MOPO: Model-based Offline Policy Optimization

Tianhe Yu[*1], **Garrett Thomas**[*1], **Lantao Yu**[1], **Stefano Ermon**[1], **James Zou**[1],
Sergey Levine[2], **Chelsea Finn**[†1], **Tengyu Ma**[†1]
Stanford University[1], UC Berkeley[2]
{tianheyu,gwthomas}@cs.stanford.edu

Abstract

Offline reinforcement learning (RL) refers to the problem of learning policies entirely from a large batch of previously collected data. This problem setting offers the promise of utilizing such datasets to acquire policies without any costly or dangerous active exploration. However, it is also challenging, due to the distributional shift between the offline training data and those states visited by the learned policy. Despite significant recent progress, the most successful prior methods are model-free and constrain the policy to the support of data, precluding generalization to unseen states. In this paper, we first observe that an existing model-based RL algorithm already produces significant gains in the offline setting compared to model-free approaches. However, standard model-based RL methods, designed for the online setting, do not provide an explicit mechanism to avoid the offline setting's distributional shift issue. Instead, we propose to modify the existing model-based RL methods by applying them with rewards artificially penalized by the uncertainty of the dynamics. We theoretically show that the algorithm maximizes a lower bound of the policy's return under the true MDP. We also characterize the trade-off between the gain and risk of leaving the support of the batch data. Our algorithm, Model-based Offline Policy Optimization (MOPO), outperforms standard model-based RL algorithms and prior state-of-the-art model-free offline RL algorithms on existing offline RL benchmarks and two challenging continuous control tasks that require generalizing from data collected for a different task.

1 Introduction

Recent advances in machine learning using deep neural networks have shown significant successes in scaling to large realistic datasets, such as ImageNet [13] in computer vision, SQuAD [55] in NLP, and RoboNet [10] in robot learning. Reinforcement learning (RL) methods, in contrast, struggle to scale to many real-world applications, e.g., autonomous driving [74] and healthcare [22], because they rely on costly online trial-and-error. However, pre-recorded datasets in domains like these can be large and diverse. Hence, designing RL algorithms that can learn from those diverse, static datasets would both enable more practical RL training in the real world and lead to more effective generalization.

While off-policy RL algorithms [43, 27, 20] can in principle utilize previously collected datasets, they perform poorly without online data collection. These failures are generally caused by large extrapolation error when the Q-function is evaluated on out-of-distribution actions [19, 36], which can lead to unstable learning and divergence. Offline RL methods propose to mitigate bootstrapped error by constraining the learned policy to the behavior policy induced by the dataset [19, 36, 72, 30, 49, 52, 58]. While these methods achieve reasonable performances in some settings, their learning is limited to behaviors within the data manifold. Specifically, these methods estimate error with respect to out-of-distribution *actions*, but only consider *states* that lie within the offline dataset and do not

---

[*]equal contribution. † equal advising. Orders randomized.

consider those that are out-of-distribution. We argue that it is important for an offline RL algorithm to be equipped with the ability to leave the data support to learn a better policy for two reasons: (1) the provided batch dataset is usually sub-optimal in terms of both the states and actions covered by the dataset, and (2) the target task can be different from the tasks performed in the batch data for various reasons, e.g., because data is not available or hard to collect for the target task. Hence, the central question that this work is trying to answer is: can we develop an offline RL algorithm that generalizes beyond the state and action support of the offline data?

To approach this question, we first hypothesize that model-based RL methods [64, 12, 42, 38, 29, 44] make a natural choice for enabling generalization, for a number of reasons. First, model-based RL algorithms effectively receive more supervision, since the model is trained on every transition, even in sparse-reward settings. Second, they are trained with supervised learning, which provides more stable and less noisy gradients than bootstrapping. Lastly, uncertainty estimation techniques, such as bootstrap ensembles, are well developed for supervised learning methods [40, 35, 60] and are known to perform poorly for value-based RL methods [72]. All
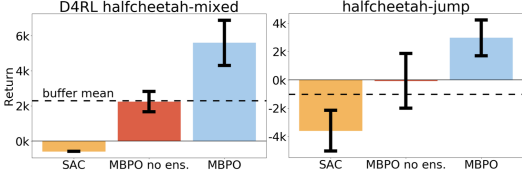

Figure 1: Comparison between vanilla model-based RL (MBPO [29]) with or without model ensembles and vanilla model-free RL (SAC [27]) on two offline RL tasks: one from the D4RL benchmark [18] and one that demands out-of-distribution generalization. We find that MBPO substantially outperforms SAC, providing some evidence that model-based approaches are well-suited for batch RL. For experiment details, see Section 5.

of these attributes have the potential to improve or control generalization. As a proof-of-concept experiment, we evaluate two state-of-the-art off-policy model-based and model-free algorithms, MBPO [29] and SAC [27], in Figure 1. Although neither method is designed for the batch setting, we find that the model-based method and its variant without ensembles show surprisingly large gains. This finding corroborates our hypothesis, suggesting that model-based methods are particularly well-suited for the batch setting, motivating their use in this paper.

Despite these promising preliminary results, we expect significant headroom for improvement. In particular, because offline model-based algorithms cannot improve the dynamics model using additional experience, we expect that such algorithms require careful use of the model in regions outside of the data support. Quantifying the risk imposed by imperfect dynamics and appropriately trading off that risk with the return is a key ingredient towards building a strong offline model-based RL algorithm. To do so, we modify MBPO to incorporate a *reward penalty* based on an estimate of the model error. Crucially, this estimate is model-dependent, and does not necessarily penalize all out-of-distribution states and actions equally, but rather prescribes penalties based on the estimated magnitude of model error. Further, this estimation is done both on *states* and *actions*, allowing generalization to both, in contrast to model-free approaches that only reason about uncertainty with respect to actions.

The primary contribution of this work is an offline model-based RL algorithm that optimizes a policy in an uncertainty-penalized MDP, where the reward function is penalized by an estimate of the model's error. Under this new MDP, we theoretically show that we maximize a lower bound of the return in the true MDP, and find the optimal trade-off between the return and the risk. Based on our analysis, we develop a practical method that estimates model error using the predicted variance of a learned model, uses this uncertainty estimate as a reward penalty, and trains a policy using MBPO in this uncertainty-penalized MDP. We empirically compare this approach, model-based offline policy optimization (MOPO), to both MBPO and existing state-of-the-art model-free offline RL algorithms. Our results suggest that MOPO substantially outperforms these prior methods on the offline RL benchmark D4RL [18] as well as on offline RL problems where the agent must generalize to out-of-distribution states in order to succeed.

2   Related Work

Reinforcement learning algorithms are well-known for their ability to acquire behaviors through online trial-and-error in the environment [3, 65]. However, such online data collection can incur high sample complexity [46, 56, 57], limit the power of generalization to unseen random initialization [9, 76, 4], and pose risks in safety-critical settings [68]. These requirements often make real-world applications of RL less feasible. To overcome some of these challenges, we study the batch offline

RL setting [41]. While many off-policy RL algorithms [53, 11, 31, 48, 43, 27, 20, 24, 25] can in principle be applied to a batch offline setting, they perform poorly in practice [19].

Model-free Offline RL. Many model-free batch RL methods are designed with two main ingredients: (1) constraining the learned policy to be closer to the behavioral policy either explicitly [19, 36, 72, 30, 49] or implicitly [52, 58], and (2) applying uncertainty quantification techniques, such as ensembles, to stabilize Q-functions [1, 36, 72]. In contrast, our *model-based* method does not rely on constraining the policy to the behavioral distribution, allowing the policy to potentially benefit from taking actions outside of it. Furthermore, we utilize uncertainty quantification to quantify the risk of leaving the behavioral distribution and trade it off with the gains of exploring diverse states.

Model-based Online RL. Our approach builds upon the wealth of prior work on model-based online RL methods that model the dynamics by Gaussian processes [12], local linear models [42, 38], neural network function approximators [15, 21, 14], and neural video prediction models [16, 32]. Our work is orthogonal to the choice of model. While prior approaches have used these models to select actions using planning [67, 17, 54, 51, 59, 70], we choose to build upon Dyna-style approaches that optimize for a policy [64, 66, 73, 32, 26, 28, 44], specifically MBPO [29]. See [71] for an empirical evaluation of several model-based RL algorithms. Uncertainty quantification, a key ingredient to our approach, is critical to good performance in model-based RL both theoretically [63, 75, 44] and empirically [12, 7, 50, 39, 8], and in optimal control [62, 2, 34]. Unlike these works, we develop and leverage proper uncertainty estimates that particularly suit the offline setting.

