Joint Model-Policy Optimization of a Lower Bound for Model-Based RL

Benjamin Eysenbach[*1,2] **Alexander Khazatsky**[*3] **Sergey Levine**[2,3] **Ruslan Salakhutdinov**[1]

[1]Carnegie Mellon University, [2]Google Brain, [3]UC Berkeley
beysenba@cs.cmu.edu, khazatsky@cs.stanford.edu

Abstract

Many model-based reinforcement learning (RL) methods follow a similar template: fit a model to previously observed data, and then use data from that model for RL or planning. However, models that achieve better training performance (e.g., lower MSE) are not necessarily better for control: an RL agent may seek out the small fraction of states where an accurate model makes mistakes, or it might act in ways that do not expose the errors of an inaccurate model. As noted in prior work, there is an objective mismatch: models are useful if they yield good policies, but they are trained to maximize their accuracy, rather than the performance of the policies that result from them. In this work, we propose a single objective for jointly training the model and the policy, such that updates to either component increase a lower bound on expected return. To the best of our knowledge, this is the first lower bound for model-based RL that holds globally and can be efficiently estimated in continuous settings; it is the only lower bound that mends the objective mismatch problem. A version of this bound becomes tight under certain assumptions. Optimizing this bound resembles a GAN: a classifier distinguishes between real and fake transitions, the model is updated to produce transitions that look realistic, and the policy is updated to avoid states where the model predictions are unrealistic. Numerical simulations demonstrate that optimizing this bound yields reward maximizing policies and yields dynamics that (perhaps surprisingly) can aid in exploration. We also show that a deep RL algorithm loosely based on our lower bound can achieve performance competitive with prior model-based methods, and better performance on certain hard exploration tasks.

1 Introduction

Much of the appeal of model-based RL is that model learning is a simple and scalable supervised learning problem. However, even after learning a very accurate model, it is hard to say whether that model will actually be useful for model-based RL [17, 29]. For example, a model might make small mistakes in critical states that cause a policy to take suboptimal actions. Alternatively, a model with large errors may yield a policy that attains high return if the model errors occur in states that the policy never visits.

The underlying problem is that dynamics models are trained differently from how they are used. Typical model-based methods train a model using data sampled from the *real* dynamics (e.g., using maximum likelihood), but apply these models by using data sampled from the *learned* dynamics [11, 23, 24, 48]. Prior work has identified this *objective mismatch* issue [17, 29, 30]: the model is trained using one objective, but the policy is trained using a different objective. Designing an objective for model training that is guaranteed to improve the expected reward remains an open problem.

[*]Equal contribution.

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

So, *how should we train a dynamics model so that it produces high-return policies when used for model-based RL?*

The key idea in this paper is to view model-based RL as a latent-variable problem: the latent variable is the trajectory and the cumulative reward is interpreted as the probability that the trajectory solves the task. Inferring the latent variable corresponds to learning a dynamics model. Latent variable models are typically learned via an evidence lower bound, and we show how a similar evidence lower bound provides yields a new objective for model-based RL. In the same way that the evidence lower bound is a joint optimization problem over two variables, our objective will *jointly* optimize the model and the policy using the same objective: to produce realistic and high-return trajectories. Our objective differs from standard model-based RL objectives, where it is more common to pit the model *against* the policy [5, 33, 37]. A consequence of maximizing the lower bound is that the dynamics model does not learn the true dynamics, but rather learns optimistic dynamics that facilitate exploration.

The main contribution of this work is an objective for model-based RL. To the best of our knowledge, this is the first lower bound for model-based RL that holds globally (unlike Luo et al. [30]) and can be efficiently estimated in continuous settings (unlike Kearns and Singh [26]). It is the first lower bound that jointly optimizes the model and policy using the same objective. We also present a more complex version of this bound that becomes tight under some assumptions. Through numerical simulations in simple tasks, we demonstrate that optimizing the bound yields reward-maximizing policies and yields an optimistic dynamics model that can aid exploration. We also demonstrate that our bound gracefully accounts for function approximation error in the model – the policy is penalized for taking transitions that the model cannot represent. Finally, we show that we can use parts of our theoretically-motivated objective to design a practically-applicable deep RL method that, despite deviating from the theory, can match the performance of prior model-based methods.

2 Related Work

Most model-based RL methods use maximum likelihood to fit the dynamics model and then use RL to maximize the expected return under samples from that model [10, 11, 23, 24, 48]. As noted in prior work, this maximum likelihood objective is not aligned with the RL objective: models that achieve higher likelihood do not necessarily produce better policies [17, 29, 30, 45, 52]. This issue is referred to as the *objective mismatch* problem: the model and policy (or planner) are optimized using different objectives. This problem arises in almost all model-based RL approaches, including those that train the model to predict the value function [34, 41] or that perform planning [10, 41].

Some prior work addresses this problem by decreasing the rewards at states where the model is inaccurate [27, 30, 42, 49, 51], a strategy that our method will also employ through a discriminator. While some of these methods also use discriminators in this manner, ours is the first to optimize a lower bound on returns. Other work modifies the model objective to include multi-step rollouts [3, 4, 17, 25, 45, 47]. Our method will modify the model objective in a different way, so that the model objective is exactly the same as the policy objective. Some prior work directly optimizes the model to produce good policies [2, 13, 32, 35, 43], as theoretically analyzed in Grimm et al. [20]. While our aim is the same as these prior methods, our approach will not require differentiating through unrolled model updates or optimization procedures.

Table 1: **Lower bounds for model-based RL.**

	Luo et al. [30]	Kearns and Singh [26]	MnM (Eq. 2)	MnM (Eq. 10)
holds globally	✗	✓	✓	✓
efficient to compute	✓	✗	✓	✓
unified objective	✗	✗	✓	✓
tight at optimality	✗[†]	✓	✗	✓

[†] Discrepancy measure is non-zero in stochastic environments.

Our work builds on prior lower bounds for model-based RL. Kearns and Singh [26] provide a lower bound that holds globally, but that is only computable in tabular settings. Luo et al. [30] provide a lower bound that can be efficiently estimated, but which only holds for nearby policies and models. As shown in Table 1, our bound combines the strengths of these prior works, providing a lower bound that holds globally and can be efficiently estimated in MDPs with continuous states and actions. Unlike prior work, our bound also mends the objective mismatch problem.

Our theoretical derivation builds on prior work that casts model-based RL as a two-player game between a model-player and a policy-player [5, 33, 36, 38]. Whereas prior work pits model and policy against one another, our formulation will result in a cooperative game: the model and policy cooperate to optimize the *same* objective (a lower bound on the expected return). Our approach, though structurally resembling a GAN, is different from prior work that replaces a maximum likelihood model with a GAN model [6, 8, 28].

The most similar prior work is VMBPO [9], which also jointly optimizes the model and the policy using the same objective. However, while our objective is a lower bound on expected return, VMBPO maximizes a different, risk-seeking objective, which is an *upper* bound on expected return (see Appendix B.1). This different objective can be expressed as the expected return plus the variance of the return, so VMBPO has the undesirable property of preferring policies that receive slight lower return if the variance of the return is much larger (see Appendix B.1). Indeed, while most of the components of our method (e.g., classifiers, GAN-like models) have been used in prior work, our paper is the first to provide a precise recipe for combining these components into an objective that is a provable lower bound.

3 A Unified Objective for Model-Based RL

Notation. We focus on the Markov decision process with states s_t, actions a_t, initial state distribution $p_0(s_0)$, positive reward function $r(s_t, a_t) > 0$, and dynamics $p(s_{t+1} \mid s_t, a_t)$. Our aim is to learn a control policy $\pi_\theta(a_t \mid s_t)$ with parameters θ that maximizes the expected discounted return:

$$\max_\theta \mathbb{E}_{\pi_\theta}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\Big]. \tag{1}$$

We use transitions (s_t, a_t, r_t, s_{t+1}) collected from the (real) environment to train the dynamics model $q_\theta(s_{t+1} \mid s_t, a_t)$, and use transitions sampled from this learned model to train the policy. To simplify notation, we will define a trajectory $\tau \triangleq (s_0, a_0, s_1, a_1, \cdots)$ as a sequence of states and actions visited in an episode. We define $R(\tau) \triangleq \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$ as the discounted return of a trajectory. We define two distributions over trajectories: $p^\pi(\tau)$ and $q^\pi(\tau)$ for when policy π_θ interacts with dynamics $p(s_{t+1} \mid s_t, a_t)$ and $q_\theta(s_{t+1} \mid s_t, a_t)$, respectively:

$$p^\pi(\tau) = p_0(s_0) \prod_{t=0}^{\infty} p(s_{t+1} \mid s_t, a_t)\pi_\theta(a_t \mid s_t), \quad q^\pi(\tau) = p_0(s_0) \prod_{t=0}^{\infty} q_\theta(s_{t+1} \mid s_t, a_t)\pi_\theta(a_t \mid s_t).$$

Desiderata. Our aim is to design an objective $\mathcal{L}(\theta)$ with two properties. *First*, this objective should be a lower bound on the expected return in the true environment: if the policy does well in the learned model, we are guaranteed that the policy will also do well in the true environment. The expected return under the learned model, which most prior model-based RL methods use to train the policy, is not a lower bound on the expected return [26, 30]. While prior work has made strides in developing lower bounds for model-based RL, even the best lower bounds do not hold for all models [30] or are limited to tabular settings [26].

