ABSTRACT

Value function estimation is an indispensable subroutine in reinforcement learning, which becomes more challenging in the offline setting. In this paper, we propose Hybrid Value Estimation (HVE) to reduce value estimation error, which trades off bias and variance by balancing between the value estimation from offline data and the learned model. Theoretical analysis discloses that HVE enjoys a better error bound than the direct methods. HVE can be leveraged in both off-policy evaluation and offline reinforcement learning settings. We, therefore, provide two concrete algorithms Off-policy HVE (OPHVE) and Model-based Offline HVE (MOHVE), respectively. Empirical evaluations on MuJoCo tasks corroborate the theoretical claim. OPHVE outperforms other off-policy evaluation methods in all three metrics measuring the estimation effectiveness, while MOHVE achieves better or comparable performance with state-of-the-art offline reinforcement learning algorithms. We hope that HVE could shed some light on further research on reinforcement learning from fixed data.

1 Introduction

Reinforcement learning (RL) \([1,2,3,4]\) has demonstrated great success in various sequential decision making problems, e.g., sequential recommendation systems \([5,6]\) and robotic locomotion skill learning \([7,8]\). Current reinforcement learning algorithms fall into two major categories: online RL algorithms and offline RL algorithms. The former category requires iteratively collecting samples by interacting with the environment and usually achieves optimal performance. The latter category learns a policy from a fixed dataset, making it suitable for risk-sensitive applications. However, the learned policy may still be sub-optimal. In both online setting and offline setting, value function estimation is an indispensable subroutine, which can be seen as the precondition of policy evaluation and policy improvement.

However, value function estimation was found to perform poorly without online interaction \([9]\), which is an important source of the sub-optimality in current offline RL algorithms. This failure is generally attributed to the extrapolation error when the value function is evaluated on out-of-distribution actions \([10,11]\). Model-based methods \([12,13,14]\) are a natural choice to approach this problem for two reasons. First, models are mainly trained with supervised learning with a fixed target, e.g., \([15]\), which is more stable than bootstrapping. Second, dynamics tend to be simpler than value function, so the learned model enjoys a better smoothness and generalization ability \([16]\).

Based on these existing promising benefits of model-based value estimation, we expect to improve it further. There are three kinds of errors in vanilla model-based value estimation, which are optimization error, projection error and statistical error. They are caused by the limitation of optimization algorithms, expressivity of neural networks and quantity of data. Such errors will accumulate along the trajectory and lead to the so-called compounding error \([17,18]\), which significantly degrades the performance of model-based value estimation. Inspired by the fact that the data in the offline dataset does not suffer from model error, we use the offline data to correct the error in value estimation caused by the inaccurate model.
The proposed value estimation method is named Hybrid Value Estimation (HVE), which uses both model-generated data and offline dataset to perform estimation. Model induces bias because of model error while the offline dataset induces bias and variance because of the policy divergence and data limitation respectively. HVE achieves a trade-off between bias and variance by automatically choosing the step length in Bellman update, which balances value estimation from offline data and the learned model. We implement two concrete algorithms OPHVE and MOHVE that apply HVE in off-policy evaluation and offline reinforcement learning respectively.

We provide a theoretical proof that HVE has a better error bound than vanilla model-based value estimation. The results on OPE benchmark DOPE [19] verify this theory empirically. The experiments suggest that MOHVE outperforms previous offline RL algorithms [20, 21, 22, 23] on the offline RL benchmark D4RL [24]. We also design an experiment to analyze the effect of step length. The results show its significance in the performance of OPHVE and MOHVE, and our automatic step length adjustment mechanism can find the near-optimal solution. Furthermore, we integrate MOHVE with OPHVE, improving the training and evaluation pipeline of offline RL.

2 Background and Related Work

Consider the standard Markov decision process (MDP), in which the environment is defined by a tuple $M = (S, A, T, r, \rho_0, \gamma)$. $S$ and $A$ denote the state space and action space respectively, both of which can be continuous or discrete. $T(s', a, s)$ is the dynamics or transition distribution and $\rho_0(s)$ is the initial state distribution. $r(s, a) \in [0, R_{\text{max}}]$ denotes the reward function, and $\gamma \in (0, 1)$ the discount factor. The goal of reinforcement learning is to find the optimal policy $\pi^*(a|s)$ that maximizes the expected discounted cumulative rewards:

$$\arg \max_{\pi} J_M(\pi) := \mathbb{E}_{\pi, T, \rho_0} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right].$$

Given a state-action pair $(s, a)$, define the state-action value function $Q^\pi(s, a)$ as $\mathbb{E}_{\pi, T, \rho_0} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a$, which represents the expected discounted return starting from $(s, a)$. Similarly, define state value function $V^\pi(s)$ as $\mathbb{E}_{\pi, T} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s$, the expected discounted return under $\pi$ when starting from state $s$. To facilitate our proof, we denote the discounted state visitation distribution of a policy $\pi$ under MDP $M$ as $d_M^\pi(s) := (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s | \pi)$, where $P(s_t = s | \pi)$ is the probability of reaching state $s$ at time $t$ by rolling out $\pi$ in $M$. Furthermore, we denote the discounted state-action visitation distribution $d_M^\pi(s, a) := d_M^\pi(s) \pi(a|s)$.

**Off-policy Evaluation.** OPE aims to evaluate a set of policies using data collected by other behavior policies without interacting with the environment. The behavior policies are sometimes known.

A straightforward approach to OPE is to fit the Q function directly from data using standard approximate dynamic programming (ADP) techniques, e.g., Fitted Q Evaluation (FQE) [25, 26]. Such an evaluation incurs high bias. Importance Sampling (IS) performs evaluation by calculating the distribution ratio of trajectories generated by behavior policy and target policy [27, 28]. Although being consistent and unbiased in theory, this class of estimators suffers from high variance. Marginalized Importance Sampling (MIS) [29, 30, 31, 32, 33] reduces the variance of IS by reweighting each state-action pair rather than reweighting the entire trajectory. Doubly Robust (DR) [34, 35] leverages an approximate Q function to decrease the variance of the unbiased estimates provided by IS, where MAGIC [35] uses a model to obtain the approximate Q function and aims to trade off between bias and variance based on MSE minimization. However, the trade-off needs the knowledge of true value function, which is not available in OPE setting. OPHVE achieves bias-variance trade-off by minimizing Mean Absolute Error and does not require the knowledge of true value function.

**Model-based RL.** Model-based reinforcement learning is featured with learned dynamic models. Recent model-based RL algorithms [17, 20, 21] use a bootstrap ensemble of dynamic models. The learned dynamic transition and (learned) reward function establish a model MDP $\tilde{M} = (S, A, \tilde{T}, \tilde{r}, \rho_0, \gamma)$. The agent can interact with $\tilde{M}$ to generate more data. Subsequently, we can use any learning or planning algorithms [36, 37] to recover the optimal policy in the model MDP as $\tilde{\pi} = \arg \max_{\pi} J_{\tilde{M}}(\pi)$. Besides generating more data, models are also useful for learning transferable policies [38].

**Offline RL.** In the offline reinforcement learning setting, agents cannot interact with a real environment like online RL. Instead, it only has access to a static dataset $D_{\text{env}} = \{(s, a, r, s')\}$ collected beforehand by one or a mixture of behavior policies, denoted by $\tilde{\pi}_\beta$.

Most recent offline RL works learn a lower bound of the ground truth Q function [20, 21, 22, 39, 40, 14], while MOHVE tries to learn the true value. BCQ [10], BEAR [11], TD3+BC [23] and EMaQ [41] are model free algorithms and constrain policy to select action similar to behavior policy. Model-based methods are also introduced for offline RL. For examples, MOPO [20] learns a conservative Q function by using model uncertainty; COMBO [21] avoids uncertainty.
Hybrid Value Estimation for Off-policy Evaluation and Offline Reinforcement Learning

quantification by adding a Q-value minimization term alongside a standard Bellman error objective. MAPLE [14] first introduced transfer meta-policy learning for model-based offline RL, which naturally overcame the out-of-distribution issue. In contrast with these approaches, MOHVE need not learn a conservative Q function because of its high value estimation accuracy.

Apart from the learning algorithms, some recent works turn to improve the entire training and evaluation pipeline of offline RL. NeoRL [9], a near real-world benchmark, points out the importance of offline policy selection. In [42], a new evaluation paradigm is proposed that uses a limited budget to evaluate the trained policy to select the best set of hyper-parameters. This paper takes a step in this direction by evaluating the performance of the trained policy in an offline manner and choosing the best training step to stop.

3 Reducing Value Estimation Error by Hybrid Value Estimation

In this section, we analyze the value estimation error of our method and the vanilla model-based method. Let us recall the procedure of value function estimation using a learned model. Let $\hat{Q}$ be the Q function induced by the learned model $\hat{M}$ and policy $\pi$. Bellman equation can be written as

$$\hat{Q}(s, a) = E_{s', a' \sim \hat{M}} \left[ \hat{r}(s, a) + \gamma \hat{Q}(s', a') \right].$$  

Without loss of generality, we assume that the expected TV-distance between two transition distributions is bounded by $\epsilon_m$, i.e., $\epsilon_m = E_{D} E_{s,a} D(T(\cdot|s,a), \hat{T}(\cdot|s,a))$ and $\epsilon_\pi = E_{D} E_{s,a} D(\pi(\cdot|s), \pi_\beta(\cdot|s))$, where $E_D$ is the expectation with respect to the generation of dataset $D$. Because the learned reward function is much more accurate than learned transition function, we ignore the error of learned reward function, and the value estimation error is bounded.

