ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230597

Model-Based Offline Policy Optimization with Adversarial Network

Junming Yang[a,c], **Xingguo Chen**[a,b], **Shengyuan Wang**[b] **and Bolei Zhang** [b;*]

[a]Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, China
[b]School of Computer Science, Nanjing University of Posts and Telecommunications, China
[c]School of Modern Posts, Nanjing University of Posts and Telecommunications, China

Abstract. Model-based offline reinforcement learning (RL), which builds a supervised transition model with logging dataset to avoid costly interactions with the online environment, has been a promising approach for offline policy optimization. As the discrepancy between the logging data and online environment may result in a distributional shift problem, many prior works have studied how to build robust transition models conservatively and estimate the model uncertainty accurately. However, the over-conservatism can limit the exploration of the agent, and the uncertainty estimates may be unreliable. In this work, we propose a novel **M**odel-based **O**ffline policy optimization framework with **A**dversarial **N**etwork (MOAN). The key idea is to use adversarial learning to build a transition model with better generalization, where an adversary is introduced to distinguish between in-distribution and out-of-distribution samples. Moreover, the adversary can naturally provide a quantification of the model's uncertainty with theoretical guarantees. Extensive experiments showed that our approach outperforms existing state-of-the-art baselines on widely studied offline RL benchmarks. It can also generate diverse in-distribution samples, and quantify the uncertainty more accurately.

1 Introduction

Over the last few years, reinforcement learning (RL) has achieved great success in a variety of simulation domains, e.g., Game of Go [31], Atari [32], MuJoCo [27]. However, vanilla RL methods have been struggling in many real-world applications, as they require frequent interactions with the environment, which are often costly or even dangerous. Offline RL provides an alternative approach that leverages logging datasets collected by another behavior policy for optimization without interacting with the online environment. With this innovative approach, RL can transcend its traditional boundaries and find application in an extensive array of real-world scenarios, becoming a practical and transformative solution in diverse fields like autonomous driving [8], recommendation [40], healthcare [5], etc.

A straightforward approach for offline RL is adopting off-policy methods over the logging dataset directly. However, previous researches have demonstrated that the performance can be quite poor due to the *distributional shift* problem [20, 19, 36], which occurs when the state-action distribution of the learning policy induced by the offline logging dataset differs from the distribution of the behavior policy. As a result, the off-policy algorithm cannot accurately evaluate the value function when selecting state-action pairs that are not covered in the offline dataset, as illustrated in Figure 1.

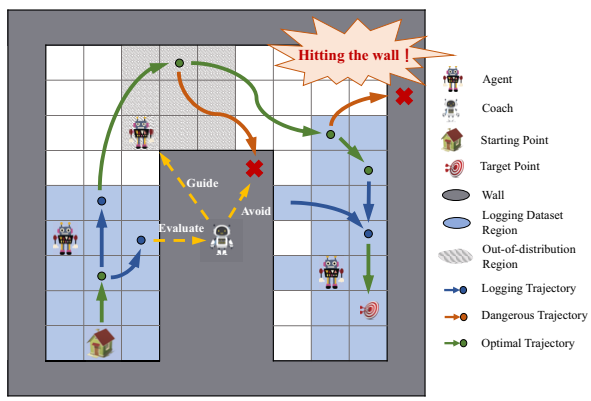


Figure 1. An agent optimizes its policy within fixed logging trajectories. Due to the limited scope of the dataset (blue), the agent may encounter new and unseen situations (white) during deployment, especially in out-of-distribution regions (grey slash). These situations can include obstacles such as hitting walls, leading to dangerous trajectories. The agent can have a higher success ratio to reach the target point if a coach can evaluate the agent's performance and guide the agent to optimize its trajectory.

To correct the distributional shift problem, the dominant paradigm is to introduce conservatism or uncertainty estimation in the offline RL algorithms. The related works can be broadly categorized as model-free offline RL and model-based offline RL. Model-free offline RL methods directly optimize the policy over the dataset and avoid the distributional shift problem by estimating conservative value functions [36, 19, 24]. There may be a sample inefficiency problem as this conservatism can also limit generalization beyond the offline dataset. Model-based approaches improve sample efficiency by first building a supervised transition model for interaction. The distributional shift problem can be mitigated by penalizing the reward with estimates of model uncertainty [39, 16, 34, 26]. However, existing works typically rely on heuristic uncertainty estimation, which is unreliable for complex datasets. To train an offline policy that can be deployed to the online environment, it is crucial to generate diverse in-distribution samples and quantify the model uncertainty accurately at the same time. As depicted in Figure 1, the agent has a higher success ratio to reach the target point if a coach can guide the agent to avoid the dangerous regions and evaluate the

* Corresponding Author. Email: bolei.zhang@njupt.edu.cn

Please check ArXiv or contact the authors for any appendices or supplementary material mentioned in the paper.

value function correctly.

In this paper, we propose a novel offline RL framework, named **M**odel-based **O**ffline policy optimization with **A**dversarial **N**etwork (MOAN). The basic idea is to introduce a two-player game in the model-based offline RL framework: one player is responsible for generating diverse transitions, and the adversary acts as a coach to accurately quantify the uncertainty. When the two players' policies converge, it is expected to generate diverse samples with accurate uncertainty quantification. In practice, the two-player game is implemented with a generative adversarial network. The discriminator learns to distinguish in-distribution data from out-of-distribution data, and the generator is trained to generate diverse samples that can confuse the discriminator. The output of the discriminator can be naturally implemented as uncertainty quantification for reward penalty. The lower bound of the policy performance in the real environment can be proved theoretically.

We conducted extensive experiments on the offline RL benchmark D4RL [10]. The results demonstrate that our proposed MOAN can achieve higher performance than state-of-the-art algorithms in most cases. In addition, our method has better generalization and can accurately quantify the model uncertainty.

The main contributions of this work are:

- We propose a novel framework MOAN, to model the offline RL as a zero-sum two player game. The two players are expected to generate diverse in-distribution samples.
- Based on the proposed framework, we devise an accurate penalty to reshape the reward, with performance guarantee in the real online environment.
- Extensive experiments are conducted. The results show that our method outperforms state-of-the-art works in most cases.

2 Related work

The problem of offline RL has been extensively studied in the last few years. Prior works can be broadly categorized as model-free offline RL and model-based offline RL.