Second, this objective should be the same for the policy and the dynamics model, such that updates to the model would improve the policy, and vice versa. This desiderata is important because prior work has found that training the model to be more accurate (increase likelihood) can decrease the policy's expected return under that model [17, 29, 30, 45, 52]. One prior method (VMBPO [9]) does train the model and policy with the same objective, but this objective corresponds to a risk-seeking, upper-bound on the expected returns.

An objective for model-based RL. We now introduce an objective that achieves these aims. The key idea to deriving this result is to take a probabilistic perspective on decision making: we view the trajectory as an unobserved random variable, and the reward function as the probability that a trajectory solves a task. Then, the problem of inferring this random variable is equivalent to learning the dynamics model. Using an evidence lower bound, we obtain an objective for jointly training the model and policy. We provide the full derivation in Appendix B.3.

Our resulting objective is the policy's reward when interacting with the learned model, but using a new reward function. The new reward function combines the task reward with a term that measures

the difference between the learned model and the real environment. We define our objective

$$\mathcal{L}(\theta) \triangleq \mathbb{E}_{q^{\pi_\theta}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t \tilde{r}(s_t, a_t, s_{t+1})\right], \tag{2}$$

where the modified reward function is defined as

$$\tilde{r}(s_t, a_t, s_{t+1}) \triangleq (1-\gamma)\log r(s_t, a_t) + \log\left(\frac{p(s_{t+1} \mid s_t, a_t)}{q(s_{t+1} \mid s_t, a_t)}\right) - (1-\gamma)\log(1-\gamma). \tag{3}$$

Intuitively, the new reward function $\tilde{r}$ penalizes the policy for taking transitions that are unlikely under the true dynamics model, similar to prior work [15, 46, 49]. Later, we will show that we can estimate this augmented reward *without knowing the true environment dynamics* by using a classifier.

We will optimize this lower bound with respect to both the policy $\pi_\theta(a_t \mid s_t)$ and the dynamics model $q_\theta(s_{t+1} \mid s_t, a_t)$. For the policy, we maximize the modified reward using samples from the learned model; the only difference from prior work is the modification to the reward function. Training the dynamics model using this objective is very different from standard maximum likelihood training. The model is optimized to sample trajectories that are similar to real dynamics (like a GAN) and that have high reward (unlike a GAN). This objective differs from VMBPO [9] by taking the $\log(\cdot)$ of the original reward functions; our experiments demonstrate that excluding this component invalidates our lower bound and results in learning suboptimal policies (Fig. 3a).

Our objective achieves the two desiderata. *First*, our objective is a lower bound on the expected return. To state this result formally, we will take the logarithm of the expected return. Of course, maximizing the $\log(\cdot)$ of the expected return is equivalent to maximizing the expected return.

Theorem 3.1. *The following bound holds for **any** dynamics $q(s_{t+1} \mid s_t, a_t)$ and policy $\pi(a_t \mid s_t)$:*

$$\log \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \geq \mathcal{L}(\theta).$$

The proof is presented in Appendix B.3. To the best of our knowledge, this is the first global (unlike Luo et al. [30]) and efficiently-computable (unlike Kearns and Singh [26]) lower bound for model-based RL. The model and the policy are trained using the same objective: updating the model not only increases the objective for the model, but also increases the objective for the policy.

Sec. 4 will introduce an algorithm to maximize this lower bound. While this lower bound may not be tight, experiments in Sec. 5 demonstrate that optimizing this lower bound yields policies that achieve high reward across a wide range of tasks. In Appendix A, we propose a variant of this bound that does become tight at convergence. Because this tight bound is more complex, we focus on the simple bound in this paper.

The optimal dynamics are optimistic. We now return to analyzing the simpler lower bound ($\mathcal{L}(\theta)$ in Eq. 2). In stochastic environments, the dynamics model that optimizes this lower bound is not equal to the true environment dynamics. Rather, it is biased towards sampling trajectories with high return. Ignoring parametrization constraints, the dynamics model that optimizes our lower bound is $q^*(\tau) = \frac{p(\tau)R(\tau)}{\int p(\tau')R(\tau')d\tau'}$ (proof in Appendix B.4.). While it may seem surprising that the objective-optimizing dynamics would differ from the true dynamics, this result is analogous to a VAE, where the ELBO-optimizing encoder differs from the prior. The optimism in the dynamics model may accelerate policy optimization, a hypothesis we will test in Sec. 5.1.

Would the optimistic dynamics overestimate the policy's return, violating Theorem 3.1? While using the optimistic dynamics with the *original* reward function will overestimate the true return, using the optimistic dynamics with the *augmented* reward function yields a valid lower bound. We demonstrate this effect in Fig. 3b.

4 Practical Optimization of the Lower Bound

The previous section presented a single (global) lower bound ($\mathcal{L}$ from Eq. 2) for jointly optimizing the policy and the dynamics model. In this section, we develop a practical algorithm for optimizing

this lower bound. The main challenge in optimizing this bound is that the augmented reward function depends on the transition probabilities of the real environment, $p(s_{t+1} \mid s_t, a_t)$, which are unknown. We address this challenge by learning a classifier (Sec. 4.1). Of course, Theorem 3.1 only holds if the classifier is Bayes-optimal. We then describe the precise update rules for the policy, dynamics model, and classifier (Sec. 4.2).

4.1 Estimating the Augmented Reward Function

To estimate the augmented reward function, which depends on the transition probabilities of the real environment, we learn a classifier that distinguishes real transitions from fake transitions. This approach is similar to GANs [19] and similar to prior work in RL [15, 49]. We use $C_\phi(s_t, a_t, s_{t+1}) \in [0, 1]$ to denote the classifier, which we train to distinguish real versus model transitions using the standard cross entropy loss:

$$\max_\phi \mathcal{L}_C(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{real}}, s_{t+1}^{\text{model}}; \phi) \triangleq \log C_\phi(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{real}}) + \log\left(1 - C_\phi(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{model}})\right). \tag{4}$$

Note that the real transition $(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{real}})$ and model transition $(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{model}})$ have the same initial state and initial action. Once trained, we can use the classifier's predictions to estimate the augmented reward function:

$$\tilde{r}(s_t, a_t, s_{t+1}) \approx \log r(s_t, a_t) + \log\left(\frac{C_\phi(s_t, a_t, s_{t+1})}{1 - C_\phi(s_t, a_t, s_{t+1})}\right). \tag{5}$$

The approximation above reflects function approximation error in learning the classifier. Now that we can estimate the augmented reward function, we can apply *any* RL method to maximize the augmented reward under transitions sampled from the dynamics model. The following section describes a particular instantiation using an off-policy RL algorithm.

4.2 Updating the Model and Policy

We now present our complete method, which trains three components: a classifier, a policy, and a dynamics model. Our method alternates between *(1)* updating the policy (by performing RL using model experience with augmented rewards) and *(2)* updating the dynamics model and classifier (using a GAN-like objective). In describing the loss functions below, we use the superscripts $(\cdot)^{\text{real}}$ and $(\cdot)^{\text{model}}$ to denote transitions that have been sampled from the true environment dynamics or the learned dynamics function. To reduce clutter, we omit the superscripts when unambiguous.

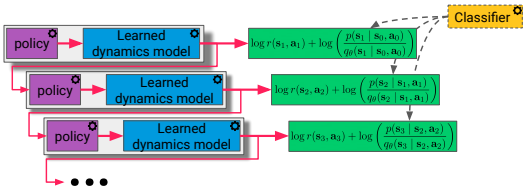


Figure 1: **Mismatched No More** is a model-based RL algorithm that learns a policy, dynamics model, and classifier. The classifier distinguishes real transitions from model transitions. The policy and dynamics are jointly optimized to sample transitions that yield high returns and look realistic, as estimated by the classifier.

Updating the policy. The policy is optimized to maximize the augmented reward on transitions sampled from the learned dynamics model. While this optimization can be done using any RL algorithm, including on-policy methods, we will focus on an off-policy actor-critic method.

