$ can be estimated via the action-value function similar to standard offline MBRL algorithms [*e.g.*, 15, 17, 18]. Thus, when the dynamic model $\widehat{P}$ is fixed, the main difficulty is to estimate the regularizer $D_\pi(P^*, \widehat{P})$ for policy π.

When the policy π is Gaussian, direct estimation of $D_\pi(P^*, \widehat{P})$ is possible. Empirically, however, it is helpful to learn the policy π in the class of implicit distribution, which is a richer distribution class and can better maximize the action-value function. Specifically, given a noise distribution $p_z(z)$, action $a = \pi_{\boldsymbol{\phi}}(s, z)$ with $z \sim p_z(\cdot)$, where $\pi_{\boldsymbol{\phi}}$ is a deterministic network.

Unfortunately, we can not directly estimate the KL term in $D_\pi(P^*, \widehat{P})$ if π is an implicit policy, since we can only draw samples from π and the density is unknown. A potential solution is to use the dual representation of KL divergence $\mathrm{KL}(p \,||\, q) = \sup_T \mathbb{E}_p[T] - \log(\mathbb{E}_q[e^T])$ [33], which can be estimated with samples from the distributions p and q. However, the exponential function therein makes the estimation unstable in practice [29]. We instead use the dual representation of Jensen–Shannon divergence (JSD) to approximate $D_\pi(P^*, \widehat{P})$, which can be approximately minimized using the GAN structure [34]. Our framework can thus utilize the many stabilization techniques developed in the GAN community (Appendix D.2.1).

Besides, we remove the MIW $w(s, a)$ during the policy training since we do not observe its empirical benefits, which will be discussed in Section 4.2. Further applying the replacement of KL with JSD and ignoring the $\sqrt{\cdot}$ for numerical stability, we get an *approximated new regularization* for policy π:

$$\widetilde{D}_\pi(P^*, \widehat{P}) \triangleq \mathrm{JSD}\left(P^*(s' \mid s, a)\pi_b(a' \mid s') d^{P^*}_{\pi_b,\gamma}(s, a) \,||\, \widehat{P}(s' \mid s, a)\pi(a' \mid s') d^{P^*}_{\pi_b,\gamma}(s)\pi(a \mid s)\right). \tag{8}$$

Informally speaking, Eq. (8) regularizes the imaginary rollouts of π on $\widehat{P}$ towards state-action pairs from the offline dataset. Intuitively, $\widetilde{D}_\pi(P^*, \widehat{P})$ is a more effective regularizer for policy training than the original $D_\pi(P^*, \widehat{P})$, since $\widetilde{D}_\pi(P^*, \widehat{P})$ regularizes action choices at both s and s'. Appendix B.3 discusses how we change from $D_\pi(P^*, \widehat{P})$ to $\widetilde{D}_\pi(P^*, \widehat{P})$ in detail.

3.3 Marginal importance weight training

A number of recent works have been proposed to estimate the marginal importance weight ω [26, 28, 29]. These methods typically require to solve complex saddle-point optimization, casting doubts on their training stability especially when combined with policy learning on continuous-control offline MBRL problems. In this section, we mimic the Bellman backup to derive an iterative method for estimating the MIW.

Define the true MIW as $\omega^*(s, a) \triangleq \frac{d^{P^*}_{\pi,\gamma}(s,a)}{d^{P^*}_{\pi_b,\gamma}(s,a)}$, we have $d^{P^*}_{\pi_b,\gamma}(s, a) \cdot \omega^*(s, a) = d^{P^*}_{\pi,\gamma}(s, a)$. Expanding the RHS, $\forall\, s', a'$,

$$d^{P^*}_{\pi_b,\gamma}(s', a')\omega^*(s', a') = \gamma \sum_{s,a} \pi(a' \mid s')P^*(s' \mid s, a)\omega^*(s, a)d^{P^*}_{\pi_b,\gamma}(s, a) + (1-\gamma)\mu_0(s')\pi(a' \mid s'). \tag{9}$$

The derivation is deferred to Appendix B.4. Therefore, the *"Bellman equation"* for $\omega(s', a')$ is

$$\omega(s', a') = \mathcal{T}\omega(s', a')\,,$$

$$\mathcal{T}\omega(s', a') \triangleq \frac{\gamma \sum_{s,a} \pi(a' \mid s')P^*(s' \mid s, a)\omega(s, a)d^{P^*}_{\pi_b,\gamma}(s, a) + (1-\gamma)\mu_0(s')\pi(a' \mid s')}{d^{P^*}_{\pi_b,\gamma}(s', a')}\,.$$

Here $\mathcal{T}$ can be viewed as the *"Bellman" operator* for ω. The update iterate defined by $\mathcal{T}$ has the following convergence property, which is proved in Appendix B.5.

Proposition 2. *On finite state-action space, if the current policy π is close to the behavior policy π_b, then the iterate for ω defined by $\mathcal{T}$ converges geometrically.*

The assumption that π is close to π_b coincides with the regularization term in the policy learning objective discussed in Section 3.2.

Unfortunately, the RHS of Eq. (9) is not estimable since we do not know the density values therein. We therefore multiply both sides of Eq. (9) by some test function and then sum over (s', a') on both sides to get a tractable objective that only requires samples from the offline dataset and the initial state-distribution μ_0. We hope to choose a test function that can distinguish the difference between the left and the right hand side. A potential choice is the action-value function of policy π, due to the primal-dual relationship between the stationary state-action density-(ratio) ($d^{P^*}_{\pi,\gamma}(s,a)$ or $w^*(s,a)$) and the action-value function [35–37]. A detailed discussion on the choice of test function can be found in Appendix I.

Practically we use $Q^{\widehat{P}}_\pi$ as the test function. Note that multiplying both sides of Eq. (9) by the same $Q^{\widehat{P}}_\pi$ does not undermine the convergence property (under mild conditions on Q). Mimicking the Bellman backup to sum over (s', a') on both sides, with the notation $d^{P^*}_{\pi_b,\gamma}(s,a,s') = d^{P^*}_{\pi_b,\gamma}(s,a)P^*(s'|s,a)$,

$$\overbrace{\mathbb{E}_{(s,a)\sim d^{P^*}_{\pi_b,\gamma}}\left[\omega^*(s,a)\cdot Q^{\widehat{P}}_\pi(s,a)\right]}^{\ell_1(\omega^*)} = \overbrace{\gamma\mathbb{E}_{\substack{(s,a,s')\sim d^{P^*}_{\pi_b,\gamma}\\ a'\sim\pi(\cdot\,|\,s')}}\left[\omega^*(s,a)\cdot Q^{\widehat{P}}_\pi(s',a')\right] + (1-\gamma)\mathbb{E}_{\substack{s\sim\mu_0(\cdot)\\ a\sim\pi(\cdot\,|\,s)}}\left[Q^{\widehat{P}}_\pi(s,a)\right]}^{\ell_2(\omega^*)}. \quad (10)$$

So for a given ω, we can optimize ω by minimizing the difference between the RHS and LHS of Eq. (10). We replace the RHS with a target network $\omega'(s,a)$, and the final objective for learning ω is

$$\left(\ell_1(\omega) - \ell_2(\omega')\right)^2, \quad (11)$$

where the target network $\omega'(s,a)$ is soft-updated after each gradient step, motivated by $Q_{\boldsymbol{\theta}'_j}$ and $\pi_{\boldsymbol{\phi}'}$.

Our proposed training approach is closely related to VPM [38]. By setting the MIW $\omega(s,a)$ itself as the test function, VPM leverage variational power iteration methods to train MIW iteratively. Instead, our approach set current action-value function as the test function, which is motivated by the primal-dual relationship between density ratio and action-value function. We discuss and compare empirical performance of these approaches in Section 4.2 and Appendix H.

3.4 Practical implementation

In this section we briefly discuss some implementation details of our method, whose main steps are summarized in Algorithm 1. Further details are deferred to Appendix D.1.

Dynamic model training. We adopt common practice in offline MBRL [*e.g.*, 15, 17] to use an ensemble of Gaussian probabilistic networks $\widehat{P}(\cdot\,|\,s,a)$ and $\hat{r}(s,a)$ to parameterize the stochastic transition and reward. We initialize the model by standard MLE training, and periodically update the model by minimizing Eq. (6).

Critic training. We use the conservative target in offline RL literature [*e.g.*, 11, 12]:

$$\widetilde{Q}(s,a) \triangleq r(s,a) + \gamma\mathbb{E}_{a'\sim\pi_{\boldsymbol{\phi}'}(\cdot\,|\,s')}\left[c\min_{j=1,2} Q_{\boldsymbol{\theta}'_j}(s',a') + (1-c)\max_{j=1,2} Q_{\boldsymbol{\theta}'_j}(s',a')\right], \quad (12)$$

where we set $c = 0.75$. With mini-batch $\mathcal{B}$ sampled from the augmented dataset $\mathcal{D}$, both critic networks are trained as

$$\forall\, j = 1,2, \quad \arg\min_{\boldsymbol{\theta}_j} \tfrac{1}{|\mathcal{B}|}\textstyle\sum_{(s,a)\in\mathcal{B}} \mathrm{Huber}(Q_{\boldsymbol{\theta}_j}(s,a), \widetilde{Q}(s,a)), \quad (13)$$

where the Huber loss $\mathrm{Huber}(\cdot)$ is used in lieu of the classical MSE for training stability [39].

Estimating $\widetilde{D}_\pi(P^*, \widehat{P})$. In $\widetilde{D}_\pi(P^*, \widehat{P})$, using the notations of GAN, we denote the sample set from the left distribution of JSD by $\mathcal{B}_{\text{true}}$ (*i.e.*, "true" samples), and the sample set from the right distribution of JSD by $\mathcal{B}_{\text{fake}}$ (*i.e.*, "fake" samples).

Algorithm 1 Main steps of the proposed alternating model-policy training framework.

Initialize: Dynamics model $\widehat{P}$ and $\hat{r}$, policy $\pi_{\boldsymbol{\phi}}$, critics $Q_{\boldsymbol{\theta}_1}$ and $Q_{\boldsymbol{\theta}_2}$, discriminator $D_{\boldsymbol{\psi}}$, MIW ω.
Initialize $\widehat{P}$ and $\hat{r}$ via the MLE (Eq. (4)).
for iteration $\in \{1, \ldots, \texttt{total_iterations}\}$ **do**
  if iteration $\%$ `model_retrain_period` == 0 **then**
    Estimate $\omega(s, a)$ via Eq. (11); train $\widehat{P}$ and $\hat{r}$ by weighted MLE (Eq. (6)) with $\omega(s, a)$.
  end if
  Rollout synthetic data with $\pi_{\boldsymbol{\phi}}$, $\widehat{P}$ and $\hat{r}$, and add the data to $\mathcal{D}_{\text{model}}$.
  Sample mini-batch $\mathcal{B} \sim \mathcal{D} = f\mathcal{D}_{\text{env}} + (1-f)\mathcal{D}_{\text{model}}$.
  Optimize $Q_{\boldsymbol{\theta}_1}, Q_{\boldsymbol{\theta}_2}$ via Eqs. (12)-(13).
  Train $D_{\boldsymbol{\psi}}$ to maximize Eq. (16).
  Optimize $\pi_{\boldsymbol{\phi}}$ by Eq. (17).
end for

The "true" set $\mathcal{B}_{\text{true}}$ consists of samples from $\mathcal{D}_{\text{env}}$. The "fake" set $\mathcal{B}_{\text{fake}}$ is formed by first sampling $s \sim \mathcal{D}$, followed by $a \sim \pi_{\boldsymbol{\phi}}(\cdot \,|\, s), s' \sim \widehat{P}(\cdot \,|\, s, a), a' \sim \pi_{\boldsymbol{\phi}}(\cdot \,|\, s')$. Concretely, the "fake" set $\mathcal{B}_{\text{fake}}$, generator loss $\mathcal{L}_g(\boldsymbol{\phi})$, and the discriminator loss $\mathcal{L}_D(\boldsymbol{\psi})$ can be described as

$$\mathcal{B}_{\text{fake}} \triangleq \begin{bmatrix} (s, & a) \\ (s', & a') \end{bmatrix}, \quad (14) \qquad \mathcal{L}_g(\boldsymbol{\phi}) \triangleq \frac{1}{|\mathcal{B}_{\text{fake}}|} \sum_{(s,a)\in\mathcal{B}_{\text{fake}}} \left[\log\left(1 - D_{\boldsymbol{\psi}}(s,a)\right)\right], \quad (15)$$

$$\mathcal{L}_D(\boldsymbol{\psi}) \triangleq \frac{1}{|\mathcal{B}_{\text{true}}|} \sum_{(s,a)\sim\mathcal{B}_{\text{true}}} \left[\log D_{\boldsymbol{\psi}}(s,a)\right] + \frac{1}{|\mathcal{B}_{\text{fake}}|} \sum_{(s,a)\sim\mathcal{B}_{\text{fake}}} \left[\log\left(1 - D_{\boldsymbol{\psi}}(s,a)\right)\right], \quad (16)$$

where $\mathcal{L}_g(\boldsymbol{\phi})$ is the empirical policy-learning regularizer implied by $\widetilde{D}_\pi(P^*, \widehat{P})$.

Policy training. Since the constant multiplier in Eq. (7) is unknown, we treat it as a hyperparameter. Adding the regularization term of policy training, our policy optimization objective is

$$\arg\min_{\boldsymbol{\phi}} -\lambda \cdot \frac{1}{|\mathcal{B}|} \sum_{s\in\mathcal{B}, a\sim\pi_{\boldsymbol{\phi}}(\cdot\,|\,s)} \left[\min_{j=1,2} Q_{\boldsymbol{\theta}_j}(s,a)\right] + \mathcal{L}_g(\boldsymbol{\phi}), \quad (17)$$

where the regularization coefficient $\lambda \triangleq \lambda'/Q_{avg}$, with soft-updated Q_{avg} similar to Fujimoto and Gu [40]. We use $\lambda' = 10$ across all datasets in our experiments (Section 4.1).

4 Experiments

In this section, we first evaluate our method on continuous-control offline-RL datasets in Section 4.1. Then we conduct ablation study on Section 4.2 to understand the efficacy of some proposed designs.

4.1 Continuous-control results

Our experiments are conducted on a diverse set of datasets in the D4RL benchmark [30], ranging across the Gym-Mojoco, Maze2D, and Adroit domains therein. We use the latest version of the datasets, *i.e.*, "v2" version for the Gym-Mojoco domain and "v1" version for the Maze2D and Adroit domains. Details of our dataset are discussed in Appendix D.2.

We compare our method with two DICE methods — AlgaeDICE [41] and OptiDICE [42] — which directly utilize MIWs to improve policy learning. We further consider three state-of-the-art (SOTA) offline model-free RL algorithms: CQL [43], FisherBRC [44], and TD3+BC [40]; and three SOTA offline MBRL methods: MOPO [15], COMBO [17], and WMOPO [45]. Experimental details and hyperparameter settings are discussed in Appendix D.2. We run baseline methods using the official implementation under the recommended hyperparameter, except for AlgaeDICE, for which we use the offline version provided by Fu et al. [30]. Table 1 shows the mean and standard deviation of our method and baselines over five random seeds.