Theorem 3.1 (Theorem 4.1 in [17]). The error bound of the value function induced by Eq. (2) is:

$$E_D E_{s,a} D \left[ Q^\pi(s, a) - \hat{Q}(s, a) \right] \leq \frac{2\gamma R_{\max} (2\epsilon_\pi + \epsilon_m)}{(1 - \gamma)^2} + \frac{4R_{\max} \epsilon_\pi}{1 - \gamma}.$$  

The theorem states that the value estimation error is related to the model learning error and policy discrepancy. Although this bound has been improved in [13] with the compounding error reduced, we consider the old result in this paper for simplicity. These two quantities both induce compounding error proportional to the effective horizon (i.e., $\frac{1}{1-\gamma}$) of the MDP. Since $\gamma$ is close to 1, it suggests that the influence of model learning error and policy divergence is seriously magnified in value function estimation.

This issue is not severe in the online setting as $\epsilon_m$ and $\epsilon_\pi$ can be reduced via online interaction. However, it cannot be neglected in the offline setting. Because the model can only access the dataset during the training process, it suffers from higher statistical error due to data limitation. Furthermore, since the distribution of data in the dataset may deviate from the distribution of the current policy, the policy divergence $\epsilon_\pi$ is much larger in the offline setting [10].

Essentially, the value estimation error in model-based methods is due to the wrong transitions provided by the model. Note that the transitions in the offline dataset are correct, the model-induced value estimation error may be mitigated by directly utilizing the offline dataset. Concretely, we can integrate the true transitions $\{s_t, a_t\}_{t=0}^{H+1} \sim D$ into the Bellman equation, i.e.,

$$\hat{Q}(s_0, a_0) = E_{s_t, a_t \sim D} \left[ \sum_{t=0}^{H} \gamma^t r(s_t, a_t) + \gamma^{H+1} \hat{Q}(s_{H+1}, a_{H+1}) \right].$$  

This method is named Hybrid Value Estimation (HVE), which uses both offline data and the model. $\hat{Q}$ is the Q function learned with HVE.

For brevity, we first define $J_D(s_0, a_0, H) := E_{s_t, a_t \sim D} \left[ \sum_{t=0}^{H} \gamma^t r(s_t, a_t) \right]$ and $J_\pi(s_0, a_0, H) := E_{s_t, a_t \sim \pi(s)} \left[ \sum_{t=0}^{H} \gamma^t r(s_t, a_t) \right]$. Then we derive the following error bound.

Theorem 3.2. The error bound of the value function induced by Eq. (3) is:

$$E_D E_{s,a} D \left[ Q^\pi(s, a) - \hat{Q}(s, a) \right] \leq \sqrt{V(J_D(s, a, H))} + f(H)R_{\max} \epsilon_\pi + \gamma^{H+1} \epsilon_{M, \hat{M}}.$$  

3
where $f(H) = \left(1 - \gamma^{H+1} \right) - (H+1)\gamma^{H+1} \right) \right) - \left(1 - \gamma^{H+1} \right) - (H+1)\gamma^{H+1} \right) $ and $\epsilon_{m,M} = \left(2\gamma R_{\max}(2\epsilon_x + \epsilon_m) \right) + \frac{4R_{\max}\epsilon_m}{1 - \gamma}$. 

The variance term $\mathbb{V}(J_D(s, a, H))$ in Thm. 3.2 is further quantified by the following theorem. 

**Theorem 3.3.** If there are $n$ trajectories in dataset $\mathcal{D}$ starting from $(s, a)$, then 

$$\mathbb{V}(J_D(s, a, H)) = \frac{1}{n} \left( \mathbb{V}(\text{rew}) + \mathbb{V}(\text{trans}) \right),$$

where 

$$\mathbb{V}(\text{rew}) = \sum_{t=0}^{H} \mathbb{E}_{s_t, a_t \sim \mathcal{D}} \left[ \sum_{t'=0}^{H} \rho_t c_t \gamma^t r(s_t, a_t) \right],$$

and 

$$\mathbb{V}(\text{trans}) = \sum_{t=0}^{H} \mathbb{E}_{s_t, a_t, H \sim \mathcal{D}} \left[ \sum_{t'=0}^{H} \rho_t c_t \gamma^t r(s_t, a_t) \right].$$

**Remark 1.** $H$ can be seen as the parameter that trades off between the first two terms and the last term in the bound in Thm. 3.2. When $H = -1$, we only use the data generated by the model. The first two terms are equal to zero, and the bound degenerates to that of Thm. 3.1. When $H \to \infty$, the last term vanishes and the first two terms are dominating. It implies the balance is usually obtained when $H$ is neither too small nor too large.

**Remark 2.** To better understand the bound in Thm. 3.2 and Thm. 3.1, we give an example to illustrate the improvement. Consider a MDP with deterministic reward function, and let $R_{\max} = 1$ and $\gamma = 0.9$. Assume policy $\pi$ and model $M$ satisfy $\epsilon_x = 0.1$ and $\epsilon_m = 0.05$. For $H = 5$, we can calculate $\mathbb{V}(J_D(s, a, H))$ using Thm. 3.3. If $n = 9$, we have $\mathbb{V}(J_D(s, a, 5)) \leq 1.723$. Then the bound in Thm. 3.2 equals 4.55. While the bound in Thm. 3.3 equals 5.8.

We hope to further reduce value estimation error based on Thm. 3.3. Considering importance sampling [27] can adjust the value estimation of one policy into that of another policy, we use it to modify the reward provided by the offline dataset. Then we can avoid the influence of policy divergence. With importance sampling, Eq. (3) is reformulated as

$$\tilde{Q}^\pi(s_0, a_0) = \mathbb{E}_{s_t, a_t \sim \pi(s_{H+1})} \left[ \sum_{t=0}^{H} \rho_t c_t \gamma^t r(s_t, a_t) + H+1 \tilde{Q}^\pi(s_{H+1}, a_{H+1}) \right],$$

where $\rho_{t+1} = \rho_1 \rho_2 \cdots \rho_t$ for $t > 0$, $\rho_{1:0} = 1$ and $\rho_t = \frac{\pi(a_t | s_t)}{\pi_0(a_t | s_t)}$.

In this way, the second term $f(H)R_{\max}\epsilon_\pi$ equals zero. However, the first term will increase and become $\mathbb{V}(J_D^\pi(s, a, H))$, where $J_D^\pi(s, a, H) := \mathbb{E}_{s_t, a_t \sim \mathcal{D}} \left[ \sum_{t=0}^{H} \rho_{t+1} \gamma^t r(s_t, a_t) \right]$. The following theorem shows that the variance will be amplified by $\rho_{1:t}$.

**Theorem 3.4.** Under the conditions of Thm. 3.3, the variance of Eq. (4) is

$$\mathbb{V}(J_D^\pi(s, a, H)) = \frac{1}{n} \left( \mathbb{V}_{\pi}(\text{rew}) + \mathbb{V}_{\pi}(\text{trans}) \right),$$

where

$$\mathbb{V}_{\pi}(\text{rew}) = \sum_{t=0}^{H} \mathbb{E}_{s_t, a_t \sim \mathcal{D}} \left[ \rho_{t+1}^2 \gamma^t r(s_t, a_t) | s_t \right],$$

and

$$\mathbb{V}_{\pi}(\text{trans}) = \sum_{t=0}^{H} \mathbb{E}_{s_t, a_t, H \sim \mathcal{D}} \left[ \rho_{t+1}^2 \gamma^t r(s_t, a_t, H - t) | s_t, a_t = a \right].$$

Thm. 3.4 shows that the variance will be large if $\rho_{1:t}$ is large, which suggests that we need to balance the first and second term in Thm. 3.2 when considering $\rho_{1:t}$.

## 4 Practical Implementations of HVE

Previous sections propose the general framework of HVE, while there are several remaining aspects to discuss for a practical implementation. In this section, we discuss (1) how to obtain the behavior policy and dynamic model, (2) how to better balance bias and variance, and (3) how to choose the step length $H$ that minimizes the error.

### 4.1 Behavior Policy Learning and Model Learning

If the behavior policy $\pi_0(a | s)$ is available, $\rho_{1:t}$ in Eq. (4) and $\epsilon_\pi$ can be computed directly. Otherwise, we can approximate the behavior policy by maximizing the log likelihood of observed data like [43]:

$$\theta_0^* = \arg \max_{\theta_0} \mathbb{E}_{(s, a) \sim \mathcal{D}_m} \left[ \log \pi_0(a | s) \right],$$

(5)

(5)
where $\theta_b$ are the parameters of the behavior policy.