We define the Q function as sum of *augmented* rewards under the learned dynamics model:

$$Q(s_t, a_t) \triangleq \mathbb{E}_{\substack{\pi(a_t|s_t), \\ q_\theta(s_{t+1}|s_t, a_t)}} \left[\sum_{t'=t}^{\infty} \gamma^{t'-t} \tilde{r}(s_{t'}, a_{t'}) \mid \substack{s_t = s_t, \\ a_t = a_t}\right]. \tag{6}$$

We approximate the Q function using a neural network $Q_\psi(s_t, a_t)$ with parameters ϕ. We train the Q function using the TD loss on transitions sampled from the *learned* dynamics model:

$$\mathcal{L}_Q(s_t, a_t, r_t, s_{t+1}^{\text{model}}; \psi) = \left(Q_\psi(s_t, a_t) - \lfloor y_t \rfloor_{\text{sg}}\right)^2, \tag{7}$$

where $\lfloor \cdot \rfloor_{\text{sg}}$ is the stop-gradient operator and $y_t = \tilde{r}(s_t, a_t, s_{t+1}^{\text{model}}) + \gamma \mathbb{E}_{\pi(a_{t+1}|s_{t+1}^{\text{model}})}\left[Q_\psi(s_{t+1}^{\text{model}}, a_{t+1})\right]$ is the TD target. The augmented reward $\tilde{r}$ is estimated using the learned classifier (Eq. 5). To estimate

Algorithm 1 **Mismatched no More (MnM)** is an algorithm for model-based RL. The method alternates between training the policy on experience from the learned dynamics model with augmented rewards and updating the model+classifier using a GAN-like loss. While we use an off-policy RL algorithm on L4, any other RL algorithm can be substituted.

1: **while** not converged **do**
2: Sample experience from the learned model.
3: Modify rewards using the classifier (Eq. 5).
4: Update the policy and Q function using the model experience and modified rewards (Eq.s 8 and 7).
5: Update model and classifier using GAN-like losses (Eq.s 4 and 9).
6: (Infrequently) Sample experience from the real model.
7: **return** policy $\pi_\theta(a_t \mid s_t)$.

the corresponding value function, we use a 1-sample approximation: $V_\psi(s_t) = Q_\psi(s_t, a_t)$ where $a_t \sim \pi_\theta(a_t \mid s_t))$. The policy is trained to maximize the Q function:

$$\max_\theta \mathcal{L}_\pi(s_t; \theta) \triangleq \mathbb{E}_{\pi_\theta(a_t|s_t)} \left[Q_\psi(s_t, a_t) \right]. \tag{8}$$

In our implementation, we regularize the policy by adding an additional entropy regularizer. Following prior work [18], we maintain two Q functions and two target Q functions, using the minimum of the two to compute the TD target. See Appendix D for details.

Updating the dynamics model. To optimize the dynamics model, we rewrite the lower bound in terms of a single transition (derivation in Appendix B.6):

$$\mathcal{L}_q(s_t^{\text{real}}, a_t^{\text{real}}; \theta) = \mathbb{E}_{s_{t+1}^{\text{model}} \sim q_\theta(s_{t+1}|s_t^{\text{real}}, a_t^{\text{real}})} \left[V_\psi(s_{t+1}^{\text{model}}) + \log \left(\frac{C_\phi(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{model}})}{1 - C_\phi(s_t^{\text{real}}, a_t^{\text{real}}, s_{t+1}^{\text{model}})} \right) \right]. \tag{9}$$

This loss is an approximation of our original lower bound (Eq. 2) because we estimate the difference in dynamics using a classifier. This approximation is standard in prior work on GANs [19] and adversarial inference [12, 14].

Intuitively, the procedure for optimizing the dynamics model and the classifier resembles a GAN [19]: the classifier is optimized to distinguish real transitions from model transitions, and the model is updated to fool the classifier (and increase rewards). However, *our method is not equivalent to simply replacing a maximum likelihood model with a GAN model*. Indeed, such an approach would not optimize a lower bound on expected return. Rather, our model objective includes an additional value term and our policy objective includes an additional classifier term. These changes enable the model and policy to optimize the same objective, which is a lower bound on expected return.

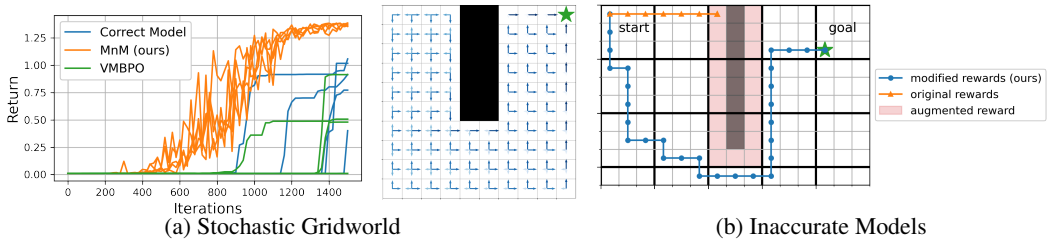
Algorithm summary. We summarize the method in Alg. 1 and provide an illustration in Fig. 1. We call the method MISMATCHED NO MORE (MnM) because the policy and model optimize the same objective, thereby resolving the objective mismatch problem noted in prior work. While the model and policy are optimized using the same objective, that objective can stop being a lower bound if the learned classifier is not Bayes-optimal.

Implementing MnM on top of a standard model-based RL algorithm is straightforward. First, create an additional classifier network. Second, instead of using the maximum likelihood objective to train the model, use the GAN-like objective in Eq. 9 to update both the model and the classifier. Third, add the classifier's logits to the predicted rewards (Eq. 5). Following prior work [24], we learn a neural network to predict the true environment rewards.

4.3 A Note about Exploration

The classifier term in the augmented reward (Eq. 3) is a double edged sword. Our theoretical derivation suggests that this term should appear, and our didactic experiments demonstrate that removing this term results in suboptimal behavior. Experiments also show that this term can effectively combat errors in the learned model. Prior work in offline RL has found similar model-error reward terms critical for achieving good performance in the offline setting [27, 42, 49, 51].

However, when scaling MnM to continuous control tasks in the online setting, we found that including this term hurts performance (Fig. 10). This makes sense: this classifier term is exactly the opposite



(a) Stochastic Gridworld (b) Inaccurate Models

Figure 2: **Gridworld experiments.** *(Left)* We apply MnM to a navigation task with transition noise that moves the agent to neighboring states with equal probability. MnM solves this task more quickly than Q-learning and VMBPO. The dynamics learned by MnM are different from the real dynamics, changing the transition noise (blue arrows) to point towards the goal. *(Right)* We simulate function approximation by a learning model that makes the same predictions for groups of 3×3 states, resulting in a model that is inaccurate around obstacles. The modified reward compensates for these errors by penalizing the policy for navigating near obstacles.

of exploration objectives based on model error (e.g., [44]), and penalizes the policy for performing exploration. Our continuous control experiments in Sec. 5.2 will deviate from our theoretical derivation: they will use the model objective suggested by our theory, but the same policy objective as prior. That is, the policy maximizes $r(s_t, a_t)$ instead of $\tilde{r}(s_t, a_t)$ (Eq. 3). We call this method "MnM-approx." We note that many theoretical model-based RL papers also find it necessary to implement practical algorithms that differ from the theory [30, 46, 49].

5 Numerical Simulations

The primary aim of our experiments is to verify that our objective is a valid lower bound, and that maximizing the bound yields reward-maximizing policies. We study these questions in Sec. 5.1, where we will use tabular problems so that we can study the objective in the absence of function approximation error. Then, in Sec. 5.2, we show adapting part of our objective into a scalable model-based RL algorithm yields a method that, while deviating from the theory, can achieve competitive results on benchmark tasks.

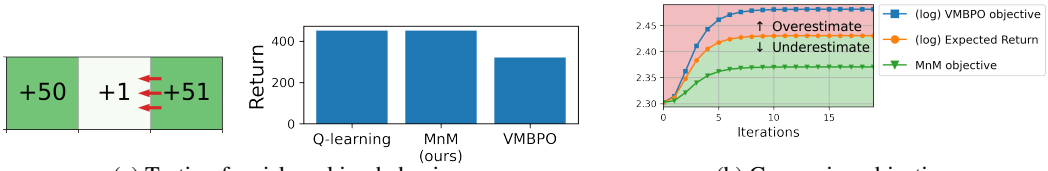
5.1 Understanding the Lower Bound and the Learned Dynamics

Our didactic experiments test whether optimizing the lower bound produces optimal policies and study how the components of the lower bound. We use tabular domains in this section, as they allow us to analytically compute the optimal policy for comparison, and allow us to evaluate the lower bound in the absence of function approximation error.