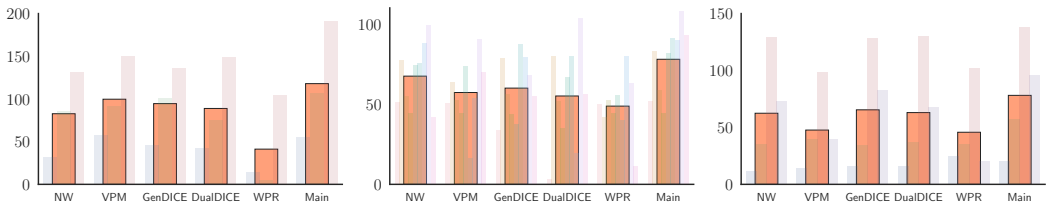
As shown in Table 1, our method performs comparably well and is relatively stable across the sixteen tested datasets. In particular, our method generally performs better than the baselines on the Maze2D and Adroit datasets. The Maze2D and Adroit datasets are considered as more challenging than the MuJoCo datasets [30], since the Maze2D datasets are collected by non-Markovian policies, and the amount of data in the high dimensional Adroit datasets is limited.

Table 1: Normalized returns for experiments on the D4RL tasks. Mean and standard deviation across five random seeds are reported. High average score and low average rank are desirable. We bold the best result over all methods and underline the best of model-based methods if different. Here, "hcheetah" denotes "halfcheetah," "med" denotes "medium," "rep" denotes "replay," and "exp" denotes "expert."

Task Name	AlgaeDICE	OptiDICE	CQL	FisherBRC	TD3+BC	MOPO	COMBO	WMOPO	Ours
maze2d-large	-2.2 ± 0.6	101.7 ± 50.7	1.5 ± 6.4	0.9 ± 6.4	107.1 ± 45.9	-0.5 ± 2.6	138.5 ± 82.2	1.8 ± 8.6	**190.8** ± 34.2
maze2d-med	2.5 ± 14.2	**119.4** ± 52.3	6.3 ± 9.1	16.1 ± 30.3	61.4 ± 45.5	12.5 ± 18.3	103.9 ± 42.1	12.7 ± 38.3	107.2 ± 45.0
maze2d-umaze	-15.3 ± 0.8	**114.0** ± 39.7	37.5 ± 7.2	3.6 ± 16.4	38.6 ± 14.4	-15.4 ± 1.9	112.1 ± 56.8	-11.5 ± 3.1	55.8 ± 3.9
hcheetah-med	-0.9 ± 0.7	42.0 ± 3.5	48.6 ± 0.2	47.8 ± 0.3	48.0 ± 0.3	69.2 ± 3.4	**73.0** ± 3.6	72.0 ± 4.7	51.7 ± 0.4
walker2d-med	1.0 ± 2.1	55.7 ± 15.7	81.8 ± 1.7	80.7 ± 2.1	83.0 ± 1.4	-0.1 ± 0.0	0.5 ± 0.9	64.8 ± 30.0	**83.1** ± 1.8
hopper-med	0.9 ± 0.2	57.5 ± 7.5	68.0 ± 6.0	94.2 ± 4.5	59.1 ± 4.5	44.8 ± 41.5	22.6 ± 38.1	**99.7** ± 2.9	58.9 ± 7.9
hcheetah-med-rep	-3.4 ± 2.3	40.5 ± 3.3	38.7 ± 18.6	34.9 ± 18.2	44.5 ± 0.6	62.7 ± 7.5	66.0 ± 1.8	**66.4** ± 4.9	44.6 ± 0.7
walker2d-med-rep	0.5 ± 0.6	37.3 ± 21.8	80.8 ± 4.0	**84.6** ± 6.7	74.7 ± 8.1	53.8 ± 34.6	60.1 ± 18.3	71.9 ± 15.0	81.5 ± 3.0
hopper-med-rep	1.5 ± 0.7	28.1 ± 12.4	**96.0** ± 4.7	94.4 ± 1.9	58.9 ± 19.7	84.8 ± 30.0	53.6 ± 29.5	93.8 ± 8.5	91.1 ± 9.5
hcheetah-med-exp	-1.8 ± 2.9	66.7 ± 25.8	53.8 ± 13.6	**94.5** ± 0.6	91.4 ± 5.2	64.0 ± 20.8	52.5 ± 32.0	68.8 ± 37.5	90.2 ± 2.0
walker2d-med-exp	-0.2 ± 0.1	79.4 ± 16.4	110.3 ± 0.6	109.3 ± 0.1	**110.4** ± 0.6	-0.2 ± 0.0	1.1 ± 0.6	98.8 ± 16.2	107.7 ± 2.1
hopper-med-exp	2.2 ± 1.9	52.6 ± 9.3	79.3 ± 20.5	**110.0** ± 4.0	98.7 ± 9.6	15.6 ± 7.4	63.7 ± 51.9	62.6 ± 26.4	93.0 ± 19.7
pen-human	-3.0 ± 0.8	-0.9 ± 3.1	-1.5 ± 2.8	-0.1 ± 3.0	1.6 ± 10.2	-1.6 ± 2.6	5.9 ± 8.7	-2.1 ± 1.4	**20.6** ± 10.7
pen-cloned	-2.6 ± 1.3	-0.8 ± 3.1	39.5 ± 23.6	-1.2 ± 3.8	6.3 ± 4.6	5.2 ± 10.2	23.2 ± 19.8	-3.0 ± 2.4	**57.4** ± 19.2
pen-exp	-1.3 ± 2.4	0.4 ± 7.3	119.3 ± 15.8	2.2 ± 1.1	104.2 ± 40.6	35.4 ± 20.9	57.1 ± 42.6	10.4 ± 20.9	**138.3** ± 6.6
door-exp	0.1 ± 0.0	86.1 ± 23.2	94.0 ± 15.8	22.9 ± 28.8	-0.3 ± 0.0	-0.0 ± 0.1	-0.2 ± 0.2	-0.1 ± 0.1	**96.3** ± 9.8
Average Score	-1.4	55	59.6	49.7	61.7	26.9	52.1	44.2	**85.5**
Average Rank	8.5	5.5	4.1	4.3	3.9	6.4	4.8	5.0	**2.5**

On Maze2D datasets, the behavioral-cloning (BC) style algorithms, such as FisherBRC, are likely to fail, since these methods use Markovian policy to approximate the non-Markovian behavior policy. Meanwhile, OptiDICE performs relatively well on Maze2D datasets, mainly because it uses a Gaussian-mixture policy for BC to alleviate this approximation difficulty. This advanced BC policy still offers little help on the higher dimensional Adroit datasets, due to the challenges of learning the behavior policy. Similarly, TD3+BC performs well on relatively lower-dimensional Maze2D datasets, but does not work well on the Adroit domain, since the action-space MSE-regularizer in TD3+BC may be insufficient in high dimensional tasks. WMOPO also shows a performance drop in these two challenging task domains. The environmental dynamics on these two domains are complex, and thus may not be accurately learned to support WMOPO's full trajectory model rollouts, which are used to estimate its MIW. By contrast, our method does not require BC or long model-rollouts, leading to stable performance across task domains. Moreover, our method shows generally better performance than MOPO and COMBO, which use fixed pretrained dynamic models without considering the mismatched model objectives.

Compared with the DICE-based methods AlgaeDICE and OptiDICE, our algorithm shows generally better performance. This may indicate that maximizing a lower bound of the true expected return can be a more effective framework than explicit stationary-distribution correction, since maximizing the lower bound is more directly related to RL's goal of maximizing the policy performance.

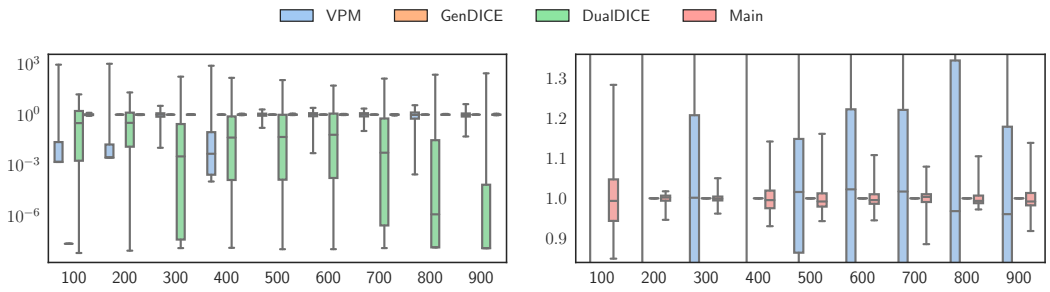


(a) Maze2D (b) MuJoCo (c) Adroit

Figure 1: Scores of each method in the ablation study (Section 4.2) on each domain of datasets. The faded bars show the scores on each datasets within the given domain, average over five seeds, where each color corresponds to a dataset. The highlighted bar shows the average scores on the entire domain. Label "Main" refers to the results of our main method in Section 4.1. Detailed numbers are on Table 2.

4.2 Ablation study

(a): *Does the MIW-weighted model (re)training help the performance?*



(a) Full scale

(b) Zoom-in and remove DualDICE

Figure 2: Box plots of the MIWs estimated by our method ("Main") and the three alternatives in Section 4.2 **(b)** on the "walker2d-medium-replay" dataset during the training process. Recall that the MIW ω is retrained every 100 epochs. The x-axis represents the number of epochs $\{100, 200, \ldots, 900\}$, and the y-axis the MIW values. For clarity, we plot the normalized MIWs whose mean on the entire dataset is 1. We use random seed 2 for this plot. Other random seeds have similar patterns.

To verify the effectiveness of our MIW-weighted model (re)training scheme, we compare our method with its variant of training the model only in the beginning using MLE, *i.e.*, No Weights (dubbed as NW). As shown in Figure 1, the performance of the NW variant is generally worse than the main method (dubbed as Main) on all three domains. The performance difference is especially significant on the Maze2D domain. In this domain, the policy is required to "stitch together collected subtrajectories to find the shortest path to the evaluation goal" [30]. As a result, the good state-action distribution learned from the policy should be different from the distribution of the dataset. Table 2 in Appendix A confirms the benefit of the MIW-weighted model (re)training. Indeed, on the majority of datasets, the NW variant not only shows worse performance, but also has a larger standard deviation relative to the mean score.

(b): *How does the algorithm perform if we change our MIW estimation to other approaches?*

We compare our method with its three variants where the MIW is instead estimated by the Variational Power Method [38], GENeralized stationary DIstribution Correction Estimation [29], and Dual stationary DIstribution Correction Estimation [28]. Three variants are simply dubbed as VPM, GenDICE, and DualDICE. For numerical stability, the estimated MIW from these three methods is clipped into $(10^{-8}, 500)$. Implementation details are provided in Appendix D.3.

As shown in Figure 1 and Table 2, the variants with these three alternative MIW estimation methods generally perform worse than our approach. A plausible explanation is that these methods can be unstable to provide good MIW estimation for the model training. Figure 2 shows the box plot of the MIW estimated by our method and three alternatives on the "walker2d-medium-replay" dataset. We train the MIW every 100 epochs and ignore the MIW initialization and outliers from the plot. The MIWs estimated by GenDICE degenerate, since most of them are 0 at the beginning and are 1 later. This may explain the relatively bad performance of GenDICE. Compared with GenDICE, DualDICE provides more diverse MIW estimates, which leads to its relatively better performance. The VPM variant performs the best among these three approaches, mainly because it provides relatively better MIW estimate in the second half of the training process. However, its estimation is poor-distributed at the beginning. Our method provides well-behaved MIW estimates in the whole training process, which may explain our better result. In general, the well-shaping of the estimated MIW is important, since the Cramer-Rao lower bound of the MSE for OPE is related to the square of the density ratio [36, 46].

Besides, on the Maze2D domain, these three variants with alternative MIW estimation methods perform generally better than the NW variant discussed in Question **(a)**, aligning with the benefit of the MIW-weighted model (re)training scheme. Generally speaking, incorporating MIW can help model training in offline MBRL.

(c): *What is the performance of a weighted regularizer for policy learning implied by Theorem 1?*

We compare our method with its variant where the policy regularizer is weighted by the MIW $\omega(s, a)$, as suggested by Theorem 1. This Weighted Policy Regularizer variant is dubbed as WPR. As shown in both Figure 1 and Table 2, the WPR variant underperforms our main method. This is because when

we estimate the regularization term in WPR, we incorporate weights into the minimax optimization of the policy and the discriminator, which may bring additional instability.

5 Related work

Offline MBRL. Most of the existing offline MBRL works focus on policy learning under a given model trained by MLE. These works typically constrain the learned policy to avoid visiting regions where the discrepancy between the true and the learned dynamics is large, thus reducing the policy evaluation error of using the learned dynamic [15, 18, 47, 48]. We derive an objective that trains both the policy and the dynamic model toward maximizing a lower bound of true expected return (simultaneously minimizing the policy evaluation error $|J(\pi, P^*) - J(\pi, \widehat{P})|$). Several recent works also propose to enhance model training, such as training a reverse dynamics model to encourage the conservative model-based imaginations [49], learning a balanced state-action representation for the model to mitigate the distributional discrepancy [50, 51], and using advanced dynamic model architecture of the GPT-Transformer [52] to achieve accurate predictions [53]. These methods are orthogonal to ours and may further improve the performance.

Objective mismatch in MBRL. This paper is related to prior online and offline MBRL that mitigate the mismatched model objectives. In online (off-policy) MBRL, Lambert et al. [22] identify the mismatched objectives between the MLE model training and the model's usage of improving the control performance. This paper, however, only proposes an impractical weighted learning loss that requires the optimal trajectory. Eysenbach et al. [23] design an objective to jointly optimize the policy and the model in online MBRL. Voelcker et al. [54] propose a per-sample diagonal scaling matrix for training deterministic model in online RL, which improves on Farahmand et al. [55] and Farahmand [56]. In offline MBRL, D'Oro et al. [57] propose a weighting scheme to learn the transition model. This method requires the behavior policy and estimates the trajectory-wise importance sampling weight, which is known to suffer from the "curse of horizon" [26]. Thus, this method may not scale to complex high-dimensional offline RL tasks. Voloshin et al. [58] propose a minimax objective for model learning, where the model is optimized over function classes of the MIWs and the state-value functions. Most similar to our work, Hishinuma and Senda [45] also use a MIW-weighted MLE objective for model training. They use a different version of the MIW estimated via multiple full-trajectory rollouts in the learned model. This strategy may suffer from the inaccuracy of the learned model [59–61] and can be time-consuming. In this paper, we develop an iterative method to train the MIW that only requires samples from the offline dataset and the initial-state distribution, which can be more stable and efficient.

