Regularizing a Model-based Policy Stationary Distribution to Stabilize Offline Reinforcement Learning

Shentao Yang [1] **Yihao Feng** [2] **Shujian Zhang** [3] **Mingyuan Zhou** [1 3]

Abstract

Offline reinforcement learning (RL) extends the paradigm of classical RL algorithms to purely learning from static datasets, without interacting with the underlying environment during the learning process. A key challenge of offline RL is the instability of policy training, caused by the mismatch between the distribution of the offline data and the undiscounted stationary state-action distribution of the learned policy. To avoid the detrimental impact of distribution mismatch, we regularize the undiscounted stationary distribution of the current policy towards the offline data during the policy optimization process. Further, we train a dynamics model to both implement this regularization and better estimate the stationary distribution of the current policy, reducing the error induced by distribution mismatch. On a wide range of continuous-control offline RL datasets, our method indicates competitive performance, which validates our algorithm. The code is publicly available.

1. Introduction

Offline reinforcement learning (RL), traditionally known as batch RL, eschews environmental interactions during the policy learning process and focuses on training policy from static dataset collected by one or more data-collecting policies, which we collectively call the behavior policy (Ernst et al., 2005; Lange et al., 2012; Levine et al., 2020). This paradigm extends the applicability of RL from trial-and-error in the simulator (Mnih et al., 2013; 2015; Silver et al., 2016; 2017) to applications where environmental interactions can be costly or even dangerous (Nie et al., 2019; Yurtsever et al., 2020), and to domains with abundant accumulated data (Gilotte et al., 2018). While classical off-policy RL algorithms (Mnih et al., 2013; Lillicrap et al., 2016; Bellemare et al., 2017; Dabney et al., 2018; Fujimoto et al., 2018; Haarnoja et al., 2018) can be directly applied to the offline setting, they often fail to learn purely from a static dataset, especially when the offline dataset exhibits a narrow distribution on the state-action space (Fujimoto et al., 2019; Kumar et al., 2019). One important cause to such a failure is the mismatch between the stationary state-action distribution of the behavior policy and that of the learned policy, resulting in possibly pathological estimates of the queried action-values during the training process and thus the instability and failure of the policy learning.

To successfully learn a policy from the offline dataset, prior model-free offline RL works try to avoid the action-distribution shift during the training process, so that the overestimated action-values in out-of-distribution (OOD) actions can be mitigated. Their approaches can be mainly characterized into three categories: **(1)** robustly train the action-value function or provide a conservative estimate of the action-value function whose numerical values distinguish in- and out-of-distribution actions (Agarwal et al., 2020; Kumar et al., 2020; Gulcehre et al., 2021; Sinha et al., 2022); **(2)** design a tactful behavior-cloning scheme to learn only from "good" actions in the dataset (Wang et al., 2020; Chen et al., 2021; Kostrikov et al., 2021a); **(3)** regularize the current policy to be close to the action choice in the offline dataset during the training process, so that the learned policy mainly takes action within the reach of the action-value function estimate (Fujimoto et al., 2019; Laroche & Trichelair, 2019; Kumar et al., 2019; Wu et al., 2019; Jaques et al., 2019; Siegel et al., 2020; Urpí et al., 2021; Kostrikov et al., 2021b; Wu et al., 2021; Fujimoto & Gu, 2021).

Model-based RL (MBRL) can provide a potential enhancement over the model-free RL. By learning an approximate dynamic model and subsequently using that for policy learning, MBRL can provide a sample-efficient solution to answer the counterfactual question pertaining to action-value estimation in the online setting (Janner et al., 2019; Rajeswaran et al., 2020), and to provide an augmentation to the

[1]McCombs School of Business, [2]Department of Computer Science, [3]Department of Statistics & Data Science, The University of Texas at Austin. Correspondence to: Shentao Yang <shentao.yang@mccombs.utexas.edu>, Mingyuan Zhou <mingyuan.zhou@mccombs.utexas.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

static dataset in the offline setting (Yu et al., 2020). Directly applying model-based RL methods into the offline setting, however, still faces the challenge of mismatching stationary state-action distribution. In particular, when the offline dataset only covers a narrow region on the state-action space, the estimated dynamic model may not support far extrapolation (Hishinuma & Senda, 2021). Consequently, poor empirical performance related to the "model exploitation" issue can appear when directly using the learned model for offline policy learning (Ross & Bagnell, 2012; Clavera et al., 2018; Kurutach et al., 2018; Rajeswaran et al., 2020; Yu et al., 2020). As remedies, prior model-based offline RL methods try to avoid state-action pairs far from the offline dataset by uncertainty quantification (Yu et al., 2020; Kidambi et al., 2020; Fan et al., 2021; Swazinna et al., 2021), model-based constrained policy optimization (Matsushima et al., 2021; Cang et al., 2021), or model-based conservative action-value function estimation (Yu et al., 2021).

In this work, we propose to directly regularize the undiscounted stationary state-action distribution of the current policy towards that of the behavior policy during the policy learning process. The learned policy thus avoids visiting state-action pairs far from the offline dataset, reducing the occurrence of overestimated action-values and stabilizing policy training. We derive a tractable bound for the distance between the undiscounted stationary distributions. To implement this bound, we **(1)** train a dynamic model in a sufficiently rich function class via the maximum likelihood estimation (MLE); and **(2)** add a tractable regularizer into the policy optimization objective. This regularizer only requires samples from the offline dataset, the short-horizon model rollouts, the current policy, and the estimated dynamic. Further, model-based synthetic rollouts are used to better estimate the stationary state-action distribution of the current policy, which helps the performance of the final policy. Besides, our framework allows training an implicit policy to better approximate the maximizer of the action-value function estimate, particularly when the latter exhibits multi-modality. Without assuming any knowledge about the underlying environment, including the reward function and the termination condition, our method shows competitive performance on a wide range of continuous-control offline-RL datasets from the D4RL benchmark (Fu et al., 2020), validating the effectiveness of our algorithmic designs.

2. Background

We follow the classical RL setting (Sutton & Barto, 2018) to model the interaction between the agent and the environment as a Markov Decision Process (MDP), specified by the tuple $\mathcal{M} = (\mathbb{S}, \mathbb{A}, P, r, \gamma, \mu_0)$, where $\mathbb{S}$ denotes the state space, $\mathbb{A}$ the action space, $P(s' \,|\, s, a) : \mathbb{S} \times \mathbb{S} \times \mathbb{A} \to [0, 1]$ 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 a given policy $\pi(a \,|\, s)$, we follow the literature (Puterman, 2014; Liu et al., 2018) to denote the state-action distribution at timestep $t \geq 0$ induced by policy π on MDP $\mathcal{M}$ as $d_{\pi,t}(s, a) := \Pr(s_t = s, a_t = a \,|\, s_0 \sim \mu_0, a_t \sim \pi, s_{t+1} \sim P, \forall\, t \geq 0)$. Denote the *discounted* stationary state-action distribution induced by π as $d_{\pi,\gamma}(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t d_{\pi,t}(s, a)$ and the *undiscounted (average)* stationary state-action distribution as $d_\pi(s, a) = \lim_{T \to \infty} \sum_{t=0}^{T} \frac{1}{T+1} d_{\pi,t}(s, a)$. We have $d_{\pi,\gamma}(s, a) = d_{\pi,\gamma}(s)\pi(a \,|\, s)$ and $d_\pi(s, a) = d_\pi(s)\pi(a \,|\, s)$. In offline RL (Levine et al., 2020), one has only access to a static dataset $\mathcal{D}_{\text{env}} = \{(s, a, r, s')\}$ collected by some behavior policy π_b, which can be a mixture of several data-collecting policies.

Denote the action-value function for policy π on $\mathcal{M}$ as $Q^\pi(s, a) = \mathbb{E}_{\pi,P,r}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,|\, s_0 = s, a_0 = a\right]$. In the actor-critic algorithm (Sutton & Barto, 2018), the policy π and critic $Q^\pi(s, a)$ are typically modelled as parametric functions $\pi_{\boldsymbol{\phi}}$ and $Q_{\boldsymbol{\theta}}$, parametrized by $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, respectively. In offline RL, the critic is trained in the policy evaluation step by Bellman backup as minimizing *w.r.t.* $\boldsymbol{\theta}$:

$$\ell(\boldsymbol{\theta}) := \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}_{\text{env}}}\left[\left(Q_{\boldsymbol{\theta}}(s, a) - \hat{B}^\pi Q_{\boldsymbol{\theta}'}(s, a)\right)^2\right],$$
$$\hat{B}^\pi Q_{\boldsymbol{\theta}'}(s, a) = r(s, a) + \gamma \mathbb{E}_{a' \sim \pi(\cdot \,|\, s')}\left[Q_{\boldsymbol{\theta}'}(s', a')\right], \quad (1)$$

where $Q_{\boldsymbol{\theta}'}$ is the target network. The actor $\pi_{\boldsymbol{\phi}}$ is trained in the policy improvement step as

$$\arg\max_{\boldsymbol{\phi}} \mathbb{E}_{s \sim d_{\pi_b}(s),\, a \sim \pi_{\boldsymbol{\phi}}(\cdot \,|\, s)}\left[Q_{\boldsymbol{\theta}}(s, a)\right], \quad (2)$$

where samples from the offline dataset $\mathcal{D}_{\text{env}}$ is used to approximate the samples from $d_{\pi_b}(s)$ (Fu et al., 2019; Levine et al., 2020).

In model-based offline RL, a stochastic dynamic model $\widehat{P}(s' \,|\, s, a)$ within some function class $\mathcal{P}$ is learned to approximate the true environmental dynamic, denoted as P^*. With the offline dataset $\mathcal{D}_{\text{env}}$, $\widehat{P}$ is typically trained using MLE (Janner et al., 2019; Yu et al., 2020; 2021) as

$$\arg\max_{\widehat{P} \in \mathcal{P}} \mathbb{E}_{(s,a,s') \sim \mathcal{D}_{\text{env}}}\left[\log \widehat{P}(s' \,|\, s, a)\right]. \quad (3)$$

Models for the reward function $\hat{r}$ and the termination conditional can also be trained similarly if assumed unknown. With $\widehat{P}$ and $\hat{r}$, an approximated MDP to $\mathcal{M}$ can be constructed as $\widehat{\mathcal{M}} = (\mathbb{S}, \mathbb{A}, \widehat{P}, \hat{r}, \gamma, \mu_0)$. To distinguish the aforementioned stationary distributions induced by policy π on $\mathcal{M}$ and on $\widehat{\mathcal{M}}$, we denote the undiscounted stationary state-action distribution induced by π on the true dynamic P^* (or $\mathcal{M}$) as $d_\pi^{P^*}(s, a)$, on the learned dynamic $\widehat{P}$ (or $\widehat{M}$)

as $d^{\widehat{P}}_{\pi}(s,a)$, and similarly for the discounted stationary state-action distributions $d^{P^*}_{\pi,\gamma}(s,a)$ and $d^{\widehat{P}}_{\pi,\gamma}(s,a)$. With the estimated dynamic $\widehat{P}$, prior model-based offline RL works (Kidambi et al., 2020; Yu et al., 2020; 2021; Cang et al., 2021) typically approximate $d^{P^*}_{\pi_{\boldsymbol{\phi}}}(s,a)$ by simulating the learned policy $\pi_{\boldsymbol{\phi}}$ in $\widehat{M}$ for a short horizon h starting from state $s \in \mathcal{D}_{\text{env}}$. The resulting trajectories are stored into a replay buffer $\mathcal{D}_{\text{model}}$, similar to the replay buffer in off-policy RL (Lin, 1992; Lillicrap et al., 2016). Sampling from $\mathcal{D}_{\text{env}}$ in Eqs. (1) and (2) 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 $\mathcal{D}_{\text{env}}$ with probability f and from $\mathcal{D}_{\text{model}}$ with probability $1-f$.

We follow the offline RL literature (Liu et al., 2018; Nachum et al., 2019b; Kallus & Zhou, 2020; Zhang et al., 2020) to assume **(i)** that all the Markov chains induced by the studied (approximated) dynamics and policies are ergodic, and thus the undiscounted stationary state-action distribution $d_{\pi}(s,a)$ equals to the limiting state-action occupancy measure induced by π on the corresponding MDP; and **(ii)** that the offline dataset $\mathcal{D}_{\text{env}}$ is constructed as the rollouts of π_b on the true dynamic P^*, *i.e.*, $\mathcal{D}_{\text{env}} \sim d^{P^*}_{\pi_b}(s,a)$. We denote the state-action and state distributions in $\mathcal{D}_{\text{env}}$ as $d_{\mathcal{D}_{\text{env}}}(s,a)$ and $d_{\mathcal{D}_{\text{env}}}(s)$, which are discrete approximations to $d^{P^*}_{\pi_b}(s,a)$ and $d^{P^*}_{\pi_b}(s)$, respectively.

3. Main Method

In this section, we introduce our approach to stabilize the policy training. Specifically, except optimizing the policy π to maximize the action-value function, we add a distribution regularization into the policy optimization objective, which encourages the closeness between the stationary distribution of the learned policy π and that of the behavior π_b. Our policy optimization objective is summarized as

$$J(\pi) := \lambda \mathbb{E}_{s\sim\mathcal{D}_{\text{env}},\, a\sim\pi(\cdot|s)}\left[Q_{\boldsymbol{\theta}}(s,a)\right] - D(d^{P^*}_{\pi_b}, d^{P^*}_{\pi}),$$

where λ is the regularization coefficient, $D(\cdot,\cdot)$ is a statistical distance between two probability distributions, such as the integral probability metric (IPM, Müller (1997)) and the Jensen–Shannon divergence (JSD, Lin (1991)), $d^{P^*}_{\pi_b}(s,a)$ is the stationary distribution induced by behavior policy π_b in the true dynamic P^*, and $d^{P^*}_{\pi}(s,a)$ is the stationary distribution induced by the current policy π in P^*.

3.1. A Tractable Bound to the Regularization Term

As discussed in Section 2, the offline dataset $\mathcal{D}_{\text{env}}$ is drawn from $d^{P^*}_{\pi_b}$. Since the current policy π can not interact with the environment during the training process, we can not directly estimate $d^{P^*}_{\pi}$ using simulation. Hence, we use a learned dynamic model $\widehat{P}$ to approximate $d^{P^*}_{\pi}$. Using the triangle inequality for the distance D, we have

$$D(d^{P^*}_{\pi_b}, d^{P^*}_{\pi}) \le D(d^{P^*}_{\pi_b}, d^{\widehat{P}}_{\pi}) + D(d^{\widehat{P}}_{\pi}, d^{P^*}_{\pi}). \tag{4}$$

Next, we describe how to estimate $D(d^{P^*}_{\pi_b}, d^{\widehat{P}}_{\pi})$ and how to upper bound $D(d^{\widehat{P}}_{\pi}, d^{P^*}_{\pi})$ in the RHS of Eq. (4).

For a given dynamic model $\widehat{P}(s'|s,a)$, since we do not know the formula of $d^{\widehat{P}}_{\pi}(s,a)$, a natural way to estimate this distribution is collecting rollouts of π on the dynamic $\widehat{P}$. A major drawback of this approach is that it requires drawing many trajectories. This can be time-consuming as trajectories are typically long or even infinite, especially when they converge to the stationary distribution slowly. Even worse, since π changes during the policy-learning process, we have to repeat this expensive rollout process many times. To estimate $D(d^{P^*}_{\pi_b}, d^{\widehat{P}}_{\pi})$ in Eq. (4), we instead derive the following theorem, which avoids sampling full trajectories and only requires sampling a *single* (s',a') pair using policy π and model $\widehat{P}$ starting from (s,a) drawn from the offline dataset. This can significantly save computation and training time.