Our first experiment compares the lower bound to an oracle method that applies Q-learning to a perfect dynamics model. In comparison to this baseline, MnM learns a dynamics model using the GAN-like objective (Eq. 9) and maximizes a modified reward function (Eq. 5). We also compare to VMBPO, which learns a dynamics model similar to MnM but omits the log-transformation of the reward function; this transformation ordinarily encourages pessimistic behavior. For this experiment, we use a 10×10 gridworld with stochastic dynamics and sparse reward, shown in Fig. 2a *(Left)*. The results, shown in Fig. 2a, show that MnM outperforms both Q-learning with the correct model and VMBPO on this task. We hypothesize that VMBPO performs poorly on this task because it maximizes an upper bound on performance; and confirm this in the following experiments.

We hypothesize that MnM outperforms Q-learning with the correct model because it learns "optimistic" dynamics. We test this hypothesis by visualizing the dynamics model learned by MnM (Fig. 2a *(Left)*). While the true environment dynamics have stochasticity that moves the agent in a random direction with *equal* probability, the MnM dynamics model biases this stochasticity to lead the agent towards the goal (blue arrows point towards the goal). Of course, we use the true environment dynamics, not the optimistic dynamics model, for evaluating the policies.

Our augmented reward function contains two crucial components, *(1)* the classifier term and *(2)* the logarithmic transformation of the reward function. We test the importance of the classifier term in correcting for inaccurate models. To do this, we limit the capacity of the MnM dynamics model so that it makes "low-resolution" predictions, forcing all states in 3×3 blocks to have the same dynamics. We will use the gridworld shown in Fig. 2b *(Left)*, which contains obstacles that occur



(a) Testing for risk seeking behavior

(b) Comparing objectives.

Figure 3: **Analyzing the lower bound.** *(Left)* On a 3-state MDP with stochastic transitions in one state (red arrows), MnM learns the reward-maximizing policy while VMBPO learns a strategy with lower rewards and higher variance (as predicted by theory). *(Right)* We apply value iteration to the gridworld from Fig. 2a to analytically compute various objectives. As predicted by our theory, the MnM objective is a lower bound on the expected return, whereas the VMBPO objective overestimates the expected return.

at a finer resolution than the model can detect. When the "low resolution" dynamics model makes predictions for states near the obstacle, it will average together some states with obstacles and some states without obstacles. Thus, the model will (incorrectly) predict that the agent always has some probability of moving in each direction, even if that direction is actually blocked by an obstacle. However, the classifier (whose capacity we have also limited) detects that the dynamics model is inaccurate in these states, so the augmented reward is much lower at these states. Thus, MnM is able to solve this task despite the inaccurate model; an ablation of MnM that removes the classifier term attempts to navigate through the wall and fails to reach the goal.

Like MnM, VMBPO includes a classifier term in the reward function but omits the logarithmic transformation, a difference we expect to cause VMBPO to prefer suboptimal, risk-seeking policies. To test this hypothesis, we use the 3-state MDP in Fig. 3a *(Right)*, where numbers indicate the reward at each state. While moving to the right state yields slightly higher rewards, "wind" knocks the agent out of this state with probability 50% so the reward-maximizing strategy is to move to the left state. While MnM learns the reward-maximizing strategy, VMBPO learns a policy that goes to the right state and receives lower returns.

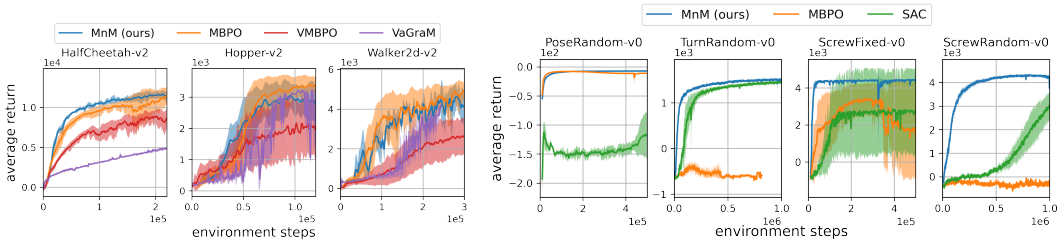
Finally, we verify Theorem 3.1 by comparing the MnM objective to the true expected return. We also compare to the objective from VMBPO, which looks similar to the MnM objective but omits the logarithmic transformation; our theory predicts that the VMBPO objective will therefore be an *upper* bound on the expected return (see Appendix B.1). We use the gridworld from Fig. 2a and use a version of MnM based on value iteration to avoid approximation error. Plotting the MnM objective in Fig. 3b *(Right)*, we observe that it is always a lower bound on the (log) expected return, as predicted by our theory. Also as predicted by the theory, the VMBPO objective overestimates the expected return, illustrating the importance of the logarithmic transformation.

Of the oft-cited benefits of model-based RL is that the learned dynamics model can be re-used to solve new tasks. MnM presents a twist on that story, because MnM does not learn the true environment dynamics but rather learns optimistic dynamics (see Sec. 3). In Appendix C (Fig. 13), we examine MnM's effectiveness at transferring dynamics to different tasks in the stochastic gridworld from Fig. 2a. We find that the (optimistic) dynamics learned by MnM do not slow learning on dissimilar tasks, but can accelerate learning of challenging, similar tasks.

5.2 Comparisons On Higher-Dimensional Tasks

Our next experiments use continuous-control robotic tasks to study whether the new model-learning objective suggested by our lower bound can be stably applied to higher-dimensional control tasks. While the MnM-approx method tested in this section deviates from our theoretical derivation (see Sec. 4.3), these experiments are nonetheless useful for providing preliminary evidence about whether the proposed model objective can be efficiently estimated and stably optimized.

We use MBPO [24] as a baseline for model-based RL because it achieves state-of-the-art results and is a prototypical example of model-based RL algorithms that use maximum likelihood models. Because our method differs from the baseline (MBPO) along only one dimension now (new model update, same policy update), these experiments will directly test the utility of our proposed model update. Experimental details are in Appendix D.



(a) OpenAI Gym benchmark

(b) ROBEL manipulation benchmark

Figure 4: **Comparison on two benchmarks**. *(Left)* On the OpenAI gym benchmark [7], MnM-approx performs on par with a prior state-of-the-art method (MBPO [24]), while consistently outperforming a recent method that addresses objective mismatch (VMBPO [9]). *(Right)* The ROBEL manipulation benchmark [1] contains complex contact dynamics that are challenging to model. MBPO performs poorly on these tasks, often worse than model-free SAC [22].

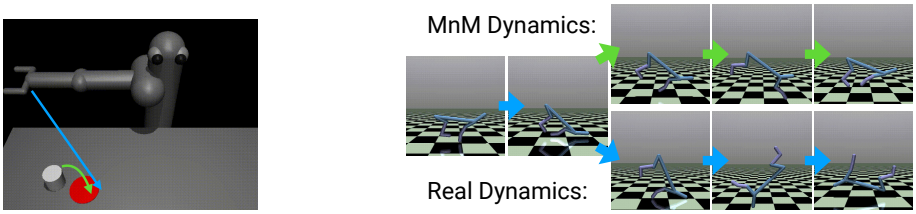


Figure 5: **Optimistic Dynamics**: *(Left)* On the `Pusher-v2` task, the MnM dynamics model makes the puck move towards the puck move towards the gripper before being grasped. *(Right)* On the `HalfCheetah-v2` task, the MnM dynamics model helps the agent stay upright after tripping.

We first use three locomotion tasks from the OpenAI Gym benchmark [7] to compare MnM-approx to MBPO and VMBPO. The VMBPO curves are taken directly from that paper. As shown in Fig. 4a, MnM-approx performs roughly on par with MBPO and outperforms [9], a more recent model-based method also addresses the objective mismatch problem by maximizing an *upper* bound on expected returns (1-sided p-values: $p = 0.04, 0.00, 0.03$).

We next use the ROBEL manipulation benchmark [1] to compare how MnM-approx and MBPO handle tasks with more complicated dynamics. As shown in Fig. 4b, MBPO struggles to learn three of the four tasks, likely because the dynamics are hard. In contrast, MnM-approx solves these tasks, likely because the GAN-like model is more accurate. MnM-approx outperforms MBPO on all tasks ($p \leq 0.03$) and outperforms the model-free baseline on 2/4 tasks ($p \leq 0.02$)

We next compare to prior methods for addressing the objective mismatch problem using the `DClawScrewFixed-v0` task. We compare to VAML [17] and the value-weighted maximum likelihood approach proposed in Lambert et al. [29]. As shown in Fig. 6, these alternative model learning objectives perform poorly on this task.