Off-policy evaluation (OPE). OPE is a well-studied problem with several lines of research. A natural way is trajectory-wise importance sampling (IS) [62]. This, however, suffers from its variance, which can grow exponentially with the horizon length [63]. Marginal importance sampling methods [26] are developed to address this problem by estimating the IS ratio between two stationary distributions, leading to estimators whose variance are polynomial *w.r.t.* horizon [64]. Though prior works can also be used to estimate the MIW, they may not be directly applicable to our framework, since they either require knowing the behavior policy [26, 35], or need to assume MIW lying in the RKHS [27]. Another approach to estimate the MIW is through saddle-point optimization [28, 29, 37, 65], which can suffer from its complex optimization. In this paper, we derive an iterative and behavior-agnostic MIW estimation approach, which is simple to train and does not require additional assumptions.

6 Conclusion

In this paper, we are motivated by the mismatched model objectives in offline MBRL to design an iterative algorithm that alternates between model training and policy optimization. Both the model and the policy are trained to maximize a lower bound of the true expected return. The proposed new algorithm performs competitively with several SOTA baselines.

Limitation. Our current approach requires training two additional networks (discriminator D_ψ and MIW ω), and the model is trained periodically to mitigate the objective mismatch, which brings additional computational costs to obtain better empirical results.

Future work. We will conduct a theoretical analysis of our proposed framework, and investigate more efficient way of unifying model training and policy learning.

References

[1] Sascha Lange, Thomas Gabel, and Martin Riedmiller. *Batch Reinforcement Learning*, pages 45–73. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27645-3. doi: 10.1007/978-3-642-27645-3_2.

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

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836.

[4] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489, January 2016. doi: 10.1038/nature16961.

[5] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the 4th International Conference on Web Search and Data Mining (WSDM)*, pages 297–306, 2011.

[6] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and K. Takeda. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access*, 8:58443–58469, 2020.

[7] Youngsoo Jang, Jongmin Lee, and Kee-Eung Kim. GPT-critic: Offline reinforcement learning for end-to-end task-oriented dialogue systems. In *International Conference on Learning Representations*, 2022.

[8] T. Lillicrap, Jonathan J. Hunt, A. Pritzel, N. Heess, T. Erez, Yuval Tassa, D. Silver, and Daan Wierstra. Continuous Control with Deep Reinforcement Learning. *CoRR*, abs/1509.02971, 2016.

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.

[10] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 10–15 Jul 2018.

[11] Scott Fujimoto, David Meger, and Doina Precup. Off-Policy Deep Reinforcement Learning without Exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. PMLR, 09–15 Jun 2019.

[12] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[13] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[14] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to Trust Your Model: Model-Based Policy Optimization. *ArXiv*, abs/1906.08253, 2019.

[15] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y. Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based Offline Policy Optimization. *ArXiv*, abs/2005.13239, 2020.

[16] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A Game Theoretic Framework for Model Based Reinforcement Learning. In *International Conference on Machine Learning*, 2020.

[17] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: Conservative Offline Model-Based Policy Optimization. *ArXiv*, abs/2102.08363, 2021.

[18] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and T. Joachims. MOReL : Model-Based Offline Reinforcement Learning. *ArXiv*, abs/2005.05951, 2020.

[19] Phillip Swazinna, Steffen Udluft, and Thomas A. Runkler. Overcoming Model Bias for Robust Offline Deep Reinforcement Learning. *Eng. Appl. Artif. Intell.*, 104:104366, 2021.

[20] Catherine Cang, Aravind Rajeswaran, P. Abbeel, and M. Laskin. Behavioral Priors and Dynamics Models: Improving Performance and Domain Transfer in Offline RL. *ArXiv*, abs/2106.09119, 2021.

[21] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees. In *International Conference on Learning Representations*, 2019.

[22] Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. *arXiv preprint arXiv:2002.04523*, 2020.

[23] Benjamin Eysenbach, Alexander Khazatsky, Sergey Levine, and Ruslan Salakhutdinov. Mismatched No More: Joint Model-Policy Optimization for Model-Based RL. *ArXiv*, abs/2110.02758, 2021.

[24] Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2:160–163, 1991.

[25] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Belmont, MA: Athena Scientific, 1995.

[26] Q. Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the Curse of Horizon: Infinite-Horizon Off-Policy Estimation. In *Advances in neural information processing systems*, 2018.

[27] A. Mousavi, Lihong Li, Qiang Liu, and Denny Zhou. Black-box Off-policy Estimation for Infinite-Horizon Reinforcement Learning. *ArXiv*, abs/2003.11126, 2020.

[28] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. DualDICE: Behavior-Agnostic Estimation of Discounted Stationary Distribution Corrections. In *Advances in neural information processing systems*, 2019.

[29] Ruiyi Zhang, Bo Dai, Lihong Li, and D. Schuurmans. GenDICE: Generalized Offline Estimation of Stationary Values. *ArXiv*, abs/2002.09072, 2020.

[30] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[31] Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing Bottlenecks in Deep Q-learning Algorithms. In *International Conference on Machine Learning*, 2019.

[32] Long-Ji Lin. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Machine Learning*, 8(3–4):293–321, 1992.

[33] Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on pure and applied mathematics*, 36 (2):183–212, 1983.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[35] Ziyang Tang, Yihao Feng, Lihong Li, Dengyong Zhou, and Qiang Liu. Doubly robust bias reduction in infinite horizon off-policy estimation. In *International Conference on Learning Representations*, 2020.

[36] Nathan Kallus and Masatoshi Uehara. Double reinforcement learning for efficient off-policy evaluation in markov decision processes. *J. Mach. Learn. Res.*, 21(167):1–63, 2020.

[37] Mengjiao Yang, Ofir Nachum, Bo Dai, Lihong Li, and Dale Schuurmans. Off-policy evaluation via the regularized lagrangian. *Advances in Neural Information Processing Systems*, 33:6551–6561, 2020.

[38] Junfeng Wen, Bo Dai, Lihong Li, and Dale Schuurmans. Batch stationary distribution estimation. *arXiv preprint arXiv:2003.00722*, 2020.

[39] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964.

[40] Scott Fujimoto and Shixiang Shane Gu. A Minimalist Approach to Offline Reinforcement Learning. *ArXiv*, abs/2106.06860, 2021.

[41] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and D. Schuurmans. AlgaeDICE: Policy Gradient from Arbitrary Experience. *ArXiv*, abs/1912.02074, 2019.

[42] Jongmin Lee, Wonseok Jeon, Byung-Jun Lee, J. Pineau, and Kee-Eung Kim. OptiDICE: Offline Policy Optimization via Stationary Distribution Correction Estimation. *ArXiv*, abs/2106.10783, 2021.

[43] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for Offline Reinforcement Learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020.

[44] Ilya Kostrikov, Jonathan Tompson, Rob Fergus, and Ofir Nachum. Offline Reinforcement Learning with Fisher Divergence Critic Regularization. In *International Conference on Machine Learning*, 2021.

[45] Toru Hishinuma and Kei Senda. Weighted model estimation for offline model-based reinforcement learning. In *Advances in neural information processing systems*, 2021.

[46] Nan Jiang and Lihong Li. Doubly robust off-policy evaluation for reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 652–661, 2016.

[47] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*, pages 1154–1168. PMLR, 2021.

[48] Cong Lu, Philip J. Ball, Jack Parker-Holder, Michael A. Osborne, and Stephen J. Roberts. Revisiting Design Choices in Model-Based Offline Reinforcement Learning. *ArXiv*, abs/2110.04135, 2021.

[49] Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34, 2021.

[50] Yao Liu, Omer Gottesman, Aniruddh Raghu, Matthieu Komorowski, Aldo A Faisal, Finale Doshi-Velez, and Emma Brunskill. Representation balancing mdps for off-policy policy evaluation. *Advances in Neural Information Processing Systems*, 31, 2018.

[51] Byung-Jun Lee, Jongmin Lee, and Kee-Eung Kim. Representation Balancing Offline Model-based Reinforcement Learning. In *International Conference on Learning Representations*, 2021.

[52] Alec Radford and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. In *arxiv*, 2018.

[53] Michael Janner, Qiyang Li, and Sergey Levine. Reinforcement Learning as One Big Sequence Modeling Problem. *ArXiv*, abs/2106.02039, 2021.

[54] Claas Voelcker, Victor Liao, Animesh Garg, and Amir-massoud Farahmand. Value Gradient weighted Model-Based Reinforcement Learning. *arXiv preprint arXiv:2204.01464*, 2022.

[55] Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, pages 1486–1494. PMLR, 2017.

[56] Amir-massoud Farahmand. Iterative value-aware model learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[57] Pierluca D'Oro, Alberto Maria Metelli, Andrea Tirinzoni, Matteo Papini, and Marcello Restelli. Gradient-aware model-based policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3801–3808, 2020.

[58] Cameron Voloshin, Nan Jiang, and Yisong Yue. Minimax model learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1612–1620. PMLR, 2021.

[59] Stéphane Ross and J. Andrew Bagnell. Agnostic System Identification for Model-Based Reinforcement Learning. *International Conference on Machine Learning*, abs/1203.1007, 2012.

[60] Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and P. Abbeel. Model-Based Reinforcement Learning via Meta-Policy Optimization. *ArXiv*, abs/1809.05214, 2018.

[61] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and P. Abbeel. Model-Ensemble Trust-Region Policy Optimization. *International Conference on Learning Representations*, abs/1802.10592, 2018.

[62] Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *International Conference on Machine Learning*, pages 417–424, 2001.

[63] Lihong Li, Rémi Munos, and Csaba Szepesvári. Toward minimax off-policy value estimation. In *Artificial Intelligence and Statistics*, pages 608–616. PMLR, 2015.

[64] Tengyang Xie, Yifei Ma, and Yu-Xiang Wang. Towards optimal off-policy evaluation for reinforcement learning with marginalized importance sampling. *Advances in Neural Information Processing Systems*, 32, 2019.

[65] Ofir Nachum and Bo Dai. Reinforcement learning via fenchel-rockafellar duality. *arXiv preprint arXiv:2001.01866*, 2020.

[66] Rémi Munos. Error bounds for approximate policy iteration. In *International Conference on Machine Learning*, volume 3, pages 560–567, 2003.

[67] Jinglin Chen and Nan Jiang. Information-Theoretic Considerations in Batch Reinforcement Learning. *ArXiv*, abs/1905.00360, 2019.

[68] Xi Chen, Ali Ghadirzadeh, Tianhe Yu, Yuan Gao, Jianhao Wang, Wenzhe Li, Bin Liang, Chelsea Finn, and Chongjie Zhang. Latent-variable advantage-weighted policy optimization for offline rl. *arXiv preprint arXiv:2203.08949*, 2022.

[69] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. In *Advances in neural information processing systems*, 2018.

[70] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Shane Gu. Deployment-Efficient Reinforcement Learning via Model-Based Offline Optimization. *International Conference on Learning Representations*, abs/2006.03647, 2021.

[71] H. V. Hasselt. Double Q-learning. In *Advances in neural information processing systems*, 2010.

[72] Chengyue Gong, Xingchao Liu, and Qiang Liu. Bi-objective Trade-off with Dynamic Barrier Gradient Descent. *Advances in Neural Information Processing Systems*, 34, 2021.

[73] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2014.

[74] Yuguang Yue, Zhendong Wang, and Mingyuan Zhou. Implicit Distributional Reinforcement Learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

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

[76] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, M. Laskin, P. Abbeel, A. Srinivas, and Igor Mordatch. Decision Transformer: Reinforcement Learning via Sequence Modeling. *ArXiv*, abs/2106.01345, 2021.

[77] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline Reinforcement Learning with Implicit Q-Learning. *ArXiv*, abs/2110.06169, 2021.

[78] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, J. Schulman, E. Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. *ArXiv*, abs/1709.10087, 2018.

[79] Tom White. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

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

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

[82] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR*, abs/1511.06434, 2016.

[83] Tim Salimans, I. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. In *Advances in neural information processing systems*, 2016.

[84] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[85] Huangjie Zheng and Mingyuan Zhou. Exploiting Chain Rule and Bayes' Theorem to Compare Probability Distributions. *Advances in Neural Information Processing Systems*, 34, 2021.

[86] Shujian Zhang, Xinjie Fan, Huangjie Zheng, Korawat Tanwisuth, and Mingyuan Zhou. Alignment attention by matching key and query distributions. *Advances in Neural Information Processing Systems*, 34, 2021.

[87] Bo Dai, Ofir Nachum, Yinlam Chow, Lihong Li, Csaba Szepesvári, and Dale Schuurmans. Coindice: Off-policy confidence interval estimation. *Advances in neural information processing systems*, 33:9398–9411, 2020.

[88] Shentao Yang, Zhendong Wang, Huangjie Zheng, Yihao Feng, and Mingyuan Zhou. A regularized implicit policy for offline reinforcement learning. *arXiv preprint arXiv:2202.09673*, 2022.

[89] Masatoshi Uehara, Jiawei Huang, and Nan Jiang. Minimax weight and q-function learning for off-policy evaluation. In *International Conference on Machine Learning*, pages 9659–9668. PMLR, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Sections 3 and 4.
   (b) Did you describe the limitations of your work? [Yes] See Sections 5 and 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix E.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3 and Appendix B.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 3 and Appendix D.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix D.2 and D.3.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Tables 1, 2, and 3.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix F.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See the GitHub pages of the assets.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We use public benckmarks.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] The used benckmarks are widely accepted in the community.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Additional table

Table 2: Normalized returns for experiments on the D4RL tasks. Mean and standard deviation across five random seeds are reported. NW denotes the No-Weights variant in Section 4.2 **(a)**. VPM, GenDICE, and DualDICE denote the variants for MIW estimation in Section 4.2 **(b)**. WPR denotes the Weighted Policy Regularizer variant in Section 4.2 **(c)**. The results of our main method in Section 4.1 is reported in column Main. Here, "hcheetah" denotes "halfcheetah", "med" denotes "medium", "rep" denotes "replay", and "exp" denotes "expert".

