When to Update Your Model: Constrained Model-based Reinforcement Learning

Tianying Ji[1], **Yu Luo**[1], **Fuchun Sun**[*,1], **Mingxuan Jing**[2], **Fengxiang He**[3], **Wenbing Huang**[4,5]
[1] Department of Computer Science and Technology, Tsinghua University
[2] Science & Technology on Integrated Information System Laboratory, Institute of Software Chinese Academy of Sciences
[3] JD Explore Academy, JD.com Inc
[4] Gaoling School of Artificial Intelligence, Renmin University of China
[5] Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China
{jity20, luoyu19}@mails.tsinghua.edu.cn; fcsun@tsinghua.edu.cn;
jingmingxuan@iscas.ac.cn; fengxiang.f.he@gmail.com; hwenbing@126.com

Abstract

Designing and analyzing model-based RL (MBRL) algorithms with guaranteed monotonic improvement has been challenging, mainly due to the interdependence between policy optimization and model learning. Existing discrepancy bounds generally ignore the impacts of model shifts, and their corresponding algorithms are prone to degrade performance by drastic model updating. In this work, we first propose a novel and general theoretical scheme for a non-decreasing performance guarantee of MBRL. Our follow-up derived bounds reveal the relationship between model shifts and performance improvement. These discoveries encourage us to formulate a constrained lower-bound optimization problem to permit the monotonicity of MBRL. A further example demonstrates that learning models from a dynamically-varying number of explorations benefit the eventual returns. Motivated by these analyses, we design a simple but effective algorithm CMLO (Constrained Model-shift Lower-bound Optimization), by introducing an event-triggered mechanism that flexibly determines when to update the model. Experiments show that CMLO surpasses other state-of-the-art methods and produces a boost when various policy optimization methods are employed.

1 Introduction

Reinforcement learning (RL) has driven impressive advances in many complex decision-making problems in recent years [36, 48]. Many of these advances are obtained by model-free (MFRL) methods, whose desirable asymptotic performance yet comes at the cost of sample efficiency. Hence their applications are mostly limited to simulation scenarios [45, 33, 35]. In contrast, Model-Based RL (MBRL) methods, which learn a transition model directly from orders-of-magnitude fewer samples and then derive the optimal policy from the learned model, have become an appealing alternative in small-data and more practical cases [37, 9, 41, 18].

In general, MBRL methods alternate between the two stages: model learning and policy optimization (*e.g.* the general Dyna-style [50, 51]). A more accurate model will lead to a better policy. Various attempts have been proposed to improve model accuracy by investigating high-capacity models (the model ensemble technique [27, 8] and better function approximators [15, 38]) or amending the policy optimization stage based on model bias [24, 39, 28, 20, 7, 57]. Even so, their resultant models are

*Corresponding authors: Fuchun Sun.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

just accurate in a local and relative sense, since the learning is conditional on a fixed number of state-action tuples explored by the policy at the current step, other than the full transition dynamics of the environment. Indeed, it is tricky to determine how much we should explore. Insufficient exploration would trap the model and the following policy optimization, whereas excessive newly-encountered state-action pairs would confuse the model and later cause policy chattering. To derive a "truly" accurate model, we need a smarter scheme to choose different numbers of explorations at different times, instead of the unchanged setting in current methods.

From an optimization point of view, the thinking of how to derive an accurate model for MBRL in a global sense also motivates us to investigate the monotonicity of the optimization target (*i.e.* the return of the learned model and policy in MBRL), which, unfortunately, is less explored and not well guaranteed in current works. However, discussing the monotonicity guarantee for MBRL is challenging by any means, arising from the coupling of the model learning and policy optimization processes. Although there has been recent interest in related subjects, most of the theoretical works seek to characterize the monotonicity in terms of a fixed model of interest [49, 34, 20, 12, 28], which does not naturally fit our case when the model is dynamically shifted.

In this paper, we study how to guarantee the optimization monotonicity theoretically, upon which we then develop an event-triggered strategy that learns the model from a dynamically-varying number of explorations. In particular, we interestingly find that the lower bound of the performance improvement between two adjacent alternation steps in MBRL is dependent on the one-step model accuracy plus the constraint of the model shifts under certain mild assumptions. This discovery encourages us to formulate a constrained optimization problem, in order to permit positive performance improvement and thus the optimization monotonicity for MBRL. We also give a feasible solution example to show that dynamical alternation between model learning and policy exploration does benefit performance monotonicity. To resolve the constrained optimization problem, we design a simple but effective algorithm CMLO (Constrained Model-shift Lower-bound) equipped with an event-triggered mechanism. This mechanism first estimates whether the model shifts meet the constraint and then decides when to train the model.

We evaluate CMLO on several continuous control benchmark tasks. The results show that CMLO learns much faster than other state-of-the-art MBRL methods and yields promising asymptotic performance compared with the model-free counterparts. Note that our optimization framework is general and can be applied to different backbones of policy optimization algorithms, which is also ablated in our experiments.

2 Related works

Model-based reinforcement learning methods have shown great potential for sequential decision-making both in simulation and in the real world due to their sample efficiency [9, 21]. Generally, these MBRL algorithms can be grouped into several categories to highlight the range of uses of predictive models [55]. And our work falls into the Dyna-style category. In Dyna-style algorithms, training alternates between model learning under policy iterations with the real environments, and policy optimization using the model rollouts [50–52, 13]. Many attempts have been devoted to improving these two stages.

For model learning stage, previous main concerns are function approximators and training objectives. The dynamics approximator has advanced from Gaussian processes [26, 9], time-varying linear dynamics [29, 30] to neural network predictive models [15, 38]. And training objectives vary from Mean Square Error (MSE) [38, 34], Negative Log Likelihood (NLL) [8, 20], *etc*. Moreover, the deep ensemble technique is appealing for improving the robustness to model error. Our method adopts an ensemble of probabilistic networks similarly as in [8, 20].

The policy optimization stage allows Dyna-style algorithms to leverage various off-the-shelf model-free methods, such as SAC [16], TRPO [44], and TD3 [14]. Much owing to the progress of model-free methods, our method is to invoke any reasonable optimization oracle for the empirical models, rather than entangle a particular policy optimization algorithm.

A consensus of MBRL is that a smart policy requires an accurate model. However, model bias cannot be eliminated because the state-action distribution of the samples in the model learning stage and policy optimization stage is quite different. Many prior works attend to this distribution mismatch

issue and then tailor the data used for policy optimization according to the model bias. Janner et al. [20], Buckman et al. [7] encourage truncated rollout lengths. Besides, the ratio of real to model-generated data can be dynamically tuned according to the model uncertainty [24, 39, 28]. Yu et al. [57] penalizes rewards by the model uncertainty. Our method incorporates the truncated model rollouts mechanism. Moreover, we further explore how real interactions affect overall performance which is rarely studied before. We construct an event-triggered mechanism to cope with overfitting in a small-data regime and suffering generalization error when facing a drastic distribution shift.

Monotonic improvement guarantee has been a fundamental concern in both model-free and model-based avenues. In MFRL methods, both CPI [22] and TRPO [44] can be understood as approximating and optimizing the performance gap by forcing the new policy to be not too far away from the current policy. However, in Model-based settings, their trust-region constraints cannot directly be satisfied because the policies highly depend on the randomness of the models. While constructing such a bound for performance gap is straightforward, it has not been explored in previous MBRL theorectical analyses, instead they [49, 34, 20, 12, 28] turn to bound the discrepancy between returns under a model and those in the real environment. Although they could guarantee that the lower bound of policy performance improves under a certain model, this guarantee may face several issues regarding model updating. In contrast, we construct the performance difference scheme for MBRL algorithms and perform monotonicity analysis under this scheme.

Another line of theoretical works focus on regret bounds [10, 23] or sample complexity properties [1, 5, 47], focusing on the convergence performance and sample complexity for model-based approaches.

3 Preliminaries

Markov Decision Process A discounted Markov Decision Process (MDP) is a quintuple $M = (\mathcal{S}, \mathcal{A}, P_M, r_M, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ represents the action space, P_M denotes the transition function, $r : \mathcal{S} \times \mathcal{A} \to [-R, R]$ stands for the reward function, and $\gamma \in (0, 1)$ is the discount factor. For a fixed policy π and model M, we define $V_M^\pi(\mu)$ as the return of the model M with the starting state distribution μ, and $V^\pi(\mu)$ denotes the returns under the real environment,

$$V_M^\pi(\mu) = \mathop{\mathbb{E}}_{\substack{a_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim P_M(\cdot|s_t, a_t)}} \Big[\sum_{t=0}^{\infty} \gamma^t r_M(s_t, a_t) | \pi, s_0 \Big]. \tag{3.1}$$

We make a mild assumption that the model M to be identical to the real MDP except the transition function. Let $d_{M_i}^{\pi_k}(s, a; \mu)$ denote the visitation probability s, a when starting at $s_0 \sim \mu$ and following π_k under the dynamics P_{M_i}. We will omit it as $d_{M_i}^{\pi_k}$ henceforth for brevity. Besides, we denote $\mathcal{M}$ as a (parameterized) family of models of interest, and let Π be a family of policies.

Generative Model Many previous works [31, 25, 1] focus on a stylized generative model. By assuming an access to the generative model, we collect N samples for each state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$: $s_{s,a}^i \overset{i.i.d}{\sim} P(\cdot|s, a)$ which allows us to construct an empirical model defined as follows:

$$\forall s' \in \mathcal{S}, \quad \hat{P}(s'|s, a) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{s_{s,a}^i = s'\}. \tag{3.2}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function. This leads to an empirical MDP $\hat{M} = (\mathcal{S}, \mathcal{A}, \hat{P}, r, \gamma)$.

4 Monotonic improvement under model shifts

This section provides theoretical analyses for monotonic improvement of MBRL, factoring in the interdependence between policies and models. We first construct a general scheme for a non-decreasing performance guarantee and follow it up by characterizing the lower bound when shifting the model. Towards a non-negative lower bound, we restrict the model shifts and then obtain a refined bound. These discoveries encourage us to translate the bound maximization to a constrained optimization problem to permit monotonicity. By deriving an instance solution under the generative model setting, we demonstrate the merits of the dynamic model learning interval.

4.1 Monotonic improvement with policy optimization oracle

Our goal is to construct a general recipe for a monotonicity guarantee. Naturally, we seek to build a performance difference scheme for model-based algorithms.

Definition 4.1 (**Performance Difference Bound Scheme**). *$V^{\pi_i|M_i}(\mu)$ denotes the return of the policy $\pi_i \in \Pi$ in the real environment, whereas π_i is derived from the dynamical model $M_i \in \mathcal{M}$. Then, the lower bound on the true return gap of π_1 and π_2 can be stated in the form,*

$$V^{\pi_2|M_2}(\mu) - V^{\pi_1|M_1}(\mu) \geq C. \tag{4.1}$$

Such a statement guarantees that the policy allows non-decreasing performance in the real environment when C is non-negative.

Although there has been interest in non-decreasing performance guarantee, previous works [34, 20] commonly derive under a "discrepancy bound" scheme disregarding the model shifts (*i.e.* $M_1 = M_2$). Their results imply that once a policy update $\pi_1 \to \pi_2$ increases the returns under the same model ($V^{\pi_2}_{M_1} > V^{\pi_1}_{M_1}$), the lower bound on the policy performance evaluated in the real environment improves accordingly, $\inf\{V^{\pi_2|M_1}\} > \inf\{V^{\pi_1|M_1}\}$.

When the model shift is introduced, establishing the performance difference bound turns out to be rather difficult, mainly due to the coupling of the policy optimization and model learning: the estimated model is generated from the policy explorations, while the policy derives from the model rollouts. Hence, we need to consider the performance gap arising from the integration of the two stages, which, unfortunately, has never been explored before. Since the performance of the policy with a fixed model is already well guaranteed, it is natural to make the following assumptions.

Assumption 4.2 (**Policy Optimization Oracle**). *The policy optimization oracle is defined as the one that takes as input a model M and returns a ϵ_{opt}-optimal policy π satisfying: $V^*_M(\mu) - \epsilon_{opt} \leq V^\pi_M(\mu) \leq V^*_M(\mu)$. We assume that our policy optimization stage always meets the policy optimization oracle given its corresponding estimated model M.*

Such assumption usually holds in practice when we explore existing off-the-shelf model-free algorithms [43, 1, 16, 44]. With this assumption, we can focus directly on the eventual performance difference under the real environment encountering the model updating.

The bound C we seek can be expressed in terms of two kinds of gaps: the inconsistency gap between the model and the environment, and the optimal returns gap between the two models. When a policy π_k samples in a model M_i, it will encounter states not consistent with those generated by the real environment. We denote this inconsistency by $\epsilon^{\pi_k}_{M_i} = \mathbb{E}_{s,a\sim d^{\pi_k}}[\mathcal{D}_{\mathrm{TV}}(P(\cdot|s,a)\|P_{M_i}(\cdot|s,a))]$. Besides, for each model $M_i \in \mathcal{M}$, a remarkable property is that there always exists a policy that maximizes the value function $V^\pi_{M_i}(\mu)$ [53]. Hence, we define the ceiling performance of model M as $V^*_{M_i}(\mu) = \sup_{\pi\in\Pi} V^\pi_{M_i}(\mu)$. We will omit μ henceforth for simplicity unless confusion exists. With these two terms well-defined, we now present our bound.

Theorem 4.3 (**Performance Difference Bound for Model-based RL**). *Let $M_i \in \mathcal{M}$ be the estimated models and π_i be the ϵ_{opt}-optimal policy for M_i. Recalling $\kappa = \frac{2R\gamma}{(1-\gamma)^2}$ where R is the bound of the reward function, we have the performance difference of π_1 and π_2 evaluated under the real environment be bounded as below,*

$$V^{\pi_2|M_2} - V^{\pi_1|M_1} \geq \kappa \cdot (\epsilon^{\pi_1}_{M_1} - \epsilon^{\pi_2}_{M_2}) + V^*_{M_2} - V^*_{M_1} - \epsilon_{opt}. \tag{4.2}$$

Proof. See Appendix B, Theorem B.1. □

This bound implies that if the model update $M_1 \to M_2$ can (1) shorten the divergence between the estimated dynamics and the true dynamics and (2) improve the ceiling performance on the model, it may guarantee overall performance improvement under the true dynamics.