The model learning process is similar to MBPO [17], which learns an ensemble of probabilistic dynamic models and uses them for short rollouts. Each model is parameterized by a neural network whose outputs represent a Gaussian distribution with diagonal covariance: $\hat{T}_\psi(s_{t+1}|s_t, a_t) = \mathcal{N}(\mu_\psi(s_t, a_t), \Sigma_\psi(s_t, a_t))$. The model is typically learned by maximum likelihood estimation:

$$
\psi^* = \arg\max_\psi \mathbb{E}_{(s,a,s') \sim D_m}[\log \hat{T}_\psi(s'|s,a)].
$$

(6)

The reward function $r$, if unknown, can be jointly learned in the same way.

4.2 Importance Ratio Clipping

As suggested by Thm. 3.4, to better balance the influence of the first two terms, we can clip the importance ratio $\rho_{t:t}$ to the range of $(1 - \epsilon, 1 + \epsilon)$ to ensure that it is not far away from 1. As $\epsilon$ increases, the first term in the bound of Thm. 3.2 increases while the second term decreases. Then we can tune $\epsilon$ to get a trade-off between the two terms. We use $\mathbb{V}(J^2_D(s, a, H))$ to denote the variance with clipped importance ratio, $\epsilon^*_\pi$ to denote the reduced policy divergence after the truncated importance sampling, the refined error bound can be represented as

$$
\sqrt{\mathbb{V}(J^2_D(s, a, H))} + f(H) R_{\max} \epsilon^*_\pi + \gamma^{H+1} \epsilon_{M,\bar{M}}
$$

(7)

In practice, $\epsilon$ is robust across environments, and we choose $\epsilon$ equals 0.1 in all experiments.

4.3 Automatic Step Length Adjustment

Sec. 3 demonstrates that the value estimation error can be reduced by properly choosing step length $H$. Since the optimal $H$ could change across environments and training stages, it is important to automatically adjust $H$ to avoid tedious hyper-parameter tuning and ensure the optimal mean-variance trade-off.

The best $H$ is the one that minimizes value estimation error, which is trivial to compute if we know the value estimation error for every $H$. However, the exact error is intractable because the true value is not known in advance. To tackle this issue, we use the error bound in Eq. (7) as a surrogate of value estimation error, and the $H$ that minimizes the error bound is chosen as the parameter, i.e.,

$$
H_b = \arg\min_H \sqrt{\mathbb{V}(J^2_D(s, a, H))} + f(H) R_{\max} \epsilon^*_\pi + \gamma^{H+1} \epsilon_{M,\bar{M}}.
$$

(8)

The error bound of $H$ is determined if we can compute $\mathbb{V}(J^2_D(s, a, H))$, $\epsilon^*_\pi$ and $\epsilon_{M,\bar{M}}$, where $\epsilon_{M,\bar{M}}$ is constituted by $\epsilon_\pi$ and $\epsilon_m$. In the experiment, we tend to select small $H_b$ because the error of small $H_b$ is not worse than vanilla...
model-based value estimation while remaining unbounded if \( H_b \) is too large. This implies that we tend to overestimate the first two terms but underestimate the last term. Take MuJoCo tasks as an example. \( \mathbb{V}(J_B^\pi(s, a, H)) \) is upper bounded by \((1 + \epsilon)^2 \sum_{t=0}^H \gamma^t P_{\text{max}}^2/4\) and \( \epsilon^2 \) is upper bounded by \( \epsilon \). We choose the upper bound as the surrogates for the reason we mentioned above. Given behavior policy \( \pi_\delta \) and current policy \( \pi \), \( \epsilon \) can be calculated directly with the data in the dataset \( \mathcal{D} \). \( \epsilon_m \) is difficult to handle because the true model \( M \) is not accessible. We assume the output of the true model satisfies Gaussian distribution, whose mean is provided by the transitions in the offline dataset and the variance is identical to that of the learned model. Fig. 1 illustrates the relationship between the approximated error bound and \( H \) in three MuJoCo tasks. The error bound decreases first and then increases as step size \( H \) increases, proving that the best \( H \) achieves a trade-off between three terms in Thm. 32. Since the error bound of vanilla model-based value estimation is the value at \( H = -1 \), the experiment also shows value error reduction of HVE.

5 Applications

We present two concrete instantiations of HVE: Off-Policy HVE (OPHVE) and Model-based Offline HVE (MOHVE) which adopt HVE to improve performance in OPE and offline RL tasks.

**OPHVE.** OPHVE uses HVE for off-policy evaluation. According to the definition of Q function, the expected return of a policy can be expressed as \( J(\pi) = \mathbb{E}_{s \sim \rho_0, a \sim \pi} [Q^\pi(s, a)] \). With the estimated Q function, to perform OPE, we only need to sample states from initial state distribution \( \rho_0 \) and sample actions from the policy to be evaluated to approximate the expectation.

**MOHVE.** MOHVE follows the actor-critic framework. The critic is modified to be updated by the HVE objective to better guide the training of the actor.

From the error bound of Eq. (7), the second term \( f(H)R_{\text{max}} \epsilon \) and the last term \( \gamma^{H+1} \epsilon_{M,H} \) of the error bound is proportional to policy divergence term \( \epsilon \). To address this problem, we regularize the learned policy to make it close to the behavior policy, which is a common practice in previous offline RL algorithms [11][44][43]. This can be achieved by the constrained optimization formulated as follows:

\[
\pi = \arg \max_{\pi} \mathbb{E}_{s \sim \mathcal{D}_{\text{mix}}, a \sim \pi(s|\cdot)} [Q^\pi(s, a)] \quad \text{s.t.} \mathbb{E}_{s \sim \mathcal{D}_{\text{mix}}} \left[ D_{\text{KL}}(\pi(\cdot|s)||\pi_\beta(\cdot|s)) \right] \leq \delta. \tag{9}
\]

The state distribution \( \mathcal{D}_{\text{mix}} \) is the mixture of the offline data and the synthetic rollouts from the model: \( \mathcal{D}_{\text{mix}} := \alpha \mathcal{D}_{\text{env}} + (1 - \alpha) \mathcal{D}_{\text{model}} \), where \( \alpha \in [0, 1] \) is a hyperparameter to control the ratio of data points drawn from the offline dataset. \( D_{\text{KL}}(\pi||\pi_\beta) \) is the Kullback–Leibler Divergence between the learned policy and behavior policy. We use KL divergence instead of TV divergence in practice as KL divergence has a closed-form solution for Gaussian distribution. This substitution is reasonable because of the following relationship between the two divergences: \( D_{\text{TV}}(\rho, \pi) \leq 2D_{\text{KL}}(\rho, \pi) \).

To solve this constrained optimization problem, we can use the method of Lagrange multiplier to convert it to an objective that can be optimized by alternating gradient descent steps on the policy parameters \( \theta \) and a penalty coefficient \( \beta \):

\[
\max_{\theta} \beta \min_{\beta} \mathbb{E}_{s \sim \mathcal{D}_{\text{mix}}, a \sim \pi_\theta(s|\cdot)} \left[ Q(s, a) + \beta(\delta - D_{\text{KL}}(\pi_\theta(\cdot|s)||\pi_\beta(\cdot|s))) \right]. \tag{10}
\]

The whole process of MOHVE is summarized in Appx. B.1.

Besides the last two terms of Eq. (7), we find the constraint also reduces the variance term \( \mathbb{V}(J_B^\pi(s, H)) \) by limiting the scale of importance ratio as shown in the following proposition.

**Proposition 5.1.** If the policy \( \pi \) satisfies \( \mathbb{E}_{s \sim \mathcal{D}_{\text{mix}}} \left[ D_{\text{KL}}(\pi(\cdot|s)||\pi_\beta(\cdot|s)) \right] \leq \delta \), and suppose \( c = \max_{s, a} \left\| \frac{\pi_\chi(s)}{\pi_\beta(s)} \right\|_{\infty} \leq \infty \), then \( \mathbb{E}_{s \sim \mathcal{D}_{\text{mix}}} \left[ \frac{\pi(s|a)}{\pi_\beta(s|a)} \right]^2 \leq c\delta + 1 \).

6 Experiments

We design several experiments to demonstrate and analyze the performance of OPHVE and MOHVE. We also analyze the effect of step length \( H \) and the accuracy of automatic step length adjustment. More experiment details are given in Appx. B.
6.1 Off-Policy Evaluation Performance of OPHVE

We follow the setting of DOPE [19] to test the off-policy evaluation performance of OPHVE: For a given dataset $D$, we estimate the values of a set of policies $\{\pi_1, \pi_2, \ldots, \pi_N\}$. The estimated values are denoted as $\{\hat{V}_1, \hat{V}_2, \ldots, \hat{V}_N\}$. We also deploy these policies in real environments and record their discounted returns as ground truth values, denoted as $\{V_1, V_2, \ldots, V_N\}$. The policies to evaluate are generated during the training process of online RL algorithms Soft Actor-Critic [3]. DOPE is based on the D4RL benchmark of Gym-MuJoCo tasks [45, 24]. The tasks are challenging for OPE methods because they have data from various sources and policies that are generated using online RL algorithms, making it less likely that the behavior and evaluation policies share similar induced state-action distributions [19]. The performance of every method is evaluated using three different metrics: absolute error, regret@$k$ and rank correlation. The details of the metrics are in Appx. B.2.