Theorem 3.1. *If the distance metric D is the IPM w.r.t some function class $\mathcal{G} = \{g : \mathbb{S}\times\mathbb{A} \to \mathbb{R}\}$, defined as*

$$D_{\mathcal{G}}(d^{P^*}_{\pi_b}, d^{\widehat{P}}_{\pi}) := \sup_{g\in\mathcal{G}} \left| \mathbb{E}_{d^{P^*}_{\pi_b}}\left[g(s,a)\right] - \mathbb{E}_{d^{\widehat{P}}_{\pi}}\left[g(s,a)\right] \right|,$$

then there exists a $\mathcal{G}$ s.t. we can rewrite $D_{\mathcal{G}}(d^{P^*}_{\pi_b}, d^{\widehat{P}}_{\pi})$ as:

$$D_{\mathcal{G}}(d^{P^*}_{\pi_b}, d^{\widehat{P}}_{\pi}) = \mathcal{R}_{\mathcal{F}}(d^{P^*}_{\pi_b}, \pi, \widehat{P})$$
$$:= \sup_{f\in\mathcal{F}} \left| \mathbb{E}_{(s,a)\sim d^{P^*}_{\pi_b}}\left[f(s,a)\right] - \mathbb{E}_{\substack{s'\sim\widehat{P}(\cdot|s,a),a'\sim\pi(\cdot|s') \\ (s,a)\sim d^{P^*}_{\pi_b}}}\left[f(s',a')\right] \right|,$$

where

$$\mathcal{F} := \left\{ f : f(s,a) = \mathbb{E}_{\pi,\widehat{P}}\left[\sum_{t=0}^{\infty}\left(g(s_t,a_t) - \widehat{\eta}^{\pi}\right) \mid \begin{matrix} a_0 = a \\ s_0 = s \end{matrix}\right], \right.$$
$$\left. \widehat{\eta}^{\pi} = \lim_{T\to\infty} \mathbb{E}_{\pi,\widehat{P}}\left[\frac{1}{T+1}\sum_{t=0}^{T} g(s_t,a_t)\right], g \in \mathcal{G} \right\}.$$

We describe the definition of the function class $\mathcal{G}$ in Appendix B.2. In Theorem 3.1, the supremum is taken over a new function class $\mathcal{F}$, which captures the *relative value function* following the policy π and the transition $\widehat{P}$ when the reward function is the deterministic function $g(s,a)$ (Puterman, 2014; Sutton & Barto, 2018). Using this new function class $\mathcal{F}$ instead of the original $\mathcal{G}$ enables the aforementioned avoidance of full-trajectory sampling. A more detailed discussion on the relation between $f(s,a)$ and $g(s,a)$ can be found in Appendix B.2.

We remark that in practice the relative value function f is approximated by neural networks with regularity, under the

same assumptions as approximating the solution of Bellman backup using neural networks. Because of the expressiveness of neural networks, we assume that f then resides in a richer function class that contains the class $\mathcal{F}$ of differential value functions. Since f is maximized over a larger function class than $\mathcal{F}$, we effectively minimize an upper bound of the original regularization term in Theorem 3.1. This relaxation avoids explicitly estimating the differential value function for each reward function g encountered in the training process. Such explicit estimation can be time-consuming.

Our next step is to bound $D(d_\pi^{\widehat{P}}, d_\pi^{P^*})$.

Theorem 3.2. *Let P^* be the true dynamic, $D_\mathcal{G}$ the IPM with the same $\mathcal{G}$ as Theorem 3.1, and $\widehat{P}$ the estimated dynamic. There exist a constant $C \geq 0$ and a function class $\Psi = \{\psi : \mathbb{S} \times \mathbb{A} \to \mathbb{R}\}$ such that for any policy π,*

$$D_\mathcal{G}(d_\pi^{\widehat{P}}, d_\pi^{P^*}) \leq C \cdot \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}} \left[\sqrt{\frac{1}{2} \mathrm{KL}\left(P^*(s'|s,a) || \widehat{P}(s'|s,a) \right)} \right] + \mathcal{R}_\Psi(d_{\pi_b}^{P^*}, \pi)\,, \tag{5}$$

where

$$\mathcal{R}_\Psi(d_{\pi_b}^{P^*}, \pi) := \sup_{\psi \in \Psi} \left| \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}}[\psi(s,a)] - \mathbb{E}_{\substack{s\sim d_{\pi_b}^{P^*} \\ a\sim\pi(\cdot|s)}}[\psi(s,a)] \right|.$$

We describe the definition of the function class Ψ in Appendix B.3. Detailed proof of Theorem 3.2 can be found in Appendix B.3.

Proposition 3.3. *An upper bound of the first term on the right-hand-side of Eq. (5) can be minimized via the MLE for $\widehat{P}$, i.e., Eq. (3).*

Informally, this proposition coincides with the intuition that estimating the environmental dynamic using MLE can be helpful for matching $d_\pi^{\widehat{P}}$ with $d_\pi^{P^*}$. This proposition is restated and proved in Proposition B.2 in Appendix B.3.

Theorem 3.4. *Combine Theorems 3.1 and 3.2 and Proposition 3.3, and suppose the dynamic model $\widehat{P}$ is trained by the MLE objective (Eq. (3)), then for the same $\mathcal{G}$ and any policy π, up to some constants w.r.t. π and $\widehat{P}$, we have*

$$D_\mathcal{G}(d_{\pi_b}^{P^*}, d_\pi^{P^*}) \leq \mathcal{R}_\mathcal{F}(d_{\pi_b}^{P^*}, \pi, \widehat{P}) + \mathcal{R}_\Psi(d_{\pi_b}^{P^*}, \pi) + \mathcal{E}_{model},$$

where $\mathcal{E}_{model}$ is the error associated with the maximum likelihood training, and is independent of the policy π.

If the function class $\mathcal{P}$ for the estimated dynamic $\widehat{P}$ is rich enough and we have sufficiently many empirical samples from $d_{\pi_b}^{P^*}$, under the classical statistical regularity condition, we can achieve a small model error $\mathcal{E}_{model}$ by MLE (Ferguson, 1996; Casella & Berger, 2001). So we focus on minimizing the first two terms in Theorem 3.4 as a regularization during the policy improvement step.

3.2. Practical Implementation

Based on the analysis in Section 3.1, we would like to minimize $D_\mathcal{G}(d_{\pi_b}^{P^*}, d_\pi^{P^*})$ as a regularization term in the policy optimization step.

Model Training. The first step is to minimize the model error $\mathcal{E}_{model}$ by pre-training the dynamic model under a sufficiently rich function class via MLE. In this paper, we take on a fully-offline perspective, assuming no prior knowledge about the reward function and the termination condition. We use neural networks to parameterize the learned transition, reward, and termination. Following prior model-based offline-RL work (Yu et al., 2020; 2021; Cang et al., 2021), we assume stochastic transition and reward, and use an ensemble of Gaussian probabilistic networks, denoted as $\widehat{P}(\cdot \mid s,a)$ and $\widehat{r}(s,a)$, to model these distributions. We use the same model structure, training strategy, and hyperparameters as in Yu et al. (2020), where the model outputs the mean and log-standard-deviation of the normal distributions of the reward and next state. The termination condition is modelled as an ensemble of deterministic sigmoid networks with the same training strategy as $\widehat{P}$ and $\widehat{r}$, and outputs the probability of the input satisfying the termination condition. For computational simplicity, the termination-condition networks share the input and hidden layers with $\widehat{P}$ and $\widehat{r}$. A default cutoff-value of 0.5 is used to decide termination. Further details on model training is in Appendix C.2.1.

Model Rollouts. In practice, we have only a finite static dataset, and we therefore replace sampling from $d_{\pi_b}^{P^*}$ with sampling from its discrete approximation $\mathcal{D}_\text{env}$. With the learned model $\widehat{P}$, we follow prior model-based RL work (Janner et al., 2019; Yu et al., 2020) to augment $\mathcal{D}_\text{env}$ with the replay buffer $\mathcal{D}_\text{model}$ consisting of h-horizon rollouts of the current policy π_ϕ on the learned model $\widehat{P}$, by branching from states in the offline dataset $\mathcal{D}_\text{env}$. As discussed in Section 2, the augmented dataset is defined as $\mathcal{D} := f \cdot \mathcal{D}_\text{env} + (1-f) \cdot \mathcal{D}_\text{model}$, with $f \in [0,1]$ denotes the percentage of real data. While theoretically not required in the derivation in Section 3.1, in our preliminary study we find that adding a significant amount of synthetic rollouts can benefit the performance. We hypothesize that adding $\mathcal{D}_\text{model}$ can better estimate the stationary state-action distribution of the current policy and hence mitigate the issue of distribution mismatch in offline policy evaluation and optimization. Besides, $\mathcal{D}_\text{model}$ can also serve as a natural data-augmentation to $\mathcal{D}_\text{env}$, reducing the sampling error in training the action-value function, as discussed in Yu et al. (2021). We therefore use $\mathcal{D}$ in lieu of $\mathcal{D}_\text{env}$ in the policy evaluation, policy optimization, and regularizer construction. We follow Yu et al. (2021) to use $f = 0.5$ for our algorithm and conduct an ablation study on f in Section 5.3.

Regularizer Construction. Though technically possible to

use different test functions under different function classes to separately estimate $\mathcal{R}_{\mathcal{F}}(d^{P^*}_{\pi_b}, \pi, \widehat{P})$ and $\mathcal{R}_{\Psi}(d^{P^*}_{\pi_b}, \pi)$, for computational simplicity, we use the same test function to directly estimate the sum $\mathcal{R}_{\mathcal{F}}(d^{P^*}_{\pi_b}, \pi, \widehat{P}) + \mathcal{R}_{\Psi}(d^{P^*}_{\pi_b}, \pi)$, and parametrize the test function using neural network. We leave the strategy of separate estimation as future work.

A sample-based estimate of $\mathcal{R}_{\mathcal{F}}(d^{P^*}_{\pi_b}, \pi, \widehat{P}) + \mathcal{R}_{\Psi}(d^{P^*}_{\pi_b}, \pi)$ requires sampling from the data distribution, which we dub as "true" samples; and sampling from the policy's distribution, dubbed as "fake" samples. The "true" samples $\mathcal{B}_{\text{true}}$ are formed by sampling from $\mathcal{D}_{\text{env}}$. To form the "fake" samples $\mathcal{B}_{\text{fake}}$, we first sample $s \sim \mathcal{D}$, followed by getting a corresponding action for each s using $a \sim \pi_{\boldsymbol{\phi}}(\cdot \,|\, s)$. Then for each (s, a) pair, we get a sample of the next state s' using the learned model $\widehat{P}$ as $s' \sim \widehat{P}(\cdot \,|\, s, a)$, followed by sampling one next action a' for each s' via $a' \sim \pi_{\boldsymbol{\phi}}(\cdot \,|\, s')$. Finally, we construct the "fake" sample $\mathcal{B}_{\text{fake}}$ as

$$\mathcal{B}_{\text{fake}} := \begin{bmatrix} (s\,, & a\,) \\ (s', & a') \end{bmatrix}. \tag{6}$$

Here, (s, a) is used to estimate $\mathbb{E}_{\substack{s\sim d^{P^*}_{\pi_b} \\ a\sim\pi(\cdot\,|\,s)}} [\psi(s, a)]$, the second term inside the $|\cdot|$ of $\mathcal{R}_{\Psi}(d^{P^*}_{\pi_b}, \pi)$. The pair (s', a') corresponds to $\mathbb{E}_{\substack{s'\sim\widehat{P}(\cdot\,|\,s,a), a'\sim\pi(\cdot\,|\,s') \\ (s,a)\sim d^{P^*}_{\pi_b}}} [f(s', a')]$, the second term inside the absolute-value of $\mathcal{R}_{\mathcal{F}}(d^{P^*}_{\pi_b}, \pi, \widehat{P})$.

Notice that both the IPM and JSD are well-defined statistical metrics, and matching *w.r.t.* one of them effectively matches the other. Meanwhile, the GAN (Goodfellow et al., 2014) framework provides a stable training scheme for approximately minimizing JSD, with easy reference to hyperparameter configuration in literature. We henceforth change the IPM structure in $\mathcal{R}_{\mathcal{F}}(d^{P^*}_{\pi_b}, \pi, \widehat{P}) + \mathcal{R}_{\Psi}(d^{P^*}_{\pi_b}, \pi)$ to JSD and approximately minimize it via GAN. This amounts to training a discriminator $D_{\boldsymbol{w}}$ to maximize

$$\begin{aligned} &\frac{1}{|\mathcal{B}_{\text{true}}|} \sum_{(s,a)\sim\mathcal{B}_{\text{true}}} [\log D_{\boldsymbol{w}}(s, a)] + \\ &\qquad \frac{1}{|\mathcal{B}_{\text{fake}}|} \sum_{(s,a)\sim\mathcal{B}_{\text{fake}}} [\log (1 - D_{\boldsymbol{w}}(s, a))], \end{aligned} \tag{7}$$

and adding the generator loss $\mathcal{L}_g(\boldsymbol{\phi})$ defined as

$$\mathcal{L}_g(\boldsymbol{\phi}) := \tfrac{1}{|\mathcal{B}_{\text{fake}}|} \textstyle\sum_{(s,a)\in\mathcal{B}_{\text{fake}}} [\log (1 - D_{\boldsymbol{w}}(s, a))] \tag{8}$$

as the regularizer into the policy optimization objective. In the ablation study (Section 5.3), we compare our JSD implementation with minimizing the dual form of Wasserstein-1 distance, a special instance of IPM.

Critic Training. Motivated by Fujimoto et al. (2019) and Kumar et al. (2019), we use the following critic target:

$$\begin{aligned} &\widetilde{Q}(s, a) \triangleq r(s, a).\text{clamp}(r_{\min}, r_{\max}) + \\ &\gamma \mathbb{E}_{a'\sim\pi_{\boldsymbol{\phi}'}(\cdot\,|\,s')} \left[c \min_{j=1,2} Q_{\boldsymbol{\theta}'_j}(s', a') + (1 - c) \max_{j=1,2} Q_{\boldsymbol{\theta}'_j}(s', a') \right] \end{aligned} \tag{9}$$

with $c = 0.75$ as in prior work, $r_{\min} = r_0 - 3\sigma_r$ and $r_{\max} = r_1 + 3\sigma_r$, where r_0, r_1, σ_r respectively denote the minimum, maximum, and standard deviation of the rewards in the offline dataset. Here we adopt similar reward-clipping strategy as Hishinuma & Senda (2021) to mitigate the inaccuracy in the reward model. Each of the double critic networks is trained to minimize the critic loss over $(s, a) \in \mathcal{B} \sim \mathcal{D}$, *i.e.,* $\forall\, j = 1, 2$,

$$\min_{\boldsymbol{\theta}_j} \tfrac{1}{|\mathcal{B}|} \textstyle\sum_{(s,a)\in\mathcal{B}} \text{Huber}\left(Q_{\boldsymbol{\theta}_j}(s, a) - \widetilde{Q}(s, a) \right), \tag{10}$$

where we replace the usual mean-square-error with the Huber loss $\text{Huber}(\cdot)$ for training stability.

Policy Training. We follow Kumar et al. (2019) to define the policy optimization objective as

$$\arg\min_{\boldsymbol{\phi}} -\lambda \cdot \frac{1}{|\mathcal{B}|} \sum_{s\in\mathcal{B}, a\sim\pi_{\boldsymbol{\phi}}(\cdot\,|\,s)} \left[\min_{j=1,2} Q_{\boldsymbol{\theta}_j}(s, a) \right] + \mathcal{L}_g(\boldsymbol{\phi}), \tag{11}$$

where the regularization coefficient λ is constructed similar to Fujimoto & Gu (2021) as

$$\lambda = \alpha / Q_{avg},$$

with soft-updated Q_{avg}. We use $\alpha = 10$ across all datasets in our main results in Section 5.2.

Implicit Policy. To better approximate the maximizer of the action-value function, instead of the classical deterministic or Gaussian policy, in this paper we use a richer function class, the class of implicit distributions, to define our policy. Specifically, similar to the generator in the conditional GAN (CGAN, Mirza & Osindero (2014)), our implicit policy uses a deterministic neural network to transform a given noise distribution $p_z(z)$ into the state-conditional action distribution. Concretely, given state s,

$$a \sim \pi_{\boldsymbol{\phi}}(\cdot \,|\, s) = \pi_{\boldsymbol{\phi}}(s, z), \text{ with } z \overset{iid}{\sim} p_z(z),$$

where $\pi_{\boldsymbol{\phi}}$ is a deterministic network. Section 5.1 conducts a toy experiment to show the efficacy of the implicit policy, particularly when fitting distributions with multimodality.

We summarize the main steps of our method Algorithm 1, which is implemented by approximately minimizing JSD via GAN and is dubbed as Stationary Distribution Matching via GAN (SDM-GAN). Further implementation details are provided in Appendix C.2.