4.2 Lower-bound optimization with model shift constraints

Theorem 4.3 provides a general performance difference bound that suggests models with higher ceiling performance and lower bias would raise the overall performance. However, finding a non-negative lower bound may face several issues regarding a drastic model shift in practice. On the one

hand, performing an abrupt model update could potentially lead to a tumble in ceiling performance and then fail to make incremental improvements. On the other hand, huge distribution divergence between model rollouts would confound the policy optimization stage and hamper its access to optimal policies. Thus, we shoot for refining the bound upon adding model shifts constraint.

First, we seek to further unfold the ceiling performance gap, which serves as the main building block toward the desired bound. During derivation, we additionally introduce the L-Lipschitz assumption.

Assumption 4.4 (**L-Lipschitzness of Value Function**). *We call a value function V_M^π on the estimated dynamical model M is L-Lipschitz w.r.t to some norm $\|\cdot\|$ in the sense that*

$$\forall s, s' \in \mathcal{S}, |V_M^\pi(s) - V_M^\pi(s')| \leq L \cdot |s - s'|. \tag{4.3}$$

We assume that our estimated model $M \in \mathcal{M}$ satisfies the Lipschiz character, inspired from [4, 34]. Under the L-Lipschitzness assumption, we can derive a bound for the ceiling return gap.

Theorem 4.5 (**Ceiling Return Gap under Model Shift**). *For an estimated model $M_i \in \mathcal{M}$, the ceiling return gap is bounded as:*

$$V_{M_2}^* - V_{M_1}^* \geq -\frac{\gamma}{1-\gamma} L \cdot \sup_{\pi \in \Pi} \mathbb{E}_{s,a \sim d_{M_2}^\pi} \Big[|P_{M_2}(\cdot|s,a) - P_{M_1}(\cdot|s,a)| \Big]. \tag{4.4}$$

Proof. See Appendix B, Theorem B.2. □

This conclusion reveals the connection between the ceiling return gap and models' disagreement. In a benign scenario, the term of ceiling performance gap in the RHS of Eq. 4.2 should be dominated by the term $\kappa \cdot (\epsilon_{M_1}^{\pi_1} - \epsilon_{M_2}^{\pi_2})$ when the model shift is sufficiently small. A sharp model shift, however, risks a massive reduction in ceiling performance that is hardly bridged by the other parts in Eq. 4.2, and therefore corrupts monotonicity. This inspires us to introduce the constraint of the model shift. We further refine our performance difference bound to better characterize the relationship between performance gap and model shift.

Theorem 4.6 (**Refined Bound with Constraint**). *Let policy $\pi_i \in \Pi$ denotes the ϵ_{opt} optimal policy under the dynamical model $M_i \in \mathcal{M}$, and σ_{M_1,M_2} be the constraint threshold for M_1 and M_2. Note that $L \geq \frac{R}{1-\gamma}$. Then we can refine performance difference lower bound under the model shifts constraint as,*

$$\begin{aligned} V^{\pi_2|M_2} - V^{\pi_1|M_1} \geq \kappa \cdot \Big\{ & \mathbb{E}_{s,a \sim d^{\pi_1}} \mathcal{D}_{\mathrm{TV}} \big[P(\cdot|s,a) \| P_{M_1}(\cdot|s,a) \big] \\ & - \mathbb{E}_{s,a \sim d^{\pi_2}} \mathcal{D}_{\mathrm{TV}} \big[P(\cdot|s,a) \| P_{M_2}(\cdot|s,a) \big] \Big\} - \frac{\gamma}{1-\gamma} L \cdot (2\sigma_{M_1,M_2}) - \epsilon_{opt}, \end{aligned} \tag{4.5}$$

$$s.t. \quad \mathcal{D}_{\mathrm{TV}}(P_{M_2}(\cdot|s,a) \| P_{M_1}(\cdot|s,a)) \leq \sigma_{M_1,M_2}, \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{4.6}$$

Proof. See AppendixB, Theorem B.3. □

Theorem 4.6 implies that policy π_2 is guaranteed to outperform policy π_1 once it makes the RHS in Eq. 4.5 greater than zero under the constraint Eq. 4.6. More specifically, to guarantee a non-decreasing performance, M_2 and π_2 should meet the following two requirements,

$$\mathop{\mathbb{E}}_{s,a \sim d^{\pi_2}} \Big[\sum_{s' \in \mathcal{S}} |P(s'|s,a) - P_{M_2}(s'|s,a)| \Big] \leq 2\epsilon_{M_1}^{\pi_1} - \frac{(1-\gamma)L}{R}(2\sigma_{M_1,M_2}) - \frac{2}{\kappa} \cdot \epsilon_{opt}, \tag{R1}$$

$$\mathcal{D}_{\mathrm{TV}}(P_{M_2}(\cdot|s,a) \| P_{M_1}(\cdot|s,a)) \leq \sigma_{M_1,M_2}, \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{R2}$$

R1 encourages us to alleviate model bias as much as possible. Moreover, when the policy π_1 samples too many states not encountered by the model M_1, it may lead to excessive generalization errors during model learning, i.e. the estimated model will suffer extrapolation error in these unexperienced regions, which further causes instability or crashes in the following derived sub-optimal policies. Thus, the introduction of the R2 constraint can help solve the above problem. Finally, we abstract these two requirements to a constrained optimization problem.

Proposition 4.7 (Constrained Lower-Bound Optimization Problem). *We reduce the issue of finding a non-negative C to the following constrained optimization problem. Here, π_i is still the sub-optimal policy under model M_i. The minimal objective in E.q. 4.7 leads to the maximum of C. Then the overall optimization problem can be formalized as,*

$$\begin{aligned} &\min_{\substack{M_2 \in \mathcal{M} \\ \pi_2 \in \Pi}} \mathop{\mathbb{E}}_{s,a \sim d^{\pi_2}} \Big[\sum_{s' \in \mathcal{S}} |P(s'|s,a) - P_{M_2}(s'|s,a)| \Big], \\ &\textit{s.t.} \quad \sup_{s \in \mathcal{S}, a \in \mathcal{A}} \mathcal{D}_{\mathrm{TV}}(P_{M_1}(\cdot|s,a) \| P_{M_2}(\cdot|s,a)) \leq \sigma_{M_1,M_2}. \end{aligned} \tag{4.7}$$

4.3 A feasible example for constrained optimization problem

We first remark that, Proposition 4.7 provides a useful guide for acquiring performance improvement through restricting the upcoming model shift into a safe zone. In this section, we give an instance for a feasible solution of the constrained optimization problem under the generative model setting [1, 31]. Specifically, we construct an empirical model M_1 from the N samples per state-action pair that stems from the generative model. Upon encountering another k samples on each state-action pair, we segue into the model updating stage, which outputs M_2 given these newly collected samples as input. Towards obtaining a non-decreasing performance, we yield the following feasible solution for model training interval through satisfying requirements R1 and R2.

Corollary 4.8. *Here, $vol(\mathcal{S})$ denotes the volume of the state coverage simplex. For simplicity, we denote $\delta_{M_i}(\cdot|s,a) = |P(\cdot|s,a) - P_{M_i}(\cdot|s,a)|$ for each model $M_i \in \mathcal{M}$. Under the generative model setting, with a probability larger than $1-\xi$, we can provide a non-negative C when given,*

$$k = \frac{2}{\epsilon^2} \log \frac{2^{vol(\mathcal{S})} - 2}{\xi} - N. \tag{4.8}$$

Here, $\epsilon = \delta_{M_1}(\cdot|s,a) - \frac{(1-\gamma)L}{R} \cdot (2\sigma_{M_1,M_2}) - \frac{(1-\gamma)^2}{R\gamma} \cdot \epsilon_{opt}$ and $\xi \in (0,1)$ is a constant.

Proof. See Appendix B, Corollary B.4. □

One can deduce from Corollary 4.8 that dynamically adjusting the model training interval according to the model bias and the model shifts constraints threshold does benefit monotonicity. Along with the model bias $\delta_{M_1}(\cdot|s,a)$ decaying, the model training interval k requires to be scaled up to obtain adequate newly encountered samples for training M_2. Besides, once gathering excessive samples, we risk violating the model shifts constraint, thus impairing performance. This instance supports the insight that determining "when to update your model" is vital for performance improvement and motivates a smarter scheme to choose different numbers of explorations at different times instead of the unchanged setting in current methods.

5 CLMO framework

It is nontrivial to tackle the proposed constrained optimization problem in Proposition 4.7, since one cannot directly assign a value to the optimization variable M_2. We decouple the optimization objective and the constraint as "how to train the model" and "when to train the model" through an event-triggered mechanism towards dynamic alternation.

Objective minimization. Minimizing the objective function involves improving the model accuracy. Specifically, we adopt the model-ensemble technique to reduce model bias. For practical implementation, the probabilistic models $\{\hat{f_{\phi_1}}, \hat{f_{\phi_2}}, \ldots, \hat{f_{\phi_K}}\}$ are fitted on shared but differently shuffled replay buffer $\mathcal{D}_e$, and the target is to optimize the Negative Log Likelihood (NLL).

Constraint estimation. The unobserved model M_2 makes the constraint function incalculable, and here we seek to design an estimator for it. Recall that $\mathcal{D}_{\mathrm{TV}}(P_{M_1}(\cdot|s,a)\|P_{M_2}(\cdot|s,a)) = \frac{1}{2}\sum_{s' \in \mathcal{S}} |P_{M_1}(s'|s,a) - P_{M_2}(s'|s,a)|$, then we can find that the distance arises from two parts, the state space coverage, and the models' disagreement. We estimate the policy coverage (state-space coverage) by computing the volume $vol(\mathcal{S}_{\mathcal{D}})$ of the convex closure $\mathcal{S}_{\mathcal{D}} = \big\{ \sum_{s_i \in \mathcal{D}} \lambda_i S_i :$

$\lambda_i \geq 0, \sum_i \lambda_i = 1\}$ constructed on the replay buffer $\mathcal{D}$. We exploit the average prediction error on these new samples data to estimate the disagreement on newly encountered data ($\Delta\mathcal{D}$) and get $\mathcal{L}(\Delta\mathcal{D}) = \mathbb{E}_{(s,a,s')\in\Delta_{\mathcal{D}}}\left[\frac{1}{K}\sum_{i=1}^{K}\|s' - \hat{f_{\phi_i}}(s,a)\|\right]$. Combining these two components, we can obtain an estimation for the model shift, *i.e.*, $vol(\mathcal{S}_{\mathcal{D}}) \cdot \mathcal{L}(\Delta\mathcal{D})$. This practical overestimation for the model shifts makes the constraint stay satisfied during objective optimization.

Event-triggered mechanism. We design an event-triggered mechanism to determine the occasion to pause collecting and turn to solve the optimization objective. Remark that although diverting to train models as long as not to violate the constraint is theoretically reasonable, we refrain from doing so in practice because performing an update on data with a minor shift in coverage and distribution is wasteful and may risk overfitting. Therefore, we trigger when the constraint boundary is touched to reduce computational cost and escape overfitting. The event-triggering mechanism is based on the following condition:

$$\frac{vol(\mathcal{S}_{\mathcal{D}\cup\Delta D(\tau)})}{vol(\mathcal{S}_{\mathcal{D}})} \cdot \mathcal{L}(\Delta D(\tau)) \geq \alpha. \tag{5.1}$$

Here, τ is the event-triggering time, and α is a given constant.

Policy optimization oracle. Clearly, we can leverage many model-free RL methods (SAC [16], TRPO [44], PPO [46] *etc.*) as our policy optimization oracle. Besides, we adopt a truncated short model rollouts technique to mitigate compounding error while encouraging model usage. Based on the fresh model rollouts, we perform the policy optimization oracle, employing SAC as an example.

Algorithm Overview. We briefly give an overview of our proposed CMLO in algorithm 1. Notably, the event-triggered mechanism subtly determines the occasion to perform model updating, promoting performance monotonicity and reducing computation load.

Algorithm 1: CMLO

initialize : policy π_θ, ensemble models $\hat{f_{\phi_1}}, \hat{f_{\phi_2}}, \ldots, \hat{f_{\phi_K}}$, environment buffer $\mathcal{D}_e$ and model buffer $\mathcal{D}_m$

repeat

  Sample $\Delta\mathcal{D}_e \sim \pi_\theta$ from real environment; add to $\mathcal{D}_e$

  Estimate model shifts by $vol(\mathcal{S}_{D_e})\mathcal{L}(\Delta\mathcal{D}_e)$

  if *Event-triggered condition (E.3) is reached* **then**

    Train all models $\hat{f_{\phi_1}}, \hat{f_{\phi_2}}, \ldots, \hat{f_{\phi_K}}$ on $\mathcal{D}_e$

  Perform h-step model rollouts using policy π_θ; add to $\mathcal{D}_m$

  Update π_θ on $\mathcal{D}_m$ through SAC [16]

until *the policy performs well in the environment*;

6 Experiments

Our experimental evaluation aims to investigate the following questions: (1) How well does our algorithm perform on standard reinforcement learning benchmarks compared to prior state-of-the-art model-based and model-free algorithms? (2) Does the performance with or without the constraint consistent with previous theoretical analyses?

6.1 Comparative evaluation

To illustrate the effectiveness of our method, we contrast several popular model-based and model-free baselines. Model-free counterparts include: (1) SAC [16], the state-of-the-art in terms of asymptotic performance. (2) PPO [46] that explores monotonic improvement as well. Model-based baselines include: (3) PETS [8], which employs models directly for planning, different from the Dyna-style. (4) SLBO [34], that explores monotonicity under the discrepancy bound scheme. (5) MBPO [20], that employs a similar design of model ensemble technique (ensemble of probabilistic dynamics networks) and policy optimization oracle (SAC) as we do. (6) AutoMBPO [28], a variant of MBPO, that uses an automatic hyperparameter controller to tune the model-training frequency but suffers from high pre-training cost and lacks theoretical analysis on parameters rationality.