To better understand why MnM-approx sometimes outperforms the maximum likelihood baseline (MBPO), we visualized the Q-values throughout training. We used `metaworld-drawer-open-v2`, a task where we found a noticeable difference in the performance between MBPO and MnM-approx. Fig. 7a shows that MnM-approx yields Q values that are more accurate and more stable than MBPO, perhaps because MBPO learns a policy that exploits inaccuracies in the learned model.

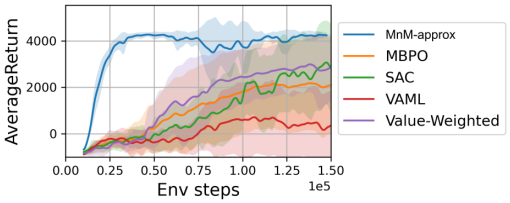
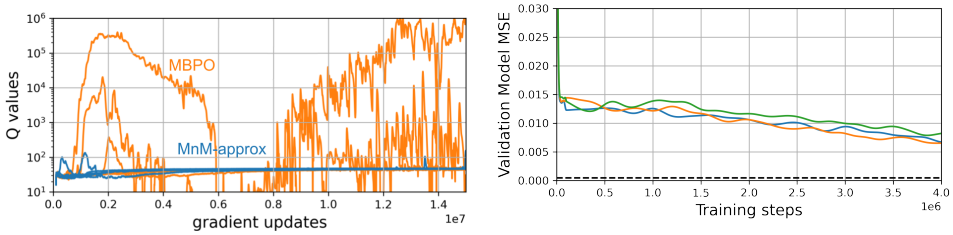


Figure 6: **Alternative model learning objectives**: Using the `DClawScrewFixed-v0` task, we compare MnM-approx and MBPO [24] to two additional model learning objectives suggested in the literature, VAML [17] and value-weighted maximum likelihood [29]. MnM-approx outperforms these alternative approaches.

To study the stability of MnM-approx, we plot the validation MSE of the MnM-approx model when training on the `DClawScrewRandom-v0` task. As shown in Fig. 7b, the MSE decreases stably, indicating that the adversarial nature of the MnM-approx model training does not create instabilities. In Appendix C (Fig. 8, 9) we visualize Q



(a) Model exploitation

(b) MnM-approx trains stably.

Figure 7: **Analyzing MnM-approx.** *(Left)* The very large Q values of MBPO suggest model exploitation, which our method appears to avoid. *(Right)* Despite resembling a GAN, the MnM-approx dynamics model trains stably. Different colors correspond to different random seeds, and the dashed line corresponds to the minimum validation MSE of an MLE dynamics model.

value stability and MSE model error on additional environments. In Appendix C (Fig. 12), we study the effect of model horizon on MnM in a continuous control task, and find that performance degrades linearly as we increase this value.

We visualize the dynamics learned by MnM on two robotic control tasks in Fig 5. These tasks have deterministic dynamics, so our theory would predict that an idealized version of MnM would learn a dynamics model exactly equal to the deterministic dynamics. However, our implementation relies on function approximation (neural networks) to learn the dynamics, and the limited capacity of function approximators makes otherwise-deterministic dynamics appear stochastic. On the `Pusher-v2` task, the MnM dynamics cause the puck to move towards the robot arm even before the arm has come in contact with the puck. While this movement is not physically realistic, it may make the exploration problem easier. On the `HalfCheetah-v2` task, the MnM dynamics increase the probability that the agent remains upright after tripping, likely making it easier for the agent to learn how to run. We expect that the implicit stochasticity caused by function approximation to be especially important for real-world tasks, where the complexity of the real dynamics often dwarfs the capacity of the learned dynamics model.

6 Conclusion

The main contribution of this paper is an objective for model-based RL that is both efficient to compute and globally valid. Moreover, the model and policy are jointly optimized using this same objective, thereby addressing the objective mismatch problem. This *joint optimization* will ease and accelerate the design of future model-based RL algorithms.

The main limitation of this paper is the classifier term. Because this term is estimated, the objective used in practice can fail to be a lower bound when the classifier is not Bayes-optimal. Additionally, this classifier term can hinder exploration and degrade performance on continuous control tasks in the online setting. We encourage future investigations into *joint optimization* based approaches that do explicitly address the role of exploration, perhaps including the dataset as an additional optimization variable.

Acknowledgements. We thank Laura Smith, Marvin Zhang, and Dibya Ghosh for helpful discussions, and thank Anusha Nagabandi and Michael Janner for help with experiments. We thank Yinlam Chow for help sharing the VMBPO baselines and thank Danijar Hafner for reviewing a draft of the paper. This material is supported by the Fannie and John Hertz Foundation and the NSF GRFP (DGE1745016).

References

[1] Ahn, M., Zhu, H., Hartikainen, K., Ponte, H., Gupta, A., Levine, S., and Kumar, V. (2020). ROBEL: Robotics benchmarks for learning with low-cost robots. In *Conference on Robot Learning*, pages 1300–1313. PMLR.

[2] Amos, B., Rodriguez, I., Sacks, J., Boots, B., and Kolter, Z. (2018). Differentiable MPC for end-to-end planning and control. *Advances in neural information processing systems*.

[3] Asadi, K., Misra, D., Kim, S., and Littman, M. L. (2019). Combating the compounding-error problem with a multi-step model. *arXiv preprint arXiv:1905.13320*.

[4] Asadi, K., Misra, D., and Littman, M. (2018). Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, pages 264–273. PMLR.

[5] Bagnell, J. A., Ng, A. Y., and Schneider, J. G. (2001). Solving uncertain Markov decision processes.

[6] Bai, X., Guan, J., and Wang, H. (2019). A model-based reinforcement learning with adversarial training for online recommendation. *Advances in Neural Information Processing Systems*, 32:10735–10746.

[7] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI gym. *arXiv preprint arXiv:1606.01540*.

[8] Chen, S., Jiang, J., Zhang, X., Wu, J., and Lu, G. (2020). Gan-based planning model in deep reinforcement learning. In *International Conference on Artificial Neural Networks*, pages 323–334. Springer.

[9] Chow, Y., Cui, B., Ryu, M., and Ghavamzadeh, M. (2020). Variational model-based policy optimization. *arXiv preprint arXiv:2006.05443*.

[10] Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765.

[11] Deisenroth, M. P. and Rasmussen, C. E. (2011). PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, pages 465–472.

[12] Donahue, J., Krähenbühl, P., and Darrell, T. (2016). Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

[13] D'Oro, P., Metelli, A. M., Tirinzoni, A., Papini, M., and Restelli, M. (2020). Gradient-aware model-based policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3801–3808.

[14] Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. (2016). Adversarially learned inference. *arXiv preprint arXiv:1606.00704*.

[15] Eysenbach, B., Chaudhari, S., Asawa, S., Levine, S., and Salakhutdinov, R. (2020a). Off-dynamics reinforcement learning: Training for transfer with domain classifiers. In *International Conference on Learning Representations*.

[16] Eysenbach, B., Salakhutdinov, R., and Levine, S. (2020b). C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*.

[17] Farahmand, A.-m., Barreto, A., and Nikovski, D. (2017). Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, pages 1486–1494.

[18] Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. *International Conference on Machine Learning (ICML)*.

[19] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NIPS)*.

[20] Grimm, C., Barreto, A., Singh, S., and Silver, D. (2020). The value equivalence principle for model-based reinforcement learning. *arXiv preprint arXiv:2011.03506*.

[21] Guadarrama, S., Korattikara, A., Ramirez, O., Castro, P., Holly, E., Fishman, S., Wang, K., Gonina, E., Wu, N., Kokiopoulou, E., Sbaiz, L., Smith, J., Bartók, G., Berent, J., Harris, C., Vanhoucke, V., and Brevdo, E. (2018). TF-Agents: A library for reinforcement learning in tensorflow. [Online; accessed 25-June-2019].

[22] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*.

[23] Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. (2019). Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*.

[24] Janner, M., Fu, J., Zhang, M., and Levine, S. (2019). When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, pages 12519–12530.

[25] Joseph, J., Geramifard, A., Roberts, J. W., How, J. P., and Roy, N. (2013). Reinforcement learning with misspecified model classes. In *2013 IEEE International Conference on Robotics and Automation*, pages 939–946. IEEE.

[26] Kearns, M. and Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232.

[27] Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. (2020). Morel : Model-based offline reinforcement learning. *ArXiv*, abs/2005.05951.

[28] Kurutach, T., Tamar, A., Yang, G., Russell, S. J., and Abbeel, P. (2018). Learning plannable representations with causal InfoGAN. In *Advances in Neural Information Processing Systems*, pages 8733–8744.

[29] Lambert, N., Amos, B., Yadan, O., and Calandra, R. (2020). Objective mismatch in model-based reinforcement learning. In *Learning for Dynamics and Control*, pages 761–770. PMLR.