Model-free offline RL Model-free offline RL methods optimize the policy directly with the logging dataset. To overcome the distributional shift problem, several approaches have been proposed. One way is to use importance sampling to give the in-distribution samples higher importance, as in BCQ [35], CQL [19], AWR [25] and VIP [22]. Alternatively, the target policy can be constrained to align with the behavior policy by adding a regularization term to the objective function that penalizes actions that deviate too much from the behavior policy [28, 36, 18, 2], thereby providing a form of safe policy improvement guarantees. Some studies provide compelling empirical evidence for the benefits of these approaches [12, 17, 30]. Based on these two directions, recent studies have also adopted ensemble models for a more robust value estimation [6, 24]. Adversarial training frameworks have also been used, where an adversary chooses the worst-case hypothesis (e.g., a value function or an MDP model) from a hypothesis class, and a policy player tries to maximize the adversarially chosen hypothesis [3, 37, 1]. In model-free methods, there may be a data inefficiency problem, as the policy can only be optimized with the logging dataset.

Model-based offline RL Compared to model-free methods that directly optimize the policy over the dataset, model-based offline RL approaches begin by building a supervised transition model to provide pseudo-exploration around the offline logging dataset [4]. The agent can then interact with the transition model to optimize policy costlessly. Therefore, model-based methods are potentially more sample efficient than model-free methods. To avoid extrapolation error induced by the distributional shift, MOPO [39] first proposed to penalize the reward with the uncertainty of transition model. They showed that policy performance could be guaranteed if the uncertainty is accurately assessed. However, the assessment is difficult since real-world state-action distributions are unavailable [38, 24]. Another line of work has tried to construct a pessimistic MDP to limit the occurrence of excessive uncertainty, such as MOReL [16], CPPO [33] and ROMI [34]. DICE [29] proposes to measure and penalty the extrapolation error by a density neural network. RAMBO [26] in a different way utilizes adversarial learning to restrict value estimation and force the policy conservatively. The difference is that our method explicitly introduces an adversary that can improve the generalization and penalize the reward at the same time. Despite the progress made by existing works, the transition models can easily overfit the offline logging dataset, and existing approaches tend to be over-conservative, making it challenging for the agent to efficiently utilize out-of-distribution regions for policy optimization. To address this, our method estimates uncertainty by introducing an adversary that learns the distance between the in-distribution and out-of-distribution samples. By doing so, the transition model can generate more diverse samples with accurate uncertainty estimation. This improves the agent's ability to utilize the out-of-distribution regions for policy optimization.

3 Preliminaries

Markov Decision Process In RL, the environment is often modeled as a Markov Decision Process (MDP), defined by a quintuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, \mathcal{T}, \gamma)$. At each step t, when observing a state $s_t \in \mathcal{S}$, an agent will take action $a_t \in \mathcal{A}$ according to the policy function $\pi(a_t|s_t)$. It will then receive an immediate reward $r(s_t, a_t) \in R$. The environment will transit to the next state $s_{t+1} \in \mathcal{S}$ according to the transition function $T(s_{t+1}, r_t|s_t, a_t) \in \mathcal{T}$. The agent aims to learn an optimal policy $\pi^*(\cdot)$ to maximize the γ-discounted cumulative reward: $\mathcal{J}(\pi, \mathcal{M}) = \mathbb{E}_{s\sim\mu_0}[\sum_{t=0}^{\infty} V_{\mathcal{M}}^{\pi}(s)]$, where μ_0 is the initial state distribution and $V_{\mathcal{M}}^{\pi}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s]$ is the expected value function from state s.

Let $\rho_{\mathcal{M}}^{\pi}(s, a) = (1-\gamma) \cdot \pi(a|s) \sum_{t=0}^{\infty} \gamma^t P_{\mathcal{M},t}^{\pi}(s)$ denote the normalized occupancy measure [15] for a policy π, where $P_{\mathcal{M},t}^{\pi}(s)$ represents the probability of state s visited by π under $\mathcal{M}$ at time step t. The discounted cumulative reward can be written as: $\mathcal{J}(\pi, \mathcal{M}) = \mathbb{E}_{(s,a)\sim\rho_{\mathcal{M}}^{\pi}}[r(s, a)]$. The normalized occupancy measure is utilized to compare the distribution of states and actions visited by different policies and dynamics.

Model-based Offline RL In offline RL, the agent cannot interact with the real environment, but only with a static logging dataset $\mathcal{D}_{env} = \{(s_i, a_i, s_i', r_i)\}_{i=1}^{N}$, collected by behavior policy π_D. To optimize the policy, model-based offline RL first builds a transition model $\hat{T}_\theta$ with supervised learning, which predicts the next state s' and reward r based on the current state s and action a. The transition model is often represented as a Gaussian distribution $\hat{T}_\theta(s', r|s, a) = \mathcal{N}(\mu_\theta(s, a), \Sigma_\theta(s, a))$, where $\mu_\theta(s, a)$ is the mean of the distribution and $\Sigma_\theta(s, a)$ is the variance. The policy can then be optimized directly by interacting with the supervised transition model. Therefore, model-based methods are more sample efficient especially when the data collection is arduous. However, the difference between the real-world and the supervised transition model will accumulate extrapolation error and lead to the distribution shift problem.

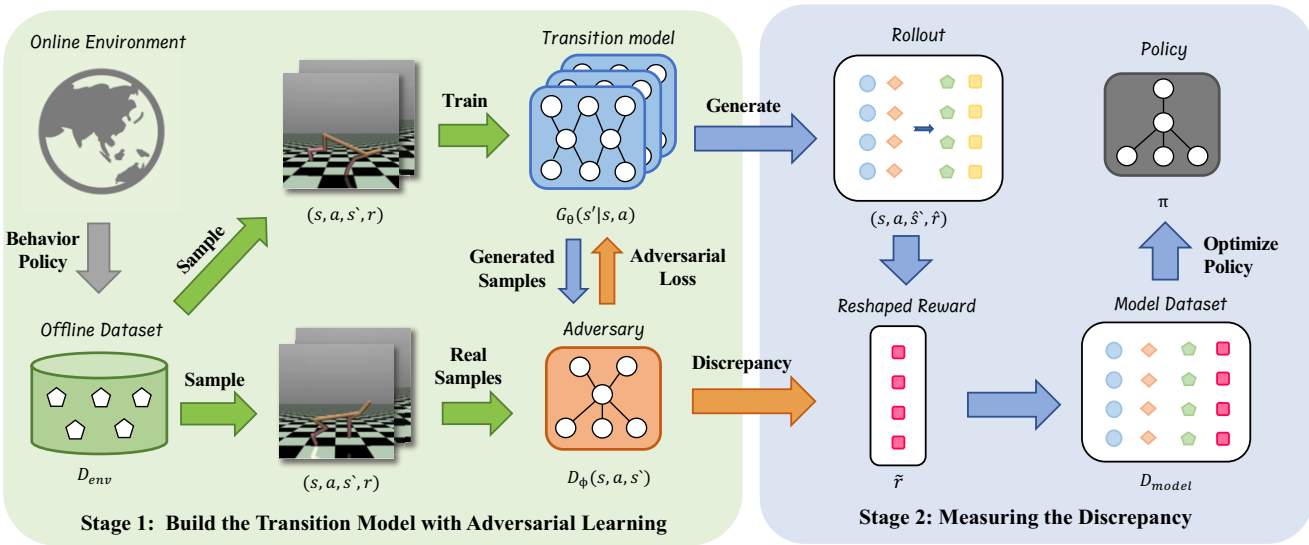


Figure 2. The framework of MOAN consists of two stages: In the first stage, the transition model is trained using adversarial learning to simulate the environment transitions. In the second stage, the agent samples rollouts from the transition model to optimize its policy. The distributional shift problem is avoided by correcting the prediction results using the adversary.