Concurrent work by Kidambi et al. [33] also develops an offline model-based RL algorithm, MOReL. Unlike MOReL, which constructs terminating states based on a hard threshold on uncertainty, MOPO uses a soft reward penalty to incorporate uncertainty. In principle, a potential benefit of a soft penalty is that the policy is allowed to take a few risky actions and then return to the confident area near the behavioral distribution without being terminated. Moreover, while Kidambi et al. [33] compares to model-free approaches, we make the further observation that even a vanilla model-based RL method outperforms model-free ones in the offline setting, opening interesting questions for future investigation. Finally, we evaluate our approach on both standard benchmarks [18] and domains that require out-of-distribution generalization, achieving positive results in both.

3 Preliminaries

We consider the standard Markov decision process (MDP) $M = (\mathcal{S}, \mathcal{A}, T, r, \mu_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state space and action space respectively, $T(s' \mid s, a)$ the transition dynamics, $r(s, a)$ the reward function, μ_0 the initial state distribution, and $\gamma \in (0, 1)$ the discount factor. The goal in RL is to optimize a policy $\pi(a \mid s)$ that maximizes the expected discounted return $\eta_M(\pi) := \mathbb{E}_{\pi, T, \mu_0} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]$. The value function $V_M^\pi(s) := \mathbb{E}_{\pi, T} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s \right]$ gives the expected discounted return under π when starting from state s.

In the *offline RL* problem, the algorithm only has access to a static dataset $\mathcal{D}_{\text{env}} = \{(s, a, r, s')\}$ collected by one or a mixture of behavior policies π^{B}, and cannot interact further with the environment. We refer to the distribution from which $\mathcal{D}_{\text{env}}$ was sampled as the *behavioral distribution*.

We also introduce the following notation for the derivation in Section 4. In the model-based approach we will have a dynamics model $\widehat{T}$ estimated from the transitions in $\mathcal{D}_{\text{env}}$. This *estimated dynamics* defines a *model MDP* $\widehat{M} = (\mathcal{S}, \mathcal{A}, \widehat{T}, r, \mu_0, \gamma)$. Let $\mathbb{P}_{\widehat{T}, t}^\pi(s)$ denote the probability of being in state s at time step t if actions are sampled according to π and transitions according to $\widehat{T}$. Let $\rho_{\widehat{T}}^\pi(s, a)$ be the discounted occupancy measure of policy π under dynamics $\widehat{T}$: $\rho_{\widehat{T}}^\pi(s, a) := \pi(a \mid s) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_{\widehat{T}, t}^\pi(s)$. Note that $\rho_{\widehat{T}}^\pi$, as defined here, is not a properly normalized probability distribution, as it integrates to $1/(1 - \gamma)$. We will denote (improper) expectations with respect to $\rho_{\widehat{T}}^\pi$ with $\bar{\mathbb{E}}$, as in $\eta_{\widehat{M}}(\pi) = \bar{\mathbb{E}}_{\rho_{\widehat{T}}^\pi}[r(s, a)]$.

We now summarize model-based policy optimization (MBPO) [29], which we build on in this work. MBPO learns a model of the transition distribution $\widehat{T}_\theta(s' \mid s, a)$ parametrized by θ, via supervised learning on the behavorial data $\mathcal{D}_{\text{env}}$. MBPO also learns a model of the reward function in the same manner. During training, MBPO performs k-step rollouts using $\widehat{T}_\theta(s' \mid s, a)$ starting from state

$s \in \mathcal{D}_{\text{env}}$, adds the generated data to a separate replay buffer $\mathcal{D}_{\text{model}}$, and finally updates the policy $\pi(a|s)$ using data sampled from $\mathcal{D}_{\text{env}} \cup \mathcal{D}_{\text{model}}$. When applied in an online setting, MBPO iteratively collects samples from the environment and uses them to further improve both the model and the policy. In our experiments in Table 1, Table 5.2 and Table 1, we observe that MBPO performs surprisingly well on the offline RL problem compared to model-free methods. In the next section, we derive MOPO, which builds upon MBPO to further improve performance.

4 MOPO: Model-Based Offline Policy Optimization

Unlike model-free methods, our goal is to design an offline model-based reinforcement learning algorithm that can take actions that are not strictly within the support of the behavioral distribution. Using a model gives us the potential to do so. However, models will become increasingly inaccurate further from the behavioral distribution, and vanilla model-based policy optimization algorithms may exploit these regions where the model is inaccurate. This concern is especially important in the offline setting, where mistakes in the dynamics will not be corrected with additional data collection.

For the algorithm to perform reliably, it's crucial to balance the return and risk: 1. the potential gain in performance by escaping the behavioral distribution and finding a better policy, and 2. the risk of overfitting to the errors of the dynamics at regions far away from the behavioral distribution. To achieve the optimal balance, we first bound the return from below by the return of a constructed model MDP penalized by the uncertainty of the dynamics (Section 4.1). Then we maximize the conservative estimation of the return by an off-the-shelf reinforcement learning algorithm, which gives MOPO, a generic model-based off-policy algorithm (Section 4.2). We discuss important practical implementation details in Section 4.3.

4.1 Quantifying the uncertainty: from the dynamics to the total return

Our key idea is to build a lower bound for the expected return of a policy π under the true dynamics and then maximize the lower bound over π. A natural estimator for the true return $\eta_M(\pi)$ is $\eta_{\widehat{M}}(\pi)$, the return under the estimated dynamics. The error of this estimator depends on, potentially in a complex fashion, the error of $\widehat{M}$, which may compound over time. In this subsection, we characterize how the error of $\widehat{M}$ influences the uncertainty of the total return. We begin by stating a lemma (adapted from [44]) that gives a precise relationship between the performance of a policy under dynamics T and dynamics $\widehat{T}$. (All proofs are given in Appendix B.)

Lemma 4.1 (Telescoping lemma). *Let M and $\widehat{M}$ be two MDPs with the same reward function r, but different dynamics T and $\widehat{T}$ respectively. Let $G_{\widehat{M}}^\pi(s,a) := \underset{s' \sim \widehat{T}(s,a)}{\mathbb{E}}[V_M^\pi(s')] - \underset{s' \sim T(s,a)}{\mathbb{E}}[V_M^\pi(s')]$.*

Then,

$$\eta_{\widehat{M}}(\pi) - \eta_M(\pi) = \gamma \, \bar{\underset{(s,a) \sim \rho_{\widehat{T}}^\pi}{\mathbb{E}}} \left[G_{\widehat{M}}^\pi(s,a) \right] \tag{1}$$

As an immediate corollary, we have

$$\eta_M(\pi) = \bar{\underset{(s,a) \sim \rho_{\widehat{T}}^\pi}{\mathbb{E}}} \left[r(s,a) - \gamma G_{\widehat{M}}^\pi(s,a) \right] \geq \bar{\underset{(s,a) \sim \rho_{\widehat{T}}^\pi}{\mathbb{E}}} \left[r(s,a) - \gamma |G_{\widehat{M}}^\pi(s,a)| \right] \tag{2}$$

Here and throughout the paper, we view T as the real dynamics and $\widehat{T}$ as the learned dynamics. We observe that the quantity $G_{\widehat{M}}^\pi(s,a)$ plays a key role linking the estimation error of the dynamics and the estimation error of the return. By definition, we have that $G_{\widehat{M}}^\pi(s,a)$ measures the difference between M and $\widehat{M}$ under the test function V^π — indeed, if $M = \widehat{M}$, then $G_{\widehat{M}}^\pi(s,a) = 0$. By equation (1), it governs the differences between the performances of π in the two MDPs. If we could estimate $G_{\widehat{M}}^\pi(s,a)$ or bound it from above, then we could use the RHS of (1) as an upper bound for the estimation error of $\eta_M(\pi)$. Moreover, equation (2) suggests that a policy that obtains high reward in the estimated MDP while also minimizing $G_{\widehat{M}}^\pi$ will obtain high reward in the real MDP.