![Graph](image)

Figure 2: The overall performance of off-policy evaluation in three metrics. The value of each bar is averaged over all tasks and the error bar correspond to the standard deviation. To aggregate across tasks, we normalize the policy values and evaluated policy values to range between 0 and 1.

In this experiment, we compare OPHVE to several previous methods including: (1) **Fitted Q-Evaluation (FQE)**: We adopt [25] that estimates the policy value by iteratively performing Bellman update and use the recent implementation [47]. (2) **Model-Based (MB)**: MB trains dynamics and reward models by maximizing the log likelihood of the next state and reward from the offline dataset as recent successful model-based RL algorithms [48, 17]. Then policy evaluation is performed by rolling out in this learned model. (3) **Importance Sampling (IS)**: We choose the weighted importance sampling method [27] with the implementation of [47]. The behavior policy is learned the same as our method. (4) **Doubly Robust (DR)**: We perform weighted doubly-robust [34], which has the least variance, and use the implementation of [47].

Fig. 2 shows the performance of OPHVE and other methods in three metrics. OPHVE consistently outperforms other methods on all metrics. The results of absolute error show that hybrid value estimation reduces the value estimation error, which verifies our theory in Section 3. Besides, OPHVE gets a higher rank correlation and lower regret. In conclusion, OPHVE can not only perform accurate evaluation but also select the competitive policies among the policies to be evaluated. Detailed OPE results of each task are in Appx. C.1.

6.2 Offline Reinforcement Learning Performance of MOHVE

We evaluate MOHVE on the D4RL benchmarks [24]. Our baselines include four offline reinforcement learning algorithms and one imitation learning algorithm. Among the four state-of-the-art offline RL algorithms, COMBO [21] and MOPO [20] are model-based methods, CQL [22] and TD3+BC [23] are model-free method. The imitation learning algorithm is behavior cloning [49].

We train MOHVE for 1 million time steps and report the average normalized score of the last ten iterations of policy update. Table 1 shows the results of offline reinforcement learning experiments. The numbers of BC are taken from the D4RL paper, while the results for COMBO, MOPO, TD3+BC, and CQL are based on their respective papers [21, 20, 23, 22]. In summary, MOHVE outperforms other methods in 7 out of the 12 tasks, and achieves comparable results in the rest 5 tasks. Previous offline RL work [20] found that model-based methods perform worse on the task with narrow data distributions (medium, medium-expert datasets) since it’s hard to learn models that generalize well. Our method performs well on these tasks because HVE reduces the value estimation error by adjusting the horizon $H$ in case that the model error is large. Besides better average performance, MOHVE generally shows a smaller variance in every task, indicating better stability and robustness. The learning curves of MOHVE are in Appx. C.2.

---

2 Despite recent exciting developments in marginalized importance sampling (MIS), it does not show strong empirical performance compared to other methods [19, 46]. Due to the difficult optimization of MIS, we defer the comparison with MIS to future work.
Hybrid Value Estimation for Off-policy Evaluation and Offline Reinforcement Learning

Table 1: Performance of MOHVE and baselines on the Gym-MuJoCo datasets, on the normalized return metric proposed by D4RL benchmark. The score roughly ranges from 0 to 100, where 0 corresponds to a random policy performance and 100 corresponds to an expert policy performance. Each result is the average score over five random seeds ± standard deviation.

| Data Type | Env.   | BC     | MOHVE (Ours) | COMBO     | MOPO     | TD3 +BC     | CQL     |
|-----------|--------|--------|--------------|-----------|----------|-------------|---------|
| random    | halfcheetah | 2.1    | 33.39±3.23   | 38.8±3.7  | 35.4±2.5 | 10.2±1.3    | 35.4    |
| random    | hopper  | 1.6    | 12.11±0.15   | 17.9±1.4  | 11.7±0.4 | 11.0±0.1    | 10.8    |
| random    | walker2d | 9.8    | 21.65±0.04   | 7.0±3.6   | 13.6±2.6 | 1.4±1.6     | 7.0     |
| medium    | halfcheetah | 36.1   | 56.48±1.22   | 54.2±1.5  | 42.3±1.6 | 42.8±0.3    | 44.4    |
| medium    | hopper  | 29.0   | 100.81±0.96  | 97.2±2.2  | 28.0±12.4| 99.5±1.0    | 58.0    |
| medium    | walker2d | 6.6    | 78.34±2.53   | 81.9±2.8  | 17.8±19.3| 79.7±1.8    | 79.2    |
| mixed     | halfcheetah | 38.4   | 57.68±0.89   | 55.1±1.0  | 53.1±2.0 | 43.3±0.5    | 46.2    |
| mixed     | hopper  | 11.8   | 93.94±2.52   | 89.5±1.8  | 67.5±24.7| 31.4±3.0    | 48.6    |
| mixed     | walker2d | 11.3   | 51.87±2.25   | 56.0±8.6  | 39.0±9.6 | 25.2±5.1    | 26.7    |
| med-exp   | halfcheetah | 35.8   | 103.06±2.72  | 90.0±5.6  | 63.3±38.0| 97.9±4.4    | 62.4    |
| med-exp   | hopper  | 111.9  | 111.69±0.19  | 111.1±2.9 | 23.7±6.0 | 112.2±0.2   | 111.0   |
| med-exp   | walker2d | 6.4    | 106.79±1.97  | 103.3±5.6 | 44.6±12.9| 105.7±2.7   | 98.7    |

Figure 3: Analysis experiments results on walker2d-random. (a): the OPE results in absolute error, rank correlation and regret@1. (b): the offline RL learning curves with different step length $H$. Note that $H$ equals $-1$ corresponds to only using the model rollout data.

6.3 Analysis of step length $H$

HVE benefits from the proposed way of hybridizing offline data and model rollout data. The step length $H$ plays an important role. We analyze the effect of the step length $H$ on OPHVE and MOHVE. The results are shown in Fig. 3. OPE results show that $H$ has a great impact on the performance of OPHVE. Compared to -1, 0, 4, 20, OPHVE performs the best in all three metrics when $H$ equals 2, which is the step length selected by the automatic adjustment. It should be noted that the variance in the experiment is induced by stochastic initialization and optimization process controlled by random seeds, while the variance term $V(J_{IS}^D(s, a, H))$ comes from the generation of the dataset, so the experiment variance cannot reflect the variance term.

Offline RL results show that our automatic adjustment of step length $H$ contributes to the performance gain of MOHVE. In detail, the vanilla model-based method (i.e., $H$ equals $-1$), and the hybrid method with $H$ equals 0 struggle to perform well. When we increase the length of $H$ to 2, the converged performance is improved. Finally, the performance decreases when $H$ is changed to $9$. Our method, MOHVE, converges to a higher normalized score more stably through automatically adjusting the parameter $H$ during the training process. In conclusion, the step length $H$ matters in OPE and offline RL, and our automatic selection method succeeds in finding the near-optimal one. Additional results and analysis on other tasks can be found in Appx. C.3.
6.4 Offline Policy Selection

It is common that the performance of the trained policy oscillates during the training process, as illustrated in Fig. 4. Therefore, simply choosing the policy in the end may not imply a good performance. To tackle this issue, we can obtain policy snapshots during the training process and select the best of them. In real-world application, we tend to limit the budget (the amount of policies deployed online after policy selection) of online evaluation to reduce risk and cost, if not completely forbid it. Since the number of the saved snapshots is usually much larger than the budget, we need to first perform offline policy selection to obtain a small enough policy set that includes good policies.

Since OPHVE achieves a remarkable performance in OPE tasks, we integrate it to MOHVE as the offline policy selection method. Such an integration forms a completely offline training and evaluation pipeline [9]. We choose uniform policy selection as the baseline, since it is a strong baseline under low online evaluation budgets [42]. We train MOHVE and save the policy every 50 epochs, and conduct offline policy selection under different online evaluation budgets. Tab. 2 shows the offline policy selection results. On one hand, OPHVE selects better policies than uniform policy selection under the same budget. On the other hand, our proposed offline RL pipeline, MOHVE+OPHVE, can further improve the performance of MOHVE, which sheds some light on designing offline RL algorithms more suitable for real word scenario.

Table 2: Offline policy selection results. TOP \( k \) reports the best online normalized score among \( k \) policies selected by corresponding offline policy selection methods. Each result is the average score over five random seeds ± standard deviation.

| Env.       | Final Iterations | Uniform Policy Selection | OPHVE          |
|------------|------------------|--------------------------|----------------|
|            |                  | TOP1 | TOP3 | TOP5 | TOP1 | TOP3 | TOP5 |
| walker2d-mixed | 51.87±2.25 | 43.09±0.81 | 55.96±1.70 | 60.05±2.69 | 56.82±7.68 | 62.22±6.00 | 66.43±5.54 |
| hopper-mixed   | 93.94±2.52 | 66.78±8.73 | 89.31±6.30 | 94.78±3.28 | 82.45±22.30 | 97.06±2.44 | 97.06±2.44 |

7 Conclusion

In this paper, we propose Hybrid Value Estimation (HVE) to perform a more accurate value function estimation in the offline setting. It automatically adjusts the step length parameter to get a bias-variance trade-off. The error bound of HVE is better than that of vanilla model-based value estimation. We provide two concrete algorithms OPHVE and MOHVE and thoroughly discuss the details of the implementation. Empirical evaluations on MuJoCo tasks corroborate the theoretical claim, showing that HVE indeed selects the best step length and contributes to the performance in
off-policy evaluation and offline RL tasks. We also test the performance of integrating MOHVE with OPHVE to select the best policy during training in an offline manner, which could shed some light on improving the training and evaluation pipeline of offline RL.