4 Methodology

In this section, we formally present our algorithm, named Model-based Offline Policy Optimization with Adversarial Network (MOAN). We begin by demonstrating how to build the transition model with adversarial learning. Next, we introduce a measurement of discrepancy between the real-world and the transition model to address the distributional shift in MOAN. The framework of our proposed method is illustrated in Figure 2.

4.1 *Build the Transition Model with Adversarial Learning*

In the first stage of MOAN, we employ adversarial learning to train a transition model that generates diverse in-distribution samples. Specifically, we model the process as a two-player game, where one player is responsible for generating new samples, and the other adversary learns to distinguish the generated samples from the real ones. When converged, the first player is expected to generate samples that are indistinguishable from the offline dataset. In practice, we use a generative model G to predict the next state, and a discriminator D as the adversary to evaluate the authenticity of the generated samples.

More concretely, the transition model is implemented as an ensemble of $\mathcal{N}$ Gaussian distributions, denoted as $G_\theta = \{\hat{T}_\theta^1, ..., \hat{T}_\theta^{\mathcal{N}}\}$, where θ is a parameter controlling the ensemble model. The transition model takes the current state-action pair (s, a) as input and predicts the next state reward pair $(\hat{s}', \hat{r})$. The data sample $(s, a, \hat{s}', \hat{r})$ is then stored to the model's buffer $\mathcal{D}_{model}$, which is used to optimize offline policy. In this paper, we propose to maximize the confidence of the generated samples to improve the model's generalization. This is achieved by training an additional discriminator to estimate the confidence of the generated samples. Specifically, the confidence is defined as the probability that the discriminator assigns to the generated samples being real. By maximizing the confidence, we encourage the generator to produce samples that are not only indistinguishable from the real ones, but also have high confidence. This approach differs from maximum likelihood training [39, 16], which may easily overfit in the in-distribution dataset region. In contrast, our approach encourages the generator to explore more diverse regions of the state-action space, which can improve the model's generalization.

During the optimization process, the transition model G_θ and discriminator model D_ϕ are updated jointly. And the discriminator aims to distinguish between the generator predictions and the true labels. Formally, the discriminator is trained to minimize the probability of the generated samples while maximizing the probability of the real samples:

$$\begin{aligned}\mathcal{L}_D(\phi) = &\mathbb{E}_{(s,a,s',r)\sim\rho_\mathcal{M}}[\log D_\phi(s, a, s', r)] \\ &+\mathbb{E}_{(\hat{s},\hat{a},\hat{s}',\hat{r})\sim G_\theta}[\log(1 - D_\phi(\hat{s}, \hat{a}, \hat{s}', \hat{r}))].\end{aligned} \tag{1}$$

For the generator, the loss function is a weighted sum of two parts: the first part is the negative log-likelihood loss of the transition model, and the second part is the adversarial loss:

$$\begin{aligned}\mathcal{L}_G(\theta) = &\mathbb{E}_{(s,a,s',r)\sim\rho_\mathcal{M}}[-\log G_\theta(s', r|s, a)] \\ &+\alpha\mathbb{E}_{(\hat{s},\hat{a},\hat{s}',\hat{r})\sim G_\theta}[\log(1 - D_\phi(\hat{s}, \hat{a}, \hat{s}', \hat{r}))],\end{aligned} \tag{2}$$

where α is a non-negative hyperparameter that balances the two parts. This hyperparameter can be tuned as a trade-off between the accuracy of the transition model and the diversity of the generated samples.

When the generator and discriminator have converged, the generator can produce data that closely resembles the in-distribution data. Moreover, this approach also maximizes the confidence from the discriminator, which encourages the generator to produce samples that are not only realistic but also diverse. By generating more diverse samples, the transition model can better explore the state-action space, especially in regions that are underrepresented in the in-distribution dataset. This leads to more effective policy optimization as the RL agent can utilize these diverse samples to learn better policies. Compared with previous model-based offline RL works, MOAN produces transition models with better generalization, which improves the agent's ability to exploit out-of-distribution regions of the state-action space.

4.2 Measure the Discrepancy

With the trained transition model, we can directly sample rollouts to optimize the policy of the RL agent. However, the discrepancy between the transition model and the real environment dynamics can lead to the failure of the policy. Measuring this discrepancy is difficult since the real environment dynamics can not be accessed. Nonetheless, the discriminator can output a probability value from $0-1$ to indicate whether the data is generated or collected from the real world to fix the distributional shift, and therefore provides a measure of the discrepancy. Formally, in the second stage of MOAN, we reshape the reward function as follows:

$$\widetilde{r}(s,a) := r(s,a) - \eta(\Sigma_\theta(s,a) + \sqrt{2D_{\phi,(\hat{s}',\hat{r})\sim G(s,a)}(s,a,\hat{s}',\hat{r})}), \quad (3)$$

where η is a hyperparameter to weight the reward penalty value. Σ_θ denotes the generator's variance and D_ϕ is the output of the discriminator. Σ_θ represents the uncertainty of the model's predictions, and can be regarded as the internal error of the transition model. D_ϕ represents the gap between the real-world model and the simulated transition model. The formulation indicates that we penalize the reward if the current state-action pairs have high uncertainty and low probability. Therefore, the policy of the agent can be constrained to the out-of-distribution region, to avoid the distributional shift problem.

The details of our algorithm are outlined in Algorithm 1. As presented, we first jointly update a discriminator D_ϕ to distinguish the real and generated data, and a transition model G_θ with adversary learning (Line 1-6). Next, we generate new samples from G_θ (Line 12) and reshape the reward with the discrepancy described above (Line 13). The generated samples are then added to the model buffer $\mathcal{D}_{model}$ (Line 14). Finally, the policy can be optimized on the joined set of $\mathcal{D}_{env} \cup \mathcal{D}_{model}$ using a soft actor-critic (SAC) [14] algorithm (Line 16). The overall goal of our method is to (1) update the transition model through adversarial learning to create a more accurate and comprehensive model and (2) optimize the policy using adversarial network penalty to correct the reward deviations, thereby reducing the distribution shift and improving the performance of offline RL.