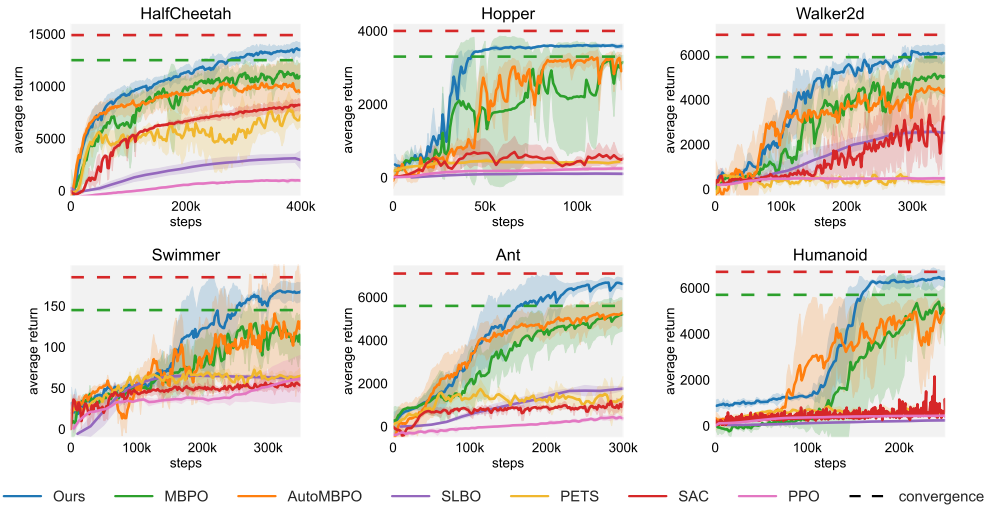


Figure 1: Comparison of learning performance on continuous control benchmarks. We evaluate each algorithm on the standard 1000-step versions. Solid curves indicate the average performance among seven trials under different random seeds, while the shade corresponds to the standard deviation over these trails. The dashed lines are the asymptotic performance of SAC (at 5M steps) and MBPO.

We evaluate CMLO and these baselines on six continuous control tasks in OpenAI Gym [6] with the MuJoCo [54] physics simulator, including HalfCheetah, Hopper, Walker2d, Swimmer, Ant, Humanoid. For fair comparison, we adopt the standard full-length version of these tasks and align the same environment settings.

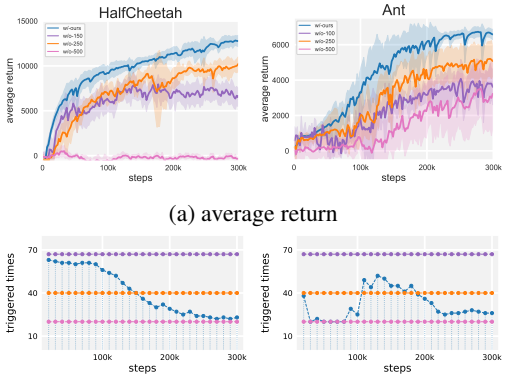
Figure 1 shows the learning curves of all compared methods, along with the asymptotic performance. These results show that our algorithm is far ahead of the model-free method in terms of sample efficiency, coupled with an asymptotic performance comparable to that of the state-of-the-art model-free counterparts SAC. Compared to model-based baselines, our method gains faster convergence speed and better eventual performance. Notably, credit to the event-triggered mechanism, our method enjoys a more stable training curve. The better monotonic property of the learning curve agrees with our previous analyses.

6.2 Ablation studies

Next, we make ablations and modifications to our method to validate the effectiveness and generalizability of the mechanism we devised.

The necessity of event-triggered mechanism. To verify the necessity of event-triggered mechanism, we compare to three unconstrained cases (given fixed model training interval) under two environments. The training curve and triggered times are shown in Figure 2. Clearly, our mechanism improves the performance while reducing the total times of model training. Besides, we notice that the performance is comparable to other MBRL baselines (MBPO *etc.*) when fixing our model training interval at 250. Still it performs worse than that equipped with a smart mechanism to decide whether to train the model at current exploration step.

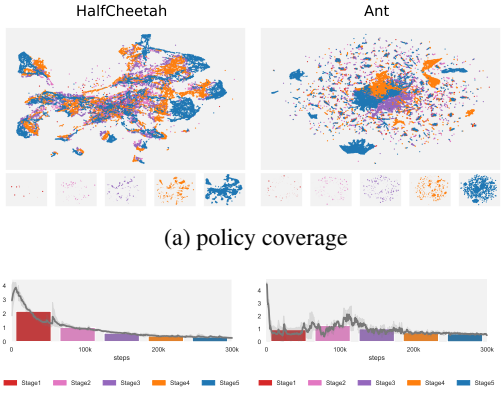
To better understand why the event-triggered mechanism brings up our outperformance, we asses its main bricks. Model shift, which reflects the current ability to digest new data, is the basis of triggered condition. And we estimate model shift from two parts, policy coverage and prediction error. In Figure 3 we observe that, gradually, as the training progresses, the policy coverage increases, which reflects our policy has new explorations at every stage without falling into a local optimum prematurely. Also, the prediction error gradually decreases, which implies that our estimated dynamics come closer to the true dynamics in the explored region. Figure 4 implies that the number of samples required to hit the constraint tends to grows and the model training frequency then goes down. This is also consistent with our intuition that, as the model refines and the exploration novelty fades, then the sample size required to reach a certain level of model shifts grows up. The result agrees with our



(a) average return

(b) triggered times

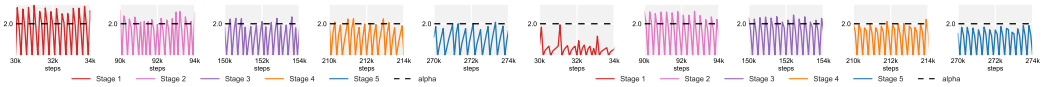
Figure 2: Ablation on event-triggered mechanism. These experiments are average over 5 random seeds. (a) shows the average return with or without event-triggered mechanism in HalfCheetah and Ant benchmarks. (b) shows the number of triggered times per 10k step.



(a) policy coverage

(b) average prediction error

Figure 3: Visualization of the policy coverage and prediction error on HalfCheetah and Ant. Each stage contains 60k steps. (a) implies that policy coverage expands over stages. (b) shows the prediction error over newly collected data. The bars are average values of each stage.



(a) Halfcheetah

(b) Ant

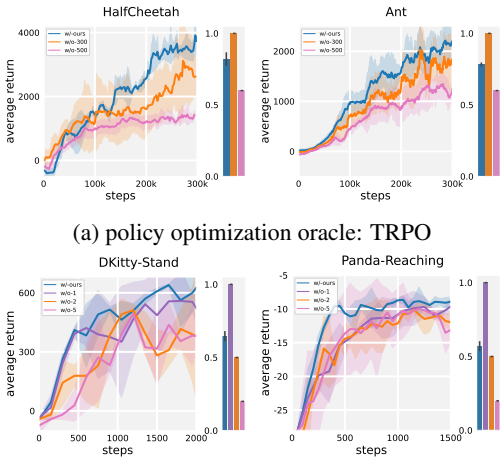
Figure 4: Visualization of our event-triggered mechanism on HalfCheetah and Ant benchmarks. Solid lines show the model shifts estimation and dotted lines reveals the threshold of triggered condition. Note that here we apply log value.

theoretical analyses which describe that a dynamic alternation subject to the model shifts constraint does benefit to monotonicity rather than an assigned one.

The generalizability of event-triggered mechanism. We further investigate the generalizability of our proposed mechanism through ablation on policy optimization oracle, and results are shown in Figure 5.

1) under Dyna-style. We adopt TRPO [44] as the policy optimization oracle and test the performance with or without event-triggered mechanism in Halfcheetah and Ant benchmarks. Observably, our mechanism can effectively guarantee the overall performance improvement and alleviate the local optimization issue.

2) jumping off Dyna-style. We utilize iLQR [3] and then conduct ablation experiments in the DKitty-Stand [2] and Panda-Reaching [19] scenarios. The result shows that adding event-triggered mechanism has a comparable asymptotic performance as the most-frequently triggered case but enjoys a lower computation cost.



(a) policy optimization oracle: TRPO

(b) policy optimization oracle: iLQR

Figure 5: Ablation on generalizability via various policy optimization oracles. Line plots reflect the average return and the standard deviation over 5 trails. Bar plots indicate the total triggered times and the y-axis is scaled to [0,1].

7 Conclusion

We have investigated the role of the decision on "when to update the model" in joint optimization procedures through theoretical and empirical lens. We devise a general novel scheme for exploring the monotonicity of MBRL methods, distinguished from the existing discrepancy bound scheme. We then derive lower bounds under the scheme, suggesting that models with higher ceiling performance and lower bias guarantee a non-decreasing performance evaluated in the real environment. An effective constrained optimization problem comes from the follow-up refined bound to seek a non-negative lower bound. Further, the instance under the generative model setting further verifies the effectiveness of learning models from a dynamically varying number of explorations. The algorithm CMLO, stemming from these analyses, has asymptotic performance rivaling the best model-free algorithms and boasts better monotonicity. Further ablation studies reveal that the proposed mechanism scales to various policy optimization oracles and benefits computation cost reduction. Currently, we observed empirically that the event-triggered condition is usually related to specific environments, which will cost a little time for tuning. Thus, one direction that merits further investigation is to construct the dual problem of our constrained optimization problem for better exploring optimality and monotonicity.

Acknowledgments and Disclosure of Funding

This work was jointly supported by the Sino-German Collaborative Research Project "Crossmodal Learning" (NSFC 62061136001/ DFG TRR169), the CAS Project for Young Scientists in Basic Research (Grant No.YSBR-040), the National Natural Science Foundation of China (No.62006137) and Beijing Outstanding Young Scientist Program (No.BJJWZYJH012019100020098). The authors would also like to thank the anonymous reviewers for their careful reading and their many insightful comments.

References

[1] Alekh Agarwal, Sham Kakade, and Lin F Yang. Model-based reinforcement learning with a generative model is minimax optimal. In *Conference on Learning Theory*, 2020.

[2] Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. Robel: Robotics benchmarks for learning with low-cost robots. In *Conference on robot learning*, 2020.

[3] Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable mpc for end-to-end planning and control. In *Advances in Neural Information Processing Systems*, 2018.

[4] Kavosh Asadi, Dipendra Misra, and Michael Littman. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, 2018.

[5] Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert Kappen. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine Learning*, 91 (3):325–349, 2013.

[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] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*.

[8] 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*.

[9] Marc Peter Deisenroth, Carl Edward Rasmussen, and Dieter Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. *Robotics: Science and Systems VII*, 7: 57–64, 2011.

[10] Yonathan Efroni, Nadav Merlis, Mohammad Ghavamzadeh, and Shie Mannor. Tight regret bounds for model-based reinforcement learning with greedy policies. In *Advances in Neural Information Processing Systems*.

[11] Ying Fan and Yifei Ming. Model-based reinforcement learning for continuous control with posterior sampling. In *International Conference on Machine Learning*, 2021.

[12] Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, 2017.

[13] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

[14] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

[15] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *ICML Workshop on Data-Efficient Machine Learning Workshop*, 2016.

[16] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[17] Wilhelmus PMH Heemels, Karl Henrik Johansson, and Paulo Tabuada. An introduction to event-triggered and self-triggered control. In *2012 ieee 51st ieee conference on decision and control (cdc)*, pages 3270–3285. IEEE, 2012.

[18] Todd Hester, Michael Quinlan, and Peter Stone. Rtmba: A real-time model-based reinforcement learning architecture for robot control. In *International Conference on Robotics and Automation*, 2012.

[19] Stephen James, Marc Freese, and Andrew J. Davison. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019.

[20] 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*, 2019.

[21] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[22] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, 2002.

[23] Sham Kakade, Akshay Krishnamurthy, Kendall Lowrey, Motoya Ohnishi, and Wen Sun. Information theoretic regret bounds for online nonlinear control. In *Advances in Neural Information Processing Systems*, 2020.

[24] Gabriel Kalweit and Joschka Boedecker. Uncertainty-driven imagination for continuous deep reinforcement learning. In *Conference on Robot Learning*, 2017.

[25] Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine learning*, 49(2):193–208, 2002.

[26] Jonathan Ko and Dieter Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90, 2009.

[27] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

[28] Hang Lai, Jian Shen, Weinan Zhang, Yimin Huang, Xing Zhang, Ruiming Tang, Yong Yu, and Zhenguo Li. On effective scheduling of model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

[29] Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, 2013.

[30] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[31] Gen Li, Yuting Wei, Yuejie Chi, Yuantao Gu, and Yuxin Chen. Breaking the sample size barrier in model-based reinforcement learning with a generative model. In *Advances in Neural Information Processing Systems*, 2020.

[32] Huiping Li and Yang Shi. Event-triggered robust model predictive control of continuous-time nonlinear systems. *Automatica*, 50(5):1507–1513, 2014.

[33] 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. In *International Conference on Learning Representations*, 2016.

[34] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations*, 2018.

[35] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2013.

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[37] Jun Morimoto and Christopher Atkeson. Minimax differential dynamic programming: An application to robust biped walking. In *Advances in Neural Information Processing Systems*, 2002.

[38] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *International Conference on Robotics and Automation*, 2018.

[39] Feiyang Pan, Jia He, Dandan Tu, and Qing He. Trust the model when it is confident: Masked model-based actor-critic. In *Advances in Neural Information Processing Systems*.

[40] Luis Pineda, Brandon Amos, Amy Zhang, Nathan O. Lambert, and Roberto Calandra. Mbrl-lib: A modular library for model-based reinforcement learning. *Arxiv*, 2021. URL https://arxiv.org/abs/2104.10159.

[41] Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

[42] pranz24. pytorch-soft-actor-critic. https://github.com/pranz24/pytorch-soft-actor-critic, 2018.

[43] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[44] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

[45] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

[47] Aaron Sidford, Mengdi Wang, Xian Wu, Lin F Yang, and Yinyu Ye. Near-optimal time and sample complexities for solving discounted markov decision process with a generative model. *arXiv preprint arXiv:1806.01492*, 2018.

[48] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[49] Wen Sun, Geoffrey J Gordon, Byron Boots, and J Bagnell. Dual policy iteration. In *Advances in Neural Information Processing Systems*.

[50] Richard S Sutton. Integrated architecture for learning, planning, and reacting based on approximating dynamic programming. In *International Conference on Machine Learning*, 1990.

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

[52] Richard S Sutton. Planning by incremental dynamic programming. In *International Conference on Machine Learning*, 1991.

[53] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, Cambridge, MA, 2018.

[54] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012.

[55] 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.

[56] Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J Weinberger. Inequalities for the l1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*, 2003.

[57] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [No] No potential negative societal impacts have been found yet.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 4 and Appendix B.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B and Appendix C.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix E and our provided code.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See in corresponding figure captions.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix E.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix E
   (b) Did you mention the license of the assets? [Yes] See Appendix E
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] Our work does not use existing data.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] No crowdsourcing/human subjects are used.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] No crowdsourcing/human subjects are used.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] No crowdsourcing/human subjects are used.