Task Name	NW	VPM	GenDICE	DualDICE	WPR	Main
maze2d-umaze	31.3 $\pm$ 25.3	57.8 $\pm$ 29.1	46.2 $\pm$ 23.1	42.6 $\pm$ 19.4	14.4 $\pm$ 21.4	55.8 $\pm$ 3.9
maze2d-med	85.4 $\pm$ 28.6	91.1 $\pm$ 54.3	101.3 $\pm$ 50.3	74.8 $\pm$ 42.6	5.0 $\pm$ 8.3	107.2 $\pm$ 45.0
maze2d-large	131.4 $\pm$ 63.0	150.2 $\pm$ 64.3	136.1 $\pm$ 63.0	149.2 $\pm$ 32.7	104.3 $\pm$ 48.1	190.8 $\pm$ 34.2
Average Maze2D	82.7	99.7	94.5	88.9	41.2	117.9
hcheetah-med	51.5 $\pm$ 0.3	50.8 $\pm$ 0.7	33.5 $\pm$ 18.8	3.5 $\pm$ 1.8	50.1 $\pm$ 0.4	51.7 $\pm$ 0.4
walker2d-med	77.3 $\pm$ 6.0	63.7 $\pm$ 28.3	78.7 $\pm$ 5.3	79.7 $\pm$ 4.7	41.8 $\pm$ 18.9	83.1 $\pm$ 1.8
hopper-med	55.0 $\pm$ 8.9	52.3 $\pm$ 6.6	56.5 $\pm$ 11.3	51.6 $\pm$ 9.1	52.5 $\pm$ 21.6	58.9 $\pm$ 7.9
hcheetah-med-rep	44.6 $\pm$ 0.3	44.3 $\pm$ 0.4	43.8 $\pm$ 1.2	34.9 $\pm$ 18.1	44.4 $\pm$ 0.4	44.6 $\pm$ 0.7
walker2d-med-rep	74.6 $\pm$ 4.2	73.5 $\pm$ 7.3	37.8 $\pm$ 32.6	66.9 $\pm$ 29.6	55.8 $\pm$ 14.3	81.5 $\pm$ 3.0
hopper-med-rep	75.4 $\pm$ 19.1	16.6 $\pm$ 20.0	87.7 $\pm$ 16.6	79.7 $\pm$ 19.8	40.1 $\pm$ 11.1	91.1 $\pm$ 9.5
hcheetah-med-exp	88.0 $\pm$ 5.1	54.0 $\pm$ 43.2	79.0 $\pm$ 18.6	19.4 $\pm$ 35.9	80.1 $\pm$ 5.9	90.2 $\pm$ 2.0
walker2d-med-exp	99.1 $\pm$ 7.9	90.3 $\pm$ 30.2	68.3 $\pm$ 32.3	103.9 $\pm$ 4.3	62.9 $\pm$ 17.6	107.7 $\pm$ 2.1
hopper-med-exp	41.7 $\pm$ 16.9	69.8 $\pm$ 9.6	54.7 $\pm$ 22.5	56.2 $\pm$ 26.5	11.3 $\pm$ 7.6	93.0 $\pm$ 19.7
Average MuJoCo	67.5	57.3	60	55.1	48.8	78
pen-human	11.4 $\pm$ 8.6	13.8 $\pm$ 12.1	15.7 $\pm$ 9.7	16.4 $\pm$ 11.9	24.8 $\pm$ 14.7	20.6 $\pm$ 10.7
pen-cloned	35.3 $\pm$ 15.7	39.5 $\pm$ 15.8	34.4 $\pm$ 17.2	37.3 $\pm$ 16.4	35.5 $\pm$ 13.8	57.4 $\pm$ 19.2
pen-exp	129.6 $\pm$ 15.4	98.1 $\pm$ 33.7	128.6 $\pm$ 16.2	130.4 $\pm$ 16.0	102.0 $\pm$ 36.3	138.3 $\pm$ 6.6
door-exp	73.5 $\pm$ 30.9	39.3 $\pm$ 45.2	83.1 $\pm$ 25.9	68.0 $\pm$ 35.0	20.7 $\pm$ 16.4	96.3 $\pm$ 9.8
Average Adroit	62.4	47.7	65.4	63	45.8	78.2
Average All	69.1	62.8	67.8	63.4	46.6	85.5

B Proofs

B.1 Preliminary on discounted stationary state-action distribution

Recall that the discounted visitation frequency for a policy π on MDP $\mathcal{M}$ with transition P^* is defined as

$$d^{P^*}_{\pi,\gamma}(s,a) \triangleq (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \Pr\left(s_t = s, a_t = a \,|\, \mu_0, \pi, P^*\right).$$

Denote $\boldsymbol{T}^*_\pi(s' \,|\, s) = p_\pi\left(s_{t+1} = s' \,|\, s_t = s\right) = \sum_{a\in\mathbb{A}} P^*\left(s_{t+1} = s' \,|\, s_t = s, a_t = a\right)\pi\left(a \,|\, s\right)$.

From Liu et al. [26], Lemma 3, for $d^{P^*}_{\pi,\gamma}(s)$ we have,

$$\gamma\sum_s \boldsymbol{T}^*_\pi(s' \,|\, s) d^{P^*}_{\pi,\gamma}(s) - d^{P^*}_{\pi,\gamma}(s') + (1-\gamma)\mu_0(s') = 0, \quad \forall\, s' \in \mathbb{S},$$

where μ_0 is the initial state-distribution. Multiply $\pi(a' \,|\, s')$ on both sides, we get, $\forall\,(\boldsymbol{s}', \boldsymbol{a}') \in \mathbb{S}\times\mathbb{A}$,

$$\gamma\sum_s \boldsymbol{T}^*_\pi(s' \,|\, s) d^{P^*}_{\pi,\gamma}(s)\pi(a' \,|\, s') - d^{P^*}_{\pi,\gamma}(s',a') + (1-\gamma)\mu_0(s')\pi(a' \,|\, s') = 0$$

$$\iff \gamma\sum_{s,a} P^*\left(s' \,|\, s, a\right) d^{P^*}_{\pi,\gamma}(s,a)\pi(a' \,|\, s') - d^{P^*}_{\pi,\gamma}(s',a') + (1-\gamma)\mu_0(s')\pi(a' \,|\, s') = 0$$

We can also multiply $\widetilde{\pi}(a' \mid s')$ on both sides to get, $\forall\, (\boldsymbol{s}', \boldsymbol{a}') \in \mathbb{S} \times \mathbb{A}$,

$$\gamma \sum_{s} \boldsymbol{T}_{\pi}^{*}(s' \mid s) d_{\pi,\gamma}^{P^*}(s) \widetilde{\pi}(a' \mid s') - d_{\pi,\gamma}^{P^*}(s') \widetilde{\pi}(a' \mid s') + (1-\gamma)\mu_0(s')\widetilde{\pi}(a' \mid s') = 0.$$

Denote $d_{\pi,\gamma}^{P^*}(s,a,s') \triangleq d_{\pi,\gamma}^{P^*}(s)\pi(a \mid s)P^*(s' \mid s,a)$. For any integrable function $f(s,a)$, we multiply both sides of the above equation by $f(s',a')$ and summing over s', a', we get

$$\gamma \mathbb{E}_{\substack{(s,a,s') \sim d_{\pi,\gamma}^{P^*} \\ a' \sim \widetilde{\pi}(\cdot \mid s')}} \left[f(s',a') \right] - \mathbb{E}_{\substack{s \sim d_{\pi,\gamma}^{P^*} \\ a \sim \widetilde{\pi}(\cdot \mid s)}} \left[f(s,a) \right] + (1-\gamma) \mathbb{E}_{\substack{s \sim \mu_0 \\ a \sim \widetilde{\pi}(\cdot \mid s)}} \left[f(s,a) \right] = 0.$$

for any given bounded $g(\boldsymbol{s}, \boldsymbol{a})$, defining function f to satisfy

$$f(s,a) = g(s,a) + \gamma \mathbb{E}_{s' \sim P^*(\cdot \mid s,a), a' \sim \pi(\cdot \mid s')} \left[f(s',a') \right], \forall\, s, a,$$

then we have

$$\begin{aligned}
\mathbb{E}_{s \sim \mu_0(\cdot), a \sim \pi(\cdot \mid s)} \left[f(s,a) \right] &= \mathbb{E}_{s \sim \mu_0(\cdot), a \sim \pi(\cdot \mid s)} \left[g(s,a) + \gamma \mathbb{E}_{s' \sim P^*(\cdot \mid s,a), a' \sim \pi(\cdot \mid s')} \left[f(s',a') \right] \right] \\
&= \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t g(s_t, a_t) \mid s_0 \sim \mu_0(\cdot), a_t \sim \pi(\cdot \mid s_t), s_{t+1} \sim P^*(\cdot \mid s_t, a_t) \right] \\
&= (1-\gamma)^{-1} \mathbb{E}_{(s,a) \sim d_{\pi,\gamma}^{P^*}} \left[g(s,a) \right],
\end{aligned}$$

based on the definition of $d_{\pi,\gamma}^{P^*}(s,a)$ stated above.

Indeed, $f(s,a)$ is the action-value function of policy π under the reward $g(s,a)$ on MDP $\mathcal{M}$, and can be approximated using neural network under the classical regularity assumptions on $g(s,a)$ and $\mathcal{M}$. Also, under these classical regularity conditions, reward function g and its action-value function f have one-to-one correspondence, with f being the unique solution to the Bellman equation.

B.2 Proofs of Theorem 1

The following Lemma will be used in the proof of Theorem 1.

Lemma 3. *For any $\widehat{P}(s' \mid s,a), \pi(a' \mid s'), P^*(s' \mid s,a), \pi_b(a' \mid s')$, we have*

$$\mathrm{KL}\left(P^*(s' \mid s,a)\pi_b(a' \mid s') \,||\, \widehat{P}(s' \mid s,a)\pi(a' \mid s') \right) \geq \mathrm{KL}\left(P^*(s' \mid s,a) \,||\, \widehat{P}(s' \mid s,a) \right).$$

Proof. Since $\mathrm{KL}(\cdot \,||\, \cdot) \geq 0$, we have

$$\begin{aligned}
&\mathrm{KL}\left(P^*(s' \mid s,a)\pi_b(a' \mid s') \,||\, \widehat{P}(s' \mid s,a)\pi(a' \mid s') \right) \\
=& \int P^*(s' \mid s,a)\pi_b(a' \mid s') \log \frac{P^*(s' \mid s,a)\pi_b(a' \mid s')}{\widehat{P}(s' \mid s,a)\pi(a' \mid s')} \,\mathrm{d}(s',a') \\
=& \int P^*(s' \mid s,a)\pi_b(a' \mid s') \left(\log \frac{P^*(s' \mid s,a)}{\widehat{P}(s' \mid s,a)} + \log \frac{\pi_b(a' \mid s')}{\pi(a' \mid s')} \right) \mathrm{d}(s',a') \\
=& \int P^*(s' \mid s,a)\pi_b(a' \mid s') \log \frac{\pi_b(a' \mid s')}{\pi(a' \mid s')} \,\mathrm{d}(s',a') + \int P^*(s' \mid s,a) \log \frac{P^*(s' \mid s,a)}{\widehat{P}(s' \mid s,a)} \,\mathrm{d}s' \\
=& \mathbb{E}_{s' \sim P^*(\cdot \mid s,a)} \left[\mathrm{KL}\left(\pi_b(a' \mid s') \,||\, \pi(a' \mid s') \right) \right] + \mathrm{KL}\left(P^*(s' \mid s,a) \,||\, \widehat{P}(s' \mid s,a) \right) \\
\geq& \mathrm{KL}\left(P^*(s' \mid s,a) \,||\, \widehat{P}(s' \mid s,a) \right),
\end{aligned}$$

as desired. □

Remark 4. It also holds that

$$\begin{aligned}
&\mathbb{E}_{(s,a) \sim d_{\pi_b,\gamma}^{P^*}} \left[\mathrm{KL}\left(P^*(s' \mid s,a)\pi_b(a' \mid s') \,||\, \widehat{P}(s' \mid s,a)\pi(a' \mid s') \right) \right] \\
\leq& \mathrm{KL}\left(P^*(s' \mid s,a)\pi_b(a' \mid s') d_{\pi_b,\gamma}^{P^*}(s,a) \,||\, \widehat{P}(s' \mid s,a)\pi(a' \mid s') d_{\pi_b,\gamma}^{P^*}(s)\pi(a \mid s) \right),
\end{aligned}$$

by using similar proof steps.

Proof of Theorem 1.

$$
\begin{aligned}
&\left|J(\pi,\widehat{P})-J(\pi,P^*)\right| \\
=&\left|(1-\gamma)\mathbb{E}_{\substack{s\sim\mu_0\\a\sim\pi(\cdot|s)}}\left[Q_\pi^{\widehat{P}}(s,a)\right]-\mathbb{E}_{d_{\pi,\gamma}^{P^*}}\left[Q_\pi^{\widehat{P}}(s,a)\right]+\mathbb{E}_{d_{\pi,\gamma}^{P^*}}\left[Q_\pi^{\widehat{P}}(s,a)\right]-\mathbb{E}_{(s,a)\sim d_{\pi,\gamma}^{P^*}}\left[r(s,a)\right]\right| \\
=&\left|(1-\gamma)\mathbb{E}_{\substack{s\sim\mu_0\\a\sim\pi(\cdot|s)}}\left[Q_\pi^{\widehat{P}}(s,a)\right]-\mathbb{E}_{d_{\pi,\gamma}^{P^*}}\left[Q_\pi^{\widehat{P}}(s,a)\right]+\mathbb{E}_{(s,a)\sim d_{\pi,\gamma}^{P^*}}\left[Q_\pi^{\widehat{P}}(s,a)-r(s,a)\right]\right| \\
=&\left|(1-\gamma)\mathbb{E}_{\substack{s\sim\mu_0\\a\sim\pi(\cdot|s)}}\left[Q_\pi^{\widehat{P}}(s,a)\right]-\mathbb{E}_{d_{\pi,\gamma}^{P^*}}\left[Q_\pi^{\widehat{P}}(s,a)\right]+\gamma\mathbb{E}_{(s,a)\sim d_{\pi,\gamma}^{P^*}}\left[\mathbb{E}_{\substack{s'\sim\widehat{P}(\cdot|s,a)\\a'\sim\pi(\cdot|s')}}\left[Q_\pi^{\widehat{P}}(s',a')\right]\right]\right| \\
=&\left|-\gamma\mathbb{E}_{(s,a)\sim d_{\pi,\gamma}^{P^*}}\left[\mathbb{E}_{\substack{s'\sim P^*(\cdot|s,a)\\a'\sim\pi(\cdot|s')}}\left[Q_\pi^{\widehat{P}}(s',a')\right]\right]+\gamma\mathbb{E}_{(s,a)\sim d_{\pi,\gamma}^{P^*}}\left[\mathbb{E}_{\substack{s'\sim\widehat{P}(\cdot|s,a)\\a'\sim\pi(\cdot|s')}}\left[Q_\pi^{\widehat{P}}(s',a')\right]\right]\right| \\
=&\gamma\left|\mathbb{E}_{(s,a)\sim d_{\pi,\gamma}^{P^*}}\left[\mathbb{E}_{\substack{s'\sim P^*(\cdot|s,a)\\a'\sim\pi(\cdot|s')}}\left[Q_\pi^{\widehat{P}}(s',a')\right]-\mathbb{E}_{\substack{s'\sim\widehat{P}(\cdot|s,a)\\a'\sim\pi(\cdot|s')}}\left[Q_\pi^{\widehat{P}}(s',a')\right]\right]\right| \\
=&\gamma\left|\mathbb{E}_{(s,a)\sim d_{\pi,\gamma}^{P^*}}\left[\mathbb{E}_{s'\sim P^*(\cdot|s,a)}\left[V_\pi^{\widehat{P}}(s')\right]-\mathbb{E}_{s'\sim\widehat{P}(\cdot|s,a)}\left[V_\pi^{\widehat{P}}(s')\right]\right]\right| \\
\leq&\gamma\mathbb{E}_{(s,a)\sim d_{\pi,\gamma}^{P^*}}\left\{\left|\mathbb{E}_{s'\sim P^*(\cdot|s,a)}\left[V_\pi^{\widehat{P}}(s')\right]-\mathbb{E}_{s'\sim\widehat{P}(\cdot|s,a)}\left[V_\pi^{\widehat{P}}(s')\right]\right|\right\} \\
\leq&\frac{\gamma\cdot r_{\max}}{1-\gamma}\mathbb{E}_{(s,a)\sim d_{\pi,\gamma}^{P^*}}\left\{\sup_{V\in\mathcal{V}}\left|\mathbb{E}_{s'\sim P^*(\cdot|s,a)}\left[V(s')\right]-\mathbb{E}_{s'\sim\widehat{P}(\cdot|s,a)}\left[V(s')\right]\right|\right\} \\
=&\frac{\gamma\cdot r_{\max}}{1-\gamma}\mathbb{E}_{(s,a)\sim d_{\pi,\gamma}^{P^*}}\left[\mathrm{TV}\left(P^*(\cdot|s,a)||\widehat{P}(\cdot|s,a)\right)\right] \\
\leq&\frac{\gamma\cdot r_{\max}}{1-\gamma}\mathbb{E}_{(s,a)\sim d_{\pi,\gamma}^{P^*}}\left[\sqrt{\frac{1}{2}\mathrm{KL}\left(P^*(\cdot|s,a)||\widehat{P}(\cdot|s,a)\right)}\right] \\
\leq&\frac{\gamma\cdot r_{\max}}{\sqrt{2}\cdot(1-\gamma)}\sqrt{\mathbb{E}_{(s,a)\sim d_{\pi,\gamma}^{P^*}}\left[\mathrm{KL}\left(P^*(\cdot|s,a)||\widehat{P}(\cdot|s,a)\right)\right]} \\
\leq&\frac{\gamma\cdot r_{\max}}{\sqrt{2}(1-\gamma)}\sqrt{\mathbb{E}_{(s,a)\sim d_{\pi,\gamma}^{P^*}}\left[\mathrm{KL}\left(P^*(s'\,|\,s,a)\pi_b(a'\,|\,s')\,||\,\widehat{P}(s'\,|\,s,a)\pi(a'\,|\,s')\right)\right]} \\
=&\frac{\gamma\cdot r_{\max}}{\sqrt{2}(1-\gamma)}\sqrt{\mathbb{E}_{(s,a)\sim d_{\pi_b,\gamma}^{P^*}}\left[\omega(s,a)\mathrm{KL}\left(P^*(s'\,|\,s,a)\pi_b(a'\,|\,s')\,||\,\widehat{P}(s'\,|\,s,a)\pi(a'\,|\,s')\right)\right]}
\end{aligned}
\tag{18}
$$

