VIPO: Value Function Inconsistency Penalized Offline Reinforcement Learning

Xuyang Chen, Keyu Yan, Guojian Wang, Lin Zhao

Department of Electrical & Computer Engineering
National University of Singapore

May 30, 2026

Reinforcement Learning. Reinforcement Learning (RL) problems are commonly formulated within the framework of a Markov Decision Process (MDP), defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \rho_0, P, \gamma)$. Here, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ represents the action space, $r(s, a) : \mathcal{S} \times \mathcal{A} \to [-r_{\max}, r_{\max}]$ is a bounded reward function, $\rho_0(s)$ specifies the initial state distribution, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ defines the transition kernel, and $\gamma \in [0, 1)$ is the discount factor. Given a policy $\pi(\cdot \mid s)$, its performance can be evaluated by the expected cumulative long-term reward, defined as

$$J(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a_t \sim \pi, s_{t+1} \sim P} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \tag{1}$$

The goal of RL is to learn a policy $\pi(\cdot \mid s)$ that maximizes the expected cumulative long-term reward, i.e., $\pi^* = \arg\max_\pi J(\pi)$.

Offline Reinforcement Learning. As shown in

$$J(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a_t \sim \pi, s_{t+1} \sim P} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right],$$

the classical RL framework requires online interactions with the environment P during training. In contrast, the offline RL learns only from a fixed offline dataset $D = \{(s_i, a_i, r_i, s_i')\}_{i=1}^{N}$ collected by the behavior policy $\mu(\cdot \mid s)$, where s, a, r, and s' denote the state, action, reward, and next state, respectively. That is, it aims to find the *best possible* policy solely from $\mathcal{D}$ without additional interactions with the environment.

Model-based Offline RL Algorithms

Model-based offline RL aims to derive the optimal policy by utilizing a learned dynamics model. These methods define a pessimistic MDP $\widehat{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, \hat{r}, \rho_0, \hat{P}, \gamma \rangle$, which shares the same state and action spaces as the original MDP but employs the learned transition dynamics $\hat{P}(s' \mid s, a)$ and reward function $\hat{r}(s, a)$. $\hat{P}$ and $\hat{r}$ are typically estimated through supervised regression on the dataset $\mathcal{D}$, with additional incorporation of uncertainty penalties to discourage exploration of OOD actions. These pessimistic models then serve as proxies for the real environment, enabling the simulation of transitions and subsequent use for planning.

The success of model-based RL algorithms hinges on the ability to learn an accurate model. Once the model is learned, various approaches, such as model predictive control (MPC), dynamic programming, or model-based policy optimization, can be employed to recover the optimal policy by solving the pessimistic MDP $\widehat{\mathcal{M}}$.

Value Function Learned from the Dataset. We first learn an approximated value function of the behavior policy directly from the dataset. In the MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \rho_0, P, \gamma)$, under the behavior policy $\mu(\cdot \mid s)$, the Bellman backup for obtaining the corresponding value function V is defined as

$$\mathcal{T}^{\mu} V(s) := \mathbb{E}_{a \sim \mu(\cdot \mid s), s' \sim P(\cdot \mid s, a)} \big[r(s, a) + \gamma V(s') \big].$$

Therefore, from the offline dataset $\mathcal{D}$, we can define the following empirical Bellman operator:

$$\widehat{\mathcal{T}}_d^{\mu} V(s) := \begin{cases} \dfrac{1}{|\mathcal{D}_s|} \displaystyle\sum_{(s,a,r,s') \in \mathcal{D}_s} \big[r + \gamma V(s') \big], & \text{if } |\mathcal{D}_s| > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where $\mathcal{D}_s \subseteq \mathcal{D}$ is the set of transitions in the dataset $\mathcal{D}$ that begin in state s, $|\mathcal{D}_s|$ is the number of such transitions, and γ is the discount factor.

We then show that the empirical Bellman operator has a unique fixed point.

Proposition 1

The empirical Bellman operator $\widehat{\mathcal{T}}_d^{\mu}$ has a unique fixed point $V_d^{\mu}(s)$ such that

$$\widehat{\mathcal{T}}_d^{\mu} V_d^{\mu}(s) = V_d^{\mu}(s). \tag{3}$$

From Proposition 1, we can learn a value function $V_d^{\mu}(s)$ from the offline dataset $\mathcal{D}$. Intuitively, when $\mathcal{D}$ is a nearly full-coverage offline dataset, the learned $V_d^{\mu}(s)$ provides an accurate approximation of the true value function $V^{\mu}(s)$, such that

$$V_d^{\mu}(s) \approx V^{\mu}(s).$$

Value Function Learned from the Model. We then train another approximated value function corresponding to the behavior policy with the help of a learned model. We learn an approximate dynamics model $P_\theta(s', r \mid s, a)$ through maximum likelihood estimation and define the Bellman operator associated with P_θ as follows:

$$\widehat{\mathcal{T}}_m^\mu V(s) := \mathbb{E}_{a\sim\mu(\cdot \mid s),(r,s')\sim P_\theta}\left[r + \gamma V(s')\right]. \tag{4}$$