Appendices

A Sketch of Theoretical Analyses

Here, we present our sketch of theoretical analyses. We first construct a general scheme (Definition 4.1) for non-decreasing performance guarantee and follow it up by characterizing the lower bound under model shifts (Theorem 4.3). Towards seeking a non-negative lower bound, we restrict model shift and refine the bound (Theorem 4.6), then further reduce this issue to a constrained optimization problem (Proposition 4.7). With an instance under the generative model setting (Corollary 4.8), we demonstrate the merits of the dynamic model learning interval.

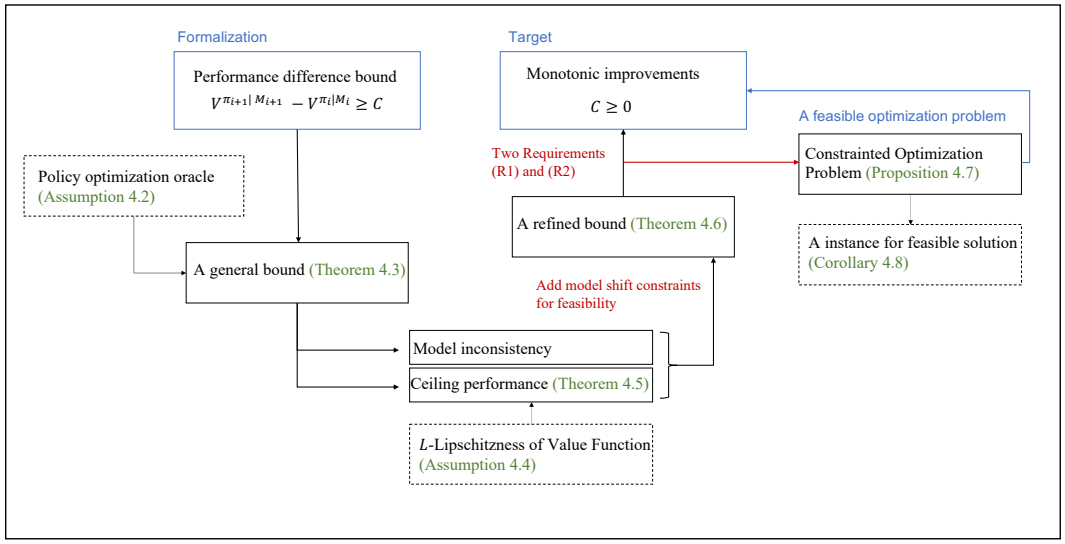


Figure 6: Theoretical sketch of CMLO

B Omitted Proofs

Theorem B.1 (Performance difference bound for Model-based RL)**.** *Let* $\epsilon_{M_i}^{\pi_j}$ *denote the inconsistency between the learned dynamics* P_{M_i} *and the true dynamics, i.e.* $\epsilon_{M_i}^{\pi_j} = \mathbb{E}_{s,a\sim d^{\pi_j}(s,a;\mu)}[\mathcal{D}_{\mathrm{TV}}(P(\cdot|s,a)\|P_{M_i}(\cdot|s,a))]$*, where* $d^{\pi}(s,a;\mu)$ *is the probability of visiting* s,a *after starting at state* $s_0 \sim \mu$ *and following* $\pi_j \in \Pi$ *thereafter under the true dynamics. Let policy* π_i *be the* ϵ_{opt}*-optimal policy under model* M_i*. Assume the performance discrepancy of policy* π *between the estimated model* M_1*and the true dynamics be approximated as* $V_{M_1}^{\pi_1}(\mu) - V^{\pi_1}(\mu) = \kappa \cdot \epsilon_{M_1}^{\pi_1}$*. Recall* $\kappa = \frac{2R\gamma}{(1-\gamma)^2}$*, then the performance gap between* π_1 *and* π_2 *evaluated in the true MDP can be bounded by:*

$$V^{\pi_2|M_2} - V^{\pi_1|M_1} \geq -\kappa \cdot (\epsilon_{M_2}^{\pi_2} - \epsilon_{M_1}^{\pi_1}) + V_{M_2}^* - V_{M_1}^* - \epsilon_{opt}$$

Proof. We overload notation $V^{\pi_i|M_i}$ and write V^{π_i} for simplicity.

$$V^{\pi_2} - V^{\pi_1} = \underbrace{V^{\pi_2} - V_{M_2}^{\pi_2}}_{L_1} + \underbrace{V_{M_2}^{\pi_2} - V_{M_1}^{\pi_1}}_{L_2} - \underbrace{(V^{\pi_1} - V_{M_1}^{\pi_1})}_{L_3}$$

We can bound $L_1 - L_3$ using Lemma C.2, and bound L_2 using the property of ϵ_{opt}-optimal.

For $L_1 - L_3$, with the performance gap approximation of M_1 and π_1, we apply Lemma C.2, and obtain: $L_1 - L_3 \geq -\kappa \cdot (\epsilon_{M_2}^{\pi_2} - \epsilon_{M_1}^{\pi_1})$.

We call a policy π ϵ-optimal under the dynamical model M, if $V_M^*(s) \geq V_M^{\pi}(s) \geq V_M^*(s) - \epsilon_{opt}$ for all $s \in \mathcal{S}$. Under the assumptions of black-box optimization oracle, we obtain:

$$L_2 \geq V_{M_2}^* - V_{M_1}^* - \epsilon_{opt}$$

Adding these two bounds together yields the desired result.

Remark that, when $V^{\pi_1}_{M_1} - V^{\pi_1}$ gets a value far away from $\kappa \cdot \epsilon^{\pi_1}_{M_1}$, it indicates the performance discrepancy that evaluated between the model M_1 and the environment is near to zero, which further indicates that the inconsistency between P_{M_1} and the true dynamics is quite small and thus the optimization process has reached a stopping point. □

Theorem B.2 (Ceiling return gap Under model shifts)**.** *For a dynamical model $M_i \in \mathcal{M}$, $V^*_{M_i}(\mu)$ denotes the maximal returns on dynamics P_{M_i}. Then the gap of optimal returns under these two models M_1, M_2 can be bounded as:*

$$V^*_{M_2} - V^*_{M_1} \geq -\frac{\gamma}{1-\gamma} L \cdot \sup_{\pi \in \Pi} \mathbb{E}_{s,a \sim d^\pi_{M_2}} \Big[|P_{M_2}(\cdot|s,a) - P_{M_1}(\cdot|s,a)| \Big]$$

Proof. Let $G^\pi_{M_i,M_j}(s,a)$ be the discrepancy between M_i and M_j on a single state-action pair (s,a), i.e. $G^\pi_{M_i,M_j}(s,a) = \mathbb{E}_{\tilde{s}' \sim P_{M_j}(\cdot|s,a)}[V^\pi_{M_j}(\tilde{s}')] - \mathbb{E}_{s' \sim P_{M_i}(\cdot|s,a)}[V^\pi_{M_j}(s')]$. We construct Z_k be the discounted return when using π to sample in model M_i for k steps and then in M_j for the rest with a starting point $s_0 = s$, that is,

$$Z_k = \mathop{\mathbb{E}}_{\substack{\forall t, a_t \sim \pi(\cdot|s_t) \\ \forall t<k, s_{t+1} \sim P_{M_i}(\cdot|s_t,a_t) \\ \forall t \geq k, s_{t+1} \sim P_{M_j}(\cdot|s_t,a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t,a_t) s_0 = s \right]$$

Base on this definition, we have that $V^\pi_{M_i}(s) = Z_\infty$ and $V^\pi_{M_j}(s) = Z_0$. Then, we can decompose $V^\pi_{M_i}(s) - V^\pi_{M_j}(s)$ into a sum of Z_k:

$$V^\pi_{M_i}(s) - V^\pi_{M_j}(s) = \sum_{k=0}^{\infty} (Z_{k+1} - Z_k)$$

We can find that, Z_{k+1} and Z_k only differ in their dynamical model used in the k-th step rollout. And we can rewrite them to be :

$$Z_k = r_k + \mathbb{E}_{s_k,a_k \sim \pi, P_{M_i}} \Big[\mathbb{E}_{\tilde{s}_{k+1} \sim P_{M_j}(\cdot|s_k,a_k)} \Big[\gamma^{k+1} V^\pi_{M_j}(\tilde{s}_{k+1}) \Big] \Big]$$
$$Z_{k+1} = r_k + \mathbb{E}_{s_k,a_k \sim \pi, P_{M_i}} \Big[\mathbb{E}_{s_{k+1} \sim P_{M_i}(\cdot|s_k,a_k)} \Big[\gamma^{k+1} V^\pi_{M_j}(s_{k+1}) \Big] \Big]$$

Here, r_k denotes the reward from the first j steps from the real environment. Combine the two equations above together and we get:

$$Z_{k+1} - Z_k = \gamma^{k+1} \mathbb{E}_{s_k,a_k \sim \pi, P_{M_i}} \left[\mathop{\mathbb{E}}_{\substack{s_{k+1} \sim P_{M_i}(\cdot|s_k,a_k) \\ \tilde{s}_{k+1} \sim P_{M_j}(\cdot|s_k,a_k)}} \Big[V^\pi_{M_j}(s_{k+1}) - V^\pi_{M_j}(\tilde{s}_{k+1}) \Big] \right]$$

Then, we can obtain the following conclusion by adding up all $Z_{k+1} - Z_k$:

$$V^\pi_{M_i} - V^\pi_{M_j} = \frac{\gamma}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s \sim d^\pi_{M_i} \\ a \sim \pi(\cdot|s)}} \left[\mathbb{E}_{s' \sim P_{M_i}(\cdot|s,a)} \big[V^\pi_{M_j}(s') \big] - \mathbb{E}_{\tilde{s}' \sim P_{M_j}(\cdot|s,a)} \big[V^\pi_{M_j}(\tilde{s}') \big] \right]$$

Here, $d^\pi_{M_i}$ denotes the distribution of state-action pair induced by policy π under the dynamical model M_i.

For a starting state $s_0 = s$, and two dynamical models M_2, M_1, with the definition of $G^\pi_{M_1,M_2}$ we have:

$$V^\pi_{M_2}(s) - V^\pi_{M_1}(s) = \frac{\gamma}{1-\gamma} \mathbb{E}_{s,a \sim d^\pi_{M_1}} \Big[G^\pi_{M_1,M_2}(s,a) \Big]$$

From the definition: $G^\pi_{M_1,M_2}(s,a) = \mathbb{E}_{\tilde{s}' \sim P_{M_2}(\cdot|s,a)}[V^\pi_{M_2}(\tilde{s}')] - \mathbb{E}_{s' \sim P_{M_1}(\cdot|s,a)}[V^\pi_{M_2}(s')]$.

In the case of deterministic dynamics: for clarity, we write $s' = M_i(s,a)$ instead of $s' \sim P_{M_i}(s'|s,a)$, then we rewrite $G_{M_1,M_2}(s,a)$ as: $G^\pi_{M_1,M_2}(s,a) = V^\pi_{M_2}(M_2(s,a)) - V^\pi_{M_2}(M_1(s,a))$, with the Lispchitzness, we have that $|G^\pi_{M_1,M_2}(s,a)| \leq L \cdot |M_2(s,a) - M_1(s,a)|$.

In the case of stochastic dynamics: when $L \geq \frac{R}{1-\gamma}$, we have

$$\begin{aligned}
|G^{\pi}_{M_1,M_2}(s,a)| &= |\mathbb{E}_{\tilde{s}'\sim P_{M_2}(\cdot|s,a)}[V^{\pi}_{M_2}(\tilde{s}')] - \mathbb{E}_{s'\sim P_{M_1}(\cdot|s,a)}[V^{\pi}_{M_2}(s')]| \\
&= |\sum_{\tilde{s}'\in\mathcal{S}}(P_{M_2}(s'|s,a) - P_{M_1}(s'|s,a))V^{\pi}_{M_2}(s')| \\
&\leq |\max_{s'} V^{\pi}_{M_2}(s')| \cdot |P_{M_2}(\cdot|s,a) - P_{M_1}(\cdot|s,a)| \\
&\leq L \cdot |P_{M_2}(\cdot|s,a) - P_{M_1}(\cdot|s,a)|.
\end{aligned}$$

Note that we require L to be larger than $\frac{R}{1-\gamma}$ in the infinite horizon settings. Previous work [11] has found that L has a dependence on $R \cdot H$ when equipped with a maximum horizon of H.