However, computing $G_{\widehat{M}}^\pi$ remains elusive because it depends on the unknown function V_M^π. Leveraging properties of V_M^π, we will replace $G_{\widehat{M}}^\pi$ by an upper bound that depends solely on the error of the

4

dynamics $\widehat{T}$. We first note that if $\mathcal{F}$ is a set of functions mapping $\mathcal{S}$ to $\mathbb{R}$ that contains V_M^π, then,

$$|G_{\widehat{M}}^\pi(s,a)| \le \sup_{f \in \mathcal{F}} \left| \mathbb{E}_{s' \sim \widehat{T}(s,a)}[f(s')] - \mathbb{E}_{s' \sim T(s,a)}[f(s')] \right| =: d_{\mathcal{F}}(\widehat{T}(s,a), T(s,a)), \qquad (3)$$

where $d_{\mathcal{F}}$ is the integral probability metric (IPM) [47] defined by $\mathcal{F}$. IPMs are quite general and contain several other distance measures as special cases [61]. Depending on what we are willing to assume about V_M^π, there are multiple options to bound $G_{\widehat{M}}^\pi$ by some notion of error of $\widehat{T}$, discussed in greater detail in Appendix A:

(i) If $\mathcal{F} = \{f : \|f\|_\infty \le 1\}$, then $d_{\mathcal{F}}$ is the *total variation distance*. Thus, if we assume that the reward function is bounded such that $\forall(s,a),\ |r(s,a)| \le r_{\max}$, we have $\|V^\pi\|_\infty \le \sum_{t=0}^\infty \gamma^t r_{\max} = \frac{r_{\max}}{1-\gamma}$, and hence

$$|G_{\widehat{M}}^\pi(s,a)| \le \frac{r_{\max}}{1-\gamma} D_{\mathrm{TV}}(\widehat{T}(s,a), T(s,a)) \qquad (4)$$

(ii) If $\mathcal{F}$ is the set of 1-Lipschitz function w.r.t. to some distance metric, then $d_{\mathcal{F}}$ is the *1-Wasserstein distance* w.r.t. the same metric. Thus, if we assume that V_M^π is L_v-Lipschitz with respect to a norm $\|\cdot\|$, it follows that

$$|G_{\widehat{M}}^\pi(s,a)| \le L_v W_1(\widehat{T}(s,a), T(s,a)) \qquad (5)$$

Note that when $\widehat{T}$ and T are both deterministic, then $W_1(\widehat{T}(s,a), T(s,a)) = \|\widehat{T}(s,a) - T(s,a)\|$ (here $T(s,a)$ denotes the deterministic output of the model T).

Approach (ii) has the advantage that it incorporates the geometry of the state space, but at the cost of an additional assumption which is generally impossible to verify in our setting. The assumption in (i), on the other hand, is extremely mild and typically holds in practice. Therefore we will prefer (i) unless we have some prior knowledge about the MDP. We summarize the assumptions and the inequalities in the options above as follows.

Assumption 4.2. Assume a scalar c and a function class $\mathcal{F}$ such that $V_M^\pi \in c\mathcal{F}$ for all π.

As a direct corollary of Assumption 4.2 and equation (3), we have

$$|G_{\widehat{M}}^\pi(s,a)| \le c\, d_{\mathcal{F}}(\widehat{T}(s,a), T(s,a)). \qquad (6)$$

Concretely, option (i) above corresponds to $c = r_{\max}/(1-\gamma)$ and $\mathcal{F} = \{f : \|f\|_\infty \le 1\}$, and option (ii) corresponds to $c = L_v$ and $\mathcal{F} = \{f : f \text{ is 1-Lipschitz}\}$. We will analyze our framework under the assumption that we have access to an oracle uncertainty quantification module that provides an upper bound on the error of the model. In our implementation, we will estimate the error of the dynamics by heuristics (see sections 4.3 and D).

Assumption 4.3. Let $\mathcal{F}$ be the function class in Assumption 4.2. We say $u : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is an admissible error estimator for $\widehat{T}$ if $d_{\mathcal{F}}(\widehat{T}(s,a), T(s,a)) \le u(s,a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$.[2]

Given an admissible error estimator, we define the *uncertainty-penalized reward* $\tilde{r}(s,a) := r(s,a) - \lambda u(s,a)$ where $\lambda := \gamma c$, and the *uncertainty-penalized MDP* $\widetilde{M} = (\mathcal{S}, \mathcal{A}, \widehat{T}, \tilde{r}, \mu_0, \gamma)$. We observe that $\widetilde{M}$ is conservative in that the return under it bounds from below the true return:

$$\eta_M(\pi) \ge \bar{\mathbb{E}}_{(s,a) \sim \rho_{\widehat{T}}^\pi} \left[r(s,a) - \gamma |G_{\widehat{M}}^\pi(s,a)| \right] \ge \bar{\mathbb{E}}_{(s,a) \sim \rho_{\widehat{T}}^\pi} [r(s,a) - \lambda u(s,a)]$$

$$\text{(by equation (2) and (6))}$$

$$\ge \bar{\mathbb{E}}_{(s,a) \sim \rho_{\widehat{T}}^\pi} [\tilde{r}(s,a)] = \eta_{\widetilde{M}}(\pi) \qquad (7)$$

4.2 Policy optimization on uncertainty-penalized MDPs

Motivated by (7), we optimize the policy on the uncertainty-penalized MDP $\widetilde{M}$ in Algorithm 1.

---

[2]The definition here extends the definition of admissible confidence interval in [63] slightly to the setting of stochastic dynamics.

Algorithm 1 Framework for Model-based Offline Policy Optimization (MOPO) with Reward Penalty

---

Require: Dynamics model $\widehat{T}$ with admissible error estimator $u(s, a)$; constant λ.
 1: Define $\tilde{r}(s, a) = r(s, a) - \lambda u(s, a)$. Let $\widetilde{M}$ be the MDP with dynamics $\widehat{T}$ and reward $\tilde{r}$.
 2: Run any RL algorithm on $\widetilde{M}$ until convergence to obtain $\hat{\pi} = \operatorname{argmax}_\pi \eta_{\widetilde{M}}(\pi)$

Theoretical Guarantees for MOPO. We will theoretical analyze the algorithm by establishing the optimality of the learned policy $\hat{\pi}$ among a family of policies. Let $\pi^\star$ be the optimal policy on M and π^{B} be the policy that generates the batch data. Define $\epsilon_u(\pi)$ as

$$\epsilon_u(\pi) := \bar{\mathbb{E}}_{(s,a)\sim\rho_{\widehat{T}}^\pi}[u(s, a)] \tag{8}$$

Note that ϵ_u depends on $\widehat{T}$, but we omit this dependence in the notation for simplicity. We observe that $\epsilon_u(\pi)$ characterizes how erroneous the model is along trajectories induced by π. For example, consider the extreme case when $\pi = \pi^{\mathrm{B}}$. Because $\widehat{T}$ is learned on the data generated from π^{B}, we expect $\widehat{T}$ to be relatively accurate for those $(s, a) \sim \rho_{\widehat{T}}^{\pi^{\mathrm{B}}}$, and thus $u(s, a)$ tends to be small. Thus, we expect $\epsilon_u(\pi^{\mathrm{B}})$ to be quite small. On the other end of the spectrum, when π often visits states out of the batch data distribution in the real MDP, namely ρ_T^π is different from $\rho_T^{\pi^{\mathrm{B}}}$, we expect that $\rho_{\widehat{T}}^\pi$ is even more different from the batch data and therefore the error estimates $u(s, a)$ for those $(s, a) \sim \rho_{\widehat{T}}^\pi$ tend to be large. As a consequence, we have that $\epsilon_u(\pi)$ will be large.

For $\delta \geq \delta_{\min} := \min_\pi \epsilon_u(\pi)$, let π^δ be the best policy among those incurring model error at most δ:

$$\pi^\delta := \operatorname*{arg\,max}_{\pi:\epsilon_u(\pi)\leq\delta} \eta_M(\pi) \tag{9}$$

The main theorem provides a performance guarantee on the policy $\hat{\pi}$ produced by MOPO.