Algorithm 1 Model-based Offline Policy Optimization with Adversarial Network (MOAN)

Input: Offline dataset $\mathcal{D}_{env}$, generator parameters θ, discriminator parameters ϕ, learning rate β and ω.
Output: Optimized policy π.

1: ▷ building the transition model
2: **while** θ not converged **do**
3: Sample (s,a,s',r) from $\mathcal{D}_{env}$.
4: Update generator $\theta \leftarrow \theta + \beta \cdot \nabla_\theta \mathcal{L}_G$.
5: Update discriminator $\phi \leftarrow \phi + \omega \cdot \nabla_\phi \mathcal{L}_D$.
6: **end while**
7: ▷ model-based policy optimization
8: **for** m epochs **do**
9: Sample an initial state s_0 from $\mathcal{D}_{env}$.
10: **for** h horizon **do**
11: Sample an action $a \sim \pi(s)$.
12: Sample $s', r \sim G_\theta(s,a)$.
13: Reshape the reward $\widetilde{r}$ according to Eq. (3).
14: Add samples $(s,a,s',\widetilde{r})$ to model buffer $\mathcal{D}_{model}$.
15: **end for**
16: Optimize π with SAC from $\mathcal{D}_{env} \cup \mathcal{D}_{model}$.
17: **end for**

5 Theoretical Analysis

In this section, we analyze the performance of MOAN theoretically. In particular, we show that the expected return of the policy π in the real environment $\mathcal{J}(\pi,\mathcal{M})$ is at least as high as the expected return of the policy π in the simulated environment $\mathcal{J}(\pi,\hat{\mathcal{M}})$ plus a bias item, where $\hat{\mathcal{M}}$ represents the simulated MDP in Section 4.1. The lower bound can be provided with the following theorem:

Theorem 1 *Let $T(\cdot|s,a)$ and $\hat{T}(\cdot|s,a)$ be the transition functions of $\mathcal{M}$ and $\hat{\mathcal{M}}$ with the same reward [39, 21] and bounded value function. The lower bound of real environment expected return can be denoted as:*

$$\begin{aligned}&\mathcal{J}(\pi,\mathcal{M}) \geq \mathcal{J}(\pi,\hat{\mathcal{M}}) \\ &-\gamma(\underbrace{\mathbb{E}_{(s,a)\sim\rho^{\pi_D}_{\mathcal{M}}}[d_{TV}(T(\cdot|s,a),\hat{T}(\cdot|s,a))]}_{\text{transition model error}} + \underbrace{\sqrt{2d_{JS}(\rho^{\pi_D}_{\mathcal{M}},\rho^{\pi}_{\hat{\mathcal{M}}})}}_{\text{distribution discrepancy}}),\end{aligned} \quad (4)$$

where $d_{TV}(\cdot)$ is the Total Variation (TV) distance [7] and $d_{JS}(\cdot)$ is the Jenson-Shanon (JS) divergence [11].

proof. According to the definition of γ-discounted return mentioned in preliminaries, the difference between the real environment $\mathcal{M}$ and simulated environment $\hat{\mathcal{M}}$ returns can be formulated as:

$$\begin{aligned}&\mathcal{J}(\pi,\mathcal{M}) - \mathcal{J}(\pi,\hat{\mathcal{M}}) = \\ &\gamma\mathbb{E}_{(s,a)\sim\rho^{\pi}_{\hat{\mathcal{M}}}}[\mathbb{E}_{s'\sim T(\cdot|s,a)}[V^{\pi}_{\mathcal{M}}(s')] - \mathbb{E}_{s'\sim\hat{T}(\cdot|s,a)}[V^{\pi}_{\mathcal{M}}(s')]].\end{aligned} \quad (5)$$

To simplify the notations, we can use the value estimation discrepancy between real transition $T(\cdot|s,a)$ and simulated transition $\hat{T}(\cdot|s,a)$ to substitute Eq.5. $Z^{\pi}_{\hat{\mathcal{M}}}(s,a) := \mathbb{E}_{s'\sim\hat{T}(\cdot|s,a)}[V^{\pi}_{\mathcal{M}}(s')] - \mathbb{E}_{s'\sim T(\cdot|s,a)}[V^{\pi}_{\mathcal{M}}(s')]$. Then We have a more concise expression:

$$\mathcal{J}(\pi,\mathcal{M}) - \mathcal{J}(\pi,\hat{\mathcal{M}}) = -\gamma\mathbb{E}_{(s,a)\sim\rho^{\pi}_{\hat{\mathcal{M}}}}[Z^{\pi}_{\hat{\mathcal{M}}}(s,a)]. \quad (6)$$

Since we aim to measure the state-action distribution deviation between two environments and the normalized occupancy distribution $\rho^{\pi}_{\mathcal{M}}$ from the real environment by a training policy is not directly accessible, we introduce $\rho^{\pi_D}_{\mathcal{M}}$ from a offline dataset $\mathcal{D}_{env}$, as Eq.7 denoted.

$$\begin{aligned}&\mathcal{J}(\pi,\mathcal{M}) - \mathcal{J}(\pi,\hat{\mathcal{M}}) \\ =& -\gamma\mathbb{E}_{(s,a)\sim\rho^{\pi}_{\hat{\mathcal{M}}}}[Z^{\pi}_{\hat{\mathcal{M}}}(s,a)] + \gamma\mathbb{E}_{(s,a)\sim\rho^{\pi_D}_{\mathcal{M}}}[Z^{\pi}_{\hat{\mathcal{M}}}(s,a)] \\ &-\gamma\mathbb{E}_{(s,a)\sim\rho^{\pi_D}_{\mathcal{M}}}[Z^{\pi}_{\hat{\mathcal{M}}}(s,a)].\end{aligned} \quad (7)$$

Furthermore, the value estimation discrepancy $Z^{\pi}_{\hat{\mathcal{M}}}(s,a)$ can be constrained by integral probability metric (IPM)[23] that the value functions are limited within the function collection $\mathcal{F} = \{f : \mathcal{S}\times\mathcal{A}\to\mathbb{R}\big|\|f\|_\infty \leq \delta\}$. And we have:

$$\begin{aligned}Z^{\pi}_{\hat{\mathcal{M}}}(s,a) &\leq \sup_{f\in\mathcal{F}}\left|\mathbb{E}_{s'\sim T(\cdot|s,a)}[f(s')] - \mathbb{E}_{s'\sim\hat{T}(\cdot|s,a)}[f(s')]\right| \\ &=: d_{f\in\mathcal{F}}(\hat{T}(\cdot|s,a), T(\cdot|s,a)),\end{aligned} \quad (8)$$

where $d_{f\in\mathcal{F}}(\cdot)$ is the IPM defined by the function class $\mathcal{F}$. By choosing different categories of measurement $\mathcal{F}$, IPM can reduce to many different well-known distance metrics between probability distributions, such as Total Variation (TV) distance [7] and maximum mean