Thus, with the Lipschitzness of $V^{\pi}_{M_1}$ and $V^{\pi}_{M_2}$, we have that $|G^{\pi}_{M_1,M_2}(s,a)| \leq L \cdot |P_{M_2}(\cdot|s,a) - P_{M_1}(\cdot|s,a)|$.

$$\begin{aligned}
|V^{\pi}_{M_2}(s) - V^{\pi}_{M_1}(s)| &= \frac{\gamma}{1-\gamma} \cdot |\mathbb{E}_{s,a\sim d^{\pi}_{M_2}}[G^{\pi}_{M_1,M_2}(s,a)]| \\
&\leq \frac{\gamma}{1-\gamma} \cdot \mathbb{E}_{s,a\sim d^{\pi}_{M_2}}[|G^{\pi}_{M_1,M_2}(s,a)|] \\
&\leq \frac{\gamma}{1-\gamma} L \cdot \mathbb{E}_{s,a\sim d^{\pi}_{M_2}}[|P_{M_2}(\cdot|s,a) - P_{M_1}(\cdot|s,a)|]
\end{aligned}$$

Observe that $|\sup_x f(x) - \sup_x g(x)| \leq \sup_x |f(x) - g(x)|$, where f And g are real valued functions. This implies:

$$|V^{*}_{M_2}(s) - V^{*}_{M_1}(s)| = |\sup_{\pi\in\Pi} V^{\pi}_{M_2}(s) - \sup_{\pi\in\Pi} V^{\pi}_{M_1}(s)| \leq \sup_{\pi\in\Pi} |V^{\pi}_{M_2}(s) - V^{\pi}_{M_1}(s)|$$

Thus, we have that for starting state $s_0 \sim \mu$:

$$V^{*}_{M_2}(\mu) - V^{*}_{M_1}(\mu) \geq -\frac{\gamma}{1-\gamma} L \cdot \sup_{\pi\in\Pi} \mathbb{E}_{s,a\sim d^{\pi}_{M_2}}\Big[|P_{M_2}(\cdot|s,a) - P_{M_1}(\cdot|s,a)|\Big]$$

□

Theorem B.3 (Refined bound with constraints)**.** *Here, $\mathcal{S}$ denotes the state space simplex. Here, policy $\pi_i \in \Pi$ is the ϵ_{opt} optimal policy under the dynamical model $M_i \in \mathcal{M}$. For a dynamical model $M_i \in \mathcal{M}$, $V^{*}_{M_i}(\mu)$ denotes the maximal returns on dynamics P_{M_i}. We give the model shift constraints under the TV-distance. Then the performance difference bound can be refined under the model shift constraints as:*

$$\begin{aligned}
V^{\pi_2|M_2} - V^{\pi_1|M_1} \geq& \kappa \cdot (\mathbb{E}_{s,a\sim d^{\pi_1}} \mathcal{D}_{\mathrm{TV}}[P(\cdot|s,a)\|P_{M_1}(\cdot|s,a)] \\
&- \mathbb{E}_{s,a\sim d^{\pi_2}} \mathcal{D}_{\mathrm{TV}}[P(\cdot|s,a)\|P_{M_2}(\cdot|s,a)]) - \frac{\gamma}{1-\gamma} L \cdot 2\sigma_{M_1,M_2} - \epsilon_{opt} \\
\textit{s.t.} \quad & \mathcal{D}_{\mathrm{TV}}(P_{M_2}(\cdot|s,a)\|P_{M_1}(\cdot|s,a)) \leq \sigma_{M_1,M_2}, \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}
\end{aligned}$$

Proof. Let μ and v be two probability distributions on the configuration space $\mathcal{X}$, according to Lemma C.1, then we have $\mathcal{D}_{\mathrm{TV}}(\mu\|v) = \frac{1}{2}\sum_{x\in\mathcal{X}} |\mu(x) - v(x)|$.

Recall $\epsilon^{\pi}_{M_i}$ denote the discrepancy between the learned dynamics P_{M_i} and the true dynamics, i.e. $\epsilon^{\pi}_{M_i} = \mathbb{E}_{s,a\sim d^{\pi}}[\mathcal{D}_{\mathrm{TV}}(P(\cdot|s,a)\|P_{M_i}(\cdot|s,a))]$.

Under these definitions, we can yield the following intermediate outcome by applying the results from B.2 and B.1

$$\begin{aligned}
V^{\pi_2|M_2} - V^{\pi_1|M_1} \geq& -\kappa \cdot (\epsilon^{\pi_2}_{M_2} - \epsilon^{\pi_1}_{M_1}) + V^{*}_{M_2} - V^{*}_{M_1} - \epsilon_{opt} \\
\geq& -\kappa \cdot \Big\{\mathbb{E}_{s,a\sim d^{\pi_2}} \mathcal{D}_{\mathrm{TV}}[P(\cdot|s,a)\|P_{M_2}(\cdot|s,a)] \\
&- \mathbb{E}_{s,a\sim d^{\pi_1}} \mathcal{D}_{\mathrm{TV}}[P(\cdot|s,a)\|P_{M_1}(\cdot|s,a)]\Big\} \\
&- \frac{\gamma}{1-\gamma} L \cdot \sup_{\pi\in\Pi} \mathbb{E}_{s,a\sim d^{\pi}_{M_2}}\Big[|P_{M_2}(\cdot|s,a) - P_{M_1}(\cdot|s,a)|\Big] - \epsilon_{opt}
\end{aligned}$$

.

Recall the following constraint on model shift that we subject to:

$$\mathcal{D}_{\mathrm{TV}}(P_{M_2}(\cdot|s,a)\|P_{M_1}(\cdot|s,a)) \leq \sigma_{M_1,M_2}, \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}$$

Then, the ceiling performance difference can be further bounded as:

$$\begin{aligned}
V^*_{M_2} - V^*_{M_1} &\geq -\frac{\gamma}{1-\gamma} L \cdot \sup_{\pi\in\Pi} \mathbb{E}_{s,a\sim d^\pi_{M_2}} \Big[|P_{M_2}(\cdot|s,a) - P_{M_1}(\cdot|s,a)| \Big] \\
&\geq -\frac{\gamma}{1-\gamma} L \cdot \sup_{\pi\in\Pi} \mathbb{E}_{s,a\sim d^\pi_{M_2}} \Big[2 \sum_{s'\in\mathcal{S}_2} \frac{1}{2} |P_{M_2}(s'|s,a) - P_{M_1}(s'|s,a)| \Big] \\
&\geq -\frac{\gamma}{1-\gamma} L \cdot \sup_{\pi\in\Pi} \mathbb{E}_{s,a\sim d^\pi_{M_2}} \Big[2\mathcal{D}_{\mathrm{TV}}(P_{M_2}(\cdot|s,a)\|P_{M_1}(\cdot|s,a)) \Big] \\
&\geq -\frac{\gamma}{1-\gamma} L \cdot (2\sigma_{M_1,M_2})
\end{aligned}$$

And finally, we have the following refined bound:

$$\begin{aligned}
V^{\pi_2|M_2} - V^{\pi_1|M_1} \geq & \kappa \cdot (\mathbb{E}_{s,a\sim d^{\pi_1}} \mathcal{D}_{\mathrm{TV}}[P(\cdot|s,a)\|P_{M_1}(\cdot|s,a)] - \mathbb{E}_{s,a\sim d^{\pi_2}} \mathcal{D}_{\mathrm{TV}}[P(\cdot|s,a)\|P_{M_2}(\cdot|s,a)]) \\
& - \frac{\gamma}{1-\gamma} L \cdot (2\sigma_{M_1,M_2}) - \epsilon_{opt}
\end{aligned}$$

This refined bound is subject to the model shift constraint we set.

□

Corollary B.4. *Under the generative model setting, let model M_1 has already trained on N samples of each (s,a) pair and model M_2 on $N+k$ samples per (s,a) pair. Policy π_1 is the ϵ_{opt}-optimal policy on M_1 and so is π_2 on M_2. Recall $\epsilon = \delta_{M_1}(\cdot|s,a) - \frac{(1-\gamma)L}{R} \cdot (2\sigma_{M_1,M_2}) - \frac{(1-\gamma)^2}{R\gamma} \cdot \epsilon_{opt}$ and $\xi \in (0,1)$ is a constant. As the model is trained on true interaction samples, we can work out the amount of samples we need to satisfy the monotonic improvement requirements:*

$$k = \frac{2}{\epsilon^2} \log \frac{2^{vol(\mathcal{S})} - 2}{\xi} - N$$

Proof. For simplicity, we denote $\delta_{M_i}(s'|s,a) = |P(s'|s,a) - P_{M_i}(s'|s,a)|$.

Recall the lower bound for performance difference:

$$\begin{aligned}
C_{M_1,\pi_1}(M_2,\pi_2) = \frac{2R\gamma}{(1-\gamma)^2} \Big\{ & \mathop{\mathbb{E}}_{s,a\sim d^{\pi_1}} \big[\frac{1}{2} \sum_{s'\in\mathcal{S}_2} (\delta_{M_1}(s'|s,a) \big] \\
& - \mathop{\mathbb{E}}_{s,a\sim d^{\pi_2}} \big[\frac{1}{2} \sum_{s'\in\mathcal{S}_2} (\delta_{M_2}(s'|s,a) \big] \Big\} - \frac{\gamma}{1-\gamma} L \cdot (2\sigma_{M_1,M_2}) - \epsilon_{opt}
\end{aligned}$$

Towards this lower bound, we give the assumption that similar models derives similar sub-obptimal policies. To be specific, when M_1 and M_2 are close to each other in terms of L_1-norm distance at any transition pair (s,a) : $|P_{M_2}(\cdot|s,a) - P_{M_1}(\cdot|s,a)|_1 \leq \delta$, then the sub-optimal policies derived from them are close as well, *i.e.*, there exists α, subject to $|\pi_2(\cdot|s) - \pi_1(\cdot|s)|_1 \leq \alpha \cdot \delta$ for all transition pairs. Here, we show the feasibility of given the α in the control theory perspective. Let a trajectory $s_{1:N}, a_{1:N-1}$ be generated from the dynamical model M_1 that satisfies:

$$s_{t+1} = f_{M_1}(x_t, a_t, 0)$$

We call it a nominal trajectory. Here, f_{M_1} is a nonlinear function that represents the dynamics of model M_1. Then, we can formalize the inconsistency between M_2 and M_1 by disturbance, $w_i \in \mathcal{W}$. By entering w_i into f_{M_1} in a general nonlinear way we can get the dynamic of model M_2 as:

$$s_{t+1} = f_{M_1}(s_t, a_t, w_t)$$

Further, the deviations from the nominal trajectory can be calculated as when giving a disturbance sequence, $w_{1:N-1}$. Let δy denote the deviations of variable y.

$$\delta s_t = f_{M_1}(s_t + \delta x_t, u_t, +\delta u_t, w_t) - s_{t+1}$$

Assume that the deviations are computed with a linear feedback controller, that is,

$$\delta a_t = -K_t \delta s_t$$

Actually, we can utilize any reasonable linear controller. Here, we take the time-varying linear quadratic regulator as an instance for illustrating the rationality of our assumption on α. Based on the dynamic Riccati equation, we have the solution as:

$$\begin{aligned} K_t &= (R + B_t^T P_{t+1} B_t)^{-1} (B_t^T P_{t+1} A_t) \\ P_{t-1} &= Q_t + A_t^T P_t A_t - A_t^T P_t B_t (R + B_t^T P_t B_t)^{-1} (B_t^T P_t A_t) \end{aligned}$$

where $A_t = \partial f_{M_1} / \partial s|_{s_t, a_t, 0}$ and $B_t = \partial f_{M_1} / \partial a|_{s_t, a_t, 0}$. And, $Q_i \succeq 0$ and $R \succeq 0$ are state and input cost matrices. That is, we have a feasible solution for α. We then seek to explain the state-action distribution is similar thus we can use d^{π_1} as an approximation of d^{π_2}. First of all, we have the distance between two policies,

$$|\pi_2(\cdot|s) - \pi_1(\cdot|s)|_1 \leq \alpha \cdot \delta \triangleq \beta$$

Denote $\mathbb{P}_\pi^h$ as the state distribution resulting from π at time step h with μ as the initial state distribution. We consider bounding $|\mathbb{P}_h^{\pi_2} - \mathbb{P}_h^{\pi_1}|_1$ with $h > 1$.

$$\begin{aligned} &\mathbb{P}_h^{\pi_2}(s') - \mathbb{P}_h^{\pi 1}(s') \\ =&\sum_{s,a} \left(\mathbb{P}_{h-1}^{\pi_2}(s)\pi_2(a|s) - \mathbb{P}_{h-1}^{\pi_1}(s)\pi_1(a|s) \right) P(s'|s,a) \\ =&\sum_{s,a} \left(\mathbb{P}_{h-1}^{\pi_2}(s)\pi_2(a|s) - \mathbb{P}_{h-1}^{\pi_2}(s)\pi_1(a|s) + \mathbb{P}_{h-1}^{\pi_2}(s)\pi_1(a|s) - \mathbb{P}_{h-1}^{\pi_1}(s)\pi_1(a|s) \right) P(s'|s,a) \\ =&\sum_{s} \mathbb{P}_{h-1}^{\pi_2}(s) \sum_a \left(\pi_2(a|s) - \pi_1(a|s) \right) P(s'|s,a) + \sum_s \left(\mathbb{P}_{h-1}^{\pi_2}(s) - \mathbb{P}_{h-1}^{\pi_1}(s) \right) \sum_a \pi_1(a|s) P(s'|s,a). \end{aligned}$$

Apply absolute value on both sides, we then get:

$$\begin{aligned} \sum_{s'} |\mathbb{P}_h^{\pi_2}(s') - \mathbb{P}_h^{\pi_1}(s')| \leq& \sum_s \mathbb{P}_{h-1}^{\pi_2}(s) \sum_a |\pi_2(a|s) - \pi_1(a|s)| \sum_{s'} P(s'|s,a) \\ &+ \sum_s |\mathbb{P}_{h-1}^{\pi_2}(s) - \mathbb{P}_{h-1}^{\pi_1}(s)| \sum_{s'} \sum_a \pi_1(a|s) P(s'|s,a) \\ \leq& \beta + \|\mathbb{P}_{h-1}^{\pi_2} - \mathbb{P}_{h-1}^{\pi_1}\|_1 \leq 2\beta + |\mathbb{P}_{h-2}^{\pi_2} - \mathbb{P}_{h-2}^{\pi_1}|_1 = h\beta. \end{aligned}$$

Under the definition of d_μ^π, we have:

$$|d_\mu^{\pi_2} - d_\mu^{\pi_1}|_1 = |(1-\gamma) \sum_{h=0}^{\infty} \gamma^h \left(\mathbb{P}_h^{\pi_2} - \mathbb{P}_h^{\pi_1} \right)| \leq \beta\gamma/(1-\gamma).$$

Upon these analyses, we find that similar model derives similar policy, which invokes similar state-action distribution. Thus we can approximate state-action visitation density d^{π_2} by previous visitation density d^{π_1}. Also, we assume the same state space $\mathcal{S}$. Then, we get a approximation of $C_{M_1,\pi}(M_2, \hat{\pi})$ as following:

$$\begin{aligned} &\tilde{C}_{M_1,\pi_1}(M_2, \pi_2) \\ =&\frac{2R\gamma}{(1-\gamma)^2} \mathop{\mathbb{E}}_{s,a} \Big[\frac{1}{2} \sum_{s' \in \mathcal{S}} \left(\delta_{M_1}(s'|s,a) - \delta_{M_2}(s'|s,a) \right) \Big] - \frac{\gamma}{1-\gamma} L \cdot (2\sigma_{M_1,M_2}) - \epsilon_{opt} \\ =&\frac{\gamma}{(1-\gamma)} \Big\{ \frac{R}{1-\gamma} \mathop{\mathbb{E}}_{s,a} \big[\sum_{s' \in \mathcal{S}} \left(\delta_{M_1}(s'|s,a) - \delta_{M_2}(s'|s,a) \right) \big] - L \cdot \mathop{\mathbb{E}}_{s,a} \big[\sum_{s' \in \mathcal{S}} \left(\frac{1}{vol(\mathcal{S})} 2\sigma_{M_1,M_2} \right) \big] \\ &- \frac{(1-\gamma)}{\gamma} \mathop{\mathbb{E}}_{s,a} \big[\frac{1}{vol(\mathcal{S})} \sum_{s' \in \mathcal{S}} (\epsilon_{opt}) \big] \Big\} \end{aligned}$$