Theorem 4.4. *Under Assumption 4.2 and 4.3, the learned policy $\hat{\pi}$ in MOPO (Algorithm 1) satisfies*

$$\eta_M(\hat{\pi}) \geq \sup_\pi\{\eta_M(\pi) - 2\lambda\epsilon_u(\pi)\} \tag{10}$$

In particular, for all $\delta \geq \delta_{\min}$,

$$\eta_M(\hat{\pi}) \geq \eta_M(\pi^\delta) - 2\lambda\delta \tag{11}$$

Interpretation: One consequence of (10) is that $\eta_M(\hat{\pi}) \geq \eta_M(\pi^{\mathrm{B}}) - 2\lambda\epsilon_u(\pi^{\mathrm{B}})$. This suggests that $\hat{\pi}$ should perform at least as well as the behavior policy π^{B}, because, as argued before, $\epsilon_u(\pi^{\mathrm{B}})$ is expected to be small.

Equation (11) tells us that the learned policy $\hat{\pi}$ can be as good as any policy π with $\epsilon_u(\pi) \leq \delta$, or in other words, any policy that visits states with sufficiently small uncertainty as measured by $u(s, a)$. A special case of note is when $\delta = \epsilon_u(\pi^\star)$, we have $\eta_M(\hat{\pi}) \geq \eta_M(\pi^\star) - 2\lambda\epsilon_u(\pi^\star)$, which suggests that the suboptimality gap between the learned policy $\hat{\pi}$ and the optimal policy $\pi^\star$ depends on the error $\epsilon_u(\pi^\star)$. The closer $\rho_{\widehat{T}}^{\pi^\star}$ is to the batch data, the more likely the uncertainty $u(s, a)$ will be smaller on those points $(s, a) \sim \rho_{\widehat{T}}^{\pi^\star}$. On the other hand, the smaller the uncertainty error of the dynamics is, the smaller $\epsilon_u(\pi^\star)$ is. In the extreme case when $u(s, a) = 0$ (perfect dynamics and uncertainty quantification), we recover the optimal policy $\pi^\star$.

Second, by varying the choice of δ to maximize the RHS of Equation (11), we trade off the risk and the return. As δ increases, the return $\eta_M(\pi^\delta)$ increases also, since π^δ can be selected from a larger set of policies. However, the risk factor $2\lambda\delta$ increases also. ==The optimal choice of δ is achieved when the risk balances the gain from exploring policies far from the behavioral distribution.== The exact optimal choice of δ may depend on the particular problem. We note δ is only used in the analysis, and our algorithm *automatically achieves the optimal balance* because Equation (11) holds for any δ.

4.3 Practical implementation

Now we describe a practical implementation of MOPO motivated by the analysis above. The method is summarized in Algorithm 2 in Appendix C, and largely follows MBPO with a few key exceptions.

Following MBPO, we model the dynamics using a neural network that outputs a Gaussian distribution over the next state and reward[3]: $\widehat{T}_{\theta,\phi}(s_{t+1}, r | s_t, a_t) = \mathcal{N}(\mu_\theta(s_t, a_t), \Sigma_\phi(s_t, a_t))$. We learn an ensemble of N dynamics models $\{\widehat{T}^i_{\theta,\phi} = \mathcal{N}(\mu^i_\theta, \Sigma^i_\phi)\}^N_{i=1}$, with each model trained independently via maximum likelihood.

The most important distinction from MBPO is that we use uncertainty quantification following the analysis above. We aim to design the uncertainty estimator that captures both the epistemic and aleatoric uncertainty of the true dynamics. Bootstrap ensembles have been shown to give a consistent estimate of the population mean in theory [5] and empirically perform well in model-based RL [7]. Meanwhile, the learned variance of a Gaussian probabilistic model can theoretically recover the true aleatoric uncertainty when the model is well-specified. To leverage both, we design our error estimator $u(s, a) = \max^N_{i=1} \|\Sigma^i_\phi(s, a)\|_F$, the maximum standard deviation of the learned models in the ensemble. We use the maximum of the ensemble elements rather than the mean to be more conservative and robust. While this estimator lacks theoretical guarantees, we find that it is sufficiently accurate to achieve good performance in practice.[4] Hence the practical uncertainty-penalized reward of MOPO is computed as $\tilde{r}(s, a) = \hat{r}(s, a) - \lambda \max_{i=1,...,N} \|\Sigma^i_\phi(s, a)\|_F$ where $\hat{r}$ is the mean of the predicted reward output by $\widehat{T}$.

We treat the penalty coefficient λ as a user-chosen hyperparameter. Since we do not have a true admissible error estimator, the value of λ prescribed by the theory may not be an optimal choice in practice; it should be larger if our heuristic $u(s, a)$ underestimates the true error and smaller if u substantially overestimates the true error.

5 Experiments

In our experiments, we aim to study the follow questions: (1) How does MOPO perform on standard offline RL benchmarks in comparison to prior state-of-the-art approaches? (2) Can MOPO solve tasks that require generalization to out-of-distribution behaviors? (3) How does each component in MOPO affect performance?

Question (2) is particularly relevant for scenarios in which we have logged interactions with the environment but want to use those data to optimize a policy for a different reward function. To study (2) and challenge methods further, we construct two additional continuous control tasks that demand out-of-distribution generalization, as described in Section 5.2. To answer question (3), we conduct a complete ablation study to analyze the effect of each module in MOPO in Appendix D. For more details on the experimental set-up and hyperparameters, see Appendix G. For more details on the experimental set-up and hyperparameters, see Appendix G. The code is available online[5].

We compare against several baselines, including the current state-of-the-art model-free offline RL algorithms. Bootstrapping error accumulation reduction (BEAR) aims to constrain the policy's actions to lie in the support of the behavioral distribution [36]. This is implemented as a constraint on the average MMD [23] between $\pi(\cdot | s)$ and a generative model that approximates $\pi^B(\cdot | s)$. Behavior-regularized actor critic (BRAC) is a family of algorithms that operate by penalizing the value function by some measure of discrepancy (KL divergence or MMD) between $\pi(\cdot | s)$ and $\pi^B(\cdot | s)$ [72]. BRAC-v uses this penalty both when updating the critic and when updating the actor, while BRAC-p uses this penalty only when updating the actor and does not explicitly penalize the critic.

5.1 Evaluation on the D4RL benchmark

To answer question (1), we evaluate our method on a large subset of datasets in the D4RL benchmark [18] based on the MuJoCo simulator [69], including three environments (halfcheetah, hopper, and walker2d) and four dataset types (random, medium, mixed, medium-expert), yielding a total of

---

[3]If the reward function is known, we do not have to estimate the reward. The theory in Sections 4.1 and 4.2 applies to the case where the reward function is known. To extend the theory to an unknown reward function, we can consider the reward as being concatenated onto the state, so that the admissible error estimator bounds the error on (s', r), rather than just s'.

[4]Designing prediction confidence intervals with strong theoretical guarantees is challenging and beyond the scope of this work, which focuses on using uncertainty quantification properly in offline RL.

[5]Code is released at `https://github.com/tianheyu927/mopo`.

Dataset type	Environment	BC	MOPO (ours)	MBPO	SAC	BEAR	BRAC-v
random	halfcheetah	2.1	**35.4** ± 2.5	30.7 ± 3.9	30.5	25.5	28.1
random	hopper	1.6	11.7 ± 0.4	4.5 ± 6.0	11.3	9.5	**12.0**
random	walker2d	9.8	**13.6** ± 2.6	8.6 ± 8.1	4.1	6.7	0.5
medium	halfcheetah	36.1	42.3 ± 1.6	28.3 ± 22.7	-4.3	38.6	**45.5**
medium	hopper	29.0	28.0 ± 12.4	4.9 ± 3.3	0.8	**47.6**	32.3
medium	walker2d	6.6	17.8 ± 19.3	12.7 ± 7.6	0.9	33.2	**81.3**
mixed	halfcheetah	38.4	**53.1** ± 2.0	47.3 ± 12.6	-2.4	36.2	45.9
mixed	hopper	11.8	**67.5** ± 24.7	49.8 ± 30.4	1.9	10.8	0.9
mixed	walker2d	11.3	**39.0** ± 9.6	22.2 ± 12.7	3.5	25.3	0.8
med-expert	halfcheetah	35.8	**63.3** ±38.0	9.7 ± 9.5	1.8	51.7	45.3
med-expert	hopper	111.9	23.7 ± 6.0	**56.0** ± 34.5	1.6	4.0	0.8
med-expert	walker2d	6.4	44.6 ± 12.9	7.6 ± 3.7	-0.1	26.0	**66.6**

Table 1: Results for D4RL datasets. Each number is the normalized score proposed in [18] of the policy at the last iteration of training, averaged over 6 random seeds, ± standard deviation. The scores are undiscounted average returns normalized to roughly lie between 0 and 100, where a score of 0 corresponds to a random policy, and 100 corresponds to an expert. We include the performance of behavior cloning (**BC**) from the batch data for comparison. Numbers for model-free methods taken from [18], which does not report standard deviation. We omit BRAC-p in this table for space because BRAC-v obtains higher performance in 10 of these 12 tasks and is only slightly weaker on the other two. We bold the highest mean.