[30] Luo, Y., Xu, H., Li, Y., Tian, Y., Darrell, T., and Ma, T. (2019). Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *ICLR (Poster)*.

[31] Mihatsch, O. and Neuneier, R. (2002). Risk-sensitive reinforcement learning. *Machine learning*, 49(2):267–290.

[32] Nikishin, E., Abachi, R., Agarwal, R., and Bacon, P.-L. (2021). Control-oriented model-based reinforcement learning with implicit differentiation. *arXiv preprint arXiv:2106.03273*.

[33] Nilim, A. and El Ghaoui, L. (2003). Robustness in Markov decision problems with uncertain transition matrices. In *NIPS*, pages 839–846. Citeseer.

[34] Oh, J., Singh, S., and Lee, H. (2017). Value prediction network. *arXiv preprint arXiv:1707.03497*.

[35] Okada, M., Rigazio, L., and Aoshima, T. (2017). Path integral networks: End-to-end differentiable optimal control. *arXiv preprint arXiv:1706.09597*.

[36] Rajeswaran, A., Mordatch, I., and Kumar, V. (2020). A game theoretic framework for model based reinforcement learning. In *International Conference on Machine Learning*, pages 7953–7963. PMLR.

[37] Ross, S. and Bagnell, D. (2012). Agnostic system identification for model-based reinforcement learning. In *ICML*.

[38] Ross, S., Gordon, G. J., and Bagnell, J. A. (2011). A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[39] Rudner, T. G., Pong, V. H., McAllister, R., Gal, Y., and Levine, S. (2021). Outcome-driven reinforcement learning via variational inference. *arXiv preprint arXiv:2104.10190*.

[40] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training GANs. *Advances in neural information processing systems*, 29:2234–2242.

[41] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

[42] Sorg, J., Singh, S. P., and Lewis, R. L. (2010). Internal rewards mitigate agent boundedness. In *ICML*.

[43] Srinivas, A., Jabri, A., Abbeel, P., Levine, S., and Finn, C. (2018). Universal planning networks: Learning generalizable representations for visuomotor control. In *International Conference on Machine Learning*, pages 4732–4741. PMLR.

[44] Stadie, B. C., Levine, S., and Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*.

[45] Talvitie, E. (2014). Model regularization for stable sample rollouts. In *UAI*, pages 780–789.

[46] Vemula, A., Oza, Y., Bagnell, J. A., and Likhachev, M. (2020). Planning and execution using inaccurate models with provable guarantees. *arXiv preprint arXiv:2003.04394*.

[47] Venkatraman, A., Capobianco, R., Pinto, L., Hebert, M., Nardi, D., and Bagnell, J. A. (2016). Improved learning of dynamics models for control. In *International Symposium on Experimental Robotics*, pages 703–713. Springer.

[48] Williams, G., Wagener, N., Goldfain, B., Drews, P., Rehg, J. M., Boots, B., and Theodorou, E. A. (2017). Information Theoretic MPC for Model-Based Reinforcement Learning. In *International Conference on Robotics and Automation (ICRA)*.

[49] Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. (2021). Combo: Conservative offline model-based policy optimization. *arXiv preprint arXiv:2102.08363*.

[50] Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. (2020a). Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR.

[51] Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., and Ma, T. (2020b). MOPO: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*.

[52] Ziebart, B. D. (2010). Modeling purposeful adaptive behavior with the principle of maximum causal entropy.

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] Our main claim is that our proposed objective is a global lower bound on the expected return (proof in Appendix B).
   (b) Did you describe the limitations of your work? [Yes] See Sec. 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] While RL algorithms might be applied to applications with positive and negative societal implications, our theoretical analysis does not dictate these applications.
   (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]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B.
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)? [No] Experimental details are included in Appendix D. Code will be released upon publications.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Plots show the mean and standard deviation across the random seeds.
   (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)? [No] Proprietary.
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? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (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]
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]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Tightening the lower bound

We now introduce a modification to our lower bound that does make the bound tight. This new lower bound will be more complex than the one introduced above and we have not yet successfully designed an algorithm for maximizing it. Nonetheless, we believe that presenting the bound may prove useful for the design of future model-based RL algorithms.

We will use $\mathcal{L}_\gamma(\theta)$ to denote this new lower bound. In addition to the policy and dynamics, this bound will also depend on a time-varying discount, $\gamma_\theta(t)$, in place of the typical γ^t term. Similar learned discount factors have been studied in previous work on model-free RL [39]. We define this objective as follows:

$$\mathcal{L}_\gamma(\theta) \triangleq \mathbb{E}_{q^{\pi_\theta}(\tau)}\left[\sum_{t=0}^{\infty} \gamma_\theta(t)\tilde{r}_\gamma(s_t, a_t, s_{t+1})\right], \tag{10}$$

where the augmented reward is now defined as

$$\begin{aligned}\tilde{r}_\gamma(s_t,a_t, s_{t+1}) \triangleq \log r(s_t, a_t) + \log\left(\frac{\gamma^t}{\gamma_\theta(t)}\right) \\ + \frac{1-\Gamma_\theta(t-1)}{\gamma_\theta(t)} \log\left(\frac{p(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \cdots)}{q_\theta(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \cdots)}\right),\end{aligned}$$

and $\Gamma_\theta(t) = \sum_{t'=0}^{t} \gamma_\theta(t')$ is the CDF of the learned discount function (i.e., $\gamma_\theta(t)$ is a probability distribution over t.). This new lower bound, which differs from our main lower bound by the learnable discount factor, does provide a tight bound on the expected return objective:

Lemma A.1. *Let an arbitrary policy $\pi(a_t \mid s_t)$ be given. The objective $\mathcal{L}_\gamma(\theta)$ is also a lower bound on the expected return objective, $\log \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \geq \mathcal{L}_\gamma(\theta)$, and this bound becomes tight at optimality:*

$$\log \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] = \max_{q^\pi(\tau), \gamma_\theta(t)} \mathcal{L}_\gamma(\theta).$$

The proof is presented in Appendix B.4. One important limitation of this result is that the learned dynamics that maximize this lower bound to make the bound tight may be non-Markovian. Intriguingly, this analysis suggests that using non-Markovian models, such as RNNs and transformers, may accelerate learning on Markovian tasks. This paper does not propose an algorithm for optimizing this more complex lower bound.

B Proofs and Additional Analysis

B.1 VMBPO Maximizes an Upper Bound on Return

While MnM aims to maximize the (log) of the expected return, VMBPO aims to maximize the expected *exponentiated* return:

$$\text{MnM:} \quad \log \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \qquad \text{VMBPO:} \quad \log \mathbb{E}_\pi\left[e^{\eta \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)}\right],$$

where $\eta > 0$ is a temperature term used by VMBPO. Note that maximizing the log of the expected return, as done by MnM, is equivalent to maximizing the expected return, as the function $\log(\cdot)$ is monotone increasing. However, maximizing the log of the expected *exponentiated* return, as done by VMBPO, is not equivalent to maximizing the expected return. Rather, it corresponds to maximizing a sum of the expected return and the *variance* of the return [31, Page 272]:

$$\frac{1}{\eta} \log \mathbb{E}_\pi\left[e^{\eta \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)}\right] = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] + \frac{\eta}{2}\mathrm{Var}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] + \mathcal{O}(\eta^2).$$

Thus, in environments with stochastic dynamics or rewards (e.g., the didactic example in Fig. 3a), VMBPO will prefer to receive lower returns if the variance of the returns is much higher. We note that the expected *exponentiated* return is an *upper* bound on the expected return:

$$\log \mathbb{E}_\pi\left[e^{\eta \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)}\right] \geq \eta \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right].$$

This statement is a direct application of Jensen's inequality. The bound holds with a strict inequality in almost all MDPs. The one exception is trivial MDPs where all trajectories have exactly the same return. Of course, even a random policy is optimal for these trivial MDPs.

B.2 Helper Lemmas

We start by introducing a simple identity that will help handle discount factors in our analysis.

Lemma B.1. *Define $p(H) = \textsc{Geom}(1-\gamma)$ as the geometric distribution. Let discount factor $\gamma \in (0,1)$ and random variable x_t be given. Then the following identity holds:*

$$\mathbb{E}_{p(H)}\left[\sum_{t=0}^{H} x_t\right] = \sum_{t=0}^{\infty} \gamma^t x_t.$$

The proof involves substituting the definition of the Geometric distribution and then rearranging terms.