where $\mathcal{V}$ is the set of all functions bounded by 1, $\omega(s,a)\triangleq\frac{d_{\pi,\gamma}^{P^*}(s,a)}{d_{\pi_b,\gamma}^{P^*}(s,a)}$ is the marginal importance weight (MIW). Here we use the assumption that $\forall\, s,a,\ |r(s,a)|\leq r_{\max}$, and hence $|V_\pi^{\widehat{P}}(\cdot)|\leq\frac{r_{\max}}{1-\gamma}$. We use Lemma 3 to introduce π and π_b in to $\mathrm{KL}\left(\cdot\,||\,\cdot\right)$. □

B.3 Derivation of Eq. (8)

Derivarion of Eq. (8). We follow the literature [*e.g.*, 66, 67] to assume that $\forall\, s,a,\ \omega(s,a)\leq\omega_{\max}$ for some unknown finite constant $\omega_{\max}$. Using Remark 4 and the assumption, we have

$$
\begin{aligned}
D_\pi(P^*,\widehat{P})&=\mathbb{E}_{(s,a)\sim d_{\pi_b,\gamma}^{P^*}}\left[\omega(s,a)\mathrm{KL}\left(P^*(s'\,|\,s,a)\pi_b(a'\,|\,s')\,||\,\widehat{P}(s'\,|\,s,a)\pi(a'\,|\,s')\right)\right] \\
&\leq\omega_{\max}\cdot\mathbb{E}_{(s,a)\sim d_{\pi_b,\gamma}^{P^*}}\left[\mathrm{KL}\left(P^*(s'\,|\,s,a)\pi_b(a'\,|\,s')\,||\,\widehat{P}(s'\,|\,s,a)\pi(a'\,|\,s')\right)\right] \\
&\leq\omega_{\max}\cdot\mathrm{KL}\left(P^*(s'\,|\,s,a)\pi_b(a'\,|\,s')d_{\pi_b,\gamma}^{P^*}(s,a)\,||\,\widehat{P}(s'\,|\,s,a)\pi(a'\,|\,s')d_{\pi_b,\gamma}^{P^*}(s)\pi(a\,|\,s)\right) \\
&\approx\omega_{\max}\cdot\mathrm{JSD}\left(P^*(s'\,|\,s,a)\pi_b(a'\,|\,s')d_{\pi_b,\gamma}^{P^*}(s,a)\,||\,\widehat{P}(s'\,|\,s,a)\pi(a'\,|\,s')d_{\pi_b,\gamma}^{P^*}(s)\pi(a\,|\,s)\right) \\
&\triangleq\omega_{\max}\cdot\widetilde{D}_\pi(P^*,\widehat{P}),
\end{aligned}
$$

where we approximate the KL divergence by the JSD. □

B.4 Derivation of Eq. (9)

Derivarion of Eq. (9). We have, $\forall\,(s,a)\in\mathbb{S}\times\mathbb{A}$,

$$
\begin{aligned}
& d^{P^*}_{\pi_b,\gamma}(s,a)\cdot\omega(s,a)=d^{P^*}_{\pi,\gamma}(s,a)\iff \\
& d^{P^*}_{\pi_b,\gamma}(s',a')\cdot\omega(s',a')=d^{P^*}_{\pi,\gamma}(s',a') \\
=&\gamma\sum_{s}T^*_{\pi}(s'|s)d^{P^*}_{\pi,\gamma}(s)\pi(a'|s')+(1-\gamma)\mu_0(s')\pi(a'|s') \\
=&\gamma\sum_{s,a}P^*(s'|s,a)d^{P^*}_{\pi,\gamma}(s,a)\pi(a'|s')+(1-\gamma)\mu_0(s')\pi(a'|s') \\
=&\gamma\sum_{s,a}P^*(s'|s,a)\omega(s,a)d^{P^*}_{\pi_b,\gamma}(s,a)\pi(a'|s')+(1-\gamma)\mu_0(s')\pi(a'|s'),
\end{aligned}
\tag{19}
$$

where $T^*_{\pi}(s'|s)=\sum_{a\in\mathbb{A}}P^*(s'|s,a)\pi(a|s)$ is the state-transition kernel. Here we change (s,a) into (s',a') in the " $\iff$ " for notation simplicity. □

B.5 Proof of Proposition 2

Proof of Proposition 2. With the assumed conditions, we would like to show that $\mathcal{T}$ is a contraction mapping under $\|\cdot\|_\infty$, *i.e.*, $\|\mathcal{T}\omega-\mathcal{T}u\|_\infty\le c\cdot\|\omega-u\|_\infty$, for all MIWs $\omega,u:\mathbb{S}\times\mathbb{A}\to\mathbb{R}$, for some constant $c<1$. We have,

$$
\begin{aligned}
& \|\mathcal{T}\omega-\mathcal{T}u\|_\infty \\
=&\max_{s',a'}|(\mathcal{T}\omega-\mathcal{T}u)(s',a')|=\max_{s',a'}|\mathcal{T}\omega(s',a')-\mathcal{T}u(s',a')| \\
=&\max_{s',a'}\left|\frac{\gamma\sum_{s,a}\pi(a'|s')P^*(s'|s,a)d^{P^*}_{\pi_b,\gamma}(s,a)(\omega(s,a)-u(s,a))}{d^{P^*}_{\pi_b,\gamma}(s',a')}\right| \\
\le&\max_{s',a'}\frac{\gamma\sum_{s,a}\pi(a'|s')P^*(s'|s,a)d^{P^*}_{\pi_b,\gamma}(s,a)|\omega(s,a)-u(s,a)|}{d^{P^*}_{\pi_b,\gamma}(s',a')} \\
\le&\max_{s',a'}\frac{\gamma\sum_{s,a}\pi(a'|s')P^*(s'|s,a)d^{P^*}_{\pi_b,\gamma}(s,a)}{d^{P^*}_{\pi_b,\gamma}(s',a')}\cdot\|\omega-u\|_\infty \\
=&\left\{\max_{s',a'}\frac{\gamma\sum_{s,a}\pi(a'|s')P^*(s'|s,a)d^{P^*}_{\pi_b,\gamma}(s,a)}{\gamma\sum_{s,a}\pi_b(a'|s')P^*(s'|s,a)d^{P^*}_{\pi_b,\gamma}(s,a)+(1-\gamma)\mu_0(s')\pi_b(a'|s')}\right\}\cdot\|\omega-u\|_\infty \\
=&\left\{\max_{s',a'}\frac{\pi(a'|s')}{\pi_b(a'|s')}\cdot\frac{\gamma\sum_{s,a}P^*(s'|s,a)d^{P^*}_{\pi_b,\gamma}(s,a)}{\gamma\sum_{s,a}P^*(s'|s,a)d^{P^*}_{\pi_b,\gamma}(s,a)+(1-\gamma)\mu_0(s')}\right\}\cdot\|\omega-u\|_\infty \\
=&\left\{\max_{s',a'}\frac{\pi(a'|s')}{\pi_b(a'|s')}\cdot\underbrace{\left(1-\frac{(1-\gamma)\mu_0(s')}{\gamma\sum_{s,a}P^*(s'|s,a)d^{P^*}_{\pi_b,\gamma}(s,a)+(1-\gamma)\mu_0(s')}\right)}_{\triangleq c(s')<1}\right\}\cdot\|\omega-u\|_\infty \\
=&\left\{\max_{s',a'}\frac{\pi(a'|s')}{\pi_b(a'|s')}\cdot c(s')\right\}\cdot\|\omega-u\|_\infty\triangleq c\cdot\|\omega-u\|_\infty.
\end{aligned}
\tag{20}
$$

If the current policy π is close to the behavior policy π_b, in a sense that $\frac{\pi(a'|s')}{\pi_b(a'|s')}$ is close to 1 on the entire finite state-action space, specifically,

$$
\forall\,s',a',\quad\frac{\pi(a'|s')}{\pi_b(a'|s')}<\frac{1}{c(s')}.
$$

Then $c=\max_{s',a'}\left\{\frac{\pi(a'|s')}{\pi_b(a'|s')}\cdot c(s')\right\}<1$. Plug into Eq. (20), we have $\|\mathcal{T}\omega-\mathcal{T}u\|_\infty\le c\cdot\|\omega-u\|_\infty$ for $c<1$. Thus $\mathcal{T}$ is a c-contraction mapping under $\|\cdot\|_\infty$. It follows that there exists a unique fixed

Table 3: Normalized returns for experiments on the D4RL tasks. Mean and standard deviation across five random seeds are reported. “V-Dis" denotes the variant where the model is trained by the value-function discriminated weighted objective Eq. (21). The results of our main method in Section 4.1 is reported in column Main. Here, “hcheetah" denotes “halfcheetah", “med" denotes “medium", “rep" denotes “replay", and “exp" denotes “expert".

Task Name	V-Dis	Main
maze2d-umaze	36.7 ± 28.3	55.8 ± 3.9
maze2d-med	36.9 ± 19.0	107.2 ± 45.0
maze2d-large	23.9 ± 17.6	190.8 ± 34.2
hcheetah-med	38.3 ± 17.1	51.7 ± 0.4
walker2d-med	46.9 ± 28.3	83.1 ± 1.8
hopper-med	58.4 ± 7.4	58.9 ± 7.9
hcheetah-med-rep	21.3 ± 13.9	44.6 ± 0.7
walker2d-med-rep	16.3 ± 12.9	81.5 ± 3.0
hopper-med-rep	69.0 ± 19.6	91.1 ± 9.5
hcheetah-med-exp	39.7 ± 4.2	90.2 ± 2.0
walker2d-med-exp	51.3 ± 29.4	107.7 ± 2.1
hopper-med-exp	24.0 ± 8.0	93.0 ± 19.7
pen-human	11.4 ± 10.0	20.6 ± 10.7
pen-cloned	36.3 ± 10.5	57.4 ± 19.2
pen-exp	131.3 ± 16.7	138.3 ± 6.6
door-exp	60.4 ± 33.0	96.3 ± 9.8
Average Score	43.9	85.5

point ω^* under the mapping $\mathcal{T}$, such that $\omega^* = \mathcal{T}\omega^*$. Finally, we have

$$\|\omega_{k+1} - \omega^*\|_\infty = \|\mathcal{T}\omega_k - \mathcal{T}\omega^*\|_\infty \leq c \cdot \|\omega_k - \omega^*\|_\infty \leq \cdots \leq c^{k+1} \cdot \|\omega_0 - \omega^*\|_\infty \to 0, \text{ as } k \to \infty,$$

which shows that the iterate defined by $\mathcal{T}$ converges geometrically. □

C Using value function as the discriminator of model training

Eq. (18) of the proof of Theorem 1 motivates us to use the value function $V_\pi^{\widehat{P}}(\cdot)$ as the discriminator of the model training. Specifically, the model training objective is

$$\arg\min_{\widehat{P} \in \mathcal{P}} \mathbb{E}_{(s,a) \sim d^{P^*}_{\pi_b,\gamma}} \left\{ \omega(s,a) \cdot \left| \mathbb{E}_{s' \sim P^*(\cdot|s,a)} \left[V_\pi^{\widehat{P}}(s') \right] - \mathbb{E}_{s' \sim \widehat{P}(\cdot|s,a)} \left[V_\pi^{\widehat{P}}(s') \right] \right| \right\}, \tag{21}$$

where the discriminator $V_\pi^{\widehat{P}}$ can be implemented by the current estimate of the action-value function, which is treated as fixed during the model training. We implemented this idea as a variant where the model is trained by the weighed objective Eq. (21) that use the value-function estimate to discriminate the trainsition prediction from the real. The reward function $\hat{r}$ is still estimated by the weighted-MLE objective. The model is initialized by the MLE loss. Other technical details follow the main algorithm. We dubbed this variant as “V-Dis".

Table 3 compare this variant with our main method. We see that V-Dis underperforms our main method on almost all dataset, often by a large margin. The inferior results of V-Dis may be related to the coupled effect on model training from the inaccurate value-function estimation. In Eq. (21), this inaccuracy can affect both the value difference (the $|\cdot|$ term) and the MIW ω. Our fix of $V_\pi^{\widehat{P}}$ during the model training process may also affect the performance. Future work can investigate how to train $V_\pi^{\widehat{P}}$ together with $\widehat{P}$ in the model estimation.

D Technical details

D.1 Details for the main algorithm

Our main algorithm consists of three major components: model training (Appendix D.1.1), the marginal importance weight training (Appendix D.1.2), and RL policy training via actor-critic algorithm (Appendix D.1.3).