Thus, when meeting the following requirements for each (s, a) pair, we can guarantee the monotonic improvement for that $V^{\pi_2|M_2} - V^{\pi_1|M_1} \geq \tilde{C}_{M_1,\pi_1}(M_2, \pi_2) \geq 0$.

$$\delta_{M_2}(\cdot|s,a) \leq \delta_{M_1}(\cdot|s,a) - \frac{(1-\gamma)L}{R} \cdot (2\sigma_{M_1,M_2}) - \frac{(1-\gamma)^2}{R\gamma} \cdot \epsilon_{opt}$$

By applying Lemma C.3, the L_1 deviation of the empirical distribution $P_{M_2}(\cdot|s,a)$and true $P(\cdot|s,a)$ over $vol(\mathcal{S})$ distinct events from n samples is bounded by:

$$\Pr(|P(\cdot|s,a) - P_{M_2}(\cdot|s,a)|_1 < \epsilon) \geq 1 - (2^{vol(\mathcal{S})} - 2)\exp(-\frac{(N+k)\epsilon^2}{2})$$

Then for a fixed (s,a), with probability greater than $1-\xi$, we have:

$$|P(\cdot|s,a) - P_{M_2}(\cdot|s,a)|_1 \leq \sqrt{\frac{2}{N+k} \cdot \log \frac{2^{vol(\mathcal{S})} - 2}{\xi}}$$

Let $\epsilon = \delta_{M_1}(\cdot|s,a) - \frac{(1-\gamma)L}{R} \cdot (2\sigma_{M_1,M_2}) - \frac{(1-\gamma)^2}{R\gamma} \cdot \epsilon_{opt}$. Our requirements can be further shown in this form:

$$\sqrt{\frac{2}{N+k} \cdot \log \frac{2^{vol(\mathcal{S})} - 2}{\xi}} = \delta_{M_1}(\cdot|s,a) - \frac{(1-\gamma)L}{R} \cdot (2\sigma_{M_1,M_2}) - \frac{(1-\gamma)^2}{R\gamma} \cdot \epsilon_{opt}$$

Finally, with probability greater than $1-\xi$, we can guarantee the monotonic improvement when having:

$$k = \frac{2}{\epsilon^2} \log \frac{2^{vol(\mathcal{S})} - 2}{\xi} - N$$

□

C Toolbox

Lemma C.1 (Total variation distance)**.** *Let μ and v be two probability distributions on the configuration space $\mathcal{X}$. Then*

$$\mathcal{D}_{\mathrm{TV}}(\mu\|v) = \frac{1}{2}\sum_{x\in\mathcal{X}} |\mu(x) - v(x)|$$

Proof. Let $B = \{x \in \mathcal{X} : \mu(x) \geq v(x)\}$, and $A \subseteq \mathcal{X}$ be any event. Since $\mu(x) - v(x) < 0$ for any $x \in A \cap B^c$, we have

$$\mu(A) - v(A) \leq \mu(A \cap B) - v(A \cap B) \leq \mu(B) - v(B)$$

For all events A, $|\mu(A) - v(A)|\mu(B) - v(B)$, and the equality is achieved for $A = B$ or $A = B^c$. Thus, we get that

$$\mathcal{D}_{\mathrm{TV}}(\mu\|v) = \frac{1}{2}[\mu(B) - v(B) + v(B^c) - \mu(B^c)] = \frac{1}{2}\sum_{x\in\mathcal{X}} |\mu(x) - v(x)|$$

□

Lemma C.2 (Relationship between true returns and model returns)**.** *Let ϵ_M^π denote the inconsistency between the learned dynamics P_M and the true dynamics, $\mathbb{E}_{s,a\sim d^\pi}[\mathcal{D}_{\mathrm{TV}}(P(\cdot|s,a)\|P_M(\cdot|s,a))]$, where $d^\pi \doteq d^\pi(s,a;\mu)$ is the probability of visiting state-action pair (s,a) after starting at state $s_0 \sim \mu$ and following π thereafter under the true dynamics. Then the true returns can be represented as below:*

$$V^\pi(\mu) \geq V_M^\pi(\mu) - \frac{2R\gamma}{(1-\gamma)^2}\epsilon_M^\pi$$

Proof. Given policy π and dynamics $P(\cdot|s,a)$, we denote the density of state-action visitation after h steps from starting state $s_0 \sim \mu$ as $\rho_h^\pi(\mu;P) = \mathbb{E}_{s_0\sim\mu}[\rho_h^\pi(s_h,a_h|s_0;P)]$.

Then the discounted returns are bounded as:

$$\begin{aligned}
V^{\pi}(\mu) - V_M^{\pi}(\mu) &= \sum_{h=0}^{\infty} \mathbb{E}_{s,a\sim\rho_h^{\pi}(\mu;P)}[\gamma^h r(s,a)] - \sum_{h=0}^{\infty} \mathbb{E}_{s,a\sim\rho_h^{\pi}(\mu;P_M)}[\gamma^h r(s,a)] \\
&\geq -\sum_{h=0}^{\infty} \gamma^h |\mathbb{E}_{s,a\sim\rho_h^{\pi}(\mu;P)}[r(s,a)] - \mathbb{E}_{s,a\sim\rho_h^{\pi}(\mu;P_M)}[\gamma^h r(s,a)]| \\
&\geq -\sum_{h=0}^{\infty} \gamma^h \sum_{s,a\in\mathcal{S}\times\mathcal{A}} R|\rho_h^{\pi}(\mu;P) - \rho_h^{\pi}(\mu;P_M)| \\
&= -2R\cdot\sum_{h=0}^{\infty} \gamma^h \frac{1}{2} \sum_{s,a\in\mathcal{S},\mathcal{A}} |\rho_h^{\pi}(\mu;P) - \rho_h^{\pi}(\mu;P_M)| \\
&= -2R\cdot\sum_{h=0}^{\infty} \gamma^h \mathcal{D}_{\mathrm{TV}}(\rho_h^{\pi}(\mu;P)\|\rho_h^{\pi}(\mu;P_M))
\end{aligned}$$

Using the property of Markov chain TV distance bound, then,

$$\begin{aligned}
V^{\pi}(\mu) - V_M^{\pi}(\mu) &\geq -2R\cdot\sum_{h=0}^{\infty} \gamma^h \mathcal{D}_{\mathrm{TV}}(\rho_h^{\pi}(\mu;P)\|\rho_h^{\pi}(\mu;P_M)) \\
&\geq \sum_{h=1}^{\infty} -2R\cdot\gamma^h \Big\{ \mathcal{D}_{\mathrm{TV}}(\rho_{h-1}^{\pi}(\mu;P)\|\rho_{h-1}^{\pi}(\mu;P_M)) \\
&\quad + \mathbb{E}_{s,a\sim d^{\pi}}[\mathcal{D}_{\mathrm{TV}}[(P(\cdot|s,a)\|P_M(\cdot|s,a))]] + \mathcal{D}_{\mathrm{TV}}(\pi\|\pi) \Big\}
\end{aligned}$$

By plugging the results back, we then get,

$$V^{\pi}(\mu) - V_M^{\pi}(\mu) \geq -2R\sum_{h=0}^{\infty} \gamma^h h\cdot\epsilon_M^{\pi} = -\frac{2R\gamma}{(1-\gamma)^2}\cdot\epsilon_M^{\pi}$$

□

Lemma C.3 (Inequalities for the L_1 deviation of the empirical distribution)**.** *Let P be a probability distribution on the set $\mathcal{A} = \{1,\ldots,a\}$. For a sequence of samples $x_1,\ldots,x_m \sim P$, let $\hat{P}$ be the empirical probability distribution on $\mathcal{A}$ defined by $\hat{P}(j) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}(x_i = j)$. The L_1-deviation of the true distribution P and the empirical distribution $\hat{P}$ over $\mathcal{A}$ from m independent identically samples is bounded by,*

$$Pr(|P - \hat{P})|_1 \geq \epsilon) \leq (2^{|\mathcal{A}|} - 2)e^{-m\epsilon^2/2}.$$

Proof. For a probability distribution P on $\mathcal{A}$, we define

$$\pi_p = \max_{A\subseteq\mathcal{A}} \min(P(A), 1 - P(A)).$$

And for $p \in [0, 1/2)$, we define

$$\varphi(p) = \frac{1}{1-2p}\log\frac{1-p}{p}.$$

and, by continuity, set $\varphi(1/2) = 2$.

According to Weissman et al. [56], the L_1-deviation of the true distribution P and the empirical distribution $\hat{P}$ is bound by,

$$Pr(|P - \hat{P})|_1 \geq \epsilon) \leq (2^{|\mathcal{A}|} - 2)e^{-m\varphi(\pi_P)\epsilon^2/4}.$$

Firstly, for any P, we have

$$\pi_p = \max_{A\subseteq\mathcal{A}} \min(P(A), 1 - P(A)) \leq \max_{A\subseteq\mathcal{A}}(\frac{P(A) + 1 - P(A)}{2}) = 1/2.$$

and note that $\pi_P = 1/2$ when $P(A) = 1/2$.

Then, we claim that the function $\varphi(p)$ is strictly decreasing for $p \in [0, 1/2]$. Differentiating $\varphi(p)$ with respect to p yields

$$\varphi'(p) = \frac{1}{(1-2p)^2}\Big[-\frac{1-2p}{1-p} - \frac{1-2p}{p} + 2\log\frac{1-p}{p}\Big].$$

For $p \in (0, 1/2)$, there always exists $\frac{1}{(1-2p)^2} > 0$. Thus, to show that $\varphi'(p) < 0$ for $p \in (0, 1/2)$, it suffices to show that

$$g(p) = -\frac{1-2p}{1-p} - \frac{1-2p}{p} + 2\log\frac{1-p}{p} < 0.$$

The derivative of $g(p)$ is

$$g'(p) = (\frac{1}{1-p} - \frac{1}{p})^2 > 0, \quad p \in (0, 1/2).$$

Note that $g(1/2) = 0$, thus we have $\varphi'(p) < 0$ for $p \in (0, 1/2)$. And continuity arguments complete the claim for $p = 1/2$ and $p = 0$.

It is then no difficult to see that for any probability distribution P,

$$\varphi(\pi_P) \geq \varphi(1/2) = 2.$$

Therefore

$$Pr(|P - \hat{P})|_1 \geq \epsilon) \leq (2^{|\mathcal{A}|} - 2)e^{-m\epsilon^2/2}.$$

□

D Comparison with Prior Works

To begin with, an important fact is that the effect of model shifts on trajectories is drastic. For example, even when the system dynamics satisfy L- Lipschitz continuity, along with the policy and the initial state are the same, the difference in trajectories sampled in M_1, M_2 grows at e^{LH} with the length H of the trajectory [32]. As the model shifts decay, the trajectory discrepancy will also decrease sharply. It implies that model shift stays a substantial influence during the MBRL training process.

There are two main trends of local view analysis:

API [22] class. Their recipe for monotonicity analysis is $V^{\pi_{n+1}}(\mu) - V^{\pi_n}(\mu) \geq C(\pi_n, \pi_{n+1}, \epsilon_m)$. If policies update $\pi_n \to \pi_{n+1}$ could provide a non-negative $C(\pi_n, \pi_{n+1}, \epsilon_m)$, then the performance is guaranteed to increase. Here, $\epsilon_m = \max_{\pi\in\Pi, M\in\mathcal{M}} \mathbb{E}_{s,a\sim d^\pi}[\mathcal{D}_{TV}(P(\cdot|s,a)\|P_M(\cdot|s,a))]$. Most previous works [44, 22] were derived under model-free settings ($\epsilon_m = 0$). they use conservative policy iteration, for example, by forcing $\mathcal{D}_{TV}(\pi_n\|\pi_{n+1}) \leq \alpha$), then the state-action distribution are close as well $\mathcal{D}_{TV}(d^{\pi_n}\|d^{\pi_{n+1}}) \leq \frac{\alpha\gamma}{1-\gamma}$, so that they can optimize over their performance difference lemma, e.g., $C(\pi_n, \pi_{n+1}, 0) \approx \frac{1}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi_n}}[A^{\pi_n}(s,a)]$. When $\epsilon_m > 0$, this approximation $C(\pi_n, \pi_{n+1}, 0) \approx \frac{1}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi_n}}[A^{\pi_n}(s,a)]$ fails. It is non-trivial to apply the results to model-based settings.

DPI [49] tries to force π_{n+1} and π_n to be close, which will result in a high similarity of the data they sample. Thus, a risk arises from it, this approach would limit the growth of the policy exploration in the real environment, thereby leading the inferred models to stay optimized in a restrictive local area. For example, in the Humanoid environment, the agent struggles to achieve balance at the beginning of training. By then, an updated restricted policy will cause the exploration space to be limited in such an unbalanced distribution for a long time, and the learned model in such highly repetitive data will converge quickly with a validation loss be zero. However, the success trajectory has not been explored yet, implying that both the policy and the learned model will fall into a poor local optimum.

Besides, the definition of model accuracy (Eq.3) is a local view in DPI, i.e., $\hat{P}$ is δ-opt under d^{π_n}. If we replace model accuracy with a more general, global definition (for example, $\hat{P}$ is δ-opt under

$d^{\pi_{n+1}}$, or $\hat{P}$ is δ-opt under all (s, a, s') tuples), we find that the δ in (Eq. 3) will be large at the initial steps, making it difficult to obtain a local optimal solution in Theorem 3.1.

Finally, theoretical analysis in DPI can only guide the policy iteration process, while the update of the model is passive, which is different from our global view theory.

Discrepancy bound class [34, 20]. They mostly derive upon $V^{\pi_n}(\mu) \geq V_M^{\pi_n}(\mu) - C(\epsilon_m, \epsilon_\pi)$. As guaranteed in them, once a policy update $\pi_n \to \pi_{n+1}$ has improved returns under the same model M, i.e., $V_M^{\pi_{n+1}}(\mu) > V_M^{\pi_n}(\mu) + C(\epsilon_m, \epsilon_\pi)$, it would improve the lower bound on the performance evaluated in the real environment, i.e., $\inf\{V^{\pi_2|M}(\mu)\} > \inf\{V^{\pi_1|M(\mu)}\}$.