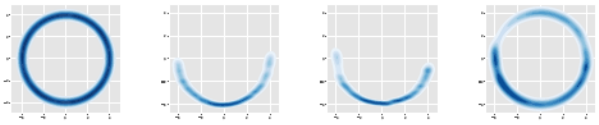
Algorithm 1 SDM-GAN, Main Steps

Initialize dynamic model $\widehat{P}$, policy network $\pi_{\boldsymbol{\phi}}$, critic network $Q_{\boldsymbol{\theta}_1}$ and $Q_{\boldsymbol{\theta}_2}$, discriminator network $D_{\boldsymbol{w}}$.
Train $\widehat{P}$ using maximum likelihood estimation Eq. (3)
for each iteration **do**
  Rollout $\pi_{\boldsymbol{\phi}}$ on $\widehat{P}$, add the synthetic data into $\mathcal{D}_{\text{model}}$.
  Sample mini-batch $\mathcal{B} \sim \mathcal{D} = f\mathcal{D}_{\text{env}} + (1-f)\mathcal{D}_{\text{model}}$.
  Get critic target via Eq. (9) and train critics by Eq. (10).
  Construct $\mathcal{B}_{\text{fake}}$ via Eq. (6) and sample $\mathcal{B}_{\text{true}} \sim \mathcal{D}_{\text{env}}$.
  Optimize the discriminator $D_{\boldsymbol{w}}$ using Eq. (7).
  Calculate generator loss $\mathcal{L}_g(\phi)$ using Eq. (8).
  Optimize policy network $\pi_{\boldsymbol{\phi}}$ by Eq. (11).
end for

4. Related Work

Model-free Offline RL. This paper is related to a classical yet persistent idea in model-free offline RL, *i.e.*, matching the learned policy with the behavior policy so that the learned policy mostly visits state-action pairs similar to the offline dataset. In literature, several designs have been proposed for this purpose, such as constraining the learned policy towards an estimated behavior policy (Kumar et al., 2019; Wu et al., 2019; Jaques et al., 2019), a tactful decomposition of the action or the action-value function into a behavior-cloning part and an offset (Fujimoto et al., 2019; Kostrikov et al., 2021b), modification of the critic-learning objective (Nachum et al., 2019b; Kumar et al., 2020). Our paper adds to the literature by directly matching the undiscounted stationary distribution of the learned policy towards the offline dataset. To achieve this, we develop the undiscounted version of the change of variable trick to derive a tractable regularizer for policy optimization, which has been explored only in the discounted case (Nachum et al., 2019a) in the off-policy evaluation (OPE) literature (Jiang & Li, 2016; Liu et al., 2018; Zhang et al., 2020). Different from the goal of estimating the performance of some policies using offline data, our goal here is to construct a tractable regularization for policy optimization, which involves a learned transition model trained by MLE. Yang et al. (2022) recently propose the design of an implicit policy via the CGAN structure, together with several additional enhancing techniques. Our paper improves on this prior work by adopting a model-based approach, which facilitates regularizer construction and mitigates distribution mismatch.

Model-based Offline RL. Similar notion of guiding the policy away from the OOD state-action pairs also persists in model-based offline RL. In literature, such guidance has been implemented by model-output-based uncertainty quantification (Yu et al., 2020; Kidambi et al., 2020), estimation of a conservative action-value function (Yu et al., 2021), and regularizing the model-based policy learning with a possibly-adaptive behavior prior (Cang et al., 2021; Matsushima et al., 2021). Recently, techniques in the OPE literarture has been adopted into policy training and model learning. Hishinuma & Senda (2021) propose an EM-style iterative training of model and policy, with a density-ratio-based weighted model estimation. Similar to our work, Lee et al. (2021a) utilize the change of variable technique to train the dynamic, which can potentially improve the quality of the representations of the dynamic during the model training. By contrast, our paper uses the model-generated rollouts to implement a tractable bound for directly constraining the undiscounted stationary distribution of the learned policy towards the offline dataset during the policy optimization step. For algorithmic simplicity, our paper does not apply the technique of uncertainty quantification, learnable behavior prior, advanced design of the dynamic model via the GPT-Transformer (Radford & Sutskever, 2018; Fan et al., 2020; Janner et al., 2021; Zhang et al., 2022), *etc.* These are orthogonal to our method and may further improve the empirical performance.



(a) Truth (b) D-CGAN (c) G-CGAN (d) CGAN

Figure 1. Performance of approximating the behavior policy on the circle dataset. A deterministic-generator conditional GAN ("D-CGAN"), a Gaussian-generator conditional GAN ("G-CGAN") and a conditional GAN ("CGAN") are fitted using the classical policy (conditional distribution) matching approach similar to Wu et al. (2019). More details are provided in Appendix C.1.

5. Experiments

In what follows, we first show the effectiveness of an implicit policy through a toy example (Section 5.1). We then present an empirical evaluation of our algorithm (Section 5.2), followed by an ablation study (Section 5.3). Source code for the experiments is publicly available.

5.1. Toy Experiments

As a motivation to train a flexible implicit policy instead of the classical deterministic or Gaussian policy, we conduct a toy behavior-cloning experiment, as shown on Figure 1. We remark that the simplest form of offline behavior cloning is essentially a supervised learning task, with the reward function and the termination condition being unnecessary. Throughout Figure 1, we use the x-axis to represent the state and y-axis the action. Figure 1a plots the state-action distribution that we seek to clone, which presents multimodality on almost all states. Such an offline dataset may come from multiple data-collecting policies or from the multiple max-

ima of the action-value function on the action space. We hence expect the learned policy to capture the targeted state-action distribution, which can translate into a better capture of the rewarding actions and an improved generalization beyond the static dataset in offline RL tasks. Figures 1b-1d plot the state-action distributions of the learned policies on the testset, in which we compare the implicit policy implemented by conditional GAN (Figure 1d) with its variants of changing the implicit generator to deterministic transformation (Figure 1b) and to the Gaussian generator (Figure 1c). Experimental details are discussed in Appendix C.1.

We see from Figures 1b and 1c that both the deterministic and the Gaussian policy fail to capture necessary action modes, which directly relates to their less flexible structures. By contrast, as shown in Figure 1d, the implicit policy defined by conditional GAN does capture all the action modes on each state and fits well onto the targeted distribution. This toy example motivates our algorithmic design of training an implicit policy via the conditional GAN structure.

5.2. Main Results

As discussed in Section 3.2, we implement our algorithm by approximately matching the JSD between the stationary distributions via the conditional GAN, which we dub as Stationary Distribution Matching via GAN (SDM-GAN). We compare our algorithm with three state-of-the-art (SOTA) model-free offline-RL algorithms: CQL (Kumar et al., 2020), FisherBRC (Kostrikov et al., 2021b), and TD3+BC (Fujimoto & Gu, 2021); three SOTA model-based offline-RL algorithms: MOPO (Yu et al., 2020), COMBO (Yu et al., 2021), and WMOPO (Hishinuma & Senda, 2021); and two "DICE-style" algorithms: AlgaeDICE (Nachum et al., 2019b) and OptiDICE (Lee et al., 2021b). For WMOPO, we use the version of $\alpha = 0.2$, which is the paper's proposal. Experimental details and hyperparameter setups are discussed in detail in Appendix C.3. We remark that unlike our algorithm, the three model-based RL baselines assume known termination function and possibly also known reward function, which potentially limits their applicability. Except AlgaeDICE, we rerun each baseline method using the official source code under the recommended hyperparameters. The results of AlgaeDICE ("aDICE") is obtained from the D4RL whitepaper (Fu et al., 2020). We compare our algorithm with the baselines on a diverse set of continuous-control offline-RL datasets (discussed in detail in Appendix C.3), ranging across the Gym-Mojoco, Maze2D, and Adroit domains in the D4RL benchmark. Note that the three model-based RL baselines do not provide hyperparameter configurations for the Maze2D and Adroit domains, and therefore we do not evaluate them on these datasets. Table 1 shows the mean and standard deviation of each algorithm on each tested dataset.

In Table 1, our algorithm shows competitive and relatively-stable performance across a diverse array of datasets. Specifically, compared to the baselines, our algorithm shows better and more robust performance on the two task domains that are traditionally considered as challenging in offline-RL (Fu et al., 2020): the Adroit domain featured by high dimensionality and narrow data-distribution; and the Maze2D domain collected by non-Markovian policies. On these two domains, behavioral-cloning (BC) based algorithms, such as FisherBRC and TD3+BC, are likely to fail. Concretely, on the Adroit datasets, a main difficulty is estimating the behavior policy and regularizing the current policy towards it in a high-dimensional space with limited data. On the Maze2D datasets, policy learning is challenged by the error of using a Markovian policy to clone the non-Markovian data-collecting policy. The complexity of these datasets and underlying environments may also call for a more flexible policy class, and thus excels our design of an implicit policy over the classical deterministic or Gaussian policy. On the Gym-Mojoco domain, our algorithm is able to learn from median-quality datasets and from samples collected by a mixture of data-collecting policies, which are closer to the real-world settings.

Comparing SDM-GAN with AlgaeDICE and OptiDICE reveals the overall benefit of our approach over the "DICE-style" method. As an example, OptiDICE requires BC, which is implemented by a Gaussian-mixture policy with several components. While this BC strategy may be beneficial on relatively-lower dimensional Maze2D datasets, it may not generalize onto the higher-dimensional Adroit datasets. This may be a direct result of the previously-stated difficulty of BC-based methods on the Adroit datasets.

Additionally, we notice that some of the designs in these SOTA baselines are orthogonal to our algorithm, such as a tactful estimation of the action-value function (CQL, FisherBRC, COMBO), an incorporation of uncertainty quantification into the estimated dynamic (MOPO), and a density-ratio-based weighted model estimation (WMOPO). These designs may be easily incorporated into our algorithm, leaving future work and potentials for further improvement.

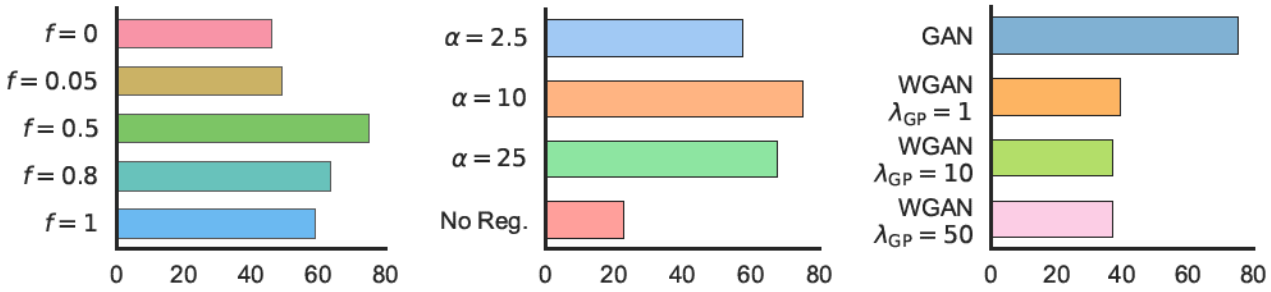
5.3. Ablation Study

In the ablation study we seek to answer the following questions: **(a)** Does the rollout dataset $\mathcal{D}_{\text{model}}$ help the performance? **(b)** Is our algorithm sensitive to the choice of the regularization coefficient α? **(c)** Does approximately matching the JSD between the stationary distributions perform better than matching the IPM? Unless stated otherwise, experimental details for all algorithmic variants follow the main results (stated in Appendix C.3).

(a). To investigate whether rollout data are helpful for policy training, we follow Yu et al. (2021) to compare our algo-

Table 1. Normalized returns for the experiments on the D4RL datasets. We use the "v0" version of the datasets in the Gym-MuJoCo and Adroit domains. On the Gym-Mojoco domain, we bold both the highest score over all algorithms and the highest score of model-based algorithms. On the Maze2D and Adroit tasks, we bold the best scores.

Task Name	aDICE	CQL	FisherBRC	TD3+BC	OptiDICE	MOPO	COMBO	WMOPO	SDM-GAN
halfcheetah-medium	-2.2	39.0 ± 0.8	41.1 ± 0.6	43.0 ± 0.5	38.2 ± 0.5	47.2 ± 1.0	53.7 ± 2.1	**55.6** ± 1.3	42.5 ± 0.5
walker2d-medium	0.3	60.2 ± 30.8	**78.4** ± 1.8	77.3 ± 4.0	14.3 ± 15.0	0.0 ± 0.1	40.9 ± 28.9	22.7 ± 27.7	**66.7** ± 1.8
hopper-medium	1.2	34.5 ± 11.7	99.2 ± 0.3	**99.6** ± 0.6	92.3 ± 16.9	23.4 ± 7.2	51.8 ± 32.8	**66.5** ± 46.0	62.8 ± 14.3
halfcheetah-medium-replay	-2.1	43.4 ± 0.8	43.2 ± 1.3	41.9 ± 2.0	39.8 ± 0.8	**52.5** ± 1.4	51.8 ± 1.6	51.8 ± 5.6	41.7 ± 0.4
walker2d-medium-replay	0.6	16.4 ± 6.6	38.4 ± 16.6	24.6 ± 6.7	20.2 ± 5.8	51.9 ± 15.8	14.2 ± 11.9	**54.8** ± 12.3	20.3 ± 4.0
hopper-medium-replay	1.1	29.5 ± 2.3	33.4 ± 2.8	31.4 ± 2.7	29.0 ± 4.9	47.1 ± 16.2	34.5 ± 2.0	**93.9** ± 1.9	30.6 ± 2.8
halfcheetah-medium-expert	-0.8	34.5 ± 15.8	**92.5** ± 8.5	90.1 ± 6.9	91.2 ± 16.6	**92.1** ± 8.3	90.0 ± 10.5	42.7 ± 13.0	89.1 ± 6.6
walker2d-medium-expert	0.4	79.8 ± 22.7	**98.2** ± 13.1	96.1 ± 15.8	67.1 ± 30.2	36.0 ± 49.6	61.3 ± 36.1	48.6 ± 37.0	**97.9** ± 4.9
hopper-medium-expert	1.1	103.5 ± 20.2	112.3 ± 0.3	111.9 ± 0.3	101.8 ± 18.5	27.8 ± 3.6	**112.6** ± 1.8	97.8 ± 19.3	104.5 ± 5.4
maze2d-large	-0.1	43.7 ± 18.6	-2.1 ± 0.4	87.6 ± 15.4	130.7 ± 56.1	-	-	-	**207.7** ± 11.7
maze2d-medium	10.0	30.7 ± 9.8	4.6 ± 20.4	59.1 ± 47.7	**140.8** ± 44.0	-	-	-	115.4 ± 34.2
maze2d-umaze	-15.7	50.5 ± 7.9	-2.3 ± 17.9	13.8 ± 22.8	**107.6** ± 33.1	-	-	-	36.1 ± 28.4
pen-human	-3.3	2.1 ± 13.7	0.0 ± 3.9	-1.7 ± 3.8	-0.1 ± 5.6	-	-	-	**17.8** ± 1.7
pen-cloned	-2.9	1.5 ± 6.2	-2.0 ± 0.8	-2.4 ± 1.4	1.4 ± 6.8	-	-	-	**40.6** ± 6.1
pen-expert	-3.5	95.9 ± 18.1	31.6 ± 24.4	32.4 ± 24.3	-1.1 ± 4.7	-	-	-	**135.8** ± 11.7
door-expert	0.0	87.9 ± 21.6	57.6 ± 37.7	-0.3 ± 0.0	87.9 ± 25.8	-	-	-	**93.5** ± 6.7



(a) Varying $f \in \{0, 0.05, 0.5, 0.8, 1\}$ (b) Varying $\alpha \in \{2.5, 10, 25\}$ and No Reg. (c) JSD *v.s.* IPM (Wasserstein-1 Dual)

Figure 2. Average score (x-axis) over the tested datasets for each of the comparison in the ablation study (Section 5.3).

rithm used in Section 5.2 ($f = 0.5$) with its variant of fewer and more rollout data, respectively $f = 0.8$ and $f = 0.05$, with all other settings kept the same. Additionally, we consider the variants of no rollout data ($f = 1$) and only rollout data ($f = 0$).

Figure 2a plots the average score over the tested datasets for each variant, with detail comparison provided in Table 3. On 11 out of 16 datasets, the $f = 0.5$ version has higher mean returns than the $f = 0.8$ version, showing the efficacy of using more rollout data. This empirical result aligns with our intuition that rollout data can better approximate the stationary state-action distribution of the learned policy, especially when the learned policy deviates from the behavior. The efficacy of model-based rollout is further corroborated by the generally worse performance of the $f = 1$ variant, compared with $f = 0.8$ and $f = 0.5$. Nevertheless, using too-many rollout data can be harmful, as shown by the comparison between the $f = 0$, $f = 0.05$, and $f = 0.5$ variants. In Table 3, the scores for the $f = 0$ and $f = 0.05$ variants are generally lower than $f = 0.5$, and the standard deviations relative to the scores are generally higher. This comparison shows the gain of adding offline data for combating the inaccuracy in the estimated dynamic.