12 problem settings. We also perform empirical evaluations on non-MuJoCo environments in Appendix F. The datasets in this benchmark have been generated as follows: **random**: roll out a randomly initialized policy for 1M steps. **medium**: partially train a policy using SAC, then roll it out for 1M steps. **mixed**: train a policy using SAC until a certain (environment-specific) performance threshold is reached, and take the replay buffer as the batch. **medium-expert**: combine 1M samples of rollouts from a fully-trained policy with another 1M samples of rollouts from a partially trained policy or a random policy.

Results are given in Table 1. MOPO is the strongest by a significant margin on all the mixed datasets and most of the medium-expert datasets, while also achieving strong performance on all of the random datasets. MOPO performs less well on the medium datasets. We hypothesize that the lack of action diversity in the medium datasets make it more difficult to learn a model that generalizes well. Fortunately, this setting is one in which model-free methods can perform well, suggesting that model-based and model-free approaches are able to perform well in complementary settings.

5.2 Evaluation on tasks requiring out-of-distribution generalization

To answer question (2), we construct two environments `halfcheetah-jump` and `ant-angle` where the agent must solve a task that is different from the purpose of the behavioral policy. The trajectories of the batch data in the these datasets are from policies trained for the original dynamics and reward functions `HalfCheetah` and `Ant` in OpenAI Gym [6] which incentivize the cheetach and ant to move forward as fast as possible. Note that for `HalfCheetah`, we set the maximum velocity to be 3. Concretely, we train SAC for 1M steps and use the entire training replay buffer as the trajectories for the batch data. Then, we assign these trajectories with new rewards that incentivize the cheetach to jump and the ant to run towards the top right corner with a 30 degree angle. Thus, to achieve good performance for the new reward functions, the policy need to leave the observational distribution, as visualized in Figure 2. We include the exact forms of the new reward functions in Appendix G. In these environments, learning the correct behaviors requires leaving the support of the data distribution; optimizing solely within the data manifold will lead to sub-optimal policies.

In Table 2, we show that MOPO significantly outperforms the state-of-the-art model-free approaches. In particular, model-free offline RL cannot outperform the best trajectory in the batch dataset, whereas MOPO exceeds the batch max by a significant margin. This validates that MOPO is able to generalize to out-of-distribution behaviors while existing model-free methods are unable to solve those challenges. Note that vanilla MBPO performs much better than SAC in the two environments, consolidating our claim that vanilla model-based methods can attain better results than model-free methods in the offline setting, especially where generalization to out-of-distribution is needed. The visualization in Figure 2 suggests indeed the policy learned MOPO can effectively solve the tasks by reaching to states unseen in the batch data. Furthermore, we test the limit of the generalization abilities of MOPO in these environments and the results are included in Appendix E.
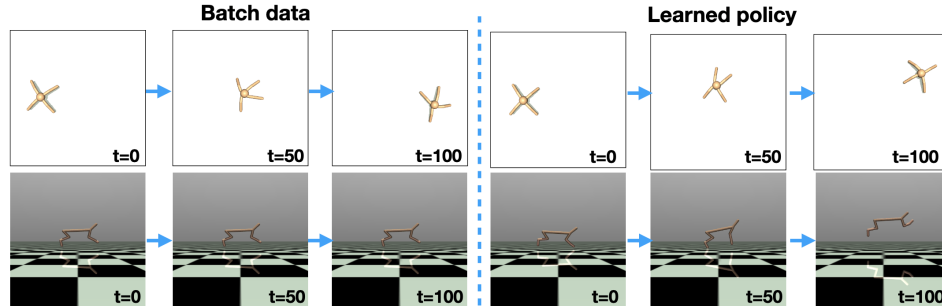
Figure 2: We visualize the two out-of-distribution generalization environments `halfcheetah-jump` (**bottom row**) and `ant-angle` (**top row**). We show the training environments that generate the batch data on the left. On the right, we show the test environments where the agents perform behaviors that require the learned policies to leave the data support. In `halfcheetah-jump`, the agent is asked to run while jumping as high as possible given an training offline dataset of halfcheetah running. In `ant-angle`, the ant is rewarded for running forward in a 30 degree angle and the corresponding training offline dataset contains data of the ant running forward directly.

Environment	Batch Mean	Batch Max	MOPO (ours)	MBPO	SAC	BEAR	BRAC-p	BRAC-v
halfcheetah-jump	-1022.6	1808.6	**4016.6±144**	2971.4±1262	-3588.2±1436	16.8±60	1069.9±232	871±41
ant-angle	866.7	2311.9	**2530.9±137**	13.6±66	-966.4±778	1658.2±16	1806.7±265	2333±139

Table 2: Average returns `halfcheetah-jump` and `ant-angle` that require out-of-distribution policy. The MOPO results are averaged over 6 random seeds, ± standard deviation, while the results of other methods are averaged over 3 random seeds. We include the mean and max undiscounted return of the episodes in the batch data (under Batch Mean and Batch Max, respectively) for comparison. Note that Batch Mean and Max are significantly lower than on-policy SAC, suggesting that the behaviors stored in the buffers are far from optimal and the agent needs to go beyond the data support in order to achieve better performance. As shown in the results, MOPO outperforms all the baselines by a large margin, indicating that MOPO is effective in generalizing to out-of-distribution states where model-free offline RL methods struggle.

6 Conclusion

In this paper, we studied model-based offline RL algorithms. We started with the observation that, in the offline setting, existing model-based methods significantly outperform vanilla model-free methods, suggesting that model-based methods are more resilient to the overestimation and overfitting issues that plague off-policy model-free RL algorithms. This phenomenon implies that model-based RL has the ability to generalize to states outside of the data support and such generalization is conducive for offline RL. However, online and offline algorithms must act differently when handling out-of-distribution states. Model error on out-of-distribution states that often drives exploration and corrective feedback in the online setting [37] can be detrimental when interaction is not allowed. Using theoretical principles, we develop an algorithm, model-based offline policy optimization (MOPO), which maximizes the policy on a MDP that penalizes states with high model uncertainty. MOPO trades off the risk of making mistakes and the benefit of diverse exploration from escaping the behavioral distribution. In our experiments, MOPO outperforms state-of-the-art offline RL methods in both standard benchmarks [18] and out-of-distribution generalization environments.

Our work opens up a number of questions and directions for future work. First, an interesting avenue for future research to incorporate the policy regularization ideas of BEAR and BRAC into the reward penalty framework to improve the performance of MOPO on narrow data distributions (such as the "medium" datasets in D4RL). Second, it's an interesting theoretical question to understand why model-based methods appear to be much better suited to the batch setting than model-free methods. Multiple potential factors include a greater supervision from the states (instead of only the reward), more stable and less noisy supervised gradient updates, or ease of uncertainty estimation. Our work suggests that uncertainty estimation plays an important role, particularly in settings that demand generalization. However, uncertainty estimation does not explain the entire difference nor does it explain why model-free methods cannot also enjoy the benefits of uncertainty estimation. For those domains where learning a model may be very difficult due to complex dynamics, developing better model-free offline RL methods may be desirable or imperative. Hence, it is crucial to conduct future research on investigating how to bring model-free offline RL methods up to the level of the performance of model-based methods, which would require further understanding where the generalization benefits come from.

Broader Impact

MOPO achieves significant strides in offline reinforcement learning, a problem setting that is particularly scalable to real-world settings. Offline reinforcement learning has a number of potential application domains, including autonomous driving, healthcare, robotics, and is notably amenable to safety-critical settings where online data collection is costly. For example, in autonomous driving, online interaction with the environment runs the risk of crashing and hurting people; offline RL methods can significantly reduce that risk by learning from a pre-recorded driving dataset collected by a safe behavioral policy. Moreover, our work opens up the possibility of learning policies offline for new tasks for which we do not already have expert data.