Their theory is based on a fixed model M, or an upper bound on the distribution shift of all models ϵ_m. It does not concern the change in model dynamics during updating, nor the performance varying due to the model shift. Moreover, The solution would be very coarse if only the upper bound of the model shift is given. Even worse, the given upper bound is likely to be too large, then it will fail to find a feasible solution for $V_M^{\pi_{n+1}}(\mu) - V_M^{\pi_n}(\mu) \geq C(\epsilon_m, \epsilon_\pi)$ in practice, thus making the monotonicity guarantee fails.

E Experimental Details

E.1 Environment Setup

We evaluate all algorithms on a set of MuJoCo [54] continuous control benchmark tasks. We adopt the standard full-length version of all these tasks. Among then, we truncate some redundant observations for Hopper, Ant and Humanoid as our model-based baselines (MBPO[20], AutoMBPO[28]) do. The details of the experimental environments are provided in Table 1.

Table 1: Overview on Environment settings. Here, θ_t denotes the joint angle at time t. and z_t denotes the height.

	State Space Dimension	Action Space Dimension	Horizon	Terminal Function
Hopper-v2	11	3	1000	$z_t \leq 0.7$ or $\theta_t \geq 0.2$
Swimmer-v2	8	2	1000	None
Walker2d-v2	17	6	1000	$z_t \geq 2.0$ or $z_t \leq 0.8$ or $\theta_t \leq -1.0$ or $\theta_t \geq 1.0$
HalfCheetah-v2	17	6	1000	None
Ant-v2	27	8	1000	$z_t < 0.2$ or $z_t > 1.0$
Humanoid-v2	45	17	1000	$z_t < 1.0$ or $z_t > 2.0$

The environment settings for the ablation study on the generalizability of event-triggered mechanism are presented in Table 2

Table 2: Overview on Environment settings in Ablation.

	State Space Dimension	Action Space Dimension	Horizon	Terminal Function
Kitty Stand	61	12	50	$u_{t,kitty} \leq 0$
Panda Reach	20	7	50	None

E.2 Baselines and implementation

MFRL algorithms. We compare to two state-of-the-art model-free baselines, SAC [16] and PPO [46]. The hyperparameters are kept the same as the authors'. Regarding the low sample efficiency of MFRL methods, we ran 5M steps for them, which is an order of magnitude more than in MBRL, to fairly evaluate the asymptotic performance of these MFRL algorithms. The implementation of SAC is based on the opensource repo (pranz24 [42], MIT License).

MBRL algorithms. As for model-based methods, we compare with several algorithms including PETS [8], SLBO [34], MBPO [20] and AutoMBPO [28]. Our algorithm CMLO is implemented based on the opensource toolbox for MBRL algorithms, MBRL-LIB [40] (MIT License). The implementation of SLBO mainly follows Wang et al. [55]. To ensure a fair comparison, we run CMLO and MBPO with the same network architectures and training configurations based on MBRL-LIB.

We report the asymptotic performance on six benchmark tasks in Table 3. Results show that our method has comparable asymptotic performance in each benchmarks to both MBRL and MFRL baselines. Each result is averaged over seven trials using different random seeds. For MBRL baselines, the performances on different tasks are capped at different timesteps when the learning curves come to converge, we choose 125k for Hopper, 350k for Walker2d and Swimmer, 400k for HalfCheetah, 300k for Ant and 250k for Humanoid.

Table 3: Comparative results. The results show the average and standard deviation on the **maximum average returns** among different trails.

		Hopper	Walker2d	Swimmer
MFRL (@5M steps)	SAC	4257.92 ± 100.23	7898.01 ± 563.45	195.60 ± 5.97
	PPO	3114.76 ± 1039.06	5740.75 ± 500.89	129.66 ± 9.78
MBRL	PETS	571.25 ± 71.14	1174.79 ± 471.39	92.61 ± 4.29
	SLBO	278.82 ± 65.83	3129.70 ± 154.16	71.02 ± 1.98
	AutoMBPO	3534.46 ± 77.53	6276.99 ± 1878.56	184.89 ± 58.84
	MBPO	2831.23 ± 1109.63	6285.64 ± 538.32	145.70 ± 18.14
	Ours	**3666.90 ± 22.71**	**7749.90 ± 523.27**	**185.14 ± 1.73**
		HalfCheetah	Ant	Humanoid
MFRL (@5M steps)	SAC	16015.64 ± 351.21	7105.49 ± 169.61	8036.12 ± 480.60
	PPO	6733.45 ± 1528.87	4427.39 ± 836.02	3068.95 ± 1600.89
MBRL	PETS	12023.84 ± 3340.02	3558.99 ± 140.76	1335.84 ± 292.27
	SLBO	3993.43 ± 127.17	2492.19 ± 92.02	644.16 ± 237.00
	AutoMBPO	12044.35 ± 1550.21	5792.35 ± 415.46	5780.14 ± 245.01
	MBPO	13171.53 ± 937.65	5894.45 ± 702.39	5905.68 ± 420.64
	Ours	**14623.45 ± 612.10**	**6798.39 ± 196.84**	**6967.54 ± 317.07**

E.3 Implementation details of CMLO

Modeling and learning the dynamical models. As inferred from the optimization objective, the minimization of the objective function can be achieved when we try to minimize the difference between M_2 and the real environment. To reduce model bias, we chose to use NLL as a loss function in our implementation, which has been shown an effective way to learn model dynamics. More specifically, CMLO adopts a bootstrap ensemble of dynamical models $\{\hat{f_{\phi_1}}, \hat{f_{\phi_2}}, \ldots, \hat{f_{\phi_K}}\}$. Specifically, each forward dynamical model f_{ϕ_i} approximates the transition function of the real environment, that is $\hat{s}_{t+1} \sim f_{\phi_i}(s_t, a_t)$. The probabilistic models are fitted on shared but differently shuffled replay buffer $\mathcal{D}_e$, and the target is to optimize the Negative Log Likelihood (NLL).

$$\mathcal{L}^H(\phi) = \sum_t^H [\mu_\phi(s_t, a_t) - s_{t+1}]^T \Sigma_\phi^{-1}(s_t, a_t)[\mu_\phi(s_t, a_t) - s_{t+1}] + \log\det \Sigma_\phi(s_t, a_t)$$

And the prediction for these ensemble models is, $\hat{s}_{t+1} = \frac{1}{K}\sum_{i=1}^{K} f_{\phi_i}(s_t, a_t)$. More details on network settings are presented in Table 4.

Model shifts estimation. Recall that we partition the incalculable model shifts into two components for estimation, one for state-space coverage and the other for model divergence.

- *state-space coverage*. State coverage (policy coverage) is the range of state spaces that our algorithm can explore in the real environment under the current policy π_i (derived from the learned model M_i). In the existing works, [1] defined the return set for two state sub-space as $\bar{R}_{ret} = \lim_{n\to\infty} R^n_{ret}(X, \bar{X})$, where $R^n_{ret}(X, \bar{X})$ means an n-step returnability from X to $\bar{X}$. Referring to this definition, the state coverage of π_i can be defined as $\mathcal{S}^{\pi_i}_{pc} : \forall s \in \mathcal{S}^{\pi_i}_{pc}, a \sim \pi_i(\cdot|s), s' \sim P(\cdot|s,a) \in \mathcal{S}^{\pi_i}_{pc}$. Besides, in the description of La Salle's Invariance Principle [2], we verify the equivalence of Invariant Set and state coverage. Intuitively, the Humanoid example in our response to your major concerns also shows that the variation of state coverage in the different training stages.

  We estimate the policy coverage (state-space coverage) by computing the volume $vol(\mathcal{S}_{\mathcal{D}})$ of the convex closure $\mathcal{S}_{\mathcal{D}}$ constructed on the replay buffer $\mathcal{D}$. Since estimation on the full historical experiences involves a huge computational burden, we instead sample N tuples (*e.g.* 1000 tuples) from the replay buffer upon each estimation. As for the convex hull, we first perform Principal Component Analysis on the states to reduce the dimension and then leverage the Graham-Scan algorithm to construct a convex hull of these N points, which only takes $\mathcal{O}(N \log N)$ for time complexity.

- *model divergence*. We estimate the model divergence by computing the average prediction error on newly encountered data. Upon it, we get the estimation for the model divergence from the K ensemble models, $\mathcal{L}(\Delta\mathcal{D}) = \mathbb{E}_{(s,a,s')\in\Delta_{\mathcal{D}}}\left[\frac{1}{K}\sum_{i=1}^{K}\|s' - \hat{f_{\phi_i}}(s,a)\|\right]$.

Event-triggered mechanism. Recall our proposed optimization problem:

$$\min_{\substack{M_2\in\mathcal{M}\\ \pi_2\in\Pi}} \mathop{\mathbb{E}}_{s,a\sim d^{\pi_2}}\Big[\sum_{s'\in\mathcal{S}}|P(s'|s,a) - P_{M_2}(s'|s,a)|\Big],$$
$$\textit{s.t.} \quad \sup_{s\in\mathcal{S},a\in\mathcal{A}} \mathcal{D}_{\mathrm{TV}}(P_{M_1}(\cdot|s,a)\|P_{M_2}(\cdot|s,a)) \le \sigma_{M_1,M_2}.$$

We design an event-triggered mechanism to determine the interval instant τ on the condition that the optimization problem is solved at step $t+\tau$. The mechanism is developed based on the difference between the current model M_1 (trained at step t) and the upcoming model M_2, which is estimated on the newly encountered data. If their model shifts reaches a certain value, it stands to reason that a new dynamic model is required to be trained. Thus, the event-triggered mechanism is based on the condition:

$$\frac{vol(\mathcal{S}_{\mathcal{D}_t\cup\Delta D(\tau)})}{vol(\mathcal{S}_{\mathcal{D}_t})}\cdot\mathcal{L}(\Delta D(\tau)) \ge \alpha.$$

Here, we adopt the fraction form for the triggered condition. Denominator $vol(\mathcal{S}_{\mathcal{D}\cup\Delta\mathcal{D}(\tau)})\cdot\mathcal{L}(\Delta\mathcal{D}(\tau))$ is used to obtain an estimation for the model shift, as detailed in Line 239-248. The numerator $vol(\mathcal{S}_{\mathcal{D}})$, on the one hand, is to reduce numerical errors; on the other hand, this fraction reflects the relative change of the policy coverage and model shift if we turn to train M_2 under different τ starting from M_1. This fraction reflects the current ability to digest new data. It can facilitate the setting of threshold, for we do not need to tune α once the policy coverage updates.

In CMLO, the condition estimation execute per F steps because excessively frequently estimation doesn't make huge difference but bring up the computation load. In order to reuse the result of intermediate computation, we apply a log value to the result of each estimation so that the log value condition function can be approximated by the sum of each estimation value within the interval. We additionally append a constant $\beta = 1.0$ for the penalty of accumulated interval steps.

$$\sum_{i=0}^{[\tau/F]}\log\Big(\frac{vol(\mathcal{S}_{\mathcal{D}_t\cup\Delta\mathcal{D}(Fi)})}{vol(\mathcal{S}_{\mathcal{D}_t})}\cdot\mathcal{L}(\Delta\mathcal{D}(Fi)) + \beta\Big) \ge \alpha$$

Remark 1: We observe from the event-triggered condition that the interevent time can be enlarged by increasing the threshold α, which implies more exploration samples will be collected by current policy and the optimization objective will be solved less frequently. In other words, the triggered threshold is environment-specific. Notably, the tuning load required for our event-triggered mechanism is not heavier than in those fixed settings, due to that only a hyperparameter α is introduced for model training frequency in CMLO, while those algorithms with fixed settings need to tune the fixed model training interval. We claim that it is crucial to dynamically adapt the numbers of explorations to update the model according to the current training and exploration status.

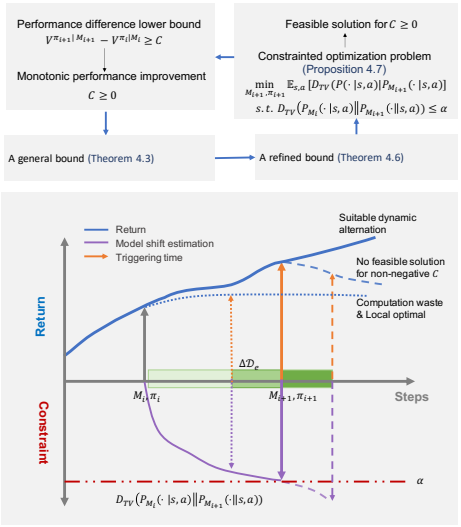


Figure 7: Illustration of the proposed event-triggered mechanism.

Remark 2: Zeno behavior [17] is common in the event-triggered mechanism, which leads to a most frequently triggering. The zeno behavior is naturallt alleviated by the introdution of β, which acts as a penalty for interevnet time. We additionaly add the lower and upper bounds of the interevent time to further keep the interval in a safe zone to aviod zeno behavior caused by some extreme situations. The minimal and maximal interevent time are given by $\inf\{\tau\} = \underline{T}$ and $\sup\{\tau\} = \overline{T}$.

Policy optimization and model rollouts. We can adopt a standard off-policy model-free RL method SAC [16] as the policy optimization oracle of CMLO. Another key concern is the way of training data generation. We adopt the truncated short model rollouts strategy inspired by some current MBRL works [20, 39, 28], which helps to escape from compounding error and encourage model usage. The main difference from the general rollouts mechanism is that we restrict our rollouts to be generated from fresh models, rather than using outdated models to generate rollout data as MBPO [20] and AutoMBPO [28] do in their implementations. Based on the dataset $\mathcal{D}_m$ of the fresh model rollouts, we perform SAC. In the policy evaluation step, SAC repeatedly apply a Bellman backup operator $\mathcal{T}^\pi$ to the soft Q-value, $\mathcal{T}^\pi Q^\pi(s,a) \triangleq r(s,a) + \gamma \mathop{\mathbb{E}}_{s'}[V^\pi(s')]$, and in the policy improvement step, SAC updates the policy according to $\pi = arg\min_{\pi \in \Pi} \mathbb{E}_{s_t \in \mathcal{D}_m} \mathcal{D}_{KL}(\pi(\cdot|s_t) \| \exp(Q^\pi - V^\pi))$.