The non-zero assumption of $Q_\pi^{\widehat{P}}$ in the derivation of Eq. (11) motivates us to scale the rewards in the offline dataset to be roughly within $(0, 1]$, similar to Chen et al. [68]. Specifically, before starting our main algorithm, we normalize the rewards r_i's in the offline dataset via $(r_i - r_{\min} + 0.001)/(r_{\max} - r_{\min})$, where $r_{\min}$ and $r_{\max}$ respectively denote the minimum and maximum reward in the offline dataset.

D.1.1 Model training

We follow the literature [69, 14, 15, 45] to assume no prior knowledge about the reward function and thus use neural network to approximate transition dynamic and the reward function. Our model is constructed as an ensemble of Gaussian probabilistic networks . Except for the weighted loss function, we use the same model architecture, ensemble size (=7), number of elite model (=5), train-test set split, optimization scheme, elite-model selection criterion, and sampling method as in Yu et al. [15]. To save computation, we define each epoch of model training as 1000 mini-batch gradient steps, instead of the original `n_train / batch_size`.

As in Yu et al. [15], the input to our dynamic model is $\left((s,a) - \mu_{(s,a)}\right)/\sigma_{(s,a)}$, where $\mu_{(s,a)}$ and $\sigma_{(s,a)}$ are respectively coordinate-wise mean and standard deviation of (s,a) in the offline dataset. We follow Kidambi et al. [18] and Matsushima et al. [70] to define the learning target as $\left((r,\Delta s) - \mu_{(r,\Delta s)}\right)/\sigma_{(r,\Delta s)}$, where Δs denote $s' - s$, $\mu_{(r,\Delta s)}$ and $\sigma_{(r,\Delta s)}$ denote the coordinate-wise mean and standard deviation of $(r,\Delta s)$ in the offline dataset. As in prior work using Gaussian probabilistic ensemble on model-based RL [69, 14, 15, 20, 17], we use a double-head architecture for our dynamic model, where the two output heads represent the mean and log-standard-deviation of the normal distribution of the predicted output, respectively. We augment the maximum likelihood loss in Yu et al. [15] as weighted maximum likelihood loss, independently for each model in the ensemble. Specifically, the dynamic model is initially trained using un-weighted maximum likelihood loss. We then employ the warm-start strategy to start each subsequent weighted re-training from the current dynamic model, with the weights $\omega(s,a)$ from the latest marginal importance weight model. To unify the learning rate across each run of model (re-)training, the weights $\omega(s,a)$'s are normalized so that their mean is 1 across the offline dataset.

To improve training stability, we are motivate by Hishinuma and Senda [45] to heuristically augment the environmental termination function by (1) $\mathrm{any}(|\hat{s}'| > 2 \cdot \max_{s,s' \in \mathcal{D}_{\mathrm{env}}} \{|s|, |s'|\})$ where all operations are coordinate-wise, *i.e.*, whether any coordinate of the predicted next state is outside of this relaxed observation-range in the offline dataset; and (2) $|\hat{r}(s,a)| > r_{\mathrm{range}} \triangleq \max\left(|r_{\min} - 10 \cdot \sigma_r|, |r_{\max} + 10 \cdot \sigma_r|\right)$ where $r_{\min}, r_{\max}, \sigma_r$ are the minimum, maximum, and standard deviation of rewards in the offline dataset, *i.e.*, whether the predicted reward is outside of this relaxed reward-range in the offline dataset. To penalize the policy for visiting out-of-distribution state-action pair, we modify $\hat{r}(s,a)$ to be $-r_{\mathrm{range}}$ if either (1) or (2) is violated. No modification for $\hat{r}(s,a)$ is performed if only the environmental termination is triggered.

D.1.2 Marginal importance weight training

We follow the OPE literature [26, 41, 29] to assume that the initial state-distribution μ_0 is known. In the implementation, we augment $\mathcal{D}_{\mathrm{env}}$ with 10^5 samples from the ground-truth μ_0. To enhance training stability, we are motivated by the double Q-learning [71] to use a target marginal importance weight network $\omega'(s,a)$ and a target critic network $Q'(s,a)$ in the implementation of MIW training. We employ the warm-start strategy so that each retraining of the marginal importance weight starts from the current $\omega(s,a)$. On each run of the weight-retraining, Q' is initialized as the current critic target $Q_{\boldsymbol{\theta}'}$. Since the weights $\omega(s,a)$'s are normalized to have mean 1 in the weighted re-training of the dynamic model, to narrow the gap between the learning and the application of the weights, we add a constraint into the training of ω that its mean on each mini-batch is upper bounded by some constant $g_{\mathrm{constraint}}$. Specifically, on each mini-batch gradient step, MIW is trained by the following

constraint optimization

$$\begin{aligned}
&\arg\min_{\omega} \left(\frac{1}{|\mathcal{B}|} \sum_{(s_i,a_i)\in\mathcal{B}} \omega\left(s_i,a_i\right) \cdot Q_{\boldsymbol{\theta}}\left(s_i,a_i\right) - y \right)^2, \\
&\text{s.t.} \quad \frac{1}{|\mathcal{B}|} \sum_{(s_i,a_i)\in\mathcal{B}} \omega\left(s_i,a_i\right) \leq g_{\text{constraint}}, \\
&\text{where} \;\; y = \gamma \cdot \frac{1}{|\mathcal{B}|} \sum_{\substack{(s_i,a_i,s_i')\in\mathcal{B} \\ a_i'\sim\pi_{\boldsymbol{\phi}}(\cdot|s_i')}} \omega'\left(s_i,a_i\right) \cdot Q'\left(s_i',a_i'\right) + (1-\gamma) \cdot \frac{1}{|\mathcal{B}_{\text{init}}|} \sum_{\substack{s_0\in\mathcal{B}_{\text{init}} \\ a_0\sim\pi_{\boldsymbol{\phi}}(\cdot|s_0)}} Q'\left(s_0,a_0\right),
\end{aligned} \tag{22}$$

where $g_{\text{constraint}}$ is conveniently chosen as 10 and $\mathcal{B}, \mathcal{B}_{\text{init}} \sim \mathcal{D}_{\text{env}}$. To optimize Eq. (22), we use the constraint optimization method in Gong et al. [72] with base optimizer Adam [73]. For training stability, the gradient norm is clipped to be bounded by 1.

As in double Q-learning, we update Q' towards $Q_{\boldsymbol{\theta}}$ and ω' towards ω via exponential moving average with rate 0.01 after each gradient step. We use batch sizes $|\mathcal{B}| = 1024$ and $|\mathcal{B}_{\text{init}}| = 2048$ to train the marginal importance weight model $\omega(s,a)$.

D.1.3 Actor-critic training

Our policy-training consists of the following six parts.

Generate synthetic data using dynamic model. We follow Janner et al. [14] and Yu et al. [15] to perform h-step rollouts branching from the offline dataset $\mathcal{D}_{\text{env}}$, using the learned dynamic $\widehat{P}$ and the current policy $\pi_{\boldsymbol{\phi}}$. The generated data is added to a separate replay buffer $\mathcal{D}_{\text{model}}$.

To save computation, we generate synthetic data every `rollout_generation_freq` iterations.

Critic training. Sample mini-batch $\mathcal{B} \sim \mathcal{D} = f \cdot \mathcal{D}_{\text{env}} + (1-f) \cdot \mathcal{D}_{\text{model}}$, with $f \in [0,1]$. For each $s' \in \mathcal{B}$, sample one corresponding actions $a' \sim \pi_{\boldsymbol{\phi}'}(\cdot \mid s')$. Calculate the estimate of the action-value of the next state-action pair as

$$\widetilde{Q}\left(s,a\right) \triangleq c \min_{j=1,2} Q_{\boldsymbol{\theta}_j'}\left(s',a'\right) + (1-c) \max_{j=1,2} Q_{\boldsymbol{\theta}_j'}\left(s',a'\right).$$

Calculate the critic-learning target defined as

$$\widetilde{Q}\left(s,a\right) \leftarrow r(s,a) + \gamma \cdot \widetilde{Q}\left(s,a\right) \cdot \left(\left| \widetilde{Q}\left(s,a\right) \right| < 2000 \right), \tag{23}$$

where 2000 is conveniently chosen to enhance the termination condition stored in $\mathcal{B}$ for training stability. Then minimize the critic loss with respect to the double critic networks, over $(s,a) \in \mathcal{B}$, with learning rate $\eta_{\boldsymbol{\theta}}$,

$$\forall\, j = 1,2,\; \arg\min_{\boldsymbol{\theta}_j} \frac{1}{|\mathcal{B}|} \sum_{(s,a)\in\mathcal{B}} \text{Huber}\left(Q_{\boldsymbol{\theta}_j}\left(s,a\right), \widetilde{Q}(s,a) \right),$$

where $\text{Huber}(\cdot)$ denote the Huber-loss with threshold conveniently chosen as 500. Finally, the norm of gradient for critic update is clipped to be bounded by 0.1 for training stability.

As a remark, the mentioned enhanced termination condition, Huber loss and gradient-norm clipping are engineering designs for stable training in combating for not knowing the true reward function. We use a unified setting for these designs across all tested datasets on all experiments, and thus a per-dataset tuning may further improve our results.

Discriminator training. The "fake" samples $\mathcal{B}_{\text{fake}}$ for discriminator training is formed by the following steps: first, sample $|\mathcal{B}|$ states $s \sim \mathcal{D}$; second, for each s get one corresponding action $a \sim \pi_{\boldsymbol{\phi}}\left(\cdot \mid s\right)$; third, for each (s,a) get an estimate of the next state s' using $\widehat{P}$ as $s' \sim \widehat{P}(\cdot \mid s,a)$, with terminal states removed; fourth, sample one next action a' for each next state s' via $a' \sim \pi_{\boldsymbol{\phi}}\left(\cdot \mid s'\right)$. The "fake" samples $\mathcal{B}_{\text{fake}}$ is subsequently formed by

$$\mathcal{B}_{\text{fake}} \triangleq \begin{bmatrix} (s, & a) \\ (s', & a') \end{bmatrix}.$$

Note that terminal states are removed since by definition, no action choices are needed on them. The "true" samples $\mathcal{B}_{\text{true}}$ consists of $|\mathcal{B}_{\text{fake}}|$ state-action samples from $\mathcal{D}_{\text{env}}$.

The discriminator $D_{\boldsymbol{\psi}}$ is optimized with learning rate $\eta_{\boldsymbol{\psi}}$ as

$$\arg\max_{\boldsymbol{\psi}} \frac{1}{|\mathcal{B}_{\text{true}}|} \sum_{(s,a)\sim\mathcal{B}_{\text{true}}} \log D_{\boldsymbol{\psi}}(s,a) + \frac{1}{|\mathcal{B}_{\text{fake}}|} \sum_{(s,a)\sim\mathcal{B}_{\text{fake}}} \log\left(1 - D_{\boldsymbol{\psi}}(s,a)\right).$$

Actor training. The policy is updated once every k updates of critic and discriminator. Using the "fake" samples $\mathcal{B}_{\text{fake}}$ and the discriminator $D_{\boldsymbol{\psi}}$, the generator loss for actor training is calculated as

$$\mathcal{L}_g(\boldsymbol{\phi}) = \frac{1}{|\mathcal{B}_{\text{fake}}|} \sum_{(s,a)\in\mathcal{B}_{\text{fake}}} \left[\log\left(1 - D_{\boldsymbol{\psi}}\left(s,a\right)\right)\right].$$

The policy is optimized with learning rate $\eta_{\boldsymbol{\phi}}$ as

$$\arg\min_{\boldsymbol{\phi}} -\lambda \cdot \frac{1}{|\mathcal{B}|} \sum_{s\in\mathcal{B}, a\sim\pi_{\boldsymbol{\phi}}(\cdot\,|\,s)} \left[\min_{j=1,2} Q_{\boldsymbol{\theta}_j}(s,a)\right] + \mathcal{L}_g(\boldsymbol{\phi}),$$

where in all tested datasets the regularization coefficient $\lambda \triangleq 10/Q_{avg}$ with soft-updated Q_{avg} and the penalty coefficient 10 conveniently chosen, similar to Fujimoto and Gu [40].

Soft updates. We follow Fujimoto et al. [10] to soft-update the target-network parameter and the Q_{avg},

$$\begin{aligned}
\boldsymbol{\phi}' &\leftarrow \beta\boldsymbol{\phi} + (1-\beta)\boldsymbol{\phi}', \\
\boldsymbol{\theta}'_j &\leftarrow \beta\boldsymbol{\theta}_j + (1-\beta)\boldsymbol{\theta}'_j, \quad \forall\, j = 1,2, \\
Q_{avg} &\leftarrow \beta\frac{1}{|\mathcal{B}|} \sum_{s\in\mathcal{B}, a\sim\pi_{\boldsymbol{\phi}}(\cdot\,|\,s)} \left|\min_{j=1,2} Q_{\boldsymbol{\theta}_j}\left(s,a\right)\right| + (1-\beta)Q_{avg}.
\end{aligned}$$

Soft-updates are performed after each iteration.

Warm-start step. We follow Kumar et al. [43] and Yue et al. [74] to warm-start our policy training in the first N_{warm} epochs. In this step, the policy is optimized *w.r.t.* the generator loss $\mathcal{L}_g(\boldsymbol{\phi})$ only, *i.e.*, $\arg\min_{\boldsymbol{\phi}} \mathcal{L}_g(\boldsymbol{\phi})$. The learning rate $\eta_{\boldsymbol{\phi}}$ in the warm-start step is the same as the normal training step.

D.2 Details for the continuous-control experiment

Datasets. We evaluate algorithms on the continuous control tasks in the D4RL benchmark [30]. Due to limited computational resources, we select therein a representative set of datasets. Specifically, **(1)** we select the "medium-expert," "medium-replay," and "medium" datasets for the Hopper, HalfCheetah, and Walker2d tasks in the Gym-MuJoCo domain, which are commonly used benchmarks in prior work [11, 12, 75, 43]. As in recent literature [20, 76, 77], we do not test on the "random" and "expert" datasets, as they are considered as less practical [70] and can be simply solved by directly applying standard off-policy RL algorithms [13] and behavior cloning on the offline dataset. We note that a data-quality agnostic setting may not align with practical offline RL applications, *i.e.*, one typically should know the quality of the offline datasets, *e.g.*, whether it is collected by random policy or by experts. **(2)** Apart from the Gym-MuJoCo domain, we further consider the Maze2D domain of tasks[1] for the non-Markovian data-collecting policy and the Adroit tasks[2] [78] for their high dimensionality and sparse rewards.

Evaluation protocol. Our algorithm is trained for 1000 epochs, where each epoch consists of 1000 mini-batch gradient descent steps. After each epoch of training, a rollout of 10 episodes is performed. Both our algorithm and baselines are run under five random seeds $\{0,1,2,3,4\}$, and we report the mean and standard deviation of the final rollouts across five seeds. We follow Fu et al. [30] to rerun the baselines under the recommended hyperparameter setting, including dataset-specific hyperparameters if available.

[1] We use the tasks "maze2d-umaze," "maze2d-medium," and "maze2d-large."

[2] We use the tasks "pen-human," "pen-cloned," "pen-expert," and "door-expert."