(b). We test our algorithm on both a smaller and a larger value of the regularization coefficient $\alpha \in \{2.5, 25\}$, together with the variant without the proposed regularization term (denoted as "No Reg."). Figure 2b plots the average scores and Table 4 presents the results. We see that on a large portion of datasets, our method is relatively robust to the choice of α, especially when using a larger value of α.

A small value of α, *e.g.*, $\alpha = 2.5$ in Fujimoto & Gu (2021), can be too conservative for our algorithm, limiting the improvement of the policy. A larger value of $\alpha = 25$, which puts more weight on policy optimization than the default $\alpha = 10$, overall does not lead to significant deterioration in the performance, though the results do in general possess higher variance. The inferior performance of the "No Reg." variant shows the necessity of proper regularization, especially on the Maze2D and Adroit datasets.

(c). We compare our default algorithm of approximately matching the JSD between the stationary distributions (SDM-GAN) with the variant of approximately matching the dual form of the Wasserstein-1 metric, which is an instance of the IPM (Müller, 1997; Sriperumbudur et al., 2009) and is dubbed as "SDM-WGAN." SDM-WGAN is implemented by changing the CGAN structure in SDM-GAN with the WGAN-GP structure (Gulrajani et al., 2017). We vary the strength of the Lipschitz-1 constraint λ_{GP} in WGAN-GP

over $\lambda_{\text{GP}} \in \{1, 10, 50\}$, with 10 being the original default. All other hyperparameters of SDM-WGAN remain the same as in SDM-GAN. Detail results are presented in Table 2.

From the plot of average scores, Figure 2c, we see that the performance of SDM-WGAN is relatively robust to the choice of λ_{GP}. Compared with SDM-GAN, the three variants of SDM-WGAN overall perform worse, and the performances exhibit relatively larger variation across datasets. Furthermore, the performances of SDM-WGAN are less stable than SDM-GAN, as shown by the relatively larger variance in the results on many tested datasets. We believe that a thorough guidance to the hyperparameter setups and training schemes for WGAN-GP could potentially improve the performance of SDM-WGAN. Such guidance is currently missing from the literature. The investigation onto other instances of IPM, such as the Maximum Mean Discrepancy (Gretton et al., 2012), is left as future work.

Table 2. Normalized returns for comparing the default algorithm with variants of approximately matching the Wasserstein-1 dual via WGAN-GP. We vary $\lambda_{\text{GP}} \in \{1, 10, 50\}$. Here "med" denotes medium, "rep" for replay, and "exp" for expert.

Tasks/SDM Variants	GAN	$\lambda_{\text{GP}} = 1$	$\lambda_{\text{GP}} = 10$	$\lambda_{\text{GP}} = 50$
maze2d-umaze	36.1 ± 28.4	26.7 ± 42.2	28.4 ± 45.2	56.7 ± 15.0
maze2d-med	115.4 ± 34.2	99.8 ± 8.0	75.1 ± 10.9	96.0 ± 10.1
maze2d-large	207.7 ± 11.7	26.2 ± 30.0	44.3 ± 63.9	1.5 ± 9.5
halfcheetah-med	42.5 ± 0.5	44.3 ± 1.0	45.5 ± 0.2	44.4 ± 0.6
walker2d-med	66.7 ± 1.8	9.6 ± 12.7	21.6 ± 10.7	11.7 ± 12.7
hopper-med	62.8 ± 14.3	9.8 ± 9.2	3.5 ± 2.9	2.7 ± 3.4
halfcheetah-med-rep	41.7 ± 0.4	49.3 ± 0.9	49.3 ± 0.6	48.6 ± 0.5
walker2d-med-rep	20.3 ± 4.0	16.2 ± 10.7	6.6 ± 8.4	8.0 ± 6.2
hopper-med-rep	30.6 ± 2.8	37.2 ± 3.1	33.3 ± 4.2	28.1 ± 3.7
halfcheetah-med-exp	89.1 ± 6.6	17.1 ± 4.7	15.9 ± 3.9	22.1 ± 6.1
walker2d-med-exp	97.9 ± 4.9	66.2 ± 18.7	56.6 ± 23.9	55.3 ± 7.8
hopper-med-exp	104.5 ± 5.4	25.5 ± 2.5	25.6 ± 5.0	21.3 ± 6.3
pen-human	17.8 ± 1.7	27.4 ± 2.6	24.0 ± 7.8	18.7 ± 2.9
pen-cloned	40.6 ± 6.1	46.9 ± 18.1	44.0 ± 17.4	44.1 ± 11.3
pen-expert	135.8 ± 11.7	63.7 ± 26.7	67.1 ± 20.7	72.4 ± 22.9
door-expert	93.5 ± 6.7	63.5 ± 27.4	50.2 ± 36.4	59.1 ± 28.8
Average Score	75.2	39.3	36.9	36.9

6. Conclusion

In this paper, we directly approach a central difficulty in offline RL, *i.e.*, the mismatch between the distribution of the offline dataset and the undiscounted stationary state-action distribution of the learned policy. Specifically, we derive a tractable bound for the difference between the stationary distribution of the behavior policy in the underlying environmental dynamic and that of the learned policy *w.r.t.* the IPM. Based on the theoretical derivation, we design a practical model-based algorithm that uses this tractable bound as a regularizer in the policy improvement step. Our practical implementation is tested on a diverse array of continuous-control offline-RL datasets and shows competitive performance compared with several state-of-the-art model-based and model-free baselines. Ablation study is also provided to validate some of the key algorithmic choices.

Acknowledgements

The authors acknowledge the support of NSF IIS1812699 and ECCS-1952193, the support of a gift fund from ByteDance Inc., and the Texas Advanced Computing Center (TACC) for providing HPC resources that have contributed to the research results reported within this paper.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020.

Bellemare, M. G., Dabney, W., and Munos, R. A Distributional Perspective on Reinforcement Learning. In *International Conference on Machine Learning*, 2017.

Binkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. Demystifying MMD GANs. *ArXiv*, abs/1801.01401, 2018.

Cang, C., Rajeswaran, A., Abbeel, P., and Laskin, M. Behavioral Priors and Dynamics Models: Improving Performance and Domain Transfer in Offline RL. *ArXiv*, abs/2106.09119, 2021.

Casella, G. and Berger, R. *Statistical Inference*. Duxbury Resource Center, June 2001.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision Transformer: Reinforcement Learning via Sequence Modeling. *ArXiv*, abs/2106.01345, 2021.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. In *Advances in neural information processing systems*, 2018.

Clavera, I., Rothfuss, J., Schulman, J., Fujita, Y., Asfour, T., and Abbeel, P. Model-Based Reinforcement Learning via Meta-Policy Optimization. *ArXiv*, abs/1809.05214, 2018.

Dabney, W., Rowland, M., Bellemare, M., and Munos, R. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-Based Batch Mode Reinforcement Learning. *J. Mach. Learn. Res.*, 6: 503–556, 2005.

Fan, X., Zhang, S., Chen, B., and Zhou, M. Bayesian attention modules. *Advances in Neural Information Processing Systems*, 33:16362–16376, 2020.

Fan, X., Zhang, S., Tanwisuth, K., Qian, X., and Zhou, M. Contextual dropout: An efficient sample-dependent dropout module. *arXiv preprint arXiv:2103.04181*, 2021.

Ferguson, T. S. *A Course in Large Sample Theory*. Chapman & Hall, London, 1996.

Fu, J., Kumar, A., Soh, M., and Levine, S. Diagnosing Bottlenecks in Deep Q-learning Algorithms. In *International Conference on Machine Learning*, 2019.

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

Fujimoto, S. and Gu, S. S. A Minimalist Approach to Offline Reinforcement Learning. *ArXiv*, abs/2106.06860, 2021.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 10–15 Jul 2018.

Fujimoto, S., Meger, D., and Precup, D. Off-Policy Deep Reinforcement Learning without Exploration. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2052–2062. PMLR, 09–15 Jun 2019.

Gilotte, A., Calauzènes, C., Nedelec, T., Abraham, A., and Dollé, S. Offline A/B Testing for Recommender Systems. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018.

Goodfellow, I. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A. A Kernel Two-Sample Test. *J. Mach. Learn. Res.*, 13:723–773, 2012.

Gulcehre, C., Colmenarejo, S. G., ziyu wang, Sygnowski, J., Paine, T., Zolna, K., Chen, Y., Hoffman, M., Pascanu, R., and de Freitas, N. Addressing Extrapolation Error in Deep Offline Reinforcement Learning. 2021.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved Training of Wasserstein GANs. In *Advances in neural information processing systems*, 2017.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft Actor-Critic Algorithms and Applications. *ArXiv*, abs/1812.05905, 2018.

Hishinuma, T. and Senda, K. Weighted model estimation for offline model-based reinforcement learning. In *Advances in neural information processing systems*, 2021.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to Trust Your Model: Model-Based Policy Optimization. *ArXiv*, abs/1906.08253, 2019.

Janner, M., Li, Q., and Levine, S. Reinforcement Learning as One Big Sequence Modeling Problem. *ArXiv*, abs/2106.02039, 2021.

Jaques, N., Ghandeharioun, A., Shen, J. H., Ferguson, C., Lapedriza, À., Jones, N. J., Gu, S., and Picard, R. W. Way Off-Policy Batch Deep Reinforcement Learning of Implicit Human Preferences in Dialog. *ArXiv*, abs/1907.00456, 2019.

Jiang, N. and Li, L. Doubly robust off-policy evaluation for reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 652–661, 2016.

Kallus, N. and Zhou, A. Confounding-robust policy evaluation in infinite-horizon reinforcement learning. *Advances in Neural Information Processing Systems*, 33:22293–22304, 2020.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. MOReL : Model-Based Offline Reinforcement Learning. *ArXiv*, abs/2005.05951, 2020.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2014.

Kostrikov, I., Nair, A., and Levine, S. Offline Reinforcement Learning with Implicit Q-Learning. *ArXiv*, abs/2110.06169, 2021a.

Kostrikov, I., Tompson, J., Fergus, R., and Nachum, O. Offline Reinforcement Learning with Fisher Divergence Critic Regularization. In *International Conference on Machine Learning*, 2021b.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-Learning for Offline Reinforcement Learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191. Curran Associates, Inc., 2020.

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. Model-Ensemble Trust-Region Policy Optimization. *International Conference on Learning Representations*, abs/1802.10592, 2018.

Lange, S., Gabel, T., and Riedmiller, M. *Batch Reinforcement Learning*, pp. 45–73. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27645-3. doi: 10.1007/978-3-642-27645-3_2.

Laroche, R. and Trichelair, P. Safe Policy Improvement with Baseline Bootstrapping. In *International Conference on Machine Learning*, 2019.

Lee, B.-J., Lee, J., and Kim, K.-E. Representation Balancing Offline Model-based Reinforcement Learning. In *International Conference on Learning Representations*, 2021a.

Lee, J., Jeon, W., Lee, B.-J., Pineau, J., and Kim, K.-E. OptiDICE: Offline Policy Optimization via Stationary Distribution Correction Estimation. *ArXiv*, abs/2106.10783, 2021b.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Liao, P., Qi, Z., Klasnja, P., and Murphy, S. Batch policy learning in average reward markov decision processes. *arXiv preprint arXiv:2007.11771*, 2020.

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

Lin, J. Divergence Measures Based on the Shannon Entropy. *IEEE Transactions on Information theory*, 37:145–151, 1991.

Lin, L.-J. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Machine Learning*, 8(3–4):293–321, 1992.

Liu, Q., Li, L., Tang, Z., and Zhou, D. Breaking the Curse of Horizon: Infinite-Horizon Off-Policy Estimation. In *Advances in neural information processing systems*, 2018.

Lu, K., Grover, A., Abbeel, P., and Mordatch, I. Reset-free lifelong learning with skill-space planning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Matsushima, T., Furuta, H., Matsuo, Y., Nachum, O., and Gu, S. S. Deployment-Efficient Reinforcement Learning via Model-Based Offline Optimization. *International Conference on Learning Representations*, abs/2006.03647, 2021.

Mirza, M. and Osindero, S. Conditional Generative Adversarial Nets. *ArXiv*, abs/1411.1784, 2014.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, February 2015. ISSN 00280836.

Müller, A. Integral Probability Metrics and Their Generating Classes of Functions. *Advances in Applied Probability*, 29:429–443, 1997.

Nachum, O., Chow, Y., Dai, B., and Li, L. DualDICE: Behavior-Agnostic Estimation of Discounted Stationary Distribution Corrections. In *Advances in neural information processing systems*, 2019a.

Nachum, O., Dai, B., Kostrikov, I., Chow, Y., Li, L., and Schuurmans, D. AlgaeDICE: Policy Gradient from Arbitrary Experience. *ArXiv*, abs/1912.02074, 2019b.

Nie, X., Brunskill, E., and Wager, S. Learning When-to-Treat Policies. *Journal of the American Statistical Association*, 116:392 – 409, 2019.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Radford, A. and Sutskever, I. Improving Language Understanding by Generative Pre-Training. In *arxiv*, 2018.

Radford, A., Metz, L., and Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR*, abs/1511.06434, 2016.

Rajeswaran, A., Kumar, V., Gupta, A., Schulman, J., Todorov, E., and Levine, S. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. *ArXiv*, abs/1709.10087, 2018.

Rajeswaran, A., Mordatch, I., and Kumar, V. A Game Theoretic Framework for Model Based Reinforcement Learning. In *International Conference on Machine Learning*, 2020.

Ross, S. and Bagnell, J. A. Agnostic System Identification for Model-Based Reinforcement Learning. *International Conference on Machine Learning*, abs/1203.1007, 2012.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved Techniques for Training GANs. In *Advances in neural information processing systems*, 2016.

Siegel, N. Y., Springenberg, J. T., Berkenkamp, F., Abdolmaleki, A., Neunert, M., Lampe, T., Hafner, R., Heess, N., and Riedmiller, M. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489, January 2016. doi: 10.1038/nature16961.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., and Hassabis, D. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *CoRR*, abs/1712.01815, 2017.

Sinha, S., Mandlekar, A., and Garg, A. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *Conference on Robot Learning*, pp. 907–917. PMLR, 2022.

Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Scholkopf, B., and Lanckriet, G. R. G. On integral probability metrics, ϕ-divergences and binary classification. *arXiv: Information Theory*, 2009.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Swazinna, P., Udluft, S., and Runkler, T. A. Overcoming Model Bias for Robust Offline Deep Reinforcement Learning. *Eng. Appl. Artif. Intell.*, 104:104366, 2021.

Tang, Z., Feng, Y., Li, L., Zhou, D., and Liu, Q. Doubly robust bias reduction in infinite horizon off-policy estimation. In *International Conference on Learning Representations*, 2020.

Urpí, N. A., Curi, S., and Krause, A. Risk-Averse Offline Reinforcement Learning. *ArXiv*, abs/2102.05371, 2021.

Wang, Z., Novikov, A., Zolna, K., Merel, J. S., Springenberg, J. T., Reed, S. E., Shahriari, B., Siegel, N., Gulcehre, C., Heess, N., and de Freitas, N. Critic Regularized Regression. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7768–7778. Curran Associates, Inc., 2020.

Wasserman, L. *All of Nonparametric Statistics*. Springer, 2006.

White, T. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Wu, Y., Zhai, S., Srivastava, N., Susskind, J., Zhang, J., Salakhutdinov, R., and Goh, H. Uncertainty Weighted Actor-Critic for Offline Reinforcement Learning. In *International Conference on Machine Learning*, 2021.

Yang, S., Wang, Z., Zheng, H., Feng, Y., and Zhou, M. A regularized implicit policy for offline reinforcement learning. *arXiv preprint arXiv:2202.09673*, 2022.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. MOPO: Model-based Offline Policy Optimization. *ArXiv*, abs/2005.13239, 2020.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. COMBO: Conservative Offline Model-Based Policy Optimization. *ArXiv*, abs/2102.08363, 2021.

Yue, Y., Wang, Z., and Zhou, M. Implicit Distributional Reinforcement Learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Yurtsever, E., Lambert, J., Carballo, A., and Takeda, K. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access*, 8:58443–58469, 2020.

Zhang, R., Dai, B., Li, L., and Schuurmans, D. Gendice: Generalized offline estimation of stationary values. In *International Conference on Learning Representations*, 2020.