However, there are still risks associated with applying learned policies to high-risk domains. We have shown the benefits of explicitly accounting for error, but without reliable out-of-distribution uncertainty estimation techniques, there is a possibility that the policy will behave unpredictably when given a scenario it has not encountered. There is also the challenge of reward design: although the reward function will typically be under the engineer's control, it can be difficult to specify a reward function that elicits the desired behavior and is aligned with human objectives. Additionally, parametric models are known to be susceptible to adversarial attacks, and bad actors can potentially exploit this vulnerability. Advances in uncertainty quantification, human-computer interaction, and robustness will improve our ability to apply learning-based methods in safety-critical domains.

Supposing we succeed at producing safe and reliable policies, there is still possibility of negative societal impact. An increased ability to automate decision-making processes may reduce companies' demand for employees in certain industries (e.g. manufacturing and logistics), thereby affecting job availability. However, historically, advances in technology have also created new jobs that did not previously exist (e.g. software engineering), and it is unclear if the net impact on jobs will be positive or negative.

Despite the aforementioned risks and challenges, we believe that offline RL is a promising setting with enormous potential for automating and improving sequential decision-making in highly impactful domains. Currently, much additional work is needed to make offline RL sufficiently robust to be applied in safety-critical settings. We encourage the research community to pursue further study in uncertainty estimation, particularly considering the complications that arise in sequential decision problems.

Acknowledgments and Disclosure of Funding

References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. Striving for simplicity in off-policy deep reinforcement learning. *arXiv preprint arXiv:1907.04543*, 2019.

[2] Andrzej Banaszuk, Vladimir A Fonoberov, Thomas A Frewen, Marin Kobilarov, George Mathew, Igor Mezic, Alessandro Pinto, Tuhin Sahai, Harshad Sane, Alberto Speranzon, et al. Scalable approach to uncertainty quantification and robust design of interconnected dynamical systems. *Annual Reviews in Control*, 35(1):77–98, 2011.

[3] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[4] Emmanuel Bengio, Joelle Pineau, and Doina Precup. Interference and generalization in temporal difference learning. *arXiv preprint arXiv:2003.06350*, 2020.

[5] Peter J Bickel and David A Freedman. Some asymptotic theory for the bootstrap. *The annals of statistics*, pages 1196–1217, 1981.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[7] 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*, pages 4754–4765, 2018.

[8] Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. *arXiv preprint arXiv:1809.05214*, 2018.

[9] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

[10] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[11] Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

[12] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *arXiv preprint arXiv:1605.07127*, 2016.

[15] Andreas Draeger, Sebastian Engell, and Horst Ranke. Model predictive control using neural networks. *IEEE Control Systems Magazine*, 15(5):61–66, 1995.

[16] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[17] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

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

[19] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

[20] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[21] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, volume 4, page 34, 2016.

[22] Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez, and Leo Anthony Celi. Guidelines for reinforcement learning in healthcare. *Nat Med*, 25(1):16–18, 2019.

[23] Arthur Gretton, Karsten M. Borgwardt, Malte Rasch, Bernhard Scholköpf, and Alexander J. Smola. A kernel approach to comparing distributions. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, page 1637–1641. AAAI Press, 2007. ISBN 9781577353232.

[24] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

[25] Shixiang Shane Gu, Timothy Lillicrap, Richard E Turner, Zoubin Ghahramani, Bernhard Schölkopf, and Sergey Levine. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In *Advances in neural information processing systems*, pages 3846–3855, 2017.

[26] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[27] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[28] G Zacharias Holland, Erin J Talvitie, and Michael Bowling. The effect of planning shape on dyna-style planning in high-dimensional state spaces. *arXiv preprint arXiv:1806.01825*, 2018.

[29] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, pages 12498–12509, 2019.

[30] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

[31] Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. *arXiv preprint arXiv:1511.03722*, 2015.

[32] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari, 2019.

[33] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

[34] Kwang-Ki K Kim, Dongying Erin Shen, Zoltan K Nagy, and Richard D Braatz. Wiener's polynomial chaos for the analysis and control of nonlinear dynamical systems with probabilistic uncertainties [historical perspectives]. *IEEE Control Systems Magazine*, 33(5):58–67, 2013.

[35] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. *arXiv preprint arXiv:1807.00263*, 2018.

[36] 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*, pages 11761–11771, 2019.

[37] Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. *arXiv preprint arXiv:2003.07305*, 2020.

[38] Vikash Kumar, Emanuel Todorov, and Sergey Levine. Optimal control with learned local models: Application to dexterous manipulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 378–383. IEEE, 2016.

[39] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

[40] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

[41] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[42] Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

[43] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[44] 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.

[45] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[46] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

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

[48] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1054–1062, 2016.

[49] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

[50] Anusha Nagabandi, Kurt Konoglie, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. *arXiv preprint arXiv:1909.11652*, 2019.

[51] Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In *Advances in Neural Information Processing Systems*, pages 6118–6128, 2017.

[52] 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.

[53] Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, pages 417–424, 2001.

[54] Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. In *Advances in neural information processing systems*, pages 5690–5701, 2017.

[55] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[56] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[57] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[58] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[59] David Silver, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. The predictron: End-to-end learning and planning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3191–3199. JMLR. org, 2017.

[60] Jasper Snoek, Yaniv Ovadia, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D Sculley, Joshua Dillon, Jie Ren, and Zachary Nado. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13969–13980, 2019.

[61] Bharath K Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert RG Lanckriet. On integral probability metrics,\phi-divergences and binary classification. *arXiv preprint arXiv:0901.2698*, 2009.

[62] Robert F Stengel. *Optimal control and estimation*. Courier Corporation, 1994.

[63] Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[64] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[65] Richard S Sutton and Andrew G Barto. Reinforcement learning, 1998.

[66] Richard S Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael P Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. *arXiv preprint arXiv:1206.3285*, 2012.

[67] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2154–2162, 2016.

[68] Philip S Thomas. *Safe reinforcement learning*. PhD thesis, University of Massachusetts Libraries, 2015.

[69] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[70] Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649*, 2019.

[71] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

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

[73] Hengshuai Yao, Shalabh Bhatnagar, Dongcui Diao, Richard S Sutton, and Csaba Szepesvári. Multi-step dyna planning for policy evaluation and control. In *Advances in neural information processing systems*, pages 2187–2195, 2009.

[74] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2018.

[75] Andrea Zanette and Emma Brunskill. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds.

[76] Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

Appendix

A Reminders about integral probability metrics

Let $(\mathcal{X}, \Sigma)$ be a measurable space. The integral probability metric associated with a class $\mathcal{F}$ of (measurable) real-valued functions on $\mathcal{X}$ is defined as

$$d_{\mathcal{F}}(P,Q) = \sup_{f \in \mathcal{F}} \left| \int_{\mathcal{X}} f \, \mathrm{d}P - \int_{\mathcal{X}} f \, \mathrm{d}Q \right| = \sup_{f \in \mathcal{F}} \left| \mathbb{E}_{X \sim P}[f(X)] - \mathbb{E}_{Y \sim Q}[f(Y)] \right|$$

where P and Q are probability measures on $\mathcal{X}$. We note the following special cases:

(i) If $\mathcal{F} = \{f : \|f\|_{\infty} \leq 1\}$, then $d_{\mathcal{F}}$ is the *total variation distance*
$$d_{\mathcal{F}}(P,Q) = D_{\mathrm{TV}}(P,Q) := \sup_{A \in \Sigma} |P(A) - Q(A)|$$

(ii) If $\mathcal{F}$ is the set of 1-Lipschitz function w.r.t. to some cost function (metric) c on $\mathcal{X}$, then $d_{\mathcal{F}}$ is the *1-Wasserstein distance* w.r.t. the same metric:
$$d_{\mathcal{F}}(P,Q) = W_1(P,Q) := \inf_{\gamma \in \Gamma(P,Q)} \int_{\mathcal{X}^2} c(x,y) \, \mathrm{d}\gamma(x,y)$$

where $\Gamma(P,Q)$ denotes the set of all *couplings* of P and Q, i.e. joint distributions on $\mathcal{X}^2$ which have marginals P and Q.