Results of CQL. We run CQL using the implementation suggested by Kostrikov et al. [77], which only provide hyperparameters for the Gym-MuJoCo and AntMaze domains. For the Maze2D and Adroit datasets, we run both sets of recommended hyperparameters and per-dataset select the better result to report. We note that hyperparameter settings for the Maze2D and Adroit datasets are also not provided in the original CQL implementation. An in-depth tuning of CQL on these two domains of datasets is beyond the scope of this paper.

Implicit policy implementation. We follow White [79] to choose the noise distribution as the isotropic normal, *i.e.*, $p_z(z) \triangleq \mathcal{N}\left(\mathbf{0}, \sigma^2_{\text{noise}} \boldsymbol{I}\right)$, where the default setting of the dimension of z is $\dim(z) = \texttt{min}(10, \texttt{state_dim}//2)$ and of the noise standard deviation is $\sigma_{\text{noise}} = 1$. In the forward pass of the implicit policy, for a given state s, a noise sample $z \sim p_z(z)$ is first independently drawn. Then s is concatenated with z and the resulting $[s, z]$ is inputted into the (deterministic) policy network to sample stochastic action.

Terminal states. In practice, the episodes in the offline dataset have finite horizon. Thus, special treatment is needed for those terminal states in calculating the Bellman update target. We combine common practice [80, 81] with our critic update target Eq. (23) as

$$\widetilde{Q}(s,a) = \begin{cases} r(s,a) + \gamma \cdot \widetilde{Q}(s,a) \cdot \left(\left|\widetilde{Q}(s,a)\right| < 2000\right) & \text{if } s' \text{ is a non-terminal state} \\ r(s,a) & \text{if } s' \text{ is a terminal state} \end{cases},$$

since the action-value function is undefined on the terminal s' due to no action choice therein.

Our network architecture is presented in Appendix D.2.1. To save computation, we use simple neural networks as in Fujimoto et al. [11]. A fine-tuning of the noise distribution $p_z(z)$, the network architecture, the optimizer hyperparameters, *etc.*, is left for future work.

D.2.1 Details for the implementation via GAN

With the training techniques for GAN suggested by previous work in generative modeling, we are able to stably and effectively train the GAN structure on data with moderate dimension, *e.g.*, the D4RL datasets we consider. In this paper, we adopt the following techniques from the literature.

- Following Goodfellow et al. [34], $\pi_{\boldsymbol{\phi}}$ is trained to maximize $\mathbb{E}_{(s,a)\in\mathcal{B}_{\text{fake}}}\left[\log\left(D_{\boldsymbol{\psi}}(s,a)\right)\right]$.
- We are motivated by Radford et al. [82] to use LeakyReLU activation function in both the generator and the discriminator, with `negative_slope` being the default value 0.01.
- For stable training, we follow Radford et al. [82] to use a reduced momentum term $\beta_1 = 0.4$ in the Adam optimizers of the policy and the discriminator networks; and to use learning rates $\eta_{\boldsymbol{\phi}} = \eta_{\boldsymbol{\psi}} = 2 \times 10^{-4}$.
- To avoid discriminator overfitting, we are motivated by Salimans et al. [83] and Goodfellow [84] to use one-sided label smoothing with soft and noisy labels. Specifically, random numbers in $[0.8, 1.0)$ are used as the labels for the "true" sample $\mathcal{B}_{\text{true}}$. No label smoothing is needed for the "fake" sample $\mathcal{B}_{\text{fake}}$, *i.e.*, their labels are all 0.
- Binary cross entropy loss between the labels and the discriminator outputs is used as the loss for discriminator training in GAN.
- We are motivated by TD3 [10] and GAN to update the policy $\pi_{\boldsymbol{\phi}}(\cdot \mid s)$ once per k updates of the critics and the discriminator.

Table 4 shows the shared hyperparameters for all datasets in our experiments. *Many of these hyperparameters follow the literature without further tuning*. For example, we use $\eta_{\boldsymbol{\phi}} = \eta_{\boldsymbol{\psi}} = 2 \times 10^{-4}$ as in Radford et al. [82], $\eta_{\boldsymbol{\theta}} = 3 \times 10^{-4}$ and $N_{\text{warm}} = 40$ as in Kumar et al. [43], $c = 0.75$ as in Fujimoto et al. [11], policy frequency $k = 2$ as in Fujimoto et al. [10], $f = 0.5$ as in Yu et al. [17] and model learning rate $\eta_{\widehat{P}} = 0.001$ as in Yu et al. [15]. Unless specified, the shared and the non-shared hyperparameters are used throughout the main results and ablation study.

Since the tested datasets possess diverse nature, we follow the common practice in offline model-based RL to perform gentle dataset-specific hyperparameter tuning. Details are at the below.

Table 4: Shared hyperparameters for the GAN implementation.

Hyperparameter	Value
Optimizer	Adam
Training iterations	10^6
Training iterations for $\omega(s,a)$	10^5
Model retrain period	per 100 epochs
Learning rate $\eta_{\boldsymbol{\theta}}$	3×10^{-4}
Learning rate $\eta_{\boldsymbol{\phi}}$, $\eta_{\boldsymbol{\psi}}$	2×10^{-4}
Learning rate $\eta_{\widehat{P}}$	1×10^{-3}
Learning rate η_{ω} for $\omega(s,a)$	1×10^{-6}
Penalty coefficient	10
Batch size	512 (as in Lee et al. [42])
Discount factor γ	0.99
Target network update rate β	0.005
Weighting for clipped double Q-learning c	0.75
Noise distribution $p_z(z)$	$\mathcal{N}\left(\mathbf{0}, \sigma^2_{\text{noise}}\boldsymbol{I}\right)$
Default value of σ_{noise}	1
Policy frequency k	2
Rollout generation frequency	per 250 iterations
Number of model-rollout samples per iteration	128
Rollout retain epochs	5
Real data percentage f	0.5
Warm start epochs N_{warm}	40
Random seeds	$\{0,1,2,3,4\}$

We are motivated by Janner et al. [14] and Yu et al. [15] to consider the rollout horizon $h \in \{1,3,5\}$. We use $h=1$ for halfcheetah-medium, hopper-medium, hopper-medium-replay, maze2d-large, pen-expert, door-expert, pen-human; $h=3$ for hopper-medium-expert, walker2d-medium, halfcheetah-medium-replay, maze2d-umaze, maze2d-medium; and $h=5$ for halfcheetah-medium-expert, walker2d-medium-expert, walker2d-medium-replay, pen-cloned.

Zheng and Zhou [85] and Zhang et al. [86] show that a larger noise dimension, *e.g.*, 50, can help learning a more flexible distribution, while a smaller noise dimension *e.g.*, 1 can make the distribution leaning towards deterministic. Hence we use the default noise dimension for the tasks: hopper-medium-expert, maze2d-umaze, door-expert; noise dimension 50 for the tasks: halfcheetah-medium-expert, walker2d-medium-expert, halfcheetah-medium, hopper-medium, walker2d-medium, halfcheetah-medium-replay, hopper-medium-replay, walker2d-medium-replay, maze2d-medium, maze2d-large, pen-expert; and noise dimension 1 for the tasks: pen-cloned, pen-human .

Due to the narrow data distribution, we use $\sigma_{\text{noise}} = 0.001$ for datasets pen-human and pen-cloned to avoid training divergence.

We state below the network architectures of actor, critic, discriminator, and MIW in the implementation via GAN. Motivated by the clipped double Q-learning, two critic networks with the same architecture are used.

Actor

Linear(state_dim + noise_dim, 400)
LeakyReLU
Linear(400, 300)
LeakyReLU
Linear(300, action_dim)
max_action $\times$ tanh

Critic

Linear(state_dim + action_dim, 400)
LeakyReLU
Linear(400, 300)
LeakyReLU
Linear(300, 1)

Discriminator	MIW
Linear(state_dim + action_dim, 400)	Linear(state_dim + action_dim, 400)
LeakyReLU	LeakyReLU
Linear(400, 300)	Linear(400, 300)
LeakyReLU	LeakyReLU
Linear(300, 1)	Linear(300, 1)
Sigmoid	(softplus($\cdot - 10^{-8}$) + 10^{-8}).power(α)

In the MIW network, α is the smoothing exponent to smooth the distribution of the weight estimates across the offline dataset, whose design is motivated by Hishinuma and Senda [45]. We fix α as 0.5 on MuJoCo datasets and 0.2 on Maze2D and Adroit datasets. The output of the MIW network prior to $\texttt{power}(\alpha)$ is lower bounded by 10^{-8} to avoid gradient explode. We follow Dai et al. [87] to initialize the weights of last layer of the MIW network from $\mathrm{Uniform}\,(-0.003, 0.003)$, and bias as 0.

D.3 Details for the implementation of alternative MIW estimation methods in Section 4.2

We implement and train the alternative methods VPM, GenDICE, and DualDICE in Section 4.2 based on the respective papers and source codes. To stabilize training, the design of target MIW network in Section D.1.2 is applied wherever applicable. The target MIW network is hard-updated, as suggested by the papers. The ν network in GenDICE and DualDICE has the same architecture as the critic network and has its own hard-updated target network. We follow the official implementation of GenDICE and DualDICE to clip the gradient values by 1. For the convex function f in DualDICE, we use $f(x) = \frac{2}{3}\,|x|^{3/2}$, as suggested by Section 5.3.

E Potential negative societal impacts

Since offline RL has strong connection with the supervised learning, offline RL methods, including this paper, are subject to the bias in the offline dataset. The data-bias can be exploited by both the dynamic-model learning and the training of the policy.

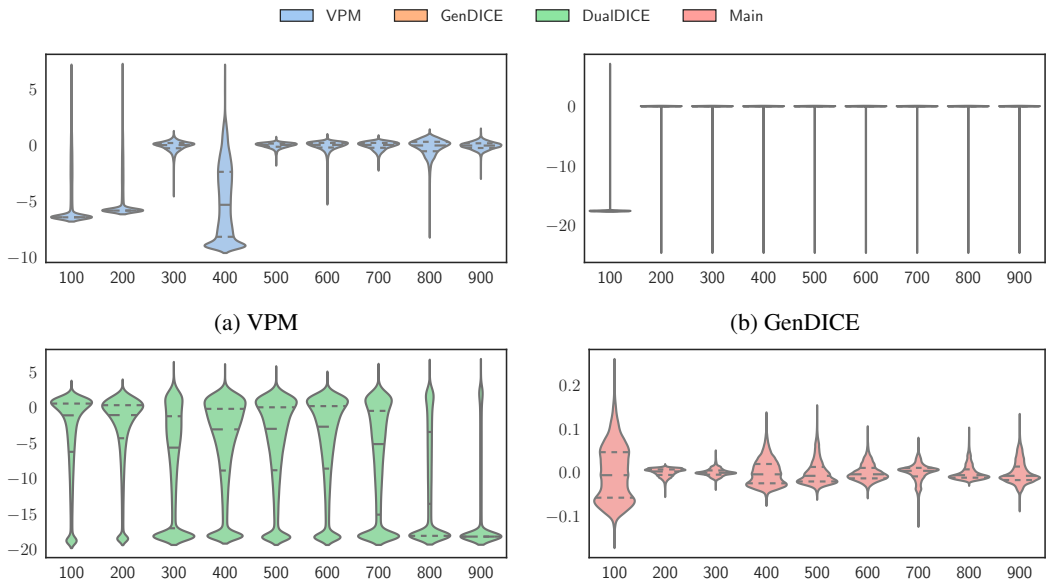
F Computing resources

We ran all experiments on 4 NVIDIA Quadro RTX 5000 GPUs.

G Distributions of the MIWs produced by our method and the alternatives

Figure 3 shows the distribution plots of the $\log\,(\mathrm{MIW})$ of the entire dataset generated by our method, and by the variants with the three alternative MIW estimation methods in Section 4.2 **(b)** over the training process on the example at Figure 2. As in Figure 2, the MIWs are normalized to have a mean on the entire dataset 1. We note that the MIWs obtained by these four methods have very different scales, which indicates the instability of the alternative methods compared with ours, as discussed in Section 4.2 **(b)**. Specifically, as shown on Figure 3b, the MIWs estimated by the GenDICE variant degenerate, in a sense that most of the MIWs are ≈ 0 at the begining and are 1 later on, with long tails on the MIW distributions. Figure 3c shows that, for the DualDICE variant, the distribution of MIWs on the entire dataset gradually concentrates on very small and very large values. This indicates the degeneration of the MIWs obtained from DualDICE. As in Section 4.2 **(b)**, Figure 3a shows that the MIWs from the VPM variant are poorly-distributed at the beginning, while behave relatively better later on. By contrast, in Figure 3d, the MIWs from our method are well-shaped and concentrate around the mean 1 [3] over the entire training process. This possibly explains why our method achieves better results, as discussed in Section 4.2 **(b)**.

[3] 0 in the $\log\,(\mathrm{MIW})$ plots.



(a) VPM (b) GenDICE

(c) DualDICE (d) Our method ("Main")

Figure 3: Plots of the distribution of the $\log$ (MIW) from our method ("Main") and from the variants with the three alternative MIW estimation methods in Section 4.2 **(b)** over the training process on the example at Figure 2. The dash line represents the quartile of the distribution. Recall that we retrain the MIW ω every 100 epochs. The x-axis represents the number of epochs, and the y-axis the $\log$ (MIW) values. Other details follows Figure 2. We note the different scales in the y-axis of these four plots.

H Additional ablation

Table 5 provides additional ablation study on several building blocks of our main method. In Table 5, we conducted following variants to demonstrate the effectiveness of our framework:

- **Main** denotes the results of our main method in Section 4.1.
- **NW** denotes the No-Weights variant in Section 4.2 **(a)**.
- **WPR** denotes the Weighted Policy Regularizer variant in Section 4.2 **(c)**.
- **KL-Dual** denote the variant changing the JSD regularizer in Main to the $\mathrm{KL}\left(P^*(s' \,|\, s,a)\pi_b(a' \,|\, s') \,||\, \widehat{P}(s' \,|\, s,a)\pi(a' \,|\, s')\right)$ term in Theorem 1, where $(s,a) \sim d^{P^*}_{\pi_b,\gamma}$ and the KL term is estimated via the dual representation $\mathrm{KL}(p \,||\, q) = \sup_T \left\{\mathbb{E}_p[T] - \log(\mathbb{E}_q[e^T])\right\}$ [33].
- **KL-Dual+WPR** denote the variant changing the JSD regularizer in Main to the weighted KL term $D_\pi(P^*, \widehat{P})$ in Theorem 1.
- **Gaussian** denote the variant changing the implicit policy in Main to the Gaussian policy.
- **No Reg.** denote the variant removing the proposed regularizer in the policy learning.
- **No model-rollout** denotes the variant of no rollout data in policy learning.