References

[1] Richard S. Sutton and Andrew G. Barto. Reinforcement learning - an introduction. Adaptive computation and machine learning. MIT Press, 1998.

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

[3] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In Proceedings of the 32nd International Conference on Machine Learning (ICML’15), pages 1889–1897, Lille, France, 2015.

[4] Xu-Hui Liu, Zhenghai Xue, Jingcheng Pang, Shengyi Jiang, Feng Xu, and Yang Yu. Regret minimization experience replay in off-policy reinforcement learning. In Proceedings of the 34th conference on Neural Information Processing Systems (NeurIPS’21), Virtual Event, 2021.

[5] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. A reinforcement learning framework for explainable recommendation. In Proceedings of the 18th International Conference on Data Mining (ICDM’18), pages 587–596, Singapore, 2018.

[6] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. In Proceedings of the 24th International Conference on Knowledge Discovery & Data Mining (KDD’18), pages 1040–1048, London, UK, 2018.

[7] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. In Proceedings of the 14th Robotics: Science and Systems (RSS’20), Virtual Event, 2020.

[8] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the 35th International Conference on Machine Learning (ICML’18), pages 1856–1865, Stockholmsmässan, Sweden, 2018.

[9] Rongjun Qin, Songyi Gao, Xingyuan Zhang, Zhen Xu, Shengkai Huang, Zewen Li, Weinan Zhang, and Yang Yu. Neorl: A near real-world benchmark for offline reinforcement learning. CoRR, abs/2102.00714, 2021.

[10] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In Proceedings of the 36th International Conference on Machine Learning (ICML’19), pages 2052–2062, Long Beach, CA, 2019.

[11] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS’19), pages 11761–11771, Vancouver, Canada, 2019.

[12] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. Virtual-Taobao: Virtualizing real-world online retail environment for reinforcement learning. In Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI’19), Honolulu, HI, 2019.

[13] Wenjie Shang, Qingyang Li, Zhiwei Qin, Yang Yu, Yiping Meng, and Jieping Ye. Partially observable environment estimation with uplift inference for reinforcement learning based recommendation. Machine Learning, (9):2603–2640, 2021.

[14] Xiong-Hui Chen, Yang Yu, Qingyang Li, Fan-Ming Luo, Zhiwei Qin, Wenjie Shang, and Jieping Ye. Offline model-based adaptable policy learning. In Proceedings of the 35th Annual Conference on Neural Information Processing Systems (NeurIPS’21), Virtual Event, 2021.

[15] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In Advances in Neural Information Processing Systems 31 (NeurIPS’18), pages 2455–2467, Montréal, Canada, 2018.

[16] Kefan Dong, Yuping Luo, Tianhe Yu, Chelsea Finn, and Tengyu Ma. On the expressivity of neural networks for deep reinforcement learning. In Proceedings of the 37th International Conference on Machine Learning (ICML’20), pages 2627–2637, Virtual Event, 2020.
[17] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS’19), pages 12498–12509, Virtual Event, 2019.

[18] Tian Xu, Ziniu Li, and Yang Yu. Error bounds of imitating policies and environments. In Advances in Neural Information Processing Systems 33 (NeurIPS’20), pages 15737–15749, 2020.

[19] Justin Fu, Mohammad Norouzi, Ofir Nachum, George Tucker, Ziyu Wang, Alexander Novikov, Mengjiao Yang, Michael R. Zhang, Yutian Chen, Aviral Kumar, Cosmin Paduraru, Sergey Levine, and Thomas Paine. Benchmarks for deep off-policy evaluation. In Proceedings of the 9th International Conference on Learning Representations (ICLR’21), Virtual Event, 2021.

[20] 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 Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS’20), pages 14129–14142, Virtual Event, 2020.

[21] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: conservative offline model-based policy optimization. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS’21), pages 28954–28967, Virtual Event, 2021.

[22] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS’20), Virtual Event, 2020.

[23] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In Proceedings of the 35th Annual Conference on Neural Information Processing Systems (NeurIPS’21), Virtual Event, 2021.

[24] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. CoRR, abs/2004.07219, 2020.

[25] Hoang Minh Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In Proceedings of the 36th International Conference on Machine Learning (ICML’19), pages 3703–3712, Long Beach, CA, 2019.

[26] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. Journal of Machine Learning Research, 6:503–556, 2005.

[27] Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In Proceedings of the 17th International Conference on Machine Learning (ICML’00), pages 759–766, Stanford, CA, 2000.

[28] Assaf Hallak and Shie Mannor. Consistent on-line off-policy evaluation. In Proceedings of the 34th International Conference on Machine Learning (ICML’17), pages 1372–1383, Sydney, Australia, 2017.

[29] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS’19), pages 2315–2325, Vancouver, Canada, 2019.

[30] Ruiyi Zhang, Bo Dai, Lihong Li, and Dale Schuurmans. Gendice: Generalized offline estimation of stationary values. In Proceedings of the 8th International Conference on Learning Representations (ICLR’20), Addis Ababa, Ethiopia, 2020.

[31] Masatoshi Uehara, Jiawei Huang, and Nan Jiang. Minimax weight and q-function learning for off-policy evaluation. In Proceedings of the 37th International Conference on Machine Learning (ICML’20), pages 9659–9668, Virtual Event, 2020.

[32] Mengjiao Yang, Ofir Nachum, Bo Dai, Lihong Li, and Dale Schuurmans. Off-policy evaluation via the regularized lagrangian. In Proceedings of the 34rd Annual Conference on Neural Information Processing Systems (NeurIPS’20), Virtual Event, 2020.

[33] Christina J. Yuan, Yash Chandak, Stephen Giguere, Philip S. Thomas, and Scott Niekum. SOPE: spectrum of off-policy estimators. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS’21), pages 18958–18969, Virtual Event, 2021.

[34] Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In Proceedings of the 33rd International Conference on Machine Learning (ICML’16), pages 652–661, New York City, NY, 2016.

[35] Philip S. Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In Proceedings of the 33rd International Conference on Machine Learning (ICML’16), pages 2139–2148, New York City, NY, 2016.
[36] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value expansion for efficient model-free reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML’18)*, Stockholmsmässan, Sweden, 2018.

[37] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS’18)*, pages 4759–4770, Montréal, Canada, 2018.

[38] Fan-Ming Luo, Shengyi Jiang, Yang Yu, Zongzhang Zhang, and Yi-Feng Zhang. Adapting environment sudden changes by learning context sensitive policy. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI’22)*, Virtual Conference, 2022.

[39] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS’20)*, Virtual Event, 2020.

[40] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems (NeurIPS’21)*, pages 7436–7447, Virtual Event, 2021.

[41] Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online RL. In *Proceedings of the 38th International Conference on Machine Learning (ICML’21)*, pages 3682–3691, Virtual Event, 2021.

[42] Vladislav Kurenkov and Sergey Kolesnikov. Showing your offline reinforcement learning work: Online evaluation budget matters. *CoRR*, abs/2110.04156, 2021.

[43] Noah Y. Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin A. Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *Proceedings of the 8th International Conference on Learning Representations (ICLR’20)*, Addis Ababa, Ethiopia, 2020.

[44] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *CoRR*, abs/1911.11361, 2019.

[45] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[46] Cameron Voloshin, Hoang Minh Le, Nan Jiang, and Yisong Yue. Empirical study of off-policy policy evaluation for reinforcement learning. *CoRR*, abs/1911.06854, 2019.

[47] Ilya Kostrikov and Ofir Nachum. Statistical bootstrapping for uncertainty estimation in off-policy evaluation. *CoRR*, abs/2007.13609, 2020.

[48] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS’18)*, pages 4759–4770, Montréal, Canada, 2018.

[49] Dean Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3:88–97, 1991.
A Proof of results in the main paper

A.1 Proof of Thm. 3.2

To provide proof of Thm. 3.2, we need Thm. 3.1, Lem. A.1 and Lem. A.2.