Note that when the learned model $P_\theta(s', r \mid s, a)$ provides an accurate approximation of the true environment model, denoted as $P^\star(s', r \mid s, a)$ (which jointly models the transition dynamics and reward), the induced Bellman operator $\widehat{\mathcal{T}}_m^\mu$ coincides with the true Bellman operator $\mathcal{T}^\mu$.

Proposition 2

The Bellman operator induced by P_θ, denoted $\widehat{\mathcal{T}}_m^\mu$, has a unique fixed point $V_m^\mu(s)$ such that

$$\widehat{\mathcal{T}}_m^\mu V_m^\mu(s) = V_m^\mu(s). \tag{5}$$

From Proposition 2, we can iteratively apply $\widehat{\mathcal{T}}_m^\mu$ to obtain the unique value function V_m^μ, representing the value function learned from the model. This value function serves as another approximation of $V^\mu(s)$, such that

$$V_m^\mu(s) \approx V^\mu(s).$$

Value Function Inconsistency. The core idea is that if the learned model P_θ is accurate, the value function $V_m^\mu(s)$—obtained from the model—should align with $V_d^\mu(s)$—derived from the dataset—as both aim to approximate the true value function $V^\mu(s)$. Therefore, we define the value function inconsistency loss as follows:

$$\mathcal{L}_{vic}(\theta) = \mathbb{E}_{s\sim\rho_0}\left[\left(V_d^\mu(s) - V_m^\mu(s)\right)^2\right], \tag{6}$$

where ρ_0 denotes the initial distribution, and $\mathcal{L}_{vic}$ depends on θ because $V_m^\mu(s)$ is derived from the learned model P_θ.

To learn the model, we begin with minimizing the negative log-likelihood, which we refer to as the original model (learning) loss,

$$\mathcal{L}_{\text{ori}}(\theta) = -\mathbb{E}_{\mathcal{D}}\left[\log P_\theta(s', r \mid s, a)\right]. \tag{7}$$

We incorporate the value function inconsistency loss into the original model loss to construct the following augmented model loss:

$$\mathcal{L}_{aug}(\theta) = \mathcal{L}_{ori}(\theta) + \lambda \mathcal{L}_{vic}(\theta), \tag{8}$$

where $\lambda > 0$ is a user-chosen hyperparameter that balances the contributions of the two loss components. Thus, $\mathcal{L}_{aug}$ serves as the overall loss for training the dynamics model which aims to optimize both the original model loss and the value function inconsistency loss.

Theorem 3 (Model Gradient Theorem)

Let θ represent the parameters of the dynamics model P_θ. Denote $V_d^\mu(\cdot)$ and $V_m^\mu(\cdot)$ as the value functions learned from the offline dataset $\mathcal{D}$, satisfying (3), and the model P_θ, satisfying (5), respectively. Then:

$$\nabla_\theta \mathcal{L}_{\text{aug}}(\theta) = \nabla_\theta \mathcal{L}_{\text{ori}}(\theta) - 2\lambda\, \mathbb{E}\Big[\big(V_d^\mu(s) - V_{m,\theta}^\mu(s)\big) \cdot \big(r' + \gamma V_{m,\theta}^\mu(s'')\big) \nabla_\theta \log P_\theta(s'', r' \mid s', a')\Big],$$

where the expectation is taken over $s \sim \rho_0$, $s' \sim d_\theta^\mu$, $a' \sim \mu(\cdot \mid s')$, and $(s'', r') \sim P_\theta(\cdot \mid s', a')$.

Empirical Results

Table: Normalized average returns on D4RL Gym tasks. The experiments are run on MuJoCo-"v2" datasets over 4 random seeds. r = random, m = medium, m-r = medium-replay, m-e = medium-expert. MOPO* indicates the results of MOPO retrained in "v2" datasets. The returns labeled with * in random tasks indicate values obtained from our training, as they were not reported in the original paper. We **bold** the highest mean.