(iii) If $\mathcal{F} = \{f : \|f\|_{\mathcal{H}} \leq 1\}$ where $\mathcal{H}$ is a reproducing kernel Hilbert space with kernel k, then $d_{\mathcal{F}}$ is the *maximum mean discrepancy*:
$$d_{\mathcal{F}}(P,Q) = \mathrm{MMD}(P,Q) := \sqrt{\mathbb{E}[k(X,X')] - 2\mathbb{E}[k(X,Y)] + \mathbb{E}[k(Y,Y')]}$$

where $X, X' \sim P$ and $Y, Y' \sim Q$.

In the context of Section 4.1, we have (at least) the following instantiations of Assumption 4.2:

(i) Assume the reward is bounded by $r_{\max}$. Then (since $\|V_M^{\pi}\|_{\infty} \leq \frac{r_{\max}}{1-\gamma}$)
$$|G_{\widehat{M}}^{\pi}(s,a)| \leq \frac{r_{\max}}{1-\gamma} D_{\mathrm{TV}}(\widehat{T}(s,a), T(s,a))$$

This corresponds to $c = \frac{r_{\max}}{1-\gamma}$ and $\mathcal{F} = \{f : \|f\|_{\infty} \leq 1\}$.

(ii) Assume V_M^{π} is L_v-Lipschitz. Then
$$|G_{\widehat{M}}^{\pi}(s,a)| \leq L_v W_1(\widehat{T}(s,a), T(s,a))$$

This corresponds to $c = L_v$ and $\mathcal{F} = \{f : f \text{ is 1-Lipschitz}\}$.

(iii) Assume $\|V_M^{\pi}\|_{\mathcal{H}} \leq \nu$. Then
$$|G_{\widehat{M}}^{\pi}(s,a)| \leq \nu \mathrm{MMD}(\widehat{T}(s,a), T(s,a))$$

This corresponds to $c = \nu$ and $\mathcal{F} = \{f : \|f\|_{\mathcal{H}} \leq 1\}$.

B Proofs

We provide a proof for Lemma 4.1 for completeness. The proof is essentially the same as that for [44, Lemma 4.3].

Proof. Let W_j be the expected return when executing π on $\widehat{T}$ for the first j steps, then switching to T for the remainder. That is,

$$W_j = \mathbb{E}_{\substack{a_t \sim \pi(s_t) \\ t < j : s_{t+1} \sim \widehat{T}(s_t, a_t) \\ t \geq j : s_{t+1} \sim T(s_t, a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]$$

Note that $W_0 = \eta_M(\pi)$ and $W_\infty = \eta_{\widehat{M}}(\pi)$, so

$$\eta_{\widehat{M}}(\pi) - \eta_M(\pi) = \sum_{j=0}^{\infty} (W_{j+1} - W_j)$$

Write

$$W_j = R_j + \mathbb{E}_{s_j,a_j \sim \pi,\widehat{T}} \left[\mathbb{E}_{s_{j+1} \sim T(s_t,a_t)} [\gamma^{j+1} V_M^\pi(s_{j+1})] \right]$$

$$W_{j+1} = R_j + \mathbb{E}_{s_j,a_j \sim \pi,\widehat{T}} \left[\mathbb{E}_{s_{j+1} \sim \widehat{T}(s_t,a_t)} [\gamma^{j+1} V_M^\pi(s_{j+1})] \right]$$

where R_j is the expected return of the first j time steps, which are taken with respect to $\widehat{T}$. Then

$$W_{j+1} - W_j = \gamma^{j+1} \mathbb{E}_{s_j,a_j \sim \pi,\widehat{T}} \left[\mathbb{E}_{s' \sim \widehat{T}(s_j,a_j)} [V_M^\pi(s')] - \mathbb{E}_{s' \sim T(s_j,a_j)} [V_M^\pi(s')] \right]$$

$$= \gamma^{j+1} \mathbb{E}_{s_j,a_j \sim \pi,\widehat{T}} \left[G_{\widehat{M}}^\pi(s_j, a_j) \right]$$

Thus

$$\eta_{\widehat{M}}(\pi) - \eta_M(\pi) = \sum_{j=0}^{\infty} (W_{j+1} - W_j)$$

$$= \sum_{j=0}^{\infty} \gamma^{j+1} \mathbb{E}_{s_j,a_j \sim \pi,\widehat{T}} \left[G_{\widehat{M}}^\pi(s_j, a_j) \right]$$

$$= \gamma \, \bar{\mathbb{E}}_{(s,a) \sim \rho_{\widehat{T}}^\pi} \left[G_{\widehat{M}}^\pi(s, a) \right]$$

as claimed. $\qquad\square$

Now we prove Theorem 4.2.

Proof. We first note that a two-sided bound follows from Lemma 4.1:

$$|\eta_{\widehat{M}}(\pi) - \eta_M(\pi)| \le \gamma \, \bar{\mathbb{E}}_{(s,a) \sim \rho_{\widehat{T}}^\pi} |G_{\widehat{M}}^\pi(s, a)| \le \lambda \, \bar{\mathbb{E}}_{(s,a) \sim \rho_{\widehat{T}}^\pi} [u(s, a)] = \lambda \epsilon_u(\pi) \qquad (12)$$

Then we have, for any policy π,

$$\eta_M(\hat{\pi}) \ge \eta_{\widetilde{M}}(\hat{\pi}) \qquad\qquad \text{(by (7))}$$

$$\ge \eta_{\widetilde{M}}(\pi) \qquad\qquad \text{(by definition of } \hat{\pi})$$

$$= \eta_{\widehat{M}}(\pi) - \lambda \epsilon_u(\pi)$$

$$\ge \eta_M(\pi) - 2\lambda \epsilon_u(\pi) \qquad\qquad \text{(by (12))}$$

$$\square$$

C MOPO Practical Algorithm Outline

We outline the practical MOPO algorithm in Algorithm 2.

D Ablation Study

To answer question (3), we conduct a thorough ablation study on MOPO. The main goal of the ablation study is to understand how the choice of reward penalty affects performance. We denote **no ens.** as a method without model ensembles, **ens. pen.** as a method that uses model ensemble disagreement as the reward penalty, **no pen.** as a method without reward penalty, and **true pen.** as

---

Algorithm 2 MOPO instantiation with regularized probabilistic dynamics and ensemble uncertainty

---
Require: reward penalty coefficient λ rollout horizon h, rollout batch size b.
1: Train on batch data $\mathcal{D}_{\text{env}}$ an ensemble of N probabilistic dynamics $\{\widehat{T}^i(s', r \mid s, a) = \mathcal{N}(\mu^i(s,a), \Sigma^i(s,a))\}_{i=1}^{N}$.
2: Initialize policy π and empty replay buffer $\mathcal{D}_{\text{model}} \leftarrow \varnothing$.
3: **for** epoch $1, 2, \ldots$ **do**   ▷ This for-loop is essentially one outer iteration of MBPO
4:      **for** $1, 2, \ldots, b$ (in parallel) **do**
5:          Sample state s_1 from $\mathcal{D}_{\text{env}}$ for the initialization of the rollout.
6:          **for** $j = 1, 2, \ldots, h$ **do**
7:              Sample an action $a_j \sim \pi(s_j)$.
8:              Randomly pick dynamics $\widehat{T}$ from $\{\widehat{T}^i\}_{i=1}^{N}$ and sample $s_{j+1}, r_j \sim \widehat{T}(s_j, a_j)$.
9:              Compute $\tilde{r}_j = r_j - \lambda \max_{i=1}^{N} \|\Sigma^i(s_j, a_j)\|_{\text{F}}$.
10:            Add sample $(s_j, a_j, \tilde{r}_j, s_{j+1})$ to $\mathcal{D}_{\text{model}}$.
11:      Drawing samples from $\mathcal{D}_{\text{env}} \cup \mathcal{D}_{\text{model}}$, use SAC to update π.

---

a method using the true model prediction error $\|\widehat{T}(s,a) - T(s,a)\|$ as the reward penalty. Note that we include **true pen.** to indicate the upper bound of our approach. Also, note that **no ens.** measures disagreement among the ensemble: precisely, if the models' mean predictions are denoted $\mu_1, \ldots, \mu_N$, we compute the average $\bar{\mu} = 1/N \sum_{i=1}^{N} \mu_i$ and then take $\max_i \|\mu_i - \bar{\mu}\|$ as the ensemble penalty.