**Lemma A.1.** Let \( J_\pi(s_w, a_w, H) = \mathbb{E}_\pi[\sum_{t=0}^{H} \gamma^t r(s_t, a_t)|s_0 = s_w, a_0 = a_w] \), then

\[
\mathbb{E}_\mathcal{D} \left| J_\pi(s_w, a_w, H) - J_{\pi_\beta}(s_w, a_w, H) \right| \leq \left( \frac{1 - \gamma^{H+1}}{(1 - \gamma)^2} - \frac{(H + 1)\gamma^{H+1}}{1 - \gamma} \right) R_{max}\epsilon_\pi
\]

**Proof.** Let \( d_{t,w}(s) \) represents the distribution of state at time \( t \) by starting with \((s_w, a_w)\) and following policy \( \pi \), \( d_{t,w}^\beta(s) \) represents the distribution of state at time \( t \) by starting with \((s_w, a_w)\) and following policy \( \pi_\beta \), and \( d'_{t,w}(s) \) represents the distribution of states at time \( t \) by following \( \pi_\beta \) conditioned on starting with \((s_w, a_w)\) and the fact \( \pi_\beta \) performs different from \( \pi \) at least one step. Then

\[
d_{t,w}^\beta(s) = p_{t-1}d_{t,w}(s) + (1 - p_{t-1})d'_{t,w}(s),
\]

where \( p_{t-1} \) is the probability \( \pi_\beta \) performs the same as \( \pi \) in \( t - 1 \) steps.

For simplification, we denote \( \pi(\cdot|s) \) as \( \pi(s) \), denote \( \mathbb{E}_\mathcal{D}\mathbb{E}_{s \sim \mathcal{D}} \) as \( \mathbb{E}_{s, \mathcal{D}} \). Notice that

\[
\mathbb{E}_{s, \mathcal{D}}\Pr(\pi(s) = \pi_\beta(s))
= \mathbb{E}_{s, \mathcal{D}} \int \min(\pi(a|s), \pi_\beta(a|s))da
= \mathbb{E}_{s, \mathcal{D}} \left( \int \pi(a|s)da - \int (\pi(a|s) - \min(\pi(a|s), \pi_\beta(a|s)))da \right)
= \mathbb{E}_{s, \mathcal{D}} \left( 1 - D_{TV}(\pi(s), \pi_\beta(s)) \right)
= 1 - \epsilon_\pi
\]

Let \( p = \mathbb{E}_{s, \mathcal{D}}\Pr(\pi(s) = \pi_\beta(s)) \), we have

\( p_t \geq p_{t-1}p \geq p^t \)

Then

\[
\mathbb{E}_{s, \mathcal{D}} \left| d_{t,w}^\beta(s) - d_{t,w}(s) \right| = \mathbb{E}_{s, \mathcal{D}}(1 - p_{t-1}) \left| d_{t,w}(s) - d_{t,w}(s) \right|
\leq \mathbb{E}_{s, \mathcal{D}}(1 - p_{t-1}) \leq 1 - p^t
\]

(11)

We calculate \( \mathbb{E}_\mathcal{D} \left| J_{\pi_\beta}(s_w, a_w, H) - J_\pi(s_w, a_w, H) \right| \) as follows.

\[
\mathbb{E}_\mathcal{D} \left| J_{\pi_\beta}(s_w, a_w, H) - J_\pi(s_w, a_w, H) \right|
= \mathbb{E}_\mathcal{D} \left| \mathbb{E}_{s \sim d_{t,w}^\pi, a_t \sim \pi, s'_{t+1} \sim d_{t,w}^\pi} \sum_{t=0}^{H} \gamma^t (r(s_t, a_t) - r(s'_t, a'_t)) \right|
= \sum_{t=0}^{H} \gamma^t \int (d_{t,w}^\beta(s \pi_{\beta}(a|s) - d_{t,w}(s) \pi(a|s))r(s,a)dsda)
\leq \sum_{t=0}^{H} \gamma^t \int |d_{t,w}^\beta(s \pi_{\beta}(a|s) - d_{t,w}(s) \pi(a|s))r(s,a)dsda|\]
Using triangle inequality,
\[
\sum_{t=0}^{H} \gamma^t \int \left| d_{t,w}^\beta(s) \pi_\beta(a|s) - d_{t,w}(s) \pi(a|s) \right| r(s,a) ds da \\
\leq \sum_{t=0}^{H} \gamma^t \int \left( \left| d_{t,w}^\beta(s) - d_{t,w}(s) \right| \pi(a|s) + \left| \pi(a|s) - \pi_\beta(a|s) \right| d_{t,w}(s) \right) r(s,a) ds da \\
\leq \sum_{t=0}^{H} \gamma^t R_{\max} \int \left( \left| d_{t,s,w}^\beta(s) - d_{t,s,w}(s) \right| + \left| \pi(a|s) - \pi_\beta(a|s) \right| \right) ds da
\]

Using Eq. (11)
\[
\mathbb{E}_D \left| J_{\pi_\beta}(s_w, a_w, H) - J_{\pi}(s_w, a_w, H) \right| \\
\leq R_{\max} \sum_{t=0}^{H} \gamma^t \left( (1 - p^t) + \epsilon_\pi \right) \\
\leq R_{\max} \sum_{t=0}^{H} \gamma^t \left( 1 - (1 - \epsilon_\pi)^t + \epsilon_\pi \right) \\
\leq R_{\max} \sum_{t=0}^{H} \gamma^t t \epsilon_\pi \\
= \left( 1 - \gamma^{H+1} \right) \left( \frac{(H+1)\gamma^{H+1}}{(1-\gamma)^2} - \frac{H+1}{1-\gamma} \right) R_{\max} \epsilon_\pi
\]

And then we get
\[
\mathbb{E}_D \left| J_{\pi}(s_w, a_w, H) - J_{\pi_\beta}(s_w, a_w, H) \right| \leq \left( 1 - \gamma^{H+1} \right) \left( \frac{(H+1)\gamma^{H+1}}{(1-\gamma)^2} - \frac{H+1}{1-\gamma} \right) R_{\max} \epsilon_\pi
\]

\[
\text{Lemma A.2. Let } J_D(s_w, a_w, H) = \mathbb{E}_{s_t,a_t \sim D} \left[ \sum_{t=0}^{H} \gamma^t r(s_t, a_t) \right] | s_0 = s_w, \text{ we have} \\
\mathbb{E}_D \left| J_{\pi_\beta}(s_w, a_w, H) - J_D(s_w, a_w, H) \right| \leq \sqrt{\mathbb{V}(J_D(s_w, a_w, H))}
\]

\textbf{Proof.} Note that \( \mathbb{E}_D J_D(s_w, a_w, H) = J_{\pi_\beta}(s_w, a_w, H). \) Using the bias-variance decomposition of MSE, we have
\[
\mathbb{E}_D \left| J_{\pi_\beta}(s_w, a_w, H) - J_D(s_w, a_w, H) \right|^2 \\
= (J_{\pi_\beta}(s_w, a_w, H) - \mathbb{E}_D J_D(s_w, a_w, H))^2 + \mathbb{V}(J_D(s_w, a_w, H)) \\
= \mathbb{V}(J_D(s_w, a_w, H))
\]

Using Jensen’s inequality,
\[
\left( \mathbb{E}_D \left| J_{\pi_\beta}(s_w, a_w, H) - J_D(s_w, a_w, H) \right| \right)^2 \leq \mathbb{E}_D \left| J_{\pi_\beta}(s_w, a_w, H) - J_D(s_w, a_w, H) \right|^2
\]

Then we conclude the proof.

\textit{Proof of Thm. 3.2} Note that
\[
Q^\pi(s, a) = J_{\pi}(s, a, H) + \gamma^{H+1} \mathbb{E}_{s' \sim P_{H+1}(s, a), a' \sim \pi} Q^\pi(s', a'),
\]
\[
\hat{Q}^\pi(s, a) = J_D(s, a, H) + \gamma^{H+1} \mathbb{E}_{s' \sim P_{H+1}(s, a), a' \sim \pi} \hat{Q}_M^\pi(s', a'),
\]
where \( P_{H+1}(s, a) \) denotes the transition starting from \((s, a)\) and following policy \( \pi \) for \( H \) steps.
We use the shorthand:

$$E_D[Q_\pi(s, a) - \tilde{Q}_\pi(s, a)]$$

(a) \[ E_D|J_\pi(s, a, H) - J_D(s, a, H)| + E_D|\gamma^{H+1}E_{s'\sim P_{\pi,H+1}(s,a)}Q_\pi(s') - E_{s'\sim P_{\pi,H+1}(s,a)}Q_\pi^M(s')| \]

(b) \[ \sqrt{V(J_D(s, a, H))} + \frac{(1 - \gamma^{H+1})(H + 1)\gamma^{H+1}}{(1 - \gamma)^2} R_{\text{max}}\epsilon \]

where (a) uses triangle inequality and (b) uses Lem. A.1 and Lem. A.2.

\[ \square \]

A.2 Proof of Thm. 3.3 and Thm. 3.4

Thm. 3.4 can be reduced to Thm. 3.3 if \( \rho_0 t = 1 \), so we only need to prove Thm. 3.4

Let \( \tilde{J}(s_w, a_w, H) = \sum_{t=0}^{H} \rho_0 t r(s_t, a_t) \) given \( s_0 = s_w, a_0 = a_w \), then

\[ \tilde{J}(s_t, a_t, H - t) = (r(s_t, a_t) + \gamma \rho_{t+1} \tilde{J}(s_{t+1}, a_{t+1}, H - t - 1)) \]

We use the shorthand: \( E_t[\cdot] = E[\cdot|s_0, a_0, \ldots, s_{t-1}, a_{t-1}] \) for conditional expectations, and \( V_t[\cdot] \) for variances similarly.