Zhang, S., Fan, X., Zheng, H., Tanwisuth, K., and Zhou, M. Alignment attention by matching key and query distributions. *Advances in Neural Information Processing Systems*, 34, 2021.

Zhang, S., Gong, C., Liu, X., He, P., Chen, W., and Zhou, M. Allsh: Active learning guided by local sensitivity and hardness. *arXiv preprint arXiv:2205.04980*, 2022.

Zheng, H. and Zhou, M. Exploiting Chain Rule and Bayes' Theorem to Compare Probability Distributions. *Advances in Neural Information Processing Systems*, 34, 2021.

Appendix

A. Additional Tables

Tables 3 - 4 correspond to the results for the first two ablation study in Section 5.3.

Table 3. Normalized returns for comparing the performance of our algorithm under different real data percentage $f \in \{0, 0.05, 0.5, 0.8, 1\}$. The reported numbers are the means and standard deviation across three random seeds $\{0, 1, 2\}$.

Task Name	SDM-GAN($f = 0$)	SDM-GAN($f = 0.05$)	SDM-GAN($f = 0.5$)	SDM-GAN($f = 0.8$)	SDM-GAN($f = 1$)
maze2d-umaze	41.1 ± 33.2	22.5 ± 29.6	36.1 ± 28.4	39.3 ± 20.8	89.8 ± 49.2
maze2d-medium	79.2 ± 27.6	71.5 ± 17.0	115.4 ± 34.2	90.8 ± 49.6	89.3 ± 38.8
maze2d-large	96.8 ± 60.8	52.0 ± 63.2	207.7 ± 11.7	127.1 ± 62.1	101.4 ± 73.6
halfcheetah-medium	39.7 ± 0.6	39.8 ± 1.3	42.5 ± 0.5	43.1 ± 0.3	43.9 ± 0.1
walker2d-medium	52.6 ± 12.9	49.6 ± 2.3	66.7 ± 1.8	58.9 ± 12.4	58.3 ± 10.7
hopper-medium	44.6 ± 19.7	44.5 ± 32.0	62.8 ± 14.3	68.2 ± 22.3	54.5 ± 6.2
halfcheetah-medium-replay	41.1 ± 0.4	41.3 ± 0.5	41.7 ± 0.4	40.5 ± 0.8	36.6 ± 2.7
walker2d-medium-replay	13.2 ± 2.7	14.0 ± 2.2	20.3 ± 4.0	18.4 ± 3.6	4.0 ± 2.0
hopper-medium-replay	21.6 ± 0.8	23.7 ± 2.2	30.6 ± 2.8	28.2 ± 1.6	34.8 ± 9.4
halfcheetah-medium-expert	69.2 ± 11.1	90.2 ± 6.2	89.1 ± 6.6	91.2 ± 7.9	71.0 ± 11.0
walker2d-medium-expert	26.6 ± 28.4	61.5 ± 21.8	97.9 ± 4.9	83.5 ± 16.1	67.8 ± 16.0
hopper-medium-expert	33.0 ± 25.4	32.8 ± 17.3	104.5 ± 5.4	99.9 ± 9.7	80.7 ± 10.6
pen-human	10.6 ± 8.0	23.8 ± 18.1	17.8 ± 1.7	19.3 ± 10.1	6.8 ± 3.8
pen-cloned	17.9 ± 14.8	30.5 ± 12.8	40.6 ± 6.1	28.7 ± 7.4	20.5 ± 11.9
pen-expert	116.3 ± 21.0	125.0 ± 7.1	135.8 ± 11.7	114.8 ± 19.2	113.5 ± 14.0
door-expert	35.8 ± 43.2	61.1 ± 42.8	93.5 ± 6.7	68.0 ± 26.2	70.3 ± 14.2
Average Score	46.2	49	75.2	63.7	59

Table 4. Normalized returns for comparing our algorithm under different regularization coefficient $\alpha \in \{2.5, 10, 25\}$ and without regularization (No Reg.). The reported number are the means and standard deviation across three random seeds $\{0, 1, 2\}$.

Task Name	SDM-GAN($\alpha = 2.5$)	SDM-GAN($\alpha = 10$)	SDM-GAN($\alpha = 25$)	SDM-GAN(No Reg.)
maze2d-umaze	37.2 ± 30.8	36.1 ± 28.4	66.1 ± 24.3	35.0 ± 64.6
maze2d-medium	95.3 ± 39.1	115.4 ± 34.2	93.0 ± 45.2	65.8 ± 53.1
maze2d-large	103.7 ± 6.6	207.7 ± 11.7	162.7 ± 46.1	-1.4 ± 1.4
halfcheetah-medium	42.2 ± 0.6	42.5 ± 0.5	43.0 ± 0.5	52.6 ± 2.9
walker2d-medium	58.7 ± 7.5	66.7 ± 1.8	58.7 ± 9.0	1.2 ± 2.4
hopper-medium	46.8 ± 23.0	62.8 ± 14.3	44.4 ± 14.6	1.5 ± 0.7
halfcheetah-medium-replay	39.6 ± 0.3	41.7 ± 0.4	43.0 ± 0.4	56.3 ± 2.4
walker2d-medium-replay	15.9 ± 3.8	20.3 ± 4.0	22.7 ± 0.8	16.3 ± 12.5
hopper-medium-replay	28.4 ± 3.0	30.6 ± 2.8	30.9 ± 3.1	44.6 ± 38.4
halfcheetah-medium-expert	62.2 ± 10.5	89.1 ± 6.6	89.6 ± 7.8	41.1 ± 10.0
walker2d-medium-expert	69.4 ± 39.4	97.9 ± 4.9	90.9 ± 12.4	25.4 ± 16.4
hopper-medium-expert	96.1 ± 15.9	104.5 ± 5.4	79.9 ± 31.3	32.9 ± 13.4
pen-human	17.1 ± 11.4	17.8 ± 1.7	23.3 ± 12.8	-2.7 ± 1.6
pen-cloned	29.8 ± 14.6	40.6 ± 6.1	34.9 ± 19.5	-0.3 ± 4.5
pen-expert	115.4 ± 14.9	135.8 ± 11.7	114.2 ± 18.6	-2.6 ± 1.8
door-expert	62.0 ± 34.4	93.5 ± 6.7	81.4 ± 28.2	-0.2 ± 0.1
Average Score	57.5	75.2	67.4	22.8

B. Theoretical Analysis and Proofs

B.1. Preliminary on Average Reward Markov Decision Process

Our proof largely relies on the preliminary knowledge on average (undiscounted) reward Markov decision process (Puterman, 2014), so we will briefly introduce the general background on the undiscounted reward infinite-horizon RL.

Assumptions on Markov Decision Process (MDP). Follow the classic RL settings (Puterman, 2014), we model RL problem as as a Markov Decision Process, which consists of $\mathcal{M} = (\mathbb{S}, \mathbb{A}, P, r)$, where $\mathbb{S}$ denots the state space, $\mathbb{A}$ the action space, $P(s'|s,a)$ the transition probability, which we assume to be time independent; $r(s,a)$ the intermediate reward function, which we assume to be deterministic and bounded ($|r(s,a)| \leq r_{\max}$).

Consider a time-stationary, Markov policy π, which specifies a distribution of actions given states, and $\pi(a|s)$ denotes the probability of selecting a given s. The average reward of the policy π is defined as

$$\eta^\pi(s) := \lim_{T\to\infty} \mathbb{E}_{\pi,P}\left[\frac{1}{T+1}\sum_{t=0}^{T} r_t \mid s_0 = s\right], \tag{12}$$

where the expectation $\mathbb{E}_{\pi,P}$ is with respect to the distribution of the trajectories where the states evolve according to P and the actions are chosen by $\pi(a|s)$. When the states of the MDP is finite and the reward is bounded, the limit in Eq. (12) always exists (Puterman, 2014). The policy induces a Markov chain of states with the transition as $P^\pi(s'|s) = \sum_a \pi(a|s)P(s'|s,a)$. When P^π is irreducible, it can be shown that the stationary distribution of P^π exists and is unique (denoted by d_π), and the average reward, $\eta^\pi(s)$ in Eq. (12) is independent of the initial state s (*e.g.*, Puterman, 2014), thus we have

$$\eta^\pi = \eta^\pi(s) = \sum_{s,a} d_\pi(s)\pi(a|s)r(s,a)\,. \tag{13}$$

Also $d_\pi(s,a) = d_\pi(s)\pi(a|s)$ is the stationary state-action distribution induced by policy π under the transition $P(s'|s,a)$, which satisfy

$$d_\pi(s',a') = \sum_{s,a} \pi(a'|s')P(s'|s,a)d_\pi(s,a), \quad \forall (s',a') \in \mathbb{S}\times\mathbb{A}\,. \tag{14}$$

From previous assumption we know the fixed point solution of Eq. (14) exists and is unique. If we introduce a discriminative test function $g(s,a) \in \mathcal{G}$, with $\mathcal{G} := \{g : \mathbb{S}\times\mathbb{A}\to\mathbb{R}, \|g\|_\infty \leq G_{\max}\}$ as previous off-policy evaluation literature (e.g., Liu et al., 2018; Zhang et al., 2020; Tang et al., 2020), we can show

$$\mathbb{E}_{(s,a)\sim d_\pi}\left[g(s,a)\right] = \mathbb{E}_{(s,a)\sim d_\pi, s'\sim P(\cdot|s,a), a'\sim\pi(\cdot|s')}\left[g(s',a')\right], \quad \forall g \in \mathcal{G}\,. \tag{15}$$

Differential Value Function. Unlike the discounted case where we define the value function $Q^\pi(s,a)$ as the expected total discounted reward when the initial state action pair is (s,a): $Q^\pi(s,a) := \mathbb{E}_{\pi,P}\left[\sum_{t=0}^\infty \gamma^t r_t | s_0 = s, a_0 = a\right]$, in the average case, $Q^\pi(s,a)$ represents the *expected total difference* between the reward and the average reward η^π under the policy π, which is called *differential (state-action) value function*[1] in Sutton & Barto (2018):

$$Q^\pi(s,a) := \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty}(r_t - \eta^\pi) \;\mid\; s_0 = s, a_0 = a\right].$$

The differential value function $Q^\pi(s,a)$ and the average reward η^π are closely related via the Bellman equation:

$$Q^\pi(s,a) = r(s,a) + \mathbb{E}_{s'\sim P(\cdot|s,a), a'\sim\pi(\cdot|s')}\left[Q^\pi(s',a')\right] - \eta^\pi, \quad \forall (s,a) \in \mathbb{S}\times\mathbb{A}\,. \tag{16}$$

For more details of the average reward RL, we refer the reader to classical RL books (*e.g.*, Puterman, 2014; Sutton & Barto, 2018).

B.2. Proof of Theorem 3.1

Theorem (Restatement of Theorem 3.1). Suppose the distance metric D is the integral probability metrics (IPM) *w.r.t* some function class $\mathcal{G} = \{g : \mathbb{S}\times\mathbb{A}\to\mathbb{R}\}$, defined as

$$D_{\mathcal{G}}\left(d_{\pi_b}^{P^*}, d_\pi^{\widehat{P}}\right) := \sup_{g\in\mathcal{G}}\left|\mathbb{E}_{d_{\pi_b}^{P^*}}\left[g(s,a)\right] - \mathbb{E}_{d_\pi^{\widehat{P}}}\left[g(s,a)\right]\right|,$$

[1] It is also referred as *relative* value function in some RL literature.

then there exists a function class $\mathcal{G}$ such that we can rewrite $D_{\mathcal{G}}(d_{\pi_b}^{P^*}, d_{\pi}^{\widehat{P}})$ as:

$$D_{\mathcal{G}}(d_{\pi_b}^{P^*}, d_{\pi}^{\widehat{P}}) = \mathcal{R}_{\mathcal{F}}(d_{\pi_b}^{P^*}, \pi, \widehat{P}) := \sup_{f \in \mathcal{F}} \left| \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}}[f(s,a)] - \mathbb{E}_{\substack{s'\sim \widehat{P}(\cdot|s,a), a'\sim\pi(\cdot|s') \\ (s,a)\sim d_{\pi_b}^{P^*}}}[f(s',a')] \right|,$$

where

$$\mathcal{F} := \left\{ f : \ f(s,a) = \mathbb{E}_{\pi,\widehat{P}} \left[\sum_{t=0}^{\infty} (g(s_t,a_t) - \hat{\eta}^{\pi}) \ \middle| \ \begin{matrix} a_0 = a \\ s_0 = s \end{matrix} \right] \hat{\eta}^{\pi} = \lim_{T\to\infty} \mathbb{E}_{\pi,\widehat{P}} \left[\frac{1}{T+1} \sum_{t=0}^{T} g(s_t,a_t) \right], g \in \mathcal{G} \right\}.$$

Proof. Based on the preliminary introduction in Section. B.1, we introduce a new MDP $\widehat{\mathcal{M}} = (\mathbb{S}, \mathbb{A}, \widehat{P}, g)$, where $\widehat{P}$ is the learned dynamics optimized by maximum likelihood estimation, and $g \in \mathcal{G}$, with $\mathcal{G} := \{g : \mathbb{S}\times\mathbb{A} \to \mathbb{R}, \|g(s,a)\|_{\infty} \le G_{\max}\}$.

Suppose the policy π is a time-stationary, Markov policy, and $\widehat{P}^{\pi}(s'|s) := \sum_a \widehat{P}(s'|s,a)\pi(a|s)$ is finite state and ergodic. $d_{\pi}^{\widehat{P}}(s,a)$ is the stationary state-action distribution induced by policy π under the learned transition $\widehat{P}$. Then we have the definition of the average reward $\widehat{\eta}^{\pi}$ and differential value function $\widehat{Q}^{\pi}(s,a)$ for the MDP $\widehat{\mathcal{M}}$:

$$\widehat{\eta}^{\pi} = \lim_{T\to\infty} \mathbb{E}_{\pi,\widehat{P}} \left[\frac{1}{T+1} \sum_{t=0}^{T} g(s_t,a_t) \right] = \mathbb{E}_{(s,a)\sim d_{\pi}^{\widehat{P}}}[g(s,a)], \tag{17}$$

$$\widehat{Q}^{\pi}(s,a) := \mathbb{E}_{\pi,\widehat{P}} \left[\sum_{t=0}^{\infty} (g(s_t,a_t) - \widehat{\eta}^{\pi}) \ \middle| \ s_0 = s, a_0 = a \right], \tag{18}$$

$$\widehat{Q}^{\pi}(s,a) = g(s,a) + \mathbb{E}_{s'\sim\widehat{P}(\cdot|s,a), a'\sim\pi(\cdot|s')} \left[\widehat{Q}^{\pi}(s',a') \right] - \widehat{\eta}^{\pi}, \qquad \forall\, (s,a) \in \mathbb{S}\times\mathbb{A}. \tag{19}$$

We next introduce a new function class $\mathcal{F}$ such that for each function $g \in \mathcal{G}$, we have a corresponding function $f \in \mathcal{F}$, which can represent the differential value function $\widehat{Q}^{\pi}$ given the reward function g and the dynamics $\widehat{P}$:

$$\mathcal{F} := \left\{ f : \ f(s,a) = \mathbb{E}_{\pi,\widehat{P}} \left[\sum_{t=0}^{\infty} (g(s_t,a_t) - \widehat{\eta}^{\pi}) \ \middle| \ \begin{matrix} a_0 = a \\ s_0 = s \end{matrix} \right], \widehat{\eta}^{\pi} = \lim_{T\to\infty} \mathbb{E}_{\pi,\widehat{P}} \left[\frac{1}{T+1} \sum_{t=0}^{T} g(s_t,a_t) \right], g \in \mathcal{G} \right\}. \tag{20}$$

Moreover, under classical regularity conditions made in Section B.1, if we fix the model dynamics $\widehat{P}$, for each reward function $g(s,a)$, we have a unique one-to-one corresponding function $f \in \mathcal{F}$ because the differential value function is the unique solution of the corresponding Bellman equation. Besides, since the reward function is bounded, we assume $\sup_{f\in\mathcal{F}} \|f\|_{\infty} \le F_{\max}$, such that $\mathcal{F}$ is a bounded function class (Liao et al., 2020). Then we have

$$\begin{aligned}
D_{\mathcal{G}}\left(d_{\pi_b}^{P^*}, d_{\pi}^{\widehat{P}}\right) &= \sup_{g\in\mathcal{G}} \left| \mathbb{E}_{d_{\pi_b}^{P^*}}[g(s,a)] - \underbrace{\mathbb{E}_{d_{\pi}^{\widehat{P}}}[g(s,a)]}_{\widehat{\eta}^{\pi}} \right| \\
&= \sup_{f\in\mathcal{F}} \left| \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}} \left[f(s,a) - \mathbb{E}_{s'\sim\widehat{P}(\cdot|s,a), a'\sim\pi(\cdot|s')}[f(s',a')] + \widehat{\eta}^{\pi} \right] - \widehat{\eta}^{\pi} \right| \\
&= \sup_{f\in\mathcal{F}} \left| \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}}[f(s,a)] - \mathbb{E}_{\substack{s'\sim\widehat{P}(\cdot|s,a), a'\sim\pi(\cdot|s') \\ (s,a)\sim d_{\pi_b}^{P^*}}}[f(s',a')] \right| \\
&:= \mathcal{R}_{\mathcal{F}}(d_{\pi_b}^{P^*}, \pi, \widehat{P}).
\end{aligned}$$

If we choose the function class $\mathcal{F}$ to be a parameterized neural network, then it should be rich enough to contain the true differential value function $\widehat{Q}^{\pi}(s,a)$ induced by dynamics $\widehat{P}$ and the deterministic reward function $g(s,a)$, because in practice we usually use a neural network to learn the value function $\widehat{Q}^{\pi}(s,a)$. □

B.3. Proof of Theorem 3.2

Before we prove Theorem 3.2, we introduce the following lemma that will be used in the latter proof.