Remark 3: The data distribution introduced by the outdated model has a drift from the data distribution introduced by the fresh model. This data shift will somehow mislead policy training and, in addition, the policy trained on the outdated model suffers from the limited sampling coverage during interacting with the real environment, which might in return cause the following models to fall into a local trap. In other words, the less the model differs from the real dynamics, the data it rolls out is more valuable.

E.4 Hyperparameters

Hyperparameters for Main Experiments. Table 4 lists the hyperparameters used in training CMLO. Here, $x \to y$ over epochs $a \to b$ denotes a threshold linear function, *i.e.*, at epoch t, $f(t) = \min(\max(x + \frac{t-a}{b-a} \cdot (y-x), x), y)$.

Hyperparameters for Ablation Studies. Note that other hyperparameters we do not mention below are the same as hyperparameter settings in Table 4.

(1) Policy optimization oracle: TRPO. For the TRPO part, the key parameters are listed below:

- Ant: $horizon = 1000, \gamma = 0.99, gae = 0.97, step\_size = 0.01, iterations = 40$
- HalfCheetah: $horizon = 1000, \gamma = 0.99, gae = 0.95, step\_size = 0.01, iterations = 40$

About Legend w/o-n, we use a data sampler with batchsize=20, thus we get $20 \times n$ real interactions during the model training interval. We compute the total triggered times and scale them to [0,1], which is shown in the bar plots.

Table 4: Hyperparameter Settings for CMLO.

	Hopper	Walker	Swimmer	HalfCheetah	Ant	Humanoid
epochs	300	125	300	300	400	250
environment steps per epoch	1000					
dynamical models network	Gaussian MLP with 4 hidden layers of size 200					
ensemble size	5					
model rollouts per policy update	400					
rollout schedule	$1 \rightarrow 15$ over epochs $20 \rightarrow 100$	1			$1 \rightarrow 25$ over epochs $20 \rightarrow 100$	$1 \rightarrow 25$ over epochs $20 \rightarrow 300$
SAC policy network	Gaussian with hidden size 512				Gaussian with hidden size 1024	
policy updates per step	40	20	20	10	20	20
event-triggered threshold α	1.2	3.0	2.0			2.5
computing frequency F	20	50				
minimal interevent time $\underline{T}$	150					
maximal interevent time $\overline{T}$	500					

(2) Policy optimization oracle: iLQR. Dynamical models network: Gaussian MLP with 3 hidden layers of size 200, batch size is 64, and the learning rate is 0.0001. For the iLQR part: $LQR\_ITER = 10, R = 0.001, Q = 1, horizon = 5$. About Legend w/o-n, we get n real interactions during the model training interval. And $\alpha = 0.5$ in w/-ours. We compute the total triggered times and scale them to $[0, 1]$, as shown in the bar plots.

E.5 Additional Ablation Study

Estimation on model shifts. The constraint function based on model shifts is incalculable due to the unobserved model M_2. We design a practical predictor for the model shifts by computing the state-space coverage and the model prediction error. Also, the decoupling of the constraint and the objective is enabled partly owing to the slightly overestimation over the true value. Recall our constraint function:

$$\mathcal{D}_{\mathrm{TV}}(P_{M_1}(\cdot|s,a)\|P_{M_2}(\cdot|s,a)) = \frac{1}{2}\sum_{s'\in\mathcal{S}}\Big[|P_{M_1}(s'|s,a) - P_{M_2}(s'|s,a)|\Big]$$

$$\leq \sum_{s'\in\mathcal{S}}\frac{1}{2}\Big[|P_{M_1}(s'|s,a) - P_M(s'|s,a)| + |P_{M_2}(s'|s,a) - P_M(s'|s,a)|\Big]$$

The updated dynamics P_{M_2} usually comes closer to the true dynamics than the previous one P_{M_1}, thus we turn to estimate $\sum_{s'\in\mathcal{S}}[|P_{M_1}(s'|s,a) - P_M(s'|s,a)|]$. Once the model M_2 is trained, we

can actually conduct a more realistic calculation for the constraint function. To show the connection between our predicted value in the absence of M_2 and the estimated value after obtaining M_2, we perform experiments on four environments and results are shown in Figure 8. The results demonstrate that our prediction is higher than the true estimation and their trends stand consistent, which indicates that our predictor is well designed. This gap helps to decouple the constrained optimization problem, and this overestimation part can be bridged by adjusting the α.

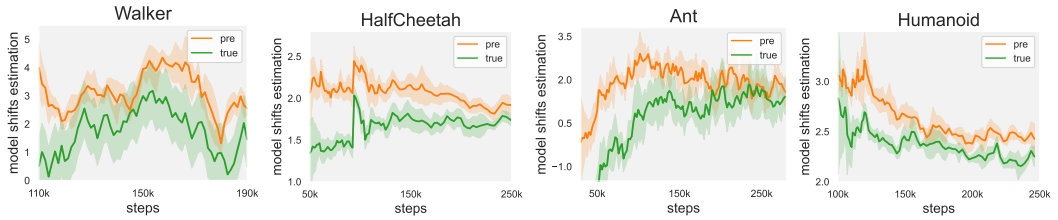


Figure 8: Estimation of model shifts in four environments. Green lines imply the estimation on model shifts after updating, which could be considered as a real value of our constraint function. And orange lines show our pre-estimation on model shifts before model updating.

Model accuracy. Figure 9 shows the one-step model error during the training under four benchmark tasks. We find that CMLO achieves a more accurate model than the state-of-the-art baseline MBPO. This result agrees with our insight that, a smarter scheme to choose different numbers of explorations at different steps instead of the unchanged setting in current methods, will promote a better model.

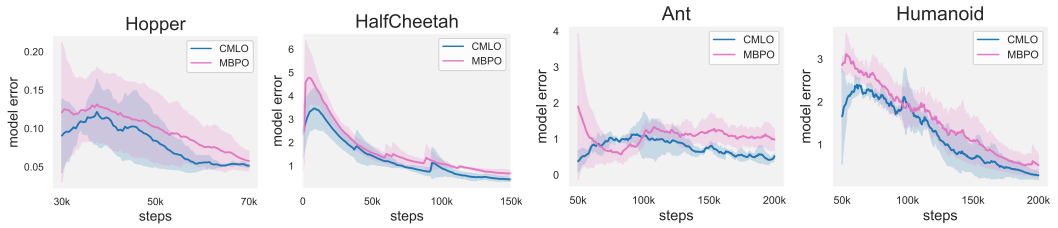


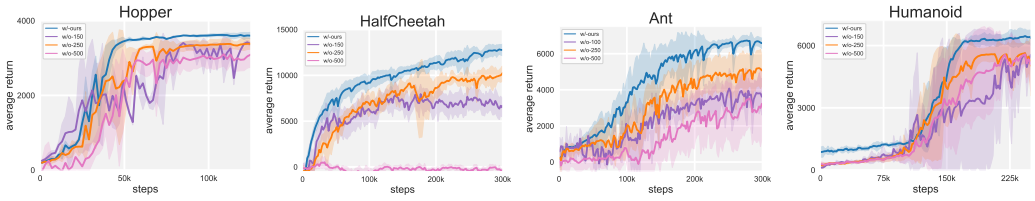
Figure 9: One-step model error in four benchmark tasks.

Policy Coverage Comparison. Policy coverage represents the exploration ability of the policy. The policy coverage increasing with the stages means that the policy has new explorations at every stage and may not fall into a local optimum. Here, we present the numerical comparison to MBPO in Table 5.

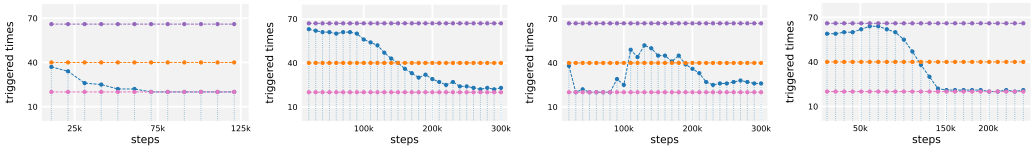
Table 5: Policy Coverage Comparison to MBPO in different stages.

		Stage1	Stage2	Stage3	Stage4	Stage5
HalfCheetah	CMLO	138.57	182.28	243.47	302.82	344.36
	MBPO	129.25	173.09	242.49	264.85	338.55
		Stage1	Stage2	Stage3	Stage4	Stage5
Ant	CMLO	354.15	744.92	849.47	876.12	909.80
	MBPO	342.13	729.30	821.66	864.93	880.25

Here, each stage i contains $(60 \times (i-1), 60 \times i]k$ steps. In HalfCheetah, we find that our policy achieves higher coverage especially in first 4 stages than MBPO. Consistently, we find that our policy enjoys higher performance, with an average return lead of about 1855.29 over MBPO in the first 300k steps. Likewise, the growth of policy coverage in Ant is also consistent with the rise in average



(a) average return



(b) triggered times

Figure 10: Ablation on event-triggered mechanism. (a,c) shows the average return with or with-out event-triggered mechanism in HalfCheetah and Ant benchmarks. (b,d) shows the average number of triggered times per 10k step. All the experiments are average over 4 random seeds.

return. The increase in policy coverage helps the policy to refrain from falling into a local optimum, thus improving performance.

Effectiveness of event-triggered mechanism. We compare applying model shift constraints to the unconstrained cases and the results are shown on Figure 10. To verify the effectiveness of adding suitable constraints, we invalidate the event-triggered mechanism and keep the other part unchanged in our method. As observed, our model is accurate enough when fixing the model training interval at 250, but it still performs worse than applying the model shift constraints. We attribute our model's out-performance to our rational model shift design. It improves the performance while minimizing the training cost of the model. Adding such a model shift can protect the model from overfitting on under-explored data, and can also save the model from fitting large data shifts.

To determine whether the event-triggered mechanism has an effect on the training process, we conduct a t-test to compare the average returns of CMLO with or without the mechanism. We compare the original CMLO to its variant with a fixed setting (w/o-250) and list the p-values in Table 6. Our p-values are much smaller than 0.05, so we say with a high degree of confidence that the smartly choosing dynamically varying number of explorations does make a difference in the overall performance.

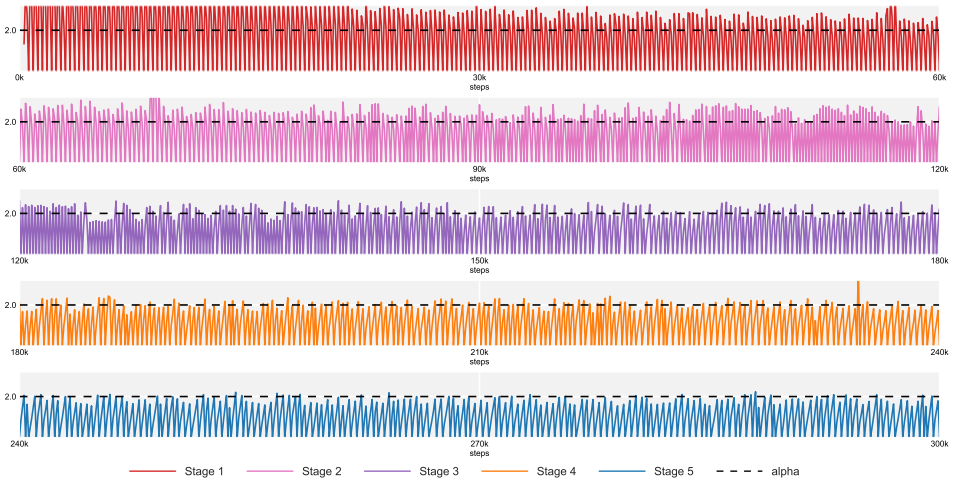
Table 6: t-test to the average returns of CMLO with or without event-triggered mechanism.

	Hopper	Walker	Swimmer	HalfCheetah	Ant	Humanoid
p-value	0.0141	4.74e-10	2.47e-5	7.48e-33	2.78e-26	1.983e-15

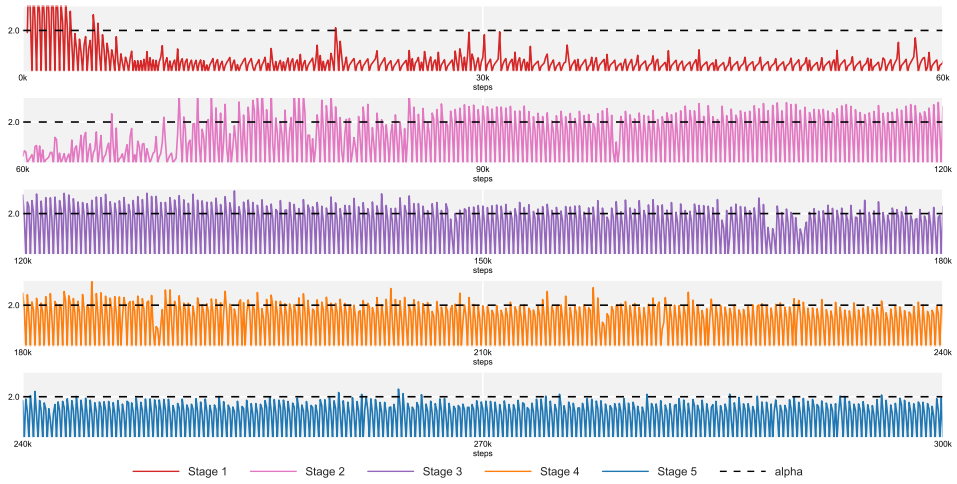
Besides, we provide visualization of event-triggered mechanism on HalfCheetah and Ant in Figure 11. The y-axis is our estimation of the triggered condition. This figure shows the constraint estimation and whether it reaches the triggered threshold (when the peak is above the threshold (dashed line), the primary trigger condition is satisfied) within different stages. Note that in the paper we have shown 4k steps for each stage, and here we present for the whole 60k.

E.6 Computing Infrastructure

Table 7 lists our computing infrastructure and the corresponding computational time used for training CMLO on the six benchmark tasks.



(a) HalfCheetah.



(b) Ant.

Figure 11: Visualization of event-triggered mechanism on HalfCheetah and Ant. Solid lines show the model shift estimation and dotted lines are the triggered threshold. Note that here we apply log value.

Table 7: Computing infrastructure and the computational time for each benchmark task.

	Hopper	Walker	Swimmer	HalfCheetah	Ant	Humanoid
CPU	Intel Core i7-6900K (16 threads)					
GPU	NVIDIA TITAN X (Pascal) x 3					
computation time in hours	20.15	19.21	31.58	35.97	29.35	33.31