**Lemma A.3.**

\[ V_t(\tilde{J}(s_t, a_t, H - t)) = E_t[\rho_t^2 V_{t+1}[r(s_t, a_t)] + E_t\left[\rho_t^2 \gamma V_{t+1}[\tilde{J}(s_{t+1}, a_{t+1}, H - t - 1)]\right] + V_t(J_\pi(s_t, a_t, H - t)) \]

**Proof.**

\[ V_t(\tilde{J}(s_t, a_t, H - t)) \]

\[ = E_t\left[\tilde{J}^2(s_t, a_t, H - t) - (E_t J_\pi(s_t, \pi(s_t), H - t))^2\right] \]

\[ = E_t\left[(r(s_t, a_t) + \gamma \rho_{t+1} \tilde{J}(s_{t+1}, a_{t+1}, H - t - 1))^2 - \tilde{J}_\pi^2(s_t, a_t, H - t)\right] + V_t(J_\pi(s_t, a_t, H - t)) \]

\[ = E_t\left[J_\pi(s_t, a_t, H - t) + r(s_t, a_t) + \gamma \rho_{t+1} \tilde{J}(s_{t+1}, a_{t+1}, H - t - 1) - J_\pi(s_t, a_t, H - t)\right]^2 \]

\[ - J_\pi^2(s_t, a_t, H - t) + V_t(J_\pi(s_t, a_t, H - t)) \]

\[ = E_t\left[(J_\pi(s_t, a_t, H - t) + (r(s_t, a_t) - R(s_t, a_t)) + \gamma \rho_{t+1} \tilde{J}(s_{t+1}, a_{t+1}, H - t - 1) \right] \]

\[ - E_{t+1}[J_\pi(s_{t+1}, a_{t+1}, H - t - 1)] \]

\[ + V_t(J_\pi(s_t, a_t, H - t)) \]

\[ = E_t\left[E_t[\gamma J_\pi^2(s_t, a_t, H - t) - J_\pi^2(s_t, \pi(s_t), H - t)|s_t] + E_t[E_t[(r(s_t, a_t) - R(s_t, a_t))^2|s_t, a_t] \right] \]

\[ + E_t\left[E_t[\gamma \rho_{t+1}^2 \tilde{J}(s_{t+1}, a_{t+1}, H - t - 1) - E_{t+1} J_\pi(s_{t+1}, a_{t+1}, H - t - 1))^2|s_t, a_t] \right] + V_t(J_\pi(s_t, a_t, H - t)) \]

\[ = E_t[V_{t+1}[r(s_t, a_t)] + E_t[\rho_{t+1}^2 V_{t+1}[\tilde{J}(s_{t+1}, a_{t+1}, H - t - 1)] + V_t(J_\pi(s_t, a_t, H - t)) \]

where \( R(s, a) = E[r(s, a)] \) and (a) uses the fact that conditioned on \( s_t \) and \( a_t \), \( r(s_t, a_t) - R(s_t, a_t) \) and \( \tilde{J}(s_{t+1}, a_{t+1}, H - t - 1) - E_{t+1} J_\pi(s_{t+1}, a_{t+1}, H - t - 1) \) are independent and have zero means, and all other terms are constants.

15
Similarly,

\[
\begin{align*}
\mathcal{V}_{t+1} & \left[ \hat{J}(s_t, a_t, H) \right] \\
& = \mathbb{E}_{t+1} \left[ \hat{J}^2(s_t, a_t, H - t) \right] - (\mathbb{E}_{t+1} [J_\pi(s_t, a_t, H - t)])^2 \\
& = \mathbb{E}_{t+1} [\mathcal{V}_{t+1} [r(s_t, a_t)]] + \mathbb{E}_{t+1} \left[ \rho_{t+1}^2 \gamma^2 \mathcal{V}_{t+1} [\hat{J}_D(s_{t+1}, a_{t+1}, H - t - 1)] \right] \\
& \quad + \mathcal{V}_{t+1} [J_\pi(s_t, a_t, H - t)] \\
& = \mathbb{E}_{t+1} [\mathcal{V}_{t+1} [r(s_t, a_t)]] + \mathbb{E}_{t+1} \left[ \rho_{t+1}^2 \gamma^2 \mathcal{V}_{t+1} [\hat{J}_D(s_{t+1}, a_{t+1}, H - t - 1)] \right].
\end{align*}
\]

The last equality follows the fact that \( \mathcal{V}_{t+1} [J_\pi(s_t, a_t, H - t)] = 0. \)

**Proof of Thm. 3.4**

\[
\begin{align*}
\mathcal{V} & (\hat{J}(s_w, a_w, H)) \\
& = \mathcal{V} [\hat{J}(s_0, a_0, H)] | s_0 = s_w, a_0 = a_w \\
& = \mathcal{V}_1 [\hat{J}(s_0, a_0, H)] \\
& \overset{(a)}{=} \mathbb{E}_1 [\mathcal{V}_1 [r(s_0, a_0)]] + \mathbb{E}_1 \left[ \rho_1^2 \gamma \mathcal{V}_1 [\hat{J}(s_1, a_1, H - 1)] \right] \\
& \overset{(b)}{=} \sum_{t=0}^{H} \mathbb{E} \left[ \rho_{t+1}^2 \gamma^{t+1} \mathcal{V}_{t+1} [r(s_t, a_t)]] | s_0 = s_w, a_0 = a_w \right] \\
& \quad + \sum_{t=1}^{H} \mathbb{E} \left[ \rho_{t+1}^2 \gamma^{t+1} \mathcal{V}_{t+1} [J_\pi(s_{t+1}, a_{t+1}, H - t)] | s_0 = s_w, a_0 = a_w \right],
\end{align*}
\]

where we define \( \rho_{1,0} = 1 \), (a) and (b) use Lem. A.3.

According to the definition of \( J_D(s_w, a_w, H) \), \( \mathcal{V}(J_D(s_w, a_w, H)) = \mathcal{V}(\hat{J}(s_w, a_w, H))/n \), where \( n \) is the number of trajectories in dataset \( D \) starting from \( (s_w, a_w) \).

**A.3 Proof of Prop. 5.1**

We first introduce the concept of the chi-square divergence.

**Definition A.4.** For r.v. \( P \) and \( Q \), the chi-square divergence of \( P \) and \( Q \), denoted as \( \chi^2(P, Q) \), is defined as

\[
\chi^2(P, Q) = \int \frac{dP^2}{dQ} - 1
\]

The following lemma reveal the relationship between chi-square divergence and KL divergence.

**Lemma A.5.** Let \( c = \left\| \frac{dP}{dQ} \right\|_\infty \leq \infty \), then

\[
cD_{KL}(P, Q) \geq \chi^2(P, Q)
\]

**Proof.** Note that \( \log x \geq \frac{x-1}{x} \),

\[
cD_{KL}(P, Q) \geq \left\| \frac{dP}{dQ} \right\|_\infty \int dP \left( \frac{dP/dQ - 1}{dP/dQ} \right) \geq \chi^2(P, Q)
\]

\[\square\]
We summarize the whole training process of MOHVE in Alg. 1.

We use the following metrics in the OPE experiments, which are the same as those used in [19].

\[ c_{D_{KL}}(\pi(\cdot|s), \pi_\beta(\cdot|s)) \geq \chi^2(\pi(\cdot|s), \pi_\beta(\cdot|s)) \]

Note that

\[ \chi^2(\pi(\cdot|s), \pi_\beta(\cdot|s)) = \int \left( \frac{\pi^2(a|s)}{\pi_\beta(a|s)} - 1 \right) da \]

\[ = \int \pi_\beta(a|s) \left( \frac{\pi^2(a|s)}{\pi_\beta^2(a|s)} \right) da - 1 \]

\[ = \mathbb{E}_{a \sim \pi} \left[ \frac{\pi^2(a|s)}{\pi_\beta^2(a|s)} \right] - 1, \]

we can get the relationship between importance ratio and policy KL divergence as

\[ c_{D_{KL}}(P, Q) \geq \mathbb{E}_{a \sim \pi_\beta} \left[ \mu^2 \right] - 1. \]

The final result can be obtained by applying the expectation of state distribution to both sides of the inequality. \( \square \)

### B Experiment Details

#### B.1 Detailed Training Procedure of MOHVE

According to the theory of Sec. 3 we need to build two separate networks to represent \( \hat{Q} \) and \( \tilde{Q} \). In practice, simply using one neural network is sufficient for good performance. Concretely, let \( Q_{\phi_k} \) be the Q function in the \( k \)-th iteration, we update Q function using Eq. (12) and (13) iteratively.

\[ \phi_{k+1} = \phi_k - \eta_2 \nabla_\phi \mathbb{E}_{s,a \sim \mathcal{D}_{env}} \left[ (Q(s,a) - \text{TIS}^H(s,a))^2 \right], \quad (12) \]

\[ \phi_{k+2} = \phi_{k+1} - \eta_2 \nabla_\phi \mathbb{E}_{s,a \sim \mathcal{D}_{model}} \left[ (Q(s,a) - \hat{B}^\pi Q^{k+1}(s,a))^2 \right], \quad (13) \]

where \( \eta \) is the learning rate and \( \text{TIS}^H(s_0,a_0) := \sum_{t=0}^H \gamma^t \rho_0 \psi(r(s_t,a_t)) + \gamma^{H+1} Q^{k+1}(s_{H+1},a_{H+1}) \).