Lemma B.1. *Denote P as an arbitrary probability measure, $\mathcal{Y}$ an arbitrary set. Let $X \sim P, y \in \mathcal{Y}$, we have*

$$\sup_{y\in\mathcal{Y}} \mathbb{E}_{X\sim P}[f(X,y)] \le \mathbb{E}_{X\sim P}\left[\sup_{y\in\mathcal{Y}} f(X,y)\right].$$

Proof. $\forall\, y \in \mathcal{Y}, f(x,y) \le \sup_{y\in\mathcal{Y}} f(x,y), \forall\, x \implies \forall\, y \in \mathcal{Y}, \mathbb{E}_{X\sim P}\left[f(X,y)\right] \le \mathbb{E}_{X\sim P}\left[\sup_{y\in\mathcal{Y}} f(X,y)\right]$. Taking sup on the left-hand-side, we have $\sup_{y\in\mathcal{Y}} \mathbb{E}_{X\sim P}\left[f(X,y)\right] \le \mathbb{E}_{X\sim P}\left[\sup_{y\in\mathcal{Y}} f(X,y)\right]$. □

Theorem (Restatement of Theorem 3.2). Let P^* be the true dynamics, $D_{\mathcal{G}}$ the IPM with the same $\mathcal{G}$ as Theorem 3.1 and $\widehat{P}$ the estimated dynamics. There exist a constant $C \ge 0$ and a function class $\Psi = \{\psi : \mathbb{S}\times\mathbb{A} \to \mathbb{R}\}$ such that for any policy π,

$$D_{\mathcal{G}}(d_\pi^{\widehat{P}}, d_\pi^{P^*}) \le C \cdot \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}}\left[\sqrt{\frac{1}{2}\mathrm{KL}\left(P^*(s'|s,a)\|\widehat{P}(s'|s,a)\right)}\right] + \mathcal{R}_\Psi(d_{\pi_b}^{P^*}, \pi),$$

where

$$\mathcal{R}_\Psi(d_{\pi_b}^{P^*}, \pi) := \sup_{\psi\in\Psi}\left|\mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}}[\psi(s,a)] - \mathbb{E}_{\substack{s\sim d_{\pi_b}^{P^*}\\ a\sim\pi(\cdot|s)}}[\psi(s,a)]\right|.$$

Proof. Still, we will use the proof technique when we prove Proposition 3.1.

We construct a new MDP $\widehat{\mathcal{M}} = (\mathbb{S}, \mathbb{A}, \widehat{P}, g)$, where $\widehat{P}$ is the learned dynamics and $g \in \mathcal{G}$ is an intermediate reward function, with $\mathcal{G} := \{g : \mathbb{S}\times\mathbb{A}\to\mathbb{R}, \|g(s,a)\|_\infty \le G_{\max}\}$. The definition of average reward $\widehat{\eta}^\pi$, stationary distribution $d_\pi^{\widehat{P}}(s,a)$, differential value function $\widehat{Q}^\pi(s,a)$, and the function class $\mathcal{F}$ are the same as in Section B.2. if the distance metric is the integral probability metric (IPM) defined on test function class $\mathcal{G}$, then we have

$$D_{\mathcal{G}}\left(d_\pi^{\widehat{P}}, d_\pi^{P^*}\right) := \sup_{g\in\mathcal{G}}\left|\mathbb{E}_{d_\pi^{\widehat{P}}}[g(s,a) - \mathbb{E}_{d_\pi^{P^*}}[g(s,a)]\right|.$$

Before we take the supremum over the function class $\mathcal{G}$, we first rewrite the term inside the sup:

$$\begin{aligned}
\left|\mathbb{E}_{d_\pi^{\widehat{P}}}[g(s,a) - \mathbb{E}_{d_\pi^{P^*}}[g(s,a)]\right| &= \left|\underbrace{\mathbb{E}_{d_\pi^{\widehat{P}}}[g(s,a)]}_{\widehat{\eta}^\pi} - \mathbb{E}_{d_\pi^{P^*}}[g(s,a)]\right| \\
&= \left|\widehat{\eta}^\pi - \mathbb{E}_{d_\pi^{P^*}}\left[\widehat{Q}^\pi(s,a)\right] + \mathbb{E}_{d_\pi^{P^*}}\left[\widehat{Q}^\pi(s,a)\right] - \mathbb{E}_{d_\pi^{P^*}}[g(s,a)]\right| \quad // \ \widehat{Q}^\pi(s,a) \text{ defined in (18)} \\
&= \left|-\mathbb{E}_{d_\pi^{P^*}}\left[\widehat{Q}^\pi(s,a)\right] + \mathbb{E}_{d_\pi^{P^*}}\left[\widehat{Q}^\pi(s,a) - g(s,a) + \widehat{\eta}^\pi\right]\right| \\
&= \left|-\mathbb{E}_{d_\pi^{P^*}}\left[\widehat{Q}^\pi(s,a)\right] + \mathbb{E}_{(s,a)\sim d_\pi^{P^*}}\left[\mathbb{E}_{s'\sim\widehat{P}(\cdot|s,a), a'\sim\pi(\cdot|s')}\left[\widehat{Q}^\pi(s',a')\right]\right]\right| \quad // \ \text{use (19)} \\
&= \left|\underbrace{-\mathbb{E}_{(s,a)\sim d_\pi^{P^*}}\left[\mathbb{E}_{\substack{a'\sim\pi(\cdot|s')\\ s'\sim P^*(\cdot|s,a)}}\left[\widehat{Q}^\pi(s',a')\right]\right]}_{\text{use (15)}} + \mathbb{E}_{(s,a)\sim d_\pi^{P^*}}\left[\mathbb{E}_{\substack{a'\sim\pi(\cdot|s')\\ s'\sim\widehat{P}(\cdot|s,a)}}\left[\widehat{Q}^\pi(s',a')\right]\right]\right| \\
&= \left|\mathbb{E}_{(s,a)\sim d_\pi^{P^*}}\left[\underbrace{\mathbb{E}_{\substack{a'\sim\pi(\cdot|s')\\ s'\sim\widehat{P}(\cdot|s,a)}}\left[\widehat{Q}^\pi(s',a')\right] - \mathbb{E}_{\substack{a'\sim\pi(\cdot|s')\\ s'\sim P^*(\cdot|s,a)}}\left[\widehat{Q}^\pi(s',a')\right]}_{\stackrel{\text{def}}{=}\mathcal{E}(s,a;\widehat{P})}\right]\right| \\
&= \left|\mathbb{E}_{(s,a)\sim d_\pi^{P^*}}\left[\mathcal{E}(s,a;\widehat{P})\right]\right|.
\end{aligned}$$

Similarly, for each reward function $g(s,a) \in \mathcal{G}$, we have a unique one-to-to corresponding function $f \in \mathcal{F}$, with the function class $\mathcal{F}$ defined in Eq. (20). Then we can change from taking $\sup$ *w.r.t.* the function class $\mathcal{G}$ to *w.r.t.* the function class $\mathcal{F}$:

$$\begin{aligned} D_{\mathcal{G}}\left(d_\pi^{\widehat{P}}, d_\pi^{P^*}\right) &= \sup_{f\in\mathcal{F}} \left| \mathbb{E}_{(s,a)\sim d_\pi^{P^*}} \left[\mathcal{E}(s,a;\widehat{P})\right] \right| \\ &= \sup_{f\in\mathcal{F}} \left| \int\int \mathcal{E}(s,a;\widehat{P}) \left(d_\pi^{P^*}(s,a) - d_{\pi_b}^{P^*}(s,a) + d_{\pi_b}^{P^*}(s,a) \right) dads \right| \\ &\le \underbrace{\sup_{f\in\mathcal{F}} \left| \int\int \mathcal{E}(s,a;\widehat{P}) d_{\pi_b}^{P^*}(s,a) dads \right|}_{\stackrel{\text{def}}{=} ①} + \underbrace{\sup_{f\in\mathcal{F}} \left| \int\int \mathcal{E}(s,a;\widehat{P}) \left(d_\pi^{P^*}(s,a) - d_{\pi_b}^{P^*}(s,a) \right) dads \right|}_{\stackrel{\text{def}}{=} ②} . \end{aligned}$$

Next we are going to analyze ① and ② separately.

Analysis on ①. Based on the definition of function class $\mathcal{F}$, we introduce a new function class $\mathcal{V}$, which is defined as

$$\mathcal{V} \triangleq \left\{ v : \; v(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[f(s,a)], f \in \mathcal{F} \right\} ,$$

where $v(s)$ can be viewed as the state value function under the MDP $\widehat{\mathcal{M}} = (\mathbb{S}, \mathbb{A}, \widehat{P}, g)$, and $f(s,a)$ is the state-action value function under the definition, and $\|v\|_\infty \le F_{\max}$ since $\|f\|_\infty \le F_{\max}$. Recall that the total variation distance is a special instance of the integral probability metrics $D_{\mathcal{G}}$ when the function class $\mathcal{G} = \{g : \|g\|_\infty \le 1\}$. (Sriperumbudur et al., 2009; Binkowski et al., 2018). Thus we have

$$\begin{aligned} ① &= \sup_{f\in\mathcal{F}} \left| \int\int \mathcal{E}(s,a;\widehat{P}) d_{\pi_b}^{P^*}(s,a) dads \right| \\ &= \sup_{f\in\mathcal{F}} \left| \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}} \left[\mathbb{E}_{\substack{a'\sim\pi(\cdot|s') \\ s'\sim\widehat{P}(\cdot|s,a)}} [f(s',a')] - \mathbb{E}_{\substack{a'\sim\pi(\cdot|s') \\ s'\sim P^*(\cdot|s,a)}} [f(s',a')] \right] \right| \\ &= \sup_{v\in\mathcal{V}} \left| \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}} \left[\mathbb{E}_{s'\sim\widehat{P}(\cdot|s,a)} [v(s')] - \mathbb{E}_{s'\sim P^*(\cdot|s,a)} [v(s')] \right] \right| \\ &\le \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}} \left[\sup_{v\in\mathcal{V}} \left| \mathbb{E}_{s'\sim\widehat{P}(\cdot|s,a)} [v(s')] - \mathbb{E}_{s'\sim P^*(\cdot|s,a)} [v(s')] \right| \right] \\ &= F_{\max} \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}} \left[\sup_{v\in\mathcal{V}} \left| \mathbb{E}_{s'\sim\widehat{P}(\cdot|s,a)} \left[\frac{v(s')}{F_{\max}} \right] - \mathbb{E}_{s'\sim P^*(\cdot|s,a)} \left[\frac{v(s')}{F_{\max}} \right] \right| \right] \\ &= F_{\max} \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}} \left[\mathrm{TV}\left(P^*(\cdot\,|\,s,a) \| \widehat{P}(\cdot\,|\,s,a) \right) \right] \\ &\le F_{\max} \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}} \left[\sqrt{\frac{1}{2} \mathrm{KL}\left(P^*(\cdot\,|\,s,a) \| \widehat{P}(\cdot\,|\,s,a) \right)} \right] . \end{aligned} \tag{21}$$

since $\sup_{v\in\mathcal{V}} \left\| \frac{v}{F_{\max}} \right\|_\infty \le 1$, where we use Lemma B.1 to exchange order between $\sup$ and $\mathbb{E}[\cdot]$, and the last bound is from the Pinsker's inequality.

Analysis on ②. From previous assumption we know $\sup_{v\in\mathcal{V}} \|v\|_\infty \le F_{\max}$, then we can bound $\mathcal{E}(s,a;\widehat{P})$ by

$$\begin{aligned} \left| \mathcal{E}(s,a;\widehat{P}) \right| &= \left| \mathbb{E}_{s'\sim\widehat{P}(\cdot|s,a)}[v(s')] - \mathbb{E}_{s'\sim P^*(\cdot|s,a)}[v(s')] \right| \\ &\le \left| \mathbb{E}_{s'\sim\widehat{P}(\cdot|s,a)}[v(s')] \right| + \left| \mathbb{E}_{s'\sim P^*(\cdot|s,a)}[v(s')] \right| \\ &\le 2F_{\max} . \end{aligned}$$

Denote the function class for $\mathcal{E}(s,a;\widehat{P})$ as $\mathcal{B} := \left\{ \mathcal{E}(s,a;\widehat{P}), \widehat{P} \in \mathcal{P} \right\}$, which is a bounded function class since

$\left|\mathcal{E}(s,a;\widehat{P})\right| \leq 2F_{\max}, \forall\, \mathcal{E}(s,a;\widehat{P}) \in \mathcal{B}$. Since $\mathcal{E}(s,a;\widehat{P})$ is linear *w.r.t.* f, we have

$$
\begin{aligned}
② &= \sup_{f\in\mathcal{F}} \left| \int\int \mathcal{E}(s,a;\widehat{P}) \left(d_{\pi}^{P^*}(s,a) - d_{\pi_b}^{P^*}(s,a) \right) dads \right| \\
&= \sup_{\mathcal{E}(s,a;\widehat{P})\in\mathcal{B}} \left| \mathbb{E}_{d_{\pi_b}^{P^*}}[\mathcal{E}(s,a;\widehat{P})] - \mathbb{E}_{d_{\pi}^{P^*}}[\mathcal{E}(s,a;\widehat{P})] \right| .
\end{aligned}
$$

Still, we will use change of the variable trick as we use in the proof of Proposition 3.1 to make ② tractable. Define function class Ψ:

$$
\Psi := \left\{ \psi :\ \psi(s,a) = \mathbb{E}_{\pi,P^*}\left[\sum_{t=0}^{\infty} \left(\mathcal{E}(s_t,a_t;\widehat{P}) - \eta^{\pi} \right) \mid \begin{matrix} a_0 = a \\ s_0 = s \end{matrix} \right], \eta^{\pi} = \lim_{T\to\infty} \mathbb{E}_{\pi,P^*}\left[\frac{1}{T+1} \sum_{t=0}^{T} \mathcal{E}(s_t,a_t;\widehat{P}) \right], \mathcal{E}(s,a;\widehat{P}) \in \mathcal{B} \right\}.
$$

Under classical regularity conditions made in Section B.1, if we fix the true dynamics P^*, for each reward function $\mathcal{E}(s,a;\widehat{P}) \in \mathcal{B}$, we have a unique one-to-one corresponding function $\psi \in \Psi$, because the differential value function is the unique solution of the corresponding Bellman equation. Besides, since the reward function is bounded, we assume $\sup_{\psi\in\Psi} \|\psi\|_{\infty} \leq \Psi_{\max}$, such that Ψ is a bounded function class (Liao et al., 2020). So we can rewrite ② as

$$
\begin{aligned}
② &= \sup_{\mathcal{E}(s,a;\widehat{P})\in\mathcal{B}} \left| \mathbb{E}_{d_{\pi_b}^{P^*}}[\mathcal{E}(s,a;\widehat{P})] - \mathbb{E}_{d_{\pi}^{P^*}}[\mathcal{E}(s,a;\widehat{P})] \right| \\
&= \sup_{\psi\in\Psi} \left| \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}} \left[\psi(s,a) - \mathbb{E}_{s'\sim P^*(\cdot|s,a), a'\sim\pi(\cdot|s')} \left[\psi(s',a') \right] + \eta^{\pi} \right] - \eta^{\pi} \right| \\
&= \sup_{\psi\in\Psi} \left| \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}} \left[\psi(s,a) \right] - \mathbb{E}_{\substack{s'\sim P^*(\cdot|s,a), a'\sim\pi(\cdot|s') \\ (s,a)\sim d_{\pi_b}^{P^*}}} \left[\psi(s',a') \right] \right| \\
&= \sup_{\psi\in\Psi} \left| \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}} \left[\psi(s,a) \right] - \mathbb{E}_{s\sim d_{\pi_b}^{P^*}, a\sim\pi(\cdot|s)} \left[\psi(s,a) \right] \right| .
\end{aligned} \tag{22}
$$

The last equality comes from the fact that for offline datasets collected by sequential rollouts, the marginal distribution of s and s' are the same. In other words, if we randomly draw $s \sim d_{\pi}^{P^*}(\cdot)$, s will almost always be the "next state" of some other state in the dataset.