Dataset type	Environment	MOAN	MOPO	MOReL	RAMBO	COMBO	CQL	IQL	TD3+BC	BC
random	halfcheetah	$\mathbf{39.2 \pm 5.2}$	35.4	25.6	**39.5**	**38.8**	19.6	-	11.0	2.1
random	hopper	31.2 ± 3.1	11.7	**53.6**	25.4	17.9	6.7	-	8.5	9.8
random	walker2d	18.2 ± 5.6	13.6	**37.3**	0.0	7.0	2.4	-	1.6	1.6
medium	halfcheetah	63.5 ± 4.7	42.3	42.1	**77.9**	54.2	49.0	47.4	48.3	36.1
medium	hopper	$\mathbf{101.3 \pm 4.7}$	28.0	95.4	87.0	**97.2**	66.6	66.3	59.3	29.0
medium	walker2d	$\mathbf{89.7 \pm 1.1}$	17.8	81.9	84.9	54.2	83.8	78.3	83.7	6.6
med-replay	halfcheetah	$\mathbf{65.3 \pm 4.4}$	53.1	40.2	**68.7**	55.1	47.1	44.2	44.6	38.4
med-replay	hopper	$\mathbf{103.9 \pm 3.3}$	67.5	93.6	**99.5**	89.5	97	94.7	60.9	11.8
med-replay	walker2d	78.1 ± 6.5	39.0	49.8	**89.2**	56.0	**88.2**	73.9	81.8	11.3
med-expert	halfcheetah	$\mathbf{95.4 \pm 2.3}$	63.3	53.3	**95.4**	**90.0**	**90.8**	86.7	**90.7**	35.8
med-expert	hopper	$\mathbf{108.3 \pm 8.7}$	23.7	**108.7**	88.2	**111.1**	**106.8**	91.5	98.0	**111.9**
med-expert	walker2d	83.4 ± 12.7	44.6	95.6	56.7	103.3	**109.4**	109.6	**110.1**	6.4

Table 1. Performance of MOAN and prior baselines on the D4RL tasks. Scores are the average returns value with 10 evaluations in the real environment, averaged over 5 random seeds. All values are normalized to lie between 0 and 100, where 0 corresponds to a random policy and 100 corresponds to an expert policy [10]. And baseline values were taken from their respective papers. Boldface denotes performance within 5% of the best performing algorithm.

discrepancy (MMD) [13].

$$\begin{aligned}
&\mathcal{J}(\pi, \mathcal{M}) - \mathcal{J}(\pi, \hat{\mathcal{M}}) \\
\geq & -(\gamma \mathbb{E}_{(s,a)\sim\rho^{\pi}_{\hat{\mathcal{M}}}}[Z^{\pi}_{\hat{\mathcal{M}}}(s,a)] - \mathbb{E}_{(s,a)\sim\rho^{\pi_D}_{\mathcal{M}}}[Z^{\pi}_{\hat{\mathcal{M}}}(s,a)]) \\
& - \gamma \mathbb{E}_{(s,a)\sim\rho^{\pi_D}_{\mathcal{M}}}[d_{f\in\mathcal{F}}(\hat{T}(\cdot|s,a), T(\cdot|s,a))] \\
= & -\gamma d_{TV}(\rho^{\pi_D}_{\mathcal{M}}, \rho^{\pi}_{\hat{\mathcal{M}}}) \\
& - \gamma \mathbb{E}_{(s,a)\sim\rho^{\pi_D}_{\mathcal{M}}}[d_{TV}(\hat{T}(\cdot|s,a), T(\cdot|s,a))],
\end{aligned} \tag{9}$$

where the distance $d_{f\in\mathcal{F}}(\cdot)$ is implemented as the TV distance in the second equation.

Since it is difficult to learn the state-action distribution $\rho^{\pi}_{\hat{\mathcal{M}}}$, we use Pinsker's Inequality [9] to transform the squared TV distance into a KL divergence equation. According to the Jenson-Shanon (JS) divergence definition, we can convert the the sum of the KL divergence into a GAN-like objective that the can be directly learned by a neural network:

$$\begin{aligned}
& d_{TV}(\rho^{\pi_D}_{\mathcal{M}}, \rho^{\pi}_{\hat{\mathcal{M}}}) \\
= & \sqrt{2(d^2_{TV}(\rho^{\pi_D}_{\mathcal{M}}, \frac{\rho^{\pi_D}_{\mathcal{M}} + \rho^{\pi}_{\hat{\mathcal{M}}}}{2}) + d^2_{TV}(\rho^{\pi}_{\hat{\mathcal{M}}}, \frac{\rho^{\pi_D}_{\mathcal{M}} + \rho^{\pi}_{\hat{\mathcal{M}}}}{2}))} \\
\leq & \sqrt{(d_{KL}(\rho^{\pi_D}_{\mathcal{M}}, \frac{\rho^{\pi_D}_{\mathcal{M}} + \rho^{\pi}_{\hat{\mathcal{M}}}}{2}) + d_{KL}(\rho^{\pi}_{\hat{\mathcal{M}}}, \frac{\rho^{\pi_D}_{\mathcal{M}} + \rho^{\pi}_{\hat{\mathcal{M}}}}{2}))} \\
= & \sqrt{2 d_{JS}(\rho^{\pi_D}_{\mathcal{M}}, \rho^{\pi}_{\hat{\mathcal{M}}})}.
\end{aligned} \tag{10}$$

In Theorem 1, the lower bound of the expected return in the real environment can be represented as a combination of 3 terms: the expected returns of π in $\hat{\mathcal{M}}$, the transition model error, and the distribution discrepancy. The second term represents the error between the real-world transition model T and the simulated transition model $\hat{T}$. The third term measures the discrepancy between the model rollout data distribution $\rho^{\pi}_{\hat{\mathcal{M}}}$ and the offline data distribution $\rho^{\pi_D}_{\mathcal{M}}$. Compared with prior methods [39, 16], we additionally introduce this discrepancy term to ensure that the model rollout distribution is closer to the true distribution. Our bound is tighter than MOPO. As presented in Eq. 4, MOPO did not consider the distribution discrepancy term. Considering this term can potentially reduce the risks of exploring out-of-distribution data.

Essentially, the theorem indicates that if the transition model is accurate and the state-action distributions are similar (as shown in Sec 4.1), and the distribution discrepancy can be precisely measured by the adversary (as shown in Sec 4.2), then the resulting performance of the policy in the real environment will be guaranteed.