We summarize the whole training process of MOHVE in Alg. 1.

**Algorithm 1** MOHVE: Model-based Offline RL with Hybrid Value Estimation

**Require:** offline data \( \mathcal{D}_{\text{env}} \), critic network \( Q_\phi \) with parameter \( \phi \), policy network \( \pi_\theta \) with parameter \( \theta \), model rollouts \( \mathcal{D}_{\text{model}} \), real data ratio \( \alpha \), KL regularization parameter \( \delta \), \( N_H \) epochs between \( H \) update.

1. Train an ensemble of \( N \) probabilistic dynamics \( \{ T^i_{\psi}(s', r|s,a) = \mathcal{N}(\mu^i_{\psi}(s,a), \Sigma^i_{\psi}(s,a)) \}^{N}_{i=1} \) on \( \mathcal{D}_{\text{env}} \) with Eq. (6).

2. Learn behavior policy \( \pi_\beta \) with Eq. (5).

3. Initialize policy \( \pi \) and empty replay buffer \( \mathcal{D}_{\text{model}} \leftarrow \emptyset \).

4. for epoch \( e \in 1, 2, \ldots \) do

5. Sample \( b \) states \( s_0 \) from \( \mathcal{D}_{\text{env}} \) as the initial states of model rollouts.

6. Rollout \( h \) steps in dynamics \( \{ T \}^{N}_{i=1} \) and add samples to \( \mathcal{D}_{\text{model}} \).

7. Update the Q-function \( Q_\phi \) via Hybrid Value Estimation by solving Eq. (12) and (13) iteratively.

8. Improve policy \( \pi_\theta \) under state marginal of \( \mathcal{D}_{\text{mix}} \) by solving Eq. (10).

9. if \( e \mod N_H = 0 \) then

10. Adjust step size \( H \) with Eq. (8).

11. end if

12. end for

#### B.2 OPE Metrics

We use the following metrics in the OPE experiments, which are the same as those used in [19]. \( V^\pi \) denotes the true value of the policy, and \( \hat{V}^\pi \) denotes the estimated value of the policy.
Hybrid Value Estimation for Off-policy Evaluation and Offline Reinforcement Learning

**Absolute Error** Absolute error is the difference between the true value and the estimated value of a policy:

\[
\text{Absolute Error} = \left| V^\pi - \hat{V}^\pi \right| \tag{14}
\]

**Rank Correlation** Rank correlation is an assessment of the degree of correlation between the estimated values and the true values. The definition of rank correlation is:

\[
\text{Rank Correlation} = \frac{\text{Cov}(V^\pi_{1:N}, \hat{V}^\pi_{1:N})}{\sigma(V^\pi_{1:N}) \sigma(\hat{V}^\pi_{1:N})} \tag{15}
\]

**Regret@k** Regret@k is the difference between the value of the best policy in the entire set, and the value of the best policy in the estimated top-k set. It can be written as:

\[
\text{Regret@k} = \max_i V^\pi_i - \max_{j \in \text{top}k(1:N)} V^\pi_j \tag{16}
\]

### B.3 Additional Experimental Setup

**Offline Dataset.** We make experiments on the D4RL benchmark of Gym-MuJoCo tasks \[45, 24\] in both OPE and offline RL experiments, including three environments (hopper, walker2d, and halfcheetah), four dataset types (random, medium, medium-replay, and medium-expert), and therefore twelve problem settings in total. The datasets are generated by different kinds of behavior policies as follows:

- **random**: roll out a randomly initialized policy for 1M steps.
- **medium**: train a policy online with SAC to approximately 1/3 the performance of the expert, then roll it out for 1M steps.
- **mixed**: uses the replay buffer of a policy trained up to the performance of the medium agent.
- **medium-expert**: combine samples from a fully-trained policy with samples from a partially trained policy.

**Hyperparameters.** Here we discuss the hyperparameters that we use for OPHVE and MOHVE.

- **Model rollout length** \( h \). Similar to most of the model-based algorithms \[17, 20, 21\], model rollout length \( h \) is an important hyperparameter in our experiments. Note that different from the hybrid step length \( H \), \( h \) is the length of rollout in the learned model. We use \( h = 10 \) in all tasks except for hopper-mixed and walker2d-mixed, unlike prior model-based methods \[17, 20, 21\], whose maximum rollout length is 5. We hypothesize the longer rollout length of HVE owing to its ability to reduce the value estimation error induced by large \( h \).
- **Maximum of the hybrid step length** \( H_{\text{max}} \). To avoid too large step length \( H \), which is computed through minimizing the approximated error bound, we clip \( H \) with parameter \( H_{\text{max}} \). We find this parameter robust, and set it to 4 in 10 out of 12 tasks.
- **KL regularization parameter** \( \delta \). The KL divergence regularization (cf. Eq. (10)) which controls the policy divergence term \( \epsilon_\pi \).
- **Real data ratio** \( \alpha \). This parameter determines the state distribution \( \mathcal{D}_{\text{mix}} \) used in the policy improvement stage. We search for \( \alpha \in \{0.2, 0.5, 0.8\} \).

The four mentioned hyper-parameters for each task are listed in Tab. 3.

**Model Training and Usage.** We train an ensemble of 7 models and use 5 models with a smaller validation error. Each model is parameterized as a 4-layer feedforward neural network with 256 hidden units. And there are two heads outputting the mean and logarithmic standard deviation of the next state. We also apply standardization to the input state and reward of the model. During each epoch, we sample 50000 states from the offline dataset as the initial state of each model rollout trajectory.

For OPHVE, we perform 100k gradient updates to learn the Q function. For MOHVE, we set \( N_H \), the frequency of updating \( H \), to 200 across all tasks, considering the stability of training. Other hyperparameters that correspond to the backbone RL algorithm SAC like learning rate and gradient steps follow the setup of COMBO \[21\].
Table 3: Hyperparameters used in experiments.

| Data Type | Env.       | h  | $H_{\text{max}}$ | $\delta$ | $\alpha$ |
|-----------|------------|----|-----------------|----------|----------|
| random    | halfcheetah| 10 | 4               | 20       | 0.2      |
| random    | hopper     | 10 | 4               | 10       | 0.8      |
| random    | walker2d   | 10 | 4               | 10       | 0.5      |
| medium    | halfcheetah| 10 | 4               | 20       | 0.5      |
| medium    | hopper     | 10 | 4               | 10       | 0.2      |
| medium    | walker2d   | 10 | 4               | 5        | 0.8      |
| mixed     | halfcheetah| 10 | 1               | 20       | 0.2      |
| mixed     | hopper     | 3  | 4               | 50       | 0.5      |
| mixed     | walker2d   | 1  | 2               | 8        | 0.5      |
| med-exp   | halfcheetah| 10 | 4               | 5        | 0.2      |
| med-exp   | hopper     | 10 | 4               | 0.5      | 0.2      |
| med-exp   | walker2d   | 10 | 4               | 5        | 0.5      |

B.4 Computational Resources

All experiments are conducted on a single NVIDIA GeForce RTX 2080 Ti. OPHVE takes about 1.5 hours to estimate 10 policies. And it takes about 10 hours to train MOHVE for 1000 epochs, including the time of online evaluation every epoch.

B.5 License of datasets

We acknowledge that D4RL datasets use the MIT license.

C Additional Results

C.1 OPE Experiments

Detailed OPE results on each task are shown in Fig. 5.
Figure 5: OPE results on each task.
C.2 Offline RL Experiments

We plot the learning curves of MOHVE on all tasks. We repeat every experiment five times in different random seeds. All results are shown in Fig. 6. MOHVE converges to relatively high scores in most datasets, which demonstrates the effectiveness of our Hybrid Value Estimation method in offline RL. Furthermore, the training process is stable with a small variance.

Figure 6: Learning curves of MOHVE on all tasks, where hc, hp and wk represent halfcheetah, hopper and walker2d respectively. The solid curves are the mean of normalized return and shadow is the standard error. The performance of MOHVE is stable and robust in most of the tasks.
C.3 Analysis of step length $H$ on walker2d-medium-expert

We analyze the effect of step length $H$ on OPHVE and MOHVE on walker2d-medium-expert task. The results of OPE are shown in Fig. 7. When $H = 0$, chosen by the automatic adjustment mechanism, OPHVE performs better than other choices on rank correlation and regret@1 and only performs no better than $H = -1$ on absolute error. Fig. 8 shows the learning curves of MOHVE with automatic adjusted $H$ (denoted by MOHVE) and fixed $H$. MOHVE converges to the highest normalized score with respect to other chosen step lengths. The two experiments verify that the automatic adjustment mechanism contributes to the performance of MOHVE.

![Figure 7: OPE results on walker2d-medium-expert with different step length $H$.](image1)

![Figure 8: Learning curves of offline RL with different step length $H$. MOHVE with automatic adjusted step length $H$ obtains the highest score.](image2)