The results of our study are shown in Table 3. For different reward penalty types, reward penalties based on learned variance perform comparably to those based on ensemble disagreement in D4RL environments while outperforming those based on ensemble disagreement in out-of-distribution domains. Both reward penalties achieve significantly better performances than no reward penalty, indicating that it is imperative to consider model uncertainty in batch model-based RL. Methods that uses oracle uncertainty obtain slightly better performance than most of our methods. Note that **MOPO** even attains the best results on halfcheetah-jump. Such results suggest that our uncertainty quantification on states is empirically successful, since there is only a small gap. We believe future work on improving uncertainty estimation may be able to bridge this gap further. Note that we do not report the results of methods with oracle uncertainty on walker2d-mixed and ant-angle as we are not able to get the true model error from the simulator based on the pre-recorded dataset.

In general, we find that performance differences are much larger for halfcheetah-jump and ant-angle than the D4RL halfcheetah-mixed and walker2d-mixed datasets, likely because halfcheetah-jump and ant-angle requires greater generalization and hence places more demands on the accuracy of the model and uncertainty estimate.

Finally, we perform another ablation study on the choice of the reward penalty. We consider the $u^{\text{mean}}(s,a) = \frac{1}{N} \sum_{i=1}^{N} \|\Sigma_\phi^i(s,a)\|_{\text{F}}$, the average standard deviation of the learned models in the ensemble, as the reward penalty instead of the max standard deviation as used in MOPO. We denote the variant of MOPO with the average learned standard deviation as **MOPO, avg. var.**. We compare **MOPO** to **MOPO, avg. var.** in the halfcheetah-jump domain. **MOPO** achieves **4140.6**±88 average return while **MOPO, avg. var.** achieves **4166.3**±228.8 where the results are averaged over 3 random seeds. The two methods did similarly, suggesting that using either mean variance or max variance would be a reasonable choice for penalizing uncertainty.

E   Empirical results on generalization capabilities

We conduct experiments in *ant-angle* to show the limit of MOPO's generalization capabilties. As shown in Table 4, we show that MOPO generalizes to Ant running at a $45°$ angle (achieving almost buffer max score), beyond the $30°$ shown in the paper, while failing to generalize to a 60 and $90°$ degree angle. This suggests that if the new task requires to explore states that are completely out of the data support, i.e. the buffer max and buffer mean both fairly bad, MOPO is unable to generalize.

Method	halfcheetah-mixed	walker2d-mixed	halfcheetah-jump	ant-angle
MOPO	6405.8 ± 35	1916.4 ± 611	4016.6 ± 144	2530.9 ± 137
MOPO, ens. pen.	6448.7 ± 115	1923.6 ± 752	3577.3 ± 461	2256.0 ± 288
MOPO, no pen.	6409.1 ± 429	1421.2 ± 359	-980.8 ± 5625	18.6 ± 49
MBPO	5598.4 ± 1285	1021.8 ± 586	2971.4 ± 1262	13.6 ± 65
MBPO, no ens.	2247.2 ± 581	500.3 ± 34	-68.7 ± 1936	-720.1 ± 728
MOPO, true pen.	6984.0 ± 148	N/A	3818.6 ± 136	N/A

Table 3: Ablation study on two D4RL tasks `halfcheetah-mixed` and `walker2d-mixed` and two out-of-distribution tasks `halfcheetah-jump` and `ant-angle`. We use average returns where the results of MOPO and its variants are averaged over 6 random seeds and MBPO results are averaged over 3 random seeds as in Table 2. We observe that different reward penalties can all lead to substantial improvement of the performance and reward penalty based on learned variance is a better choice than that based on ensemble disagreement in out-of-distribution cases. Methods that use oracle uncertainty as the reward penalty achieve marginally better performance than **MOPO**, implying that **MOPO** is effective at estimating the uncertainty.

Environment	Buffer Max	Buffer Mean	MOPO
`ant-angle-45`	3168.7	1105.5	2571.3 ± 598.1
`ant-angle-60`	1953.7	846.7	840.5 ± 1103.7
`ant-angle-90`	838.8	-901.6	-503.2 ± 803.4

Table 4: Limit of generalization on `ant-angle`.

F   Experiments on HIV domains

Beyond continous control tasks in MuJoCo, we test MOPO on an HIV treatment simulator slightly modified from the one in the whynot package. The task simulates the sequential decision making in HIV treatment, which involves determining the amounts of two anti-HIV drugs to be administered to the patient in order to maximize the immune response and minimize the amount of virus. The agent observes both of those quantities as well as the (log) number of infected and uninfected T cells and macrophages.

We evaluated MOPO with the data generated from the first 200k steps of training an online SAC agent on this environment. We show results in Table 5, where MOPO outperforms BEAR and achieves almost the buffer max score.

Buffer Max	Buffer Mean	SAC (online)	BEAR	MOPO
15986.2	6747.2	25716.3 ± 254.3	11709.1 ± 1292.1	$\mathbf{13484.6} \pm 3900.7$

Table 5: HIV treatment results, averaged over 3 random seeds.

G   Experiment Details

G.1   Details of out-of-distribution environments

For `halfcheetah-jump`, the reward function that we use to train the behavioral policy is $r(s, a) = \max\{v_x, 3\} - 0.1 * \|a\|_2^2$ where v_x denotes the velocity along the x-axis. After collecting the offline dataset, we relabel the reward function to $r(s, a) = \max\{v_x, 3\} - 0.1 * \|a\|_2^2 + 15 * (z - \text{init z})$ where z denotes the z-position of the half-cheetah and init z denotes the initial z-position.

For `ant-angle`, the reward function that we use to train the behavioral policy is $r(s, a) = v_x -$ control cost. After collecting the offline dataset, we relabel the reward function to $r(s, a) = v_x \cdot \cos\frac{\pi}{6} + v_y \cdot \sin\frac{\pi}{6} -$ control cost where v_x, v_y denote the velocity along the x, y-axis respectively.

For both out-of-distribution environments, instead of sampling actions from the learned policy during the model rollout (line 10 in Algorithm 2), we sample random actions from $\text{Unif}[-1, 1]$, which achieves better performance empirically. One potential reason is that using random actions during model rollouts leads to better exploration of the OOD states.

Dataset type	Environment	MOPO (h, λ)	MBPO h
random	halfcheetah	5, 0.5	5
random	hopper	5, 1	5
random	walker2d	1, 1	5
medium	halfcheetah	1, 1	5
medium	hopper	5, 5	5
medium	walker2d	5, 5	5
mixed	halfcheetah	5, 1	5
mixed	hopper	5, 1	5
mixed	walker2d	1, 1	1
med-expert	halfcheetah	5, 1	5
med-expert	hopper	5, 1	5
med-expert	walker2d	1, 2	1

Table 6: Hyperparameters used in the D4RL datasets.

G.2 Hyperparameters

Here we list the hyperparameters used in the experiments.

For the D4RL datasets, the rollout length h and penalty coefficient λ are given in Table 6. We search over $(h, \lambda) \in \{1, 5\}^2$ and report the best final performance, averaged over 3 seeds. The only exceptions are halfcheetah-random and walker2d-medium-expert, where other penalty coefficients were found to work better.

For the out-of-generalization tasks, we use rollout length 5 for `halfcheetah-jump` and 25 for `ant-angle`, and penalty coefficient 1 for `halfcheetah-jump` and 2 for `ant-angle`.

Across all domains, we train an ensemble of 7 models and pick the best 5 models based on their prediction error on a hold-out set of 1000 transitions in the offline dataset. Each of the model in the ensemble is parametrized as a 4-layer feedforward neural network with 200 hidden units and after the last hidden layer, the model outputs the mean and variance using a two-head architecture. Spectral normalization [45] is applied to all layers except the head that outputs the model variance.

For the SAC updates, we sample a batch of 256 transitions, 5% of them from $\mathcal{D}_{\text{env}}$ and the rest of them from $\mathcal{D}_{\text{model}}$. We also perform ablation studies on the percentage of the real data in a batch for MOPO. For simplicity, we use MBPO, which essentially MOPO without reward penalty, for this ablation study. We tried to train MBPO with data all sampled from $\mathcal{D}_{\text{model}}$ and no data from $\mathcal{D}_{\text{env}}$ and compare the performance to MBPO with 5% of data from $\mathcal{D}_{\text{env}}$ on all 12 settings in the D4RL benchmark. We find that the performances of both methods are not significantly distinct: no-real-data MBPO outperforms 5%-real-data MBPO on 6 out of 12 tasks and lies within one SD of 5%-real-data MBPO on 9 out of 12 tasks.