6 Experiments

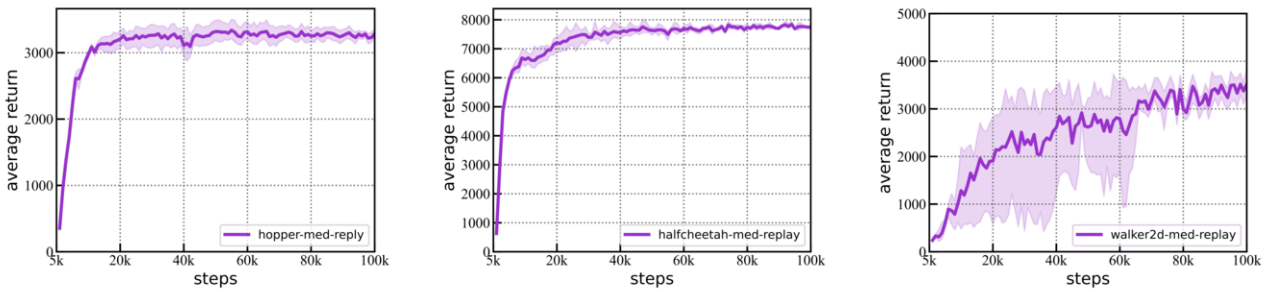
In this section, we evaluate the effectiveness of the proposed MOAN to answer the following four questions[1]: (1) How is the performance of MOAN compared to state-of-the-art offline RL algorithms on typical benchmark tasks? (2) Can MOAN generate diverse samples that closely mimic the online environment? (3) Can MOAN accurately measure the discrepancy between the real world and the simulated transition model? (4) What is the performance of offline policy optimization with the different hyperparameters for adversarial generation and discrepancy measuring?

6.1 Experiment Setting

Our experiments are conducted in three MuJoCo simulated environments (HalfCheetah, Hopper and Walker2d) using a standard offline RL benchmark D4RL [10]. Each environment has four offline logging datasets, collected by different one or mixed behavior policies: *random, medium, medium-replay, and medium-expert*. The *random* dataset is created by implementing a randomly generated policy for 10^6 steps. The *medium* dataset is generated by using a soft actor-critic policy that has been trained to achieve approximately 1/3 of the performance level of an expert. The *medium-replay* dataset is made up of all the samples in the replay buffer during the training process until the policy reaches the medium level of performance. The *medium-expert* dataset is created by combining an equal number of expert-level samples and medium-level samples. All results are evaluated 1000 episode (1M steps) runs per seed in real environment, and averaged based on five random seeds. More details on the experimental setup, different environmental hyperparameter configurations, generator and discriminator network structures, and codes are attached to Appendix A.

To thoroughly evaluate the performance of our approach on the D4RL dataset, we conduct a series of experiments and compare the results to several offline RL baselines. These baselines include Behavior Cloning method (BC) and model-free methods Constrain Q-learning (CQL) [19], Implicit Q-learning (IQL) [17], and Minimalist Offline RL Algorithm (TD3+BC) [12], as well as Model-based offline policy optimization (MOPO) [39], model-based methods Model-based offline reinforcement learning (MOReL) [16], Robust Adversarial Model-Based Offline Reinforcement Learning(RAMBO) [26] and Conservative offline model-based policy optimization (COMBO) [38].

[1] Code and appendix is available at https://github.com/junming-yang/MOAN



(a) Hopper Medium-replay Env. (b) Halfcheetach Medium-replay Env. (c) Walker2d Medium-replay Env.

Figure 3. D4RL Benchmark Result. (a) The learning curves of the three algorithms MOAN on the Hopper environment *medium-replay* setting. (b) The learning curves of the three algorithms MOAN on the Halfcheetah environment *medium-replay* setting. (c) The learning curves of the three algorithms MOAN on the Walker2d environment *medium-replay* setting. The curve represents the mean score and the shaded part represents the statistical standard deviation under multiple random seed experiments.

6.2 Evaluation on D4RL Benchmark

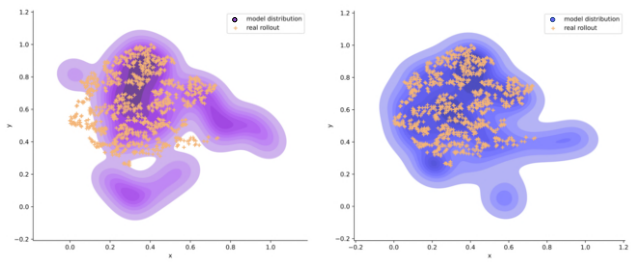
First, we evaluate the expected returns of different methods on a set of D4RL benchmarks with several settings. The evaluation results are presented in Table 1. As presented, MOAN achieves the best performance on 7 tasks across all 12 dataset settings, and achieves comparable results (the second best) in other 3 out of 5 datasets. In particular, MOAN has superior performance in the Hopper environment. This is because Hopper has lower dimensions of states and actions, which may lead to over-conservative in other algorithms.

Compared with the model-based algorithms, MOAN can outperform MOPO across all settings. MOReL has better performance in two *random* settings. The reason is that MOReL adopts a more conservative policy, which reduces the occurrence of dangerous situations in the random datasets. RAMBO performs well on *med-replay* datasets because its transition model also has a certain generalization. However, due to its inability to accurately correct the extrapolation error, RAMBO adopted a conservative approach and had a mediocre effect on other datasets. Compared with COMBO, which is the state-of-the-art model-based method, MOAN still has higher performance in 9 out of the 12 settings. When compared to model-free methods such as CQL, IQL, and TD3+BC, MOAN has better performance in almost all settings, particularly in the *random* and *medium* datasets. This is because MOAN can utilize offline datasets more efficiently by building the transition model. Overall, our evaluation results provide a clear indication of the strengths and effectiveness of the MOAN method in offline RL.

We also compare the average return of MOAN, MOPO and CQL in the Hopper environment during the training phase. MOPO has a similar model structure to MOAN, and CQL is the state-of-the-art algorithm in this setting. The learning progresses are presented in Figure 3. The curves are the mean value of the average returns from multiple runs, and the shaded areas represent the standard derivations. Obviously, MOAN can achieve the best performance across the training phase. Moreover, the results show that MOAN can converge very fast in only about 20k steps.

6.3 Performance of the Transition Model

To evaluate the effectiveness of the transition model in our MOAN algorithm, we compared its performance with that of MOPO, which has a similar transition model structure. For each task, we trained the transition models independently until convergence, and then randomly generated 10,000 samples from the real environment. These samples were fed into the different transition models to predict the next states.



(a) MOPO algorithm. (b) MOAN algorithm.