Apart from the discusstion in Section 4.2 **(a)** and Section 4.2 **(c)** on the **NW** and the **WPR** variants. We see that changing the proposed JSD regularizer in Section 3.2 to KL-dual based regularizer breaks the policy learning. This may be related to the unstable estimation of KL-dual discussed in Section 3.2. Changing the implicit policy to the Gaussian policy generally leads to worse performance. The performance difference is especially significant on the Maze2D and Adroit datasets. This aligns with the observation in Yang et al. [88] that a uni-model Gaussian policy may be insufficient to capture the necessary multiple action modes in the Maze2D and Adroit datasets, while an implicit policy is likely to be capable. Removing the proposed regularization term breaks the policy learning,

showing the efficacy of the proposed regularizer. Finally, removing rollout data in the policy learning generally leads to worse performance and larger standard deviation. This shows the benefit of adding the model-rollout data, that it can mitigate the off-policy issue by taking into account the rollouts of the learned policy.

Table 5: Normalized returns for experiments on the D4RL tasks. Mean and standard deviation across five random seeds are reported.
Main denotes the results of our main method in Section 4.1. **NW** denotes the No-Weights variant in Section 4.2 **(a)**. **WPR** denotes the Weighted Policy Regularizer variant in Section 4.2 **(c)**. **KL-Dual** denotes the variant changing the JSD regularizer in Main to the KL term in Theorem 1. **KL-Dual+WPR** denotes the variant changing the JSD regularizer in Main to the weighted KL term $D_\pi(P^*, \widehat{P})$ in Theorem 1. **Gaussian** denotes the variant changing the implicit policy in Main to the Gaussian policy. **No Reg.** denotes the variant removing the proposed regularizer in the policy learning. **No model-rollout** denotes the variant of no rollout data in policy learning.
Here, "hcheetah" denotes "halfcheetah", "med" denotes "medium", "rep" denotes "replay", and "exp" denotes "expert".

Task Name	Main	NW	WPR	KL-Dual	KL-Dual+WPR	Gaussian	No Reg.	No model-rollout
maze2d-umaze	55.8 $\pm$ 3.9	31.3 $\pm$ 25.3	14.4 $\pm$ 21.4	-3.3 $\pm$ 18.6	-0.9 $\pm$ 16.5	41.5 $\pm$ 26.3	-13.9 $\pm$ 1.3	48.4 $\pm$ 14.7
maze2d-medium	107.2 $\pm$ 45.0	85.4 $\pm$ 28.6	5.0 $\pm$ 8.3	16.9 $\pm$ 17.9	4.8 $\pm$ 23.2	59.5 $\pm$ 29.5	83.3 $\pm$ 30.1	120.7 $\pm$ 41.9
maze2d-large	190.8 $\pm$ 34.2	131.4 $\pm$ 63.0	104.3 $\pm$ 48.1	-1.0 $\pm$ 4.0	-0.5 $\pm$ 7.4	106.1 $\pm$ 51.0	0.6 $\pm$ 0.7	121.0 $\pm$ 26.5
Average Maze2D	117.9	82.7	41.2	4.2	1.1	69	23.3	96.7
hcheetah-med	51.7 $\pm$ 0.4	51.5 $\pm$ 0.3	50.1 $\pm$ 0.4	16.2 $\pm$ 13.0	28.3 $\pm$ 2.9	49.2 $\pm$ 0.4	30.5 $\pm$ 26.3	51.9 $\pm$ 0.6
walker2d-med	83.1 $\pm$ 1.8	77.3 $\pm$ 6.0	41.8 $\pm$ 18.9	0.3 $\pm$ 1.5	2.8 $\pm$ 3.4	72.5 $\pm$ 6.7	3.5 $\pm$ 4.4	83.3 $\pm$ 4.4
hopper-med	58.9 $\pm$ 7.9	55.0 $\pm$ 8.9	52.5 $\pm$ 21.6	7.6 $\pm$ 3.4	6.9 $\pm$ 7.0	52.1 $\pm$ 3.7	6.2 $\pm$ 9.0	42.8 $\pm$ 22.8
hcheetah-med-rep	44.6 $\pm$ 0.7	44.6 $\pm$ 0.3	44.4 $\pm$ 0.4	23.1 $\pm$ 5.9	18.9 $\pm$ 8.0	41.1 $\pm$ 0.7	67.3 $\pm$ 2.7	41.9 $\pm$ 1.7
walker2d-med-rep	81.5 $\pm$ 3.0	74.6 $\pm$ 4.2	55.8 $\pm$ 14.3	2.3 $\pm$ 4.4	1.1 $\pm$ 2.5	54.1 $\pm$ 5.8	10.6 $\pm$ 8.9	13.8 $\pm$ 13.4
hopper-med-rep	91.1 $\pm$ 9.5	75.4 $\pm$ 19.1	40.1 $\pm$ 11.1	6.0 $\pm$ 5.7	5.5 $\pm$ 7.3	71.3 $\pm$ 19.7	9.9 $\pm$ 6.5	85.8 $\pm$ 16.8
hcheetah-med-exp	90.2 $\pm$ 2.0	88.0 $\pm$ 5.1	80.1 $\pm$ 5.9	20.7 $\pm$ 11.1	19.1 $\pm$ 9.4	85.0 $\pm$ 2.9	7.5 $\pm$ 12.6	79.0 $\pm$ 7.2
walker2d-med-exp	107.7 $\pm$ 2.1	99.1 $\pm$ 7.9	62.9 $\pm$ 17.6	0.3 $\pm$ 1.0	1.3 $\pm$ 2.7	90.6 $\pm$ 6.6	2.6 $\pm$ 0.8	104.7 $\pm$ 3.4
hopper-med-exp	93.0 $\pm$ 19.7	41.7 $\pm$ 16.9	11.3 $\pm$ 7.6	3.1 $\pm$ 1.4	4.5 $\pm$ 3.3	59.1 $\pm$ 25.5	12.4 $\pm$ 9.6	48.7 $\pm$ 17.4
Average MuJoCo	78	67.5	48.8	8.8	9.8	63.9	16.7	61.3
pen-human	20.6 $\pm$ 10.7	11.4 $\pm$ 8.6	24.8 $\pm$ 14.7	9.3 $\pm$ 4.0	4.7 $\pm$ 7.6	8.5 $\pm$ 5.6	1.6 $\pm$ 5.4	8.4 $\pm$ 9.0
pen-cloned	57.4 $\pm$ 19.2	35.3 $\pm$ 15.7	35.5 $\pm$ 13.8	22.4 $\pm$ 12.5	35.7 $\pm$ 12.8	39.3 $\pm$ 16.4	39.1 $\pm$ 21.3	3.5 $\pm$ 6.2
pen-exp	138.3 $\pm$ 6.6	129.6 $\pm$ 15.4	102.0 $\pm$ 36.3	31.2 $\pm$ 15.5	30.1 $\pm$ 18.3	102.0 $\pm$ 14.9	5.2 $\pm$ 7.6	109.2 $\pm$ 40.9
door-exp	96.3 $\pm$ 9.8	73.5 $\pm$ 30.9	20.7 $\pm$ 16.4	0.1 $\pm$ 1.4	0.2 $\pm$ 0.4	20.3 $\pm$ 21.2	-0.3 $\pm$ 0.1	53.9 $\pm$ 41.4
Average Adroit	78.2	62.4	45.8	15.8	17.7	42.5	11.4	43.8
Average All	85.5	69.1	46.6	9.7	10.2	59.5	16.6	63.6

I More discussion on the choice of test function for MIW

We first discuss the original minimax optimization loss [26, 29, 89] for learning density ratio $\omega(s,a)$, given an offline dataset $\mathcal{D}_{\text{env}}$:

$$\min_{\omega\in W}\max_{f\in\mathcal{F}}\Big\{(1-\gamma)\mathbb{E}_{\pi,\mu_0}[f(s,a)]+ \mathbb{E}_{(s,a,s')\sim\mathcal{D}_{\text{env}},a'\sim\pi(\cdot|s')}\left[\gamma\omega(s,a)f(s',a')-\omega(s,a)f(s,a)\right]\Big\}, \tag{24}$$

where $f\in\mathcal{F}$ is a test function similar to the discriminators in adversarial training.

A number of previous works on learning $\omega(s,a)$ (*e.g.* DualDICE [28], GenDICE [29]) can be viewed as variants of the above minimax optimization, with additional regularization term on the test function $f(s,a)$. For example, there is an additional $\frac{1}{2}f^2(x)$ term in DualDICE to penalize $f(x)$ such that the output of $f(x)$ will not be too large, which avoids the divergence during the optimization process.

However, the optimization in Eq. (24) for $\omega(s,a)$ is quite unstable due to the following two reasons: 1) the minimax optimization itself is already a challenging optimization problem; 2) When we fix $f(s,a)$ and optimize $\omega(s,a)$, the gradients for ω that come from $\gamma\omega(s,a)f(s',a')$ and $w(s,a)f(s,a)$ are quite close to each other, especially when γ is close to 1 (*e.g.*, 0.99). This makes the whole

gradient for ω quite small and unstable, especially if we use stochastic gradient descent to optimize ω. The empirical distributions of MIW obtained by DualDICE and GenDICE in Fig. 3 can also indicate the optimization instability of these two methods.

As a result, we hope to get rid of the minimax optimization and stabilize the gradient estimation.

Another closely related work that also tries to address the above-mentioned issues is VPM [38], where the authors set the test function $f(s,a)$ as the density ratio function ω itself, based on the observation that the optimum test function under the f-divergence is the density ratio, and use variational power iteration to solve for ω. We refer to Wen et al. [38] for the detailed derivations. Empirically, we can observe the distribution of ω obtained by VPM in Fig. 3(a) tends to be more stable than those obtained by DualDICE and GenDICE.

Instead, our choice of test function is based on the following constrain optimization formulation of off-policy evaluation [35, 89]:

$$\begin{aligned} \min_{Q} \quad & (1-\gamma)\mathbb{E}_{s\sim\mu_0, a\sim\pi(\cdot|s)}\left[Q(s,a)\right] \\ \text{s.t.} \quad & Q(s,a) \geq r(s,a) + \gamma\mathbb{E}_{s'\sim P(\cdot|s,a), a'\sim\pi(\cdot|s')}\left[Q(s',a')\right] \quad \forall\,(s,a)\in\mathbb{S}\times\mathbb{A}\,. \end{aligned}$$

We can show that $Q^\pi(s,a)$ is the optimal solution of above primal optimization problem. The dual form of above optimization problem can be reformulated as

$$\begin{aligned} \max_{d} \quad & \mathbb{E}_d[r(s,a)] \\ \text{s.t.} \quad & d(s',a') = (1-\gamma)\mu(s')\pi(a'|s') + \gamma\sum_{s,a} d(s,a)p(s'|s,a)\pi(a'|s')\,, \quad \forall (s',a')\in\mathbb{S}\times\mathbb{A}\,. \end{aligned}$$

Also, we know that the stationary distribution $d^P_{\pi,\gamma}$ is the solution of above dual problem.

For the dual problem, we introduce the Lagrangian multiplier $Q(s,a)$ for each (s,a) tuple, and the Lagrangian function can be written as

$$L(d,Q) := \mathbb{E}_d[r(s,a)] + (1-\gamma)\mathbb{E}_{\pi,\mu_0}[Q(s,a)] + \mathbb{E}_{d,P,\pi}[\gamma Q(s',a') - Q(s,a)]\,.$$

As we know from previous observations, the *optimal Lagrangian multiplier* for the Lagrangian function is $Q^\pi(s,a)$ (also the solution for the primal problem). We change d to the density ratio function ω, we have

$$\begin{aligned} L(\omega,Q) := & \underbrace{\mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{env}}}[\omega(s,a)r(s,a)]}_{\text{①: Estimator for } J(\pi) \text{ with } \omega(s,a)} \\ & + \underbrace{(1-\gamma)\mathbb{E}_{\pi,\mu_0}[Q(s,a)] + \mathbb{E}_{(s,a,s')\sim\mathcal{D}_{\text{env}}, a'\sim\pi(\cdot|s')}[\gamma\omega(s,a)Q(s',a') - \omega(s,a)Q(s,a)]}_{\text{②: loss for } \omega(s,a)}\,, \end{aligned}$$

where the first term ① is the estimator for the expected discounted reward $J(\pi)$ using $\omega(s,a)$, and if ω is the true density ratio, then we can use ① to estimate $J(\pi)$ accurately. The second term ② is the loss for $\omega(s,a)$ (Eq. (24)) except that we replace the test function $f(s,a)$ with $Q(s,a)$. With the *optimal Lagrangian multiplier* $Q^\pi(s,a)$, we can get rid of the minimax optimization issue, by directly setting Q to be $\widehat{Q}^\pi(s,a)$ and optimizing $\omega(s,a)$ during the training.

Specifically, we can optimize w using the following two alternative approaches: 1) we can directly optimize ω with $L(\omega,\widehat{Q}^\pi(s,a))$. $L(\omega,\widehat{Q}^\pi(s,a))$ should be close to $J(\pi)$; 2) we can optimize ω with ②, because we know that if ② is close zero, then ω is close to the true density ratio, and $L(\omega,\widehat{Q}^\pi(s,a))$ is again close to $J(\pi)$ since ① is an estimator for $J(\pi)$. In theory the difference of these two approaches is that we may obtain more accurate estimation for $J(\pi)$ using ① since we optimize ω globally (the first and the second terms jointly). However, we do not know the ground truth for $J(\pi)$, so when we optimize ω using the first approach, the optimization process may be unstable as observed in previous literature where the authors directly perform minimax optimization on $L(\omega,Q)$ [35, 89]. By contrast, for the second approach, we know that ② is close to zero when ω is accurate, thus we can leverage ℓ_2 or ℓ_1 loss for optimizing $\omega(s,a)$ such that ② is shrunk towards zero. This approach would be more stable.

Another intuitive explanation of choosing $Q^\pi(s,a)$ as the test function is that the minimax loss in Eq. (5) for optimizing ω is based on the *saddle-point optimization*, and $f \in \mathcal{F}$ is the ***dual variable*** for the density ratio ω. As we learned from previous work [35], density (ratio) function and action-value function are the primal-dual variables of each other (basically we can conclude that the primal-dual variables are meaningful in the problem of off-policy evaluation, rather than meaningless in general saddle-point optimization problems), so we can directly set the dual variable f to be the optimal dual variable for ω, which would be the action-value function $Q^\pi(s,a)$. We further use the design of target network for training stability.

Remark Note that both our approach and VPM choose the test function based on some mathematical properties of the test function and the density ratio ω, such that the loss for ω could be potentially easier for optimization, and both methods are not the ***best theoretical way*** to optimize ω. The best theoretical way is just to use the saddle-point optimization based approaches such as DualDICE or GenDICE (or performing minimax optimization on $L(\omega, Q)$). However, as we mentioned above, there are practical optimization issues for the saddle-point optimization based approaches. And as we observed in Fig. 3, the empirical results indeed show that our approach and VPM are more stable than DualDICE and GenDICE. Besides, as reviewer 3XmH pointed out, it is definitely possible to use other functions as the test function (such as reward function that 3XmH suggested, or any other initialized neural networks). However, these choices do not have nice mathematical properties (either fixed point solution or primal-dual relationship), so the loss may not be easier for optimization. As a demonstration, we will use the reward function as the test function to optimize ω, whose empirical results will be included in our next version if we can finish the experiments before the end of discussion period.