Proof.

$$\begin{aligned}
\mathbb{E}_{p(H)}\left[\sum_{t=0}^{H} x_t\right] &= (1-\gamma)\sum_{H=0}^{\infty}\gamma^H \sum_{t=0}^{H} x_t \\
&= (1-\gamma)\left(x_0 + \gamma(x_0+x_1) + \gamma^2(x_0+x_1+x_2) + \cdots\right) \\
&= (1-\gamma)\left(x_0(1+\gamma+\gamma^2+\cdots) + x_1(\gamma+\gamma^2+\cdots) + \cdots\right) \\
&= (1-\gamma)\left(x_0\frac{1}{1-\gamma} + x_1\frac{\gamma}{1-\gamma} + x_2\frac{\gamma^2}{1-\gamma} + \cdots\right) \\
&= \sum_{t=0}^{\infty}\gamma^t x_t.
\end{aligned}$$

□

The second helper lemma describes how the discounted expected return objective can be written as the expected *terminal* reward of a mixture of finite-length episodes.

Lemma B.2. *Define $p(H) = \textsc{Geom}(1-\gamma)$ as the geometric distribution, and $p(\tau \mid H)$ as a distribution over length-H episodes. We can then write the expected discounted return objective as follows:*

$$\mathbb{E}_{p(\tau|H=\infty)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] = \frac{1}{1-\gamma}\mathbb{E}_{p(H)}\left[\mathbb{E}_{p(\tau|H=H)}\left[r(s_H,a_H)\right]\right] \tag{11}$$

$$= \frac{1}{1-\gamma}\iint p(H)p(\tau \mid H=H)r(s_H,a_H)d\tau dH. \tag{12}$$

Proof. The first identity follows from the definition of the geometric distribution. The second identity writes the expectations as integrals, which will make future analysis clearer. □

B.3 Proof of Theorem 3.1

Proof.

$$
\begin{aligned}
&\log \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] \overset{(a)}{=} \log\frac{1}{1-\gamma}\iint p(H)p(\tau \mid H=H)r(s_H,a_H)d\tau dH \\
&= \log\iint p(H)\frac{p(\tau \mid H=H)}{q_\theta(\tau \mid H=H)}q_\theta(\tau \mid H=H)r(s_H,a_H)d\tau dH - \log(1-\gamma) \\
&\overset{(b)}{\geq} \int p(H)\left(\log\int \frac{p(\tau \mid H=H)}{q_\theta(\tau \mid H=H)}q_\theta(\tau \mid H=H)r(s_H,a_H)d\tau\right)dH - \log(1-\gamma) \\
&\overset{(c)}{\geq} \iint p(H)q_\theta(\tau \mid H=H)\left(\log p(\tau \mid H=H) - \log q_\theta(\tau \mid H=H) + \log r(s_H,a_H)\right)d\tau dH - \log(1-\gamma) \\
&\overset{(d)}{=} \iint p(H)q_\theta(\tau \mid H=H)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1} \mid s_t,a_t) + \cancel{\log \pi_\theta(a_t \mid s_t)} - \log q_\theta(s_{t+1} \mid s_t,a_t) - \cancel{\log \pi_\theta(a_t \mid s_t)}\right) + \log r(s_H,a_H)\right)d\tau dH \\
&\qquad - \log(1-\gamma) \\
&\overset{(e)}{=} \iint p(H)q_\theta(\tau \mid H=\infty)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1} \mid s_t,a_t) - \log q_\theta(s_{t+1} \mid s_t,a_t)\right) + \log r(s_H,a_H)\right)d\tau dH - \log(1-\gamma) \\
&\overset{(f)}{=} \int q_\theta(\tau)\int p(H)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1} \mid s_t,a_t) - \log q_\theta(s_{t+1} \mid s_t,a_t)\right) + \log r(s_H,a_H)\right)dH d\tau - \log(1-\gamma) \\
&\overset{(g)}{=} \int q_\theta(\tau)\mathbb{E}_{p(H)}\left[\left(\sum_{t=0}^{H}\log p(s_{t+1} \mid s_t,a_t) - \log q_\theta(s_{t+1} \mid s_t,a_t)\right) + \log r(s_H,a_H)\right]d\tau - \log(1-\gamma) \\
&\overset{(h)}{=} \int q_\theta(\tau)\sum_{t=0}^{\infty}\gamma^t\left(\log p(s_{t+1} \mid s_t,a_t) - \log q_\theta(s_{t+1} \mid s_t,a_t) + (1-\gamma)\log r(s_H,a_H)\right)d\tau - \log(1-\gamma) \\
&\overset{(i)}{=} \mathbb{E}_{q_\theta(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t\left(\log p(s_{t+1} \mid s_t,a_t) - \log q_\theta(s_{t+1} \mid s_t,a_t) + (1-\gamma)\log r(s_H,a_H) - (1-\gamma)\log(1-\gamma)\right)\right].
\end{aligned}
$$

For *(a)*, we applied Lemma B.2. For *(b)*, we applied Jensen's inequality. For *(c)*, we applied Jensen's inequality again. For *(d)*, we substituted the definitions of $p_\theta(\tau \mid H)$ and $q_\theta(\tau \mid H)$. For *(e)*, we noted that the term inside the summation only depends on the first H steps of the trajectory, so collecting longer trajectories will not change the result. This allows us to rewrite the integral as an expectation using a single infinite-length trajectory. For *(f)*, we recalled the definition $q_\theta(\tau) = q_\theta(\tau = H = \infty)$ and swap the order of integration. For *(g)*, we express the inner integral over $p(H)$ as an expectation. For *(h)*, we applied the identity from Lemma B.1. For *(i)*, we moved the constant $\log(1-\gamma)$ back inside the integral and rewrote the integral as an expectation. We have thus obtained the desired result. □

B.4 Proof of Lemma A.1

Before presenting the proof of Theorem 3.1 itself, we show how we derived the lower bound in this more general case. While this step is not required for the proof, we include it because it sheds light on how similar lower bounds might be derived for other problems. We define $\gamma_\theta(H)$ to be a learned distribution over horizons H. We then proceed, following many of the same steps as for the proof of Theorem 3.1.

$$
\begin{aligned}
&\log \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] \overset{(a)}{=} \log \iint \frac{p(\tau,H)}{q_\theta(\tau,H)} q_\theta(\tau,H) r(s_H,a_H)\, d\tau\, dH - \log(1-\gamma) \\
&\overset{(b)}{\geq} \iint q_\theta(\tau,H)\left(\log p(\tau,H) - \log q_\theta(\tau,H) + \log r(s_H,a_H) d\tau\right) dH - \log(1-\gamma) \qquad (13)\\
&\overset{(c)}{=} \int \sum_{H=0}^{\infty} \gamma_\theta(H) q_\theta(\tau \mid H)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1}\mid s_t,a_t) - \log q_\theta(s_{t+1}\mid s_t,a_t)\right) + \log p(H) - \log\gamma_\theta(H) + \log r(s_H,a_H) d\tau\right) - \log(1-\gamma) \\
&\overset{(d)}{=} \int q_\theta(\tau \mid H=\infty) \sum_{H=0}^{\infty} \gamma_\theta(H)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1}\mid s_t,a_t) - \log q_\theta(s_{t+1}\mid s_t,a_t)\right) + \log p(H) - \log\gamma_\theta(H) + \log r(s_H,a_H) d\tau\right) - \log(1-\gamma) \\
&\overset{(e)}{=} \int q_\theta(\tau) \sum_{H=0}^{\infty} \gamma_\theta(H)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1}\mid s_t,a_t) - \log q_\theta(s_{t+1}\mid s_t,a_t)\right) + \log p(H) - \log\gamma_\theta(H) + \log r(s_H,a_H) d\tau\right) - \log(1-\gamma) \\
&\overset{(f)}{=} \mathbb{E}_{q_\theta(\tau)}\left[\sum_{H=0}^{\infty} \gamma_\theta(H)\left(\left(\sum_{t=0}^{H}\log p(s_{t+1}\mid s_t,a_t) - \log q_\theta(s_{t+1}\mid s_t,a_t)\right) + \cancel{\log(1-\gamma)} + H\log\gamma - \log\gamma_\theta(H) + \log r(s_H,a_H)\right)\right] - \cancel{\log(1-\gamma)} \\
&\overset{(g)}{=} \mathbb{E}_{q_\theta(\tau)}\left[\sum_{H=0}^{\infty}\left(\sum_{t=H}^{\infty} q(t)\right)\left(\log p(s_{H+1}\mid s_H,a_H) - \log q_\theta(s_{H+1}\mid s_H,a_H)\right) + \gamma_\theta(H)\left(H\log\gamma - \log\gamma_\theta(H) + \log r(s_H,a_H)\right)\right] \\
&\overset{(h)}{=} \mathbb{E}_{q_\theta(\tau)}\left[\sum_{H=0}^{\infty}\left(1 - \sum_{t=0}^{H-1} q(t)\right)\left(\log p(s_{H+1}\mid s_H,a_H) - \log q_\theta(s_{H+1}\mid s_H,a_H)\right) + \gamma_\theta(H)\left(H\log\gamma - \log\gamma_\theta(H) + \log r(s_H,a_H)\right)\right] \\
&\overset{(i)}{=} \mathbb{E}_{q_\theta(\tau)}\left[\sum_{H=0}^{\infty}\left(1-\Gamma_\theta(H-1)\right)\left(\log p(s_{H+1}\mid s_H,a_H) - \log q_\theta(s_{H+1}\mid s_H,a_H)\right) + \gamma_\theta(H)\left(H\log\gamma - \log\gamma_\theta(H) + \log r(s_H,a_H)\right)\right] \\
&\overset{(j)}{=} \mathbb{E}_{q_\theta(\tau)}\left[\sum_{H=0}^{\infty}\gamma_\theta(H)\left(\frac{1-\Gamma_\theta(H-1)}{\gamma_\theta(H)}\left(\log p(s_{H+1}\mid s_H,a_H) - \log q_\theta(s_{H+1}\mid s_H,a_H)\right) + H\log\gamma - \log\gamma_\theta(H) + \log r(s_H,a_H)\right)\right].
\end{aligned}
$$