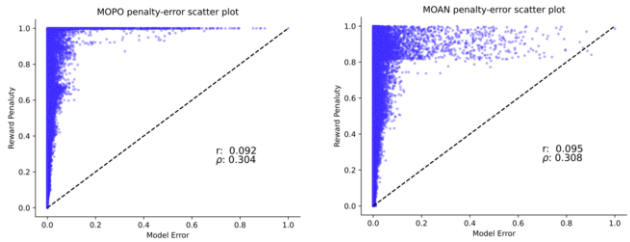
Figure 4. Visualization of the two-dimensional t-SNE state distribution learned by transition model from hopper environment. Colors correspond to different datasets. Purple: MOPO model state distribution, Blue: MOAN model state distribution, Yellow crosses: real-world rollout samples. The dark colors represent areas where model samples distribution is concentrated.

We visualized the state distributions using t-SNE and presented the results in Figure 4. The yellow crosses represent the rollouts from the real environment, while the model states generated by MOPO and MOAN are colored purple and red, respectively. To facilitate comparison, we drew the results of model prediction as distribution maps, where the darker parts represent more concentrated regions of the results.

Our results show that the real rollout points in MOPO are more likely to be outside of the model distribution, while they are always consistent with the model distribution in MOAN. This indicates that MOAN not only generates more similar states that are within the real dataset but also has a better generalization to cover the distribution. These findings suggest that the transition model of MOAN has better generalization and accuracy compared to that of the MOPO algorithm.

6.4 Performance of the Discrepancy Measure

We conducted experiments to investigate whether our proposed MOAN algorithm can more accurately quantify the discrepancy between an offline dataset and the real environment. Specifically, we



(a) MOPO algorithm. (b) MOAN algorithm.

Figure 5. Visualization of the correlation between the ground-truth MSE and reward penalty values. The reward penalty values are predicted from action-state pairs that are randomly selected from the dataset. The left figure shows the results of the MOPO method, while the right figure shows the results of the MOAN method.

compared the results in the Hopper environment using the *medium-replay* dataset. The primary objective of this experiment was to learn more about how accurately both algorithms could quantify discrepancies in the real-world environment, particularly the accuracy of the reward penalties given by each algorithm. We visualized the results in Figure 5, where the x-axis represents the mean squared error (MSE) of the ensemble model $\|\hat{T}(\cdot|s,a) - T(\cdot|s,a)\|_2$, and the y-axis represents the normalized reward penalty value. We normalized the reward penalty and model errors on a per-dimension basis to the [0, 1] interval so that the scattered points should lie along the diagonal line $y = x$ in an ideal situation where the discriminator perfectly captures the simulated dynamics error.

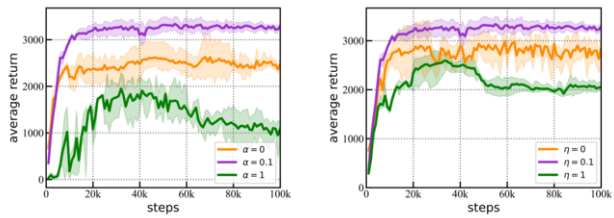
As shown in Figure 5, the results of the experiment prove effectiveness of MOAN, with the scattered points of MOAN located much closer to the diagonal line. In contrast, MOPO's scattered points were mainly located higher on the diagram, indicating an emphasis on over-conservatism and a higher reward penalty value. This result demonstrates that MOAN has greater accuracy in quantifying discrepancies in the real environment between the offline dataset.

Both MOAN and MOPO use penalty functions to encourage the online policy to explore the environment. However, MOPO uses a handcrafted penalty function, whereas MOAN designs a neural network-based penalty function that can learn to balance and optimize exploration and exploitation dynamically. MOAN uses the discriminator as the reward penalty function, which is also used in training the generator to eliminate the shift between the offline dataset and the real environment.

6.5 *Ablation Study*

In the last part of our study, two ablation studies were conducted to investigate the contributions of adversarial learning and discrepancy penalty in MOAN in an online environment. The performance of the algorithm was evaluated with different values of adversarial learning hyperparameter α and discrepancy measurement hyperparameter η.

Figure 6(a) shows the learning progress of our algorithm in the Hopper environment as a function of the hyperparameter α. As shown, when α is too large, the generator (transition model) is heavily influenced by the noise produced by the discriminator, which can cause instability in the training process. On the other hand, when α is too small, the generator may not learn to generate diverse samples, resulting in plain performance. Therefore, it is important to choose an appropriate value for α to ensure stable and effective training of MOAN.



(a) Ablation on α. (b) Ablation on η.

Figure 6. (a) Ablation study on adversarial learning hyperparameter α. We set the hyperparameter to three different values, {0, 0.1, 1}, to examine its effect on the performance of the algorithm MOAN. (b) Ablation study on discrepancy measurement hyperparameter η.

In addition, Figure 6(b) demonstrates how MOAN's performance varies with different values of the hyperparameter η, which balances the importance of the reward value from the transition model and the deviation from the discriminator. When η is too large, the reward may be too conservative for agent exploration in policy optimization. Therefore, it is important to choose an appropriate value for η to balance exploration and exploitation in policy optimization.

In terms of sample complexity, the training of the discriminator has low sample complexity, since it only needs to distinguish between real and generated data. In our experiments, the discriminator consists of only two layers of networks with a simple structure, which makes it easy to train. Regarding the stability of convergence, the input state and action are usually low-dimensional vectors in the task of offline RL, making it easier to achieve stable and efficient training compared to high-dimensional image samples. As Figure 3 shows, MOAN is stable under multiple random seeds. However, the value of hyperparameter α still affects the stability of the training process. Choosing an appropriate value for α is critical to achieving stable and effective training of MOAN.

Overall, selecting appropriate hyperparameters is crucial for MOAN's stability and performance in online environments. The results of this study show that incorporating adversarial learning and discrepancy penalty into MOAN can result in improved performance over prior algorithms. These findings have implications for the development of future reinforcement learning algorithms, particularly those designed for offline environments.

7 Conclusion

In this paper, we proposed an offline RL framework called MOAN, which introduces a two-player game to improve the generalization capability of the transition model and mitigate the negative effects of potentially problematic rollouts during offline reinforcement learning. Thus, MOAN can effectively balance the exploitation of logging dataset and exploration in out-of-distribution regions. Our experiments on several benchmark tasks demonstrated that our method achieved the highest performance in most settings. However, a major limitation of MOAN is the increased computational complexity required for the training of the discriminator network. In the future, we plan to develop more accurate and efficient strategies, such as improving the architecture of the discriminator network, to further enhance the performance and computational efficiency of MOAN. Additionally, we also plan to investigate the integration of various model-free approaches into MOAN to increase performance on certain datasets, such as the Walker2d *medium-expert*.

Acknowledgements

This paper is partially supported by National Natural Science Foundation of China under Grant No.62202238 and No.62276142; Natural Science Foundation of Jiangsu Province under Grant No.BK20200752.