Task Name	TD3+BC	CQL	IQL	DQL	MOPO*	MOReL	COMBO	RAMBO	MOBILE	VIPO-MOPO	VIPO-MOBILE
halfcheetah-r	10.2	31.3	16.1*	22.1*	38.5	25.6	38.8	39.5	39.3	$38.9_{\pm 0.7}$	$\mathbf{42.5_{\pm 0.2}}$
hopper-r	11.0	5.3	10.8*	17.5*	31.7	**53.6**	17.9	25.4	31.9	$30.2_{\pm 3.2}$	$33.4_{\pm 1.9}$
walker2d-r	1.4	5.4	7.7*	14.1*	7.4	**37.3**	7.0	0.0	17.9	$9.5_{\pm 0.8}$	$20.0_{\pm 0.1}$
halfcheetah-m	42.8	46.9	47.4	51.1	73.0	42.1	54.2	77.9	74.6	$78.6_{\pm 2.1}$	$\mathbf{80.0_{\pm 0.4}}$
hopper-m	99.5	61.9	66.3	90.5	62.8	95.4	97.2	87.0	106.6	$73.5_{\pm 1.7}$	$\mathbf{107.7_{\pm 1.0}}$
walker2d-m	79.7	79.5	78.3	87.0	84.1	77.8	81.9	84.9	87.7	$87.9_{\pm 2.9}$	$\mathbf{93.1_{\pm 1.8}}$
halfcheetah-m-r	43.4	45.3	44.2	47.8	72.1	40.2	55.1	68.7	71.7	$73.0_{\pm 1.8}$	$\mathbf{77.2_{\pm 0.4}}$
hopper-m-r	31.4	86.3	94.7	101.3	103.5	93.6	89.5	99.5	103.9	$103.0_{\pm 0.7}$	$\mathbf{109.6_{\pm 0.9}}$
walker2d-m-r	25.2	76.8	73.9	95.5	85.6	49.8	56.0	89.2	89.9	$88.2_{\pm 0.5}$	$\mathbf{98.4_{\pm 0.3}}$
halfcheetah-m-e	97.9	95.0	86.7	96.8	90.8	53.3	90.0	95.4	108.2	$93.8_{\pm 2.7}$	$\mathbf{110.0_{\pm 0.4}}$
hopper-m-e	112.2	96.9	91.5	111.1	81.6	108.7	111.1	88.2	112.6	$104.5_{\pm 3.1}$	$\mathbf{113.2_{\pm 0.1}}$
walker2d-m-e	105.7	109.1	109.6	110.1	112.9	95.6	103.3	56.7	115.2	$111.3_{\pm 1.0}$	$\mathbf{117.7_{\pm 1.0}}$
Average	54.6	61.6	60.6	70.4	70.3	64.4	66.8	67.7	80.0	74.4	**83.6**

Empirical Results

Table: Normalized average returns on NeoRL tasks. The experiments are run on MuJoCo-"v3" datasets over 4 random seeds. L = low, M = medium, H = high. We **bold** the highest mean.

Task Name	CQL	TD3+BC	MOPO	MOBILE	VIPO-MOBILE
halfcheetah-L	38.2	30.0	40.1	54.7	**58.5 ± 0.1**
hopper-L	16.0	15.8	6.2	17.4	**30.7 ± 0.3**
walker2d-L	44.7	43.0	11.6	37.6	**67.6 ± 0.7**
halfcheetah-M	54.6	52.3	62.3	77.8	**80.9 ± 0.2**
hopper-M	64.5	**70.3**	1.0	51.1	66.3 ± 0.2
walker2d-M	57.3	58.5	39.9	62.2	**76.8 ± 0.1**
halfcheetah-H	77.4	75.3	65.9	83.0	**89.4 ± 0.6**
hopper-H	76.6	75.3	11.5	87.8	**107.7 ± 0.5**
walker2d-H	75.3	69.6	18.0	74.9	**81.7 ± 1.0**
Average	56.1	54.5	28.5	60.7	**73.3**

Table: Normalized average returns on AntMaze tasks. Each score is the average success rate over 10 trials. We **bold** the highest mean.

Dataset	**MOBILE**	**IQL-TD-MPC**	**LEQ**	**VIPO-LEQ**
umaze	0.0	52.0	94.4 $\pm$ 6.3	**95.7 $\pm$ 4.9**
umaze-diverse	0.0	72.6	71.0 $\pm$ 12.3	**74.8 $\pm$ 12.7**
large-play	0.0	66.6	58.6 $\pm$ 9.1	**79.5 $\pm$ 4.8**
large-diverse	0.0	4.0	60.2 $\pm$ 18.3	**81.8 $\pm$ 8.3**
ultra-play	0.0	20.6	25.8 $\pm$ 18.2	**31.0 $\pm$ 18.7**
ultra-diverse	0.0	3.6	**55.8 $\pm$ 18.3**	26.5 $\pm$ 10.9
Average	0	36.6	61.0	**64.9**

For Interested Readers

Thank You!

Questions and discussion are welcome.

Full paper	https://arxiv.org/abs/2504.11944
Code	https://github.com/NUS-CORE/vipo