Combining analysis of ① and ②. Combing Eq. (21) and Eq. (22), we have the final results

$$
D_{\mathcal{G}}(d_{\pi}^{\widehat{P}}, d_{\pi}^{P^*}) \leq C \cdot \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}} \left[\sqrt{\tfrac{1}{2}\mathrm{KL}\left(P^*(s'|s,a) \| \widehat{P}(s'|s,a) \right)} \right] + \sup_{\psi\in\Psi} \left| \mathbb{E}_{(s,a)\sim d_{\pi_b}^{P^*}} \left[\psi(s,a) \right] - \mathbb{E}_{s\sim d_{\pi_b}^{P^*}, a\sim\pi(\cdot|s)} \left[\psi(s,a) \right] \right| ,
$$

where C is a constant, which may require tuning when implemented in practise. The proof of Theorem 3.2 is completed. □

Proposition B.2. *An upper bound of ①, the last term of Eq. (21), can be minimized via the maximum likelihood estimation for* $\widehat{P}$.

Proof. Based on Eq. (21), we would like to minimize the last equation *w.r.t.* $\widehat{P}$, and by Jensen's inequality, we have,

$$
\begin{aligned}
&\min_{\widehat{P}} F_{\max} \mathbb{E}_{(s,a)\sim d^{P^*}_{\pi_b}} \left[\sqrt{\frac{1}{2} \mathrm{KL}\left(P^*(\cdot \mid s,a) \| \widehat{P}(\cdot \mid s,a) \right)} \right] \\
\leq &\min_{\widehat{P}} F_{\max} \sqrt{\frac{1}{2} \mathbb{E}_{(s,a)\sim d^{P^*}_{\pi_b}} \left[\mathrm{KL}\left(P^*(\cdot \mid s,a) \| \widehat{P}(\cdot \mid s,a) \right) \right]} \\
= &\min_{\widehat{P}} F_{\max} \sqrt{\frac{1}{2} \mathbb{E}_{(s,a)\sim d^{P^*}_{\pi_b}} \left[\int P^*(s' \mid s,a) \left(\log P^*(s' \mid s,a) - \log \widehat{P}(s' \mid s,a) \right) ds' \right]} \\
= &\min_{\widehat{P}} F_{\max} \sqrt{\frac{1}{2} \left\{ \mathbb{E}_{(s,a)\sim d^{P^*}_{\pi_b}} \left[\int P^*(s' \mid s,a) \log P^*(s' \mid s,a) ds' \right] - \mathbb{E}_{(s,a)\sim d^{P^*}_{\pi_b}} \left[\int P^*(s' \mid s,a) \log \widehat{P}(s' \mid s,a) ds' \right] \right\}} \\
= &F_{\max} \sqrt{\frac{1}{2} \left\{ \mathbb{E}_{(s,a,s')\sim d^{P^*}_{\pi_b}} \left[\log P^*(s' \mid s,a) \right] + \min_{\widehat{P}} \left\{ -\mathbb{E}_{(s,a,s')\sim d^{P^*}_{\pi_b}} \left[\log \widehat{P}(s' \mid s,a) \right] \right\} \right\}},
\end{aligned}
$$

where both $F_{\max}$ and the first term inside $\sqrt{\cdot}$ are constants *w.r.t.* $\widehat{P}$, and

$$
\arg\min_{\widehat{P}} \left\{ -\mathbb{E}_{(s,a,s')\sim d^{P^*}_{\pi_b}} \left[\log \widehat{P}(s' \mid s,a) \right] \right\} = \arg\max_{\widehat{P}} \mathbb{E}_{(s,a,s')\sim d^{P^*}_{\pi_b}} \left[\log \widehat{P}(s' \mid s,a) \right]
$$

which is exactly the maximum-likelihood model-training objective Eq. (3). □

B.4. Proof of Theorem 3.4

Theorem (Restatement of Theorem 3.4). Combining Theorem 3.1, 3.2 and Proposition 3.3, and suppose the dynamics model $\widehat{P}$ is trained by the MLE objective (Eq. (3)), then for the same $\mathcal{G}$ and any policy π, up to some constants *w.r.t.* π and $\widehat{P}$,

$$
D_{\mathcal{G}}(d^{P^*}_{\pi_b}, d^{P^*}_{\pi}) \leq \mathcal{R}_{\mathcal{F}}(d^{P^*}_{\pi_b}, \pi, \widehat{P}) + \mathcal{R}_{\Psi}(d^{P^*}_{\pi_b}, \pi) + \mathcal{E}_{model},
$$

where $\mathcal{E}_{model}$ is the error associated with the maximum likelihood training, and is independent of the policy π.

Proof. From Eq. (4) we have

$$
D_{\mathcal{G}}(d^{P^*}_{\pi_b}, d^{P^*}_{\pi}) \leq D_{\mathcal{G}}(d^{P^*}_{\pi_b}, d^{\widehat{P}}_{\pi}) + D_{\mathcal{G}}(d^{\widehat{P}}_{\pi}, d^{P^*}_{\pi}) \,. \tag{23}
$$

From Theorem 3.1, we have

$$
D_{\mathcal{G}}(d^{P^*}_{\pi_b}, d^{\widehat{P}}_{\pi}) = \mathcal{R}_{\mathcal{F}}(d^{P^*}_{\pi_b}, \pi, \widehat{P}) := \sup_{f\in\mathcal{F}} \left| \mathbb{E}_{(s,a)\sim d^{P^*}_{\pi_b}} [f(s,a)] - \mathbb{E}_{\substack{s'\sim\widehat{P}(\cdot|s,a), a'\sim\pi(\cdot|s') \\ (s,a)\sim d^{P^*}_{\pi_b}}} [f(s',a')] \right| ,
$$

From Theorem 3.2 we have,

$$
D_{\mathcal{G}}(d^{\widehat{P}}_{\pi}, d^{P^*}_{\pi}) \leq C \cdot \mathbb{E}_{(s,a)\sim d^{P^*}_{\pi_b}} \left[\sqrt{\frac{1}{2} \mathrm{KL}\left(P^*(s'|s,a) \| \widehat{P}(s'|s,a) \right)} \right] + \mathcal{R}_{\Psi}(d^{P^*}_{\pi_b}, \pi) \,,
$$

where

$$
\mathcal{R}_{\Psi}(d^{P^*}_{\pi_b}, \pi) := \sup_{\psi\in\Psi} \left| \mathbb{E}_{(s,a)\sim d^{P^*}_{\pi_b}} [\psi(s,a)] - \mathbb{E}_{\substack{s\sim d^{P^*}_{\pi_b} \\ a\sim\pi(\cdot|s)}} [\psi(s,a)] \right| ,
$$

C is some constant depending on the function class $\mathcal{G}$ and the model error $\mathcal{E}_{model}$ is

$$
C \cdot \mathcal{E}_{model} = \mathbb{E}_{(s,a)\sim d^{P^*}_{\pi_b}} \left[\sqrt{\frac{1}{2} \mathrm{KL}\left(P^*(s'|s,a) \| \widehat{P}(s'|s,a) \right)} \right]
$$

From Proposition 3.3, we have an upper bound for $\mathcal{E}_{model}$,

$$\mathcal{E}_{model} \leq \sqrt{\frac{1}{2}\left\{\mathbb{E}_{(s,a,s')\sim d^{P^*}_{\pi_b}}\left[\log P^*(s' \mid s,a)\right] + \left\{-\mathbb{E}_{(s,a,s')\sim d^{P^*}_{\pi_b}}\left[\log \widehat{P}(s' \mid s,a)\right]\right\}\right\}}$$

where for a given true MDP $\mathcal{M}$, $\mathbb{E}_{(s,a,s')\sim d^{P^*}_{\pi_b}}\left[\log P^*(s' \mid s,a)\right]$ is a constant *w.r.t.* the policy π and the learned dynamics $\widehat{P}$ and the upper bound is essentially determined by

$$\mathcal{E}_{model} \leq O\left(\sqrt{-\mathbb{E}_{(s,a,s')\sim d^{P^*}_{\pi_b}}\left[\log \widehat{P}(s' \mid s,a)\right]}\right),$$

where minimizing this upper bound amounts to the maximum likelihood training for $\widehat{P}$ and does not involve the policy π.

Putting everything together, we get the desired inequality. □

C. Technical Details

C.1. Details for the Toy Experiment

We conduct the toy experiment following Gulrajani et al. (2017). We set up the total sample size as N_{total} as $N_{\text{total}} = 100000$, radius r as 4, and standard deviation σ for the Gaussian noise as 0.05. The dataset is randomly split into a training and testing set with the size of 5000. Details are outlined in Algorithm 2, where $\odot$ denotes element-wise multiplication.

Algorithm 2 Constructing the Circle Dataset

Input: Total sample size N_{total}, radius r.
Output: Generated dataset $\mathbb{D}_{\text{Circle}}$.
Sample angles $\boldsymbol{\theta} \sim \text{Unif}(0, 2\pi)$, sample noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2)$, $\boldsymbol{r} \leftarrow r + \boldsymbol{\epsilon}$.
Sample data point (x, y) as $(x, y) = (\boldsymbol{r} \odot \cos(\boldsymbol{\theta}), \boldsymbol{r} \odot \sin(\boldsymbol{\theta}))$.

To interpret the circle dataset as an offline reinforcement learning task, we define x as state and the corresponding y as action. Note that in the behavior cloning task, the information of reward, next state, and the episodic termination is unnecessary.

Hence, the generated dataset $\mathbb{D}_{\text{Circle}}$ can serve as an offline RL dataset which is applicable to train behavior cloning policies. the circle dataset at here is able to indicate multimodality in almost all the states, *i.e.*, the conditional distribution $p(y \mid x)$ is multi-modal in almost all x. Therefore, a good policy is expected to capture all the action modes.

To show the advantage of using a flexible implicit policy to approximate the behavior policy, we fit a conditional GAN (Mirza & Osindero, 2014) ("CGAN"), Gaussian generator conditional GAN ("G-CGAN"), and a deterministic-generator conditional GAN ("D-CGAN"). All these CGAN variants are fitted using the conditional-distribution matching approach similar to Wu et al. (2019). Following Gulrajani et al. (2017) and Goodfellow (2016), we use `DIM` = 128, `noise_dim` = 2, batch size 256, critic iteration 5, one-side label smoothing of 0.9, and total training iterations 100000. In Figure 1, we show the kernel-density-estimate plots on the test set for each method using random seed 1. We have experimented on several other random seeds and the results do not significantly alter. The detailed architecture of our generators and discriminator are shown as follows.

Implicit Generator

```
Linear(state_dim+noise_dim, DIM),
ReLU(),
Linear(DIM, DIM),
ReLU(),
Linear(DIM, DIM),
ReLU(),
Linear(DIM, action_dim)
```

Deterministic Generator

```
Linear(state_dim, DIM),
ReLU(),
Linear(DIM, DIM),
ReLU(),
Linear(DIM, DIM),
ReLU(),
Linear(DIM, action_dim)
```

Gaussian Generator

```
Linear(state_dim, DIM),
ReLU(),
Linear(DIM, DIM),
ReLU(),
Linear(DIM, DIM),
ReLU()
mean=Linear(DIM, action_dim)
logstd=Linear(DIM,action_dim).clamp(-5,5)
```

Discriminator

```
Linear(state_dim+action_dim, DIM),
ReLU(),
Linear(DIM, DIM),
ReLU(),
Linear(DIM, DIM),
ReLU(),
Linear(DIM, 1),
Sigmoid()
```

C.2. Details for the Main Algorithm

As in prior model-based offline RL algorithms, our main algorithm can be divided into two parts: model training (Appendix C.2.1) and actor-critic training (Appendix C.2.2).

C.2.1. Model Training

In this paper, we assume no prior knowledge about the reward function and the termination condition. Hence we use neural network to approximate transition dynamics, reward function and the termination condition. We follow prior work (Chua et al., 2018; Janner et al., 2019; Yu et al., 2020) to construct our model as an ensemble of Gaussian probabilistic networks. We use the same model architecture, ensemble size (=7), number of elite model (=5), train-test set split, optimizer setting, elite-model selection criterion, and sampling strategy as in Yu et al. (2020).

As in Yu et al. (2020), the input to our model is $\left((s,a)-\mu_{(s,a)}\right)/\sigma_{(s,a)}$, where $\mu_{(s,a)}$ and $\sigma_{(s,a)}$ are respectively mean and standard deviation of (s,a) in the offline dataset. We follow Kidambi et al. (2020) and Matsushima et al. (2021) 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 mean and standard deviation of $(r,\Delta s)$ in the offline dataset. As in prior model-based RL using Gaussian probabilistic ensemble (Chua et al., 2018; Janner et al., 2019; Yu et al., 2020; Cang et al., 2021; Yu et al., 2021), the output of our model is double-headed, respectively outputting the mean and log-standard-deviation of the normal distribution of the estimated output. As in Yu et al. (2020), the loss function is maximum likelihood, independently for each model in the ensemble.

We augment the mean-head of the output layer to further output the probability of the input satisfying the termination condition, which we model as a deterministic function of the input and is trained using weighted binary-cross entropy (WBCE) loss with class-weight inverse-proportional to the ratio between the number of terminated and non-terminated samples in the offline dataset. This WBCE loss is added to the losses of the probabilistic networks in both model training and elite-model selection. We use a cutoff value of 0.5 for deciding termination.

C.2.2. Actor-Critic Training

Our training process for actor and critic can be divided into the follow parts.

Generate synthetic data using dynamics model. Perform h-step rollouts using the learned model $\widehat{P}$ and the current policy $\pi_{\boldsymbol{\phi}}$ by branching from the offline dataset $\mathcal{D}_{\text{env}}$, and adding the generated data to a separate replay buffer $\mathcal{D}_{\text{model}}$, as in Janner et al. (2019) and Yu et al. (2020).

For computational efficiency, we perform model rollout every `rollout_generation_freq` iterations.

Critic Training. With hyperparameter $f\in[0,1]$, sample mini-batch $\mathcal{B}$ from $\mathcal{D}=f\cdot\mathcal{D}_{\text{env}}+(1-f)\cdot\mathcal{D}_{\text{model}}$.

To smooth the discrete $\delta_{s'}$ transition kernel recorded in the offline dataset, we apply the state-smoothing trick in Yang et al. (2022) in calculating the value of the next state. Specifically, for each $s'\in\mathcal{B}$ sample N_B $\hat{s}$ with noise standard deviation σ_B for state-smoothing via,

$$\hat{s}=s'+\epsilon,\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_B^2\boldsymbol{I}).$$

Sample N_a corresponding actions $\hat{a}\sim\pi_{\boldsymbol{\phi}'}(\cdot\,|\,\hat{s})$ for each $\hat{s}$. Calculate the estimated value of next state,

$$\widetilde{Q}\left(s,a\right)\triangleq\frac{1}{N_B\times N_a}\sum_{(\hat{s},\hat{a})}\left[c\min_{j=1,2}Q_{\boldsymbol{\theta}'_j}\left(\hat{s},\hat{a}\right)+(1-c)\max_{j=1,2}Q_{\boldsymbol{\theta}'_j}\left(\hat{s},\hat{a}\right)\right].$$

Calculate the critic-learning target defined as

$$\widetilde{Q}\left(s,a\right) \leftarrow r(s,a).\mathrm{clamp}(r_{\min}, r_{\max}) + \gamma \cdot \widetilde{Q}\left(s,a\right) \cdot \left(\left|\widetilde{Q}\left(s,a\right)\right| < 2000\right), \tag{24}$$

where $r_{\min} = r_0 - 3\sigma_r, r_{\max} = r_1 + 3\sigma_r$ for r_0, r_1, σ_r the minimum, maximum and standard deviation of rewards in the offline dataset, similar to the reward-clapping strategy in Hishinuma & Senda (2021); 2000 is some convenient number as an augmentation to the termination condition recorded in $\mathcal{B}$.