For *(a)*, we applied Lemma B.2 and multiplied the integrand by $\frac{q_\theta(\tau|H=H)\gamma_\theta(H)}{q_\theta(\tau|H=H)\gamma_\theta(H)} = 1$. For *(b)*, we applied Jensen's inequality. For *(c)*, we factored $p(\tau,H) = p(\tau,H)p(H)$ and $q_\theta(\tau,H) = q(\tau \mid H)\gamma_\theta(H)$. Note that under the joint distribution $p(\tau,H)$, the horizon $H \sim p(H) = \textsc{Geom}(1-\gamma)$ is independent of the trajectory, τ. For *(d)*, we rewrote the expectation as an expectation over a single infinite-length trajectory and simplified the summand. For *(e)*, we recall the definition $q_\theta(\tau) = q_\theta(\tau = H = \infty)$. For *(f)*, we rewrote the integral as an expectation and wrote out the definition of the geometric distribution, $p(H)$. For *(g)*, we regrouped the difference of dynamics terms. For *(h)*, we noted used the fact that $\sum_{t=0}^{H-1}\gamma_\theta(t) + \gamma_{t=H}^{\infty}\gamma_\theta(t) = 1$. For *(i)*, we substituted the definition of the CDF function. For *(j)*, we rearranged terms so that all were multiplied by the discount $\gamma_\theta(H)$. Thus, we have obtained the desired result. We now prove Lemma A.1, showing that Eq. 10 becomes tight at optimality.

Proof.

$$
\begin{aligned}
\mathcal{L}_\gamma(\theta) &\overset{(a)}{=} \iint q_\theta(\tau,H)\left(\log p(\tau,H) - \log q_\theta(\tau,H) + \log r(s_H,a_H) d\tau\right) dH - \log(1-\gamma) \\
&\overset{(b)}{=} \iint q_\theta(\tau)\gamma_\theta(H\mid\tau)\left(\log p(\tau) + \log p(H) - \log q_\theta(\tau) - \log\gamma_\theta(H\mid\tau) + \log r(s_H,a_H) d\tau\right) dH - \log(1-\gamma)
\end{aligned} \qquad (14)
$$

For *(a)*, we undo some of the simplifications above, going back to Eq. 13 For *(b)*, we factor $q_\theta(\tau,H) = q_\theta(\tau)\gamma_\theta(H\mid\tau)$ and $p(\tau,H) = p(\tau)p(H)$. At this point, we can solve analytically for the optimal discount distribution, $\gamma_\theta(H\mid\tau)$:

$$
\gamma_\theta^*(H\mid\tau) = \frac{p(H)r(s_H,a_H)}{\sum_{H'=0}^{\infty} p(H')r(s_{H'},a_{H'})} = \frac{p(H)r(s_H,a_H)}{(1-\gamma)R(\tau)} \qquad (15)
$$

In the second equality, we substitute the definition of $R(\tau)$. We then substitute Eq. 15 into our expression for $\mathcal{L}_\gamma(\theta)$ and simplify the resulting expression.

$$\begin{aligned}
\mathcal{L}_\gamma(\theta) &= \iint q_\theta(\tau)\gamma_\theta(H \mid \tau)\left(\log p(\tau) + \cancel{\log p(H)} - \log q_\theta(\tau) - \cancel{\log p(H)} - \cancel{\log r(s_H, a_H)} + \cancel{\log(1-\gamma)} + \log R(\tau) + \cancel{\log r(s_H, a_H)} d\tau\right) dH - \cancel{\log(1-\gamma)} \\
&= \iint q_\theta(\tau)\gamma_\theta(H \mid \tau)\left(\log p(\tau) - \log q_\theta(\tau) + \log R(\tau) d\tau\right) dH \\
&= \int q_\theta(\tau)\left(\log p(\tau) - \log q_\theta(\tau) + \log R(\tau)\right) d\tau.
\end{aligned}$$

In the final line we have removed the integral over H because none of the integrands depend on H. At this point, we can solve analytically for the optimal trajectory distribution, $q_\theta(\tau)$:

$$q^*(\tau) = \frac{p(\tau)R(\tau)}{\int p(\tau')R(\tau')d\tau'}. \tag{16}$$

We then substitute Eq. 16 into our expression for $\mathcal{L}_\gamma(\theta)$, and simplify the resulting expression:

$$\begin{aligned}
\mathcal{L}_\gamma(\theta) &= \int q_\theta(\tau)\left(\cancel{\log p(\tau)} - \cancel{\log p(\tau)} - \cancel{\log R(\tau)} + \log \int p(\tau')R(\tau')d\tau' + \cancel{\log R(\tau)}\right) d\tau \\
&= \log \int p(\tau)R(\tau)d\tau = \log \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right].
\end{aligned}$$

We have thus shown that the lower bound $\mathcal{L}_\gamma$ becomes tight when we use the optimal distribution over trajectories $q_\theta(\tau)$ and optimal learned discount $\gamma_\theta(H \mid \tau)$. □

B.5 A lower bound for goal-reaching tasks.

Many RL problems can be better formulated as goal-reaching problems, a formulation that does not require defining a reward function. We now introduce a variant of our method for goal-reaching tasks. Using $\rho^\pi(s_{t+})$ to denote the discounted state occupancy measure of policy π, we define the goal-reaching objective as maximizing the probability density of reaching a desired goal s_g:

$$\max_\theta \log \rho^{\pi_\theta}(s_{t+} = s_g). \tag{17}$$

We refer the reader to Eysenbach et al. [16] for a more detailed discussion of this objective. For simplicity, we assume that the goal is fixed, noting that the multi-task setting can be handled by conditioning the policy on the commanded goal. Similar to Theorem 3.1, we can construct a lower bound on the goal-conditioned RL problem:

Lemma B.3. *Let initial state distribution $p_1(s_1)$, real dynamics $p(s_{t+1} \mid s_t, a_t)$, reward function $r(s_t, a_t) > 0$, discount factor $\gamma \in (0, 1)$, and goal g be given. Then the following bound holds for* any *dynamics $q(s_{t+1} \mid s_t, a_t)$ and policy $\pi(a_t \mid s_t)$:*

$$\log p^{\pi_\theta}(s_{t+} = s_g) \geq \mathbb{E}_{q^{\pi_\theta}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t \tilde{r}(s_t, a_t)\right], \tag{18}$$

where

$$\begin{aligned}
\tilde{r}_g(s_t, a_t, s_{t+1}) \triangleq &(1-\gamma)(\log p(s_{t+1} = s_g \mid s_t, a_t) - \log q(s_{t+1} = s_g \mid s_t, a_t) - \log(1-\gamma)) \\
&+ \log p(s_{t+1} \mid s_t, a_t) - \log q(s_{t+1} \mid s_t, a_t).
\end{aligned}$$

The proof, presented below, is similar to the proof of Theorem 3.1. The first term in the reward function, the log ratio of reaching the commanded goal one time step in the future, is similar to prior work [39]. The correction term $\log p - \log q$ incentivizes the policy to avoid transitions where the model is inaccurate, and can be estimated using a separate classifier. One important aspect of this goal-reaching problem is that it is entirely data-driven, avoiding the need for any manually-designed reward functions.

Proof.

$$
\begin{aligned}
\log \rho^{\pi_\theta}(s_{t+} = s_g) &= \log \iint p(H) p(\tau \mid H = H) p(s_g \mid s_H, a_H) d\tau dH \\
&= \log \iint p(H) \frac{p(\tau \mid H = H)}{q_\