References

[1] Xueying Bai, Jian Guan, and Hongning Wang, 'A model-based reinforcement learning with adversarial training for online recommendation', in *Advances in Neural Information Processing Systems*, volume 32, (2019).

[2] Xingguo Chen, Xingzhou Ma, Yang Li, Guang Yang, Shangdong Yang, and Yang Gao, 'Modified retrace for off-policy temporal difference learning', in *Uncertainty in Artificial Intelligence*, pp. 303–312. PMLR, (2023).

[3] Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal, 'Adversarially trained actor critic for offline reinforcement learning', in *International Conference on Machine Learning*, pp. 3852–3878. PMLR, (2022).

[4] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine, 'Deep reinforcement learning in a handful of trials using probabilistic dynamics models', in *Advances in Neural Information Processing Systems*, volume 31, (2018).

[5] Antonio Coronato, Muddasar Naeem, Giuseppe De Pietro, and Giovanni Paragliola, 'Reinforcement learning for intelligent healthcare applications: A survey', *Artificial Intelligence in Medicine*, **109**, 101964, (2020).

[6] Esther Derman, Daniel Mankowitz, Timothy Mann, and Shie Mannor, 'A bayesian approach to robust reinforcement learning', in *Uncertainty in Artificial Intelligence*, pp. 648–658. PMLR, (2020).

[7] Luc Devroye, Abbas Mehrabian, and Tommy Reddad, 'The total variation distance between high-dimensional gaussians', *arXiv preprint arXiv:1810.08693*, **6**, (2018).

[8] Jingliang Duan, Shengbo Eben Li, Yang Guan, Qi Sun, and Bo Cheng, 'Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data', *IET Intelligent Transport Systems*, **14**(5), 297–305, (2020).

[9] Alexei A Fedotov, Peter Harremoës, and Flemming Topsoe, 'Refinements of pinsker's inequality', *IEEE Transactions on Information Theory*, **49**(6), 1491–1498, (2003).

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

[11] Bent Fuglede and Flemming Topsoe, 'Jensen-shannon divergence and hilbert space embedding', in *International Symposium on Information Theory*, p. 31. IEEE, (2004).

[12] Scott Fujimoto and Shixiang Shane Gu, 'A minimalist approach to offline reinforcement learning', in *Advances in Neural Information Processing Systems*, volume 34, pp. 20132–20145, (2021).

[13] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola, 'A kernel two-sample test', *The Journal of Machine Learning Research*, **13**(1), 723–773, (2012).

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine, 'Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor', in *International Conference on Machine Learning*, pp. 1861–1870. PMLR, (2018).

[15] Jonathan Ho and Stefano Ermon, 'Generative adversarial imitation learning', in *Advances in Neural Information Processing Systems*, volume 29, (2016).

[16] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims, 'Morel: Model-based offline reinforcement learning', in *Advances in Neural Information Processing Systems*, volume 33, pp. 21810–21823, (2020).

[17] Ilya Kostrikov, Ashvin Nair, and Sergey Levine, 'Offline reinforcement learning with implicit q-learning', *arXiv preprint arXiv:2110.06169*, (2021).

[18] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine, 'Stabilizing off-policy q-learning via bootstrapping error reduction', in *Advances in Neural Information Processing Systems*, volume 32, (2019).

[19] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine, 'Conservative q-learning for offline reinforcement learning', *Advances in Neural Information Processing Systems*, **33**, 1179–1191, (2020).

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

[21] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma, 'Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees', *arXiv preprint arXiv:1807.03858*, (2018).

[22] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang, 'Vip: Towards universal visual reward and representation via value-implicit pre-training', *arXiv preprint arXiv:2210.00030*, (2022).

[23] Alfred Müller, 'Integral probability metrics and their generating classes of functions', *Advances in Applied Probability*, **29**(2), 429–443, (1997).

[24] Kishan Panaganti, Zaiyan Xu, Dileep Kalathil, and Mohammad Ghavamzadeh, 'Robust reinforcement learning using offline data', *arXiv preprint arXiv:2208.05129*, (2022).

[25] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine, 'Advantage-weighted regression: Simple and scalable off-policy reinforcement learning', *arXiv preprint arXiv:1910.00177*, (2019).

[26] Marc Rigter, Bruno Lacerda, and Nick Hawes, 'Rambo-rl: Robust adversarial model-based offline reinforcement learning', *arXiv preprint arXiv:2204.12581*, (2022).

[27] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever, 'Evolution strategies as a scalable alternative to reinforcement learning', *arXiv preprint arXiv:1703.03864*, (2017).

[28] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz, 'Trust region policy optimization', in *International Conference on Machine Learning*, pp. 1889–1897. PMLR, (2015).

[29] Jian Shen, Mingcheng Chen, Zhicheng Zhang, Zhengyu Yang, Weinan Zhang, and Yong Yu, 'Model-based offline policy optimization with distribution correcting regularization', in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 174–189. Springer, (2021).

[30] Laixi Shi, Gen Li, Yuting Wei, Yuxin Chen, and Yuejie Chi, 'Pessimistic q-learning for offline reinforcement learning: Towards optimal sample complexity', in *International Conference on Machine Learning*, pp. 19967–20025. PMLR, (2022).

[31] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al., 'A general reinforcement learning algorithm that masters chess, shogi, and go through self-play', *Science*, **362**(6419), 1140–1144, (2018).

[32] Ruben Rodriguez Torrado, Philip Bontrager, Julian Togelius, Jialin Liu, and Diego Perez-Liebana, 'Deep reinforcement learning for general video game ai', in *2018 IEEE Conference on Computational Intelligence and Games*, pp. 1–8, (2018).

[33] Masatoshi Uehara and Wen Sun, 'Pessimistic model-based offline rl: Pac bounds and posterior sampling under partial coverage', *arXiv preprint arXiv:2107.06226*, (2021).

[34] Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang, 'Offline reinforcement learning with reverse model-based imagination', in *Advances in Neural Information Processing Systems*, volume 34, pp. 29420–29432, (2021).

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

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

[37] Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal, 'Bellman-consistent pessimism for offline reinforcement learning', in *Advances in Neural Information Processing Systems*, volume 34, pp. 6683–6694, (2021).

[38] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn, 'Combo: Conservative offline model-based policy optimization', in *Advances in Neural Information Processing Systems*, volume 34, pp. 28954–28967, (2021).

[39] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma, 'Mopo: Model-based offline policy optimization', in *Advances in Neural Information Processing Systems*, volume 33, pp. 14129–14142, (2020).

[40] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li, 'Drn: A deep reinforcement learning framework for news recommendation', in *World Wide Web Conference*, pp. 167–176, (2018).