Minimize the critic loss with respect to $\boldsymbol{\theta}_j, j = 1, 2$, over $(s,a) \in \mathcal{B}$, with learning rate $\eta_{\boldsymbol{\theta}}$,

$$\arg\min_{\boldsymbol{\theta}_j} \tfrac{1}{|\mathcal{B}|} \textstyle\sum_{(s,a)\in\mathcal{B}} \mathrm{Huber}\left(Q_{\boldsymbol{\theta}_j}\left(s,a\right) - \widetilde{Q}(s,a)\right),$$

where $\mathrm{Huber}(\cdot)$ denote the Huber-loss function, with threshold conveniently chosen as 500.

In actual implemention, we choose to skip critic update for the i-th iteration when its critic loss $\mathcal{L}_{\mathrm{critic},i}$ exceed the critic loss threshold ℓ_{critic}.

We remark that the mentioned reward-clapping, augmented termination condition, Huber loss and critic-update skipping are mainly engineering tricks for improving training stablity in compensation for not knowing the true reward function and the true termination condition.

Discriminator Training. To form the "fake" sample $\mathcal{B}_{\mathrm{fake}}$ for training the discriminator, we first sample $|\mathcal{B}|$ states $s \sim \mathcal{D}$, and remove the terminal states. Second, we apply the state-smoothing trick in Yang et al. (2022) to s with noise standard deviation σ_J using $\epsilon \sim \mathcal{N}\left(\mathbf{0}, \sigma_J^2 \boldsymbol{I}\right)$, $s \leftarrow s + \epsilon$. Third, get a corresponding action using $a \sim \pi_{\boldsymbol{\phi}}\left(\cdot \mid s\right)$ for each s. Fourth, get a sample of the next state s' using the learned dynamics $\widehat{P}$ as $s' \sim \widehat{P}(\cdot \mid s, a)$, with terminal states removed. Fifth, sample one next action a' for each next state s' via $a' \sim \pi_{\boldsymbol{\phi}}\left(\cdot \mid s'\right)$. Finally, the "fake" sample $\mathcal{B}_{\mathrm{fake}}$ is defined as

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

The "true" sample $\mathcal{B}_{\mathrm{true}}$ is formed by sampling $|\mathcal{B}_{\mathrm{fake}}|$ state-action tuples from $\mathcal{D}_{\mathrm{env}}$.

Optimize the discriminator $D_{\boldsymbol{w}}$ to maximize

$$\frac{1}{|\mathcal{B}_{\mathrm{true}}|} \sum_{(s,a)\sim\mathcal{B}_{\mathrm{true}}} \left[\log D_{\boldsymbol{w}}(s,a)\right] + \frac{1}{|\mathcal{B}_{\mathrm{fake}}|} \sum_{(s,a)\sim\mathcal{B}_{\mathrm{fake}}} \left[\log\left(1 - D_{\boldsymbol{w}}(s,a)\right)\right],$$

with respect to $\boldsymbol{w}$ with learning rate $\eta_{\boldsymbol{w}}$.

We remark that the state-smoothing trick is applied here to smooth the discrete state-distribution manifested in $\mathcal{D}$ and to provide a better coverage of the state space, which is similar to a kernel density approximation (Wasserman, 2006) of $d\left(s,a\right) = f \cdot d_{\pi_b}^{P^*}(s,a) + (1-f) \cdot d_{\pi_{\boldsymbol{\phi}}}^{\widehat{P}}(s,a)$ using data points $s \in \mathcal{D}$ and with radial basis kernel of bandwidth σ_J. The terminal states are removed since no action choice occurs on them by definition.

Actor Training. We update the policy every k iterations. In training the actor, we first calculate the generator loss

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

using the "fake" sample $\mathcal{B}_{\mathrm{fake}}$ and discriminator $D_{\boldsymbol{w}}$.

Then we optimize policy with learning rate $\eta_{\boldsymbol{\phi}}$ for the target

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

where the regularization coefficient λ is defined similar to Fujimoto & Gu (2021) as $\lambda = \alpha / Q_{avg}$, with soft-updated Q_{avg} and $\alpha = 10$ across all datasets.

Soft updates. Similar to Fujimoto et al. (2018), we perform soft-update on the target-network parameter, the Q_{avg} and the critic loss threshold ℓ_{critic}. The first three is soft-updated after each iteration while the last is updated after each epoch.

$$\begin{aligned}
\boldsymbol{\phi}' &\leftarrow \beta\boldsymbol{\phi} + (1-\beta)\boldsymbol{\phi}', \\
\boldsymbol{\theta}'_j &\leftarrow \beta\boldsymbol{\theta}_j + (1-\beta)\boldsymbol{\theta}'_j, \forall\, j = 1, 2, \\
Q_{avg} &\leftarrow \beta \cdot \frac{1}{|\mathcal{B}|} \sum_{(s,a)\in\mathcal{B}} |Q(s,a)| + (1-\beta)\cdot Q_{avg}, \\
\ell_{\text{critic}} &\leftarrow 0.05 \cdot \left(\mu_{\mathcal{L}_{\text{critic},i}} + 3\sigma_{\mathcal{L}_{\text{critic},i}}\right) + (1-0.05)\cdot \ell_{\text{critic}}, \text{ for } \mathcal{L}_{\text{critic},i} \text{ in current epoch},
\end{aligned}$$

where 0.05 and 3 in the last equation are some conveniently chosen numbers.

Warm-start step. We follow Kumar et al. (2020) and Yue et al. (2020) to devote the first 40 epochs of training for warm start. In this phase, we sample $\mathcal{B} \sim \mathcal{D}_{\text{env}}$. We do not train the critic during the warm-start phase and optimize policy with respect to the generator loss $\mathcal{L}_g(\boldsymbol{\phi})$ only, with the same learning rate $\eta_{\boldsymbol{\phi}}$, *i.e.*, $\arg\min_{\boldsymbol{\phi}} \mathcal{L}_g(\boldsymbol{\phi})$.

C.3. Details for the Reinforcement Learning Experiment

Datasets. We use the continuous control tasks provided by the D4RL dataset (Fu et al., 2020) to conduct algorithmic evaluations. Due to limited computational resources, we follow the literature to select the "medium-expert," "medium-replay," and "medium" datasets for the Hopper, HalfCheetah, Walker2d tasks in the Gym-MuJoCo domain, which are commonly used benchmarks in prior work (Fujimoto et al., 2019; Kumar et al., 2019; Wu et al., 2019; Kumar et al., 2020). We follow the literature (Cang et al., 2021; Chen et al., 2021; Kostrikov et al., 2021a) to not test on the "random" and "expert" datasets as these datasets are less practical (Matsushima et al., 2021) and can be respectively solved by directly using standard off-policy RL algorithms (Agarwal et al., 2020) and the behavior cloning algorithms. We note that a comprehensive benchmarking of prior offline-RL algorithms on the "expert" datasets is currently unavailable in the literature, which is out of the scope of this paper. Apart from the Gym-MuJoCo, we also consider the Maze2D tasks[2] for their non-Markovian data-collecting policy and the Adroit tasks[3] (Rajeswaran et al., 2018) for their sparse reward-signal and high dimensionality.

Due to limited computational resources, we choose not to evaluate methods on the full set of Adriot datasets. From the benchmarking results in the D4RL whitepaper, most of other Adroit datasets are out of the scope of current offline RL algorithms. As an example, we evaluate our method and some baselines on the other two "door" datasets and present the results in Table 5. We hypothesize that obtaining good performance on these datasets requires specially-designed methods and a deeper investigation onto the quality of those datasets. Similar hypothesis may also be applied to other Adroit datasets, and thus we do not test on those datasets.

Table 5. Average scores of some baselines and our SDM-GAN on the other two "door" datasets.

Task Name	FisherBRC	TD3+BC	WMOPO	SDM-GAN
door-cloned	0 ± 0.1	-0.3 ± 0	-0.1 ± 0.1	0 ± 0.1
door-human	0.1 ± 0	-0.3 ± 0	-0.2 ± 0.2	0.2 ± 0.6

Evaluation Protocol. In all the experiments, we follow Fu et al. (2020) to use 3 random seeds for evaluation and to use the "v0" version of the datasets in the Gym-MuJoCo and Adroit domains. In our preliminary study, we find that the rollout results of some existing algorithms can be unstable across epochs, even towards the end of training. To reduce the instability during the evaluation, for our algorithm, we report the mean and standard deviation of the last five rollouts, conducted at the end of the last five training epochs, across three random seeds $\{0, 1, 2\}$. We train our algorithm for 1000 epochs, where each epoch consists of 1000 mini-batch stochastic gradient descent steps. For evaluation, We rollout our agent and the baselines for 10 episodes after each epoch of training.

Source Code. The design of our source code for the RL experiments is motivated by the codebase of Lu et al. (2021).

C.3.1. Details for the Implementation via GAN

In practice, the rollouts contained in the offline dataset have finite horizon, and thus in calculating the Bellman update target, special treatment is needed per appearance of the terminal states. We follow the standard treatment (Mnih et al., 2013;

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

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

Sutton & Barto, 2018) to modify the critic update target Equation (24) as

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

This modification removes the contribution of the next state when reaching the terminal state.

Following White (2016), we choose the noise distribution $p_z(z)$ as the multivariate standard normal distribution, where the dimension of z is defaulted as $\dim(z) = \texttt{min}(10, \texttt{state\_dim}//2)$. To sample from the implicit policy, for each state s, we first independently sample $z \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. We then concatenate s with z and feed the resulting $[s, z]$ into the deterministic policy network to generate stochastic actions. To sample from a small region around the next state s' (Appendix C.2.2), we keep the original s' and repeat it additionally $N_B - 1$ times. For each of the $N_B - 1$ replications, we add an independent Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma_B^2 \boldsymbol{I})$. The original s' and its $N_B - 1$ noisy replications are then fed into the implicit policy to sample the corresponding actions.

For fair comparison, we use the same network architecture as the implementation of the BCQ (Fujimoto et al., 2019), which will be presented in details in Appendix C.3.1. Due to limited computational resources, we leave a fine-tuning of the noise distribution $p_z(z)$, the network architectures, and the optimization hyperparameters for future work.

To approximately match the Jensen–Shannon divergence between the current and the behavior policies via GAN, a crucial step is to stably and effectively train the GAN structure. Together, we adopt the following technique from the literature.

- Following Goodfellow et al. (2014), we train $\pi_{\boldsymbol{\phi}}$ to maximize $\mathbb{E}_{(s,a)\in\mathcal{B}_{\text{fake}}}\left[\log\left(D_{\boldsymbol{w}}(s,a)\right)\right]$.
- Motivated by Radford et al. (2016), we use the LeakyReLU activation in both the generator and discriminator, under the default `negative_slope=0.01`.
- To stabilize the training, we follow Radford et al. (2016) to use a reduced momentum term $\beta_1 = 0.4$ in the Adam optimizer (Kingma & Ba, 2014) of the policy network and the discriminator network, with learning rate 2×10^{-4}.
- To avoid overfitting of the discriminator, we are motivated by Salimans et al. (2016) and Goodfellow (2016) to use one-sided label smoothing with soft and noisy labels. Specifically, the labels for the "true" sample $\mathcal{B}_{\text{true}}$ is replaced with a random number between 0.8 and 1.0. No label smoothing is applied for the "fake" sample $\mathcal{B}_{\text{fake}}$, and therefore their labels are all 0.
- The loss function for training the discriminator in GAN is the binary cross entropy between the labels and the outputs from the discriminator.
- Motivated by TD3 (Fujimoto et al., 2018) and GAN, we update $\pi_{\boldsymbol{\phi}}(\cdot \mid s)$ once per k updates of the critics and discriminator.

Table 6 shows the hyperparameters shared across all datasets for our empirical study. Note that several simplifications are made to minimize hyperparameter tuning, such as fixing $\eta_{\boldsymbol{\phi}} = \eta_{\boldsymbol{w}}$ as in Radford et al. (2016) and $\sigma_B = \sigma_J$. We also fix $N_a = 1$ to ease computation. We comment that many of these hyperparameters can be set based on literature, for example, we use $\eta_{\boldsymbol{\phi}} = \eta_{\boldsymbol{w}} = 2 \times 10^{-4}$ as in Radford et al. (2016), $\eta_{\boldsymbol{\theta}} = 3 \times 10^{-4}$ and $N_{\text{warm}} = 40$ as in Kumar et al. (2020), $c = 0.75$ as in Fujimoto et al. (2019), policy frequency $k = 2$ as in Fujimoto et al. (2018), $f = 0.5$ as in Yu et al. (2021) and model learning rate $\eta_{\widehat{P}} = 0.001$ as in Yu et al. (2020). Unless specified otherwise, the same hyperparameters, shared and non-shared, are used for both the main results and the ablation study.

Due to the diverse nature of the tested datasets, we follow the common practice in model-based offline RL to perform gentle hyperparameter tuning for each task. Specifically, we tune the dimension of the noise distribution $p_z(z)$ for controlling the stochasticity of the learned policy, and the rollout horizon h for mitigating model estimation error.

As shown in Zheng & Zhou (2021) and Zhang et al. (2021), a larger noise dimension, such as 50, can facilitate learning a more flexible distribution. Hence we use the default noise dimension for the tasks: halfcheetah-medium-expert, hopper-medium-expert, halfcheetah-medium, walker2d-medium, maze2d-umaze, pen-cloned, door-expert, pen-human; and noise dimension 50 for the tasks:walker2d-medium-expert, hopper-medium, halfcheetah-medium-replay, hopper-medium-replay, walker2d-medium-replay, maze2d-medium, maze2d-large, pen-expert.

Table 6. Shared hyperparameters for the GAN implementation.

Hyperparameter	Value
Optimizer	Adam (Kingma & Ba, 2014)
Learning rate $\eta_{\boldsymbol{\theta}}$	3×10^{-4}
Learning rate $\eta_{\boldsymbol{\phi}}$, $\eta_{\boldsymbol{w}}$	2×10^{-4}
Learning rate $\eta_{\widehat{P}}$	1×10^{-3}
Penalty coefficient α	10.0
Evaluation frequency (epoch length)	10^3
Training iterations	10^6
Batch size	512
Discount factor γ	0.99
Target network update rate β	0.005
Weighting for clipped double Q-learning c	0.75
Noise distribution $p_z(z)$	$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
Standard deviations for state smoothing σ_B, σ_J	3×10^{-4}
Number of smoothed states in Bellman backup N_B	50
Number of actions $\hat{\boldsymbol{a}}$ at each $\hat{\boldsymbol{s}}$ N_a	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
Random seeds	$\{0, 1, 2\}$

Similar to Janner et al. (2019) and Yu et al. (2020) we consider the rollout horizon $h \in \{1, 3, 5\}$. We use $h = 1$ for hopper-medium, walker2d-medium, hopper-medium-replay, pen-cloned; $h = 3$ for halfcheetah-medium-expert, halfcheetah-medium, halfcheetah-medium-replay, maze2d-umaze, maze2d-medium, maze2d-large, pen-expert, door-expert, pen-human; and $h = 5$ for hopper-medium-expert, walker2d-medium-expert, walker2d-medium-replay.

Below, we state the network architectures of the actor, critic, and the discriminator in the implementation via GAN. Note that we use two critic networks with the same architecture to perform clipped double Q-learning.

Actor

```
Linear(state_dim+noise_dim, 400)
LeakyReLU
Linear(400, 300)
LeakyReLU
Linear(300, action_dim)
max_action * tanh
```

Critic

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

Discriminator in GAN

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

Discriminator in WGAN (Ablation Study in Section 5.3)

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

C.3.2. Results of CQL

We note that the official CQL GitHub repository does not provide hyperparameter settings for the Maze2D and Adroit domain of tasks. For datasets in these two domains, we train CQL agents using five hyperparameter settings: the four recommended Gym-MuJoCo settings and the one recommended Ant-Maze setting. We then calculate the average normalized-return

over the random seeds $\{0, 1, 2\}$ for each hyperparameter settings and per-dataset select the best result from these five settings. We comment that this per-dataset tuning may give CQL some advantage on the Maze2D and Adroit domains, and is a compensation for the missing of recommended hyperparameters. For the Gym-MuJoCo domain, we follow the recommendation by Kumar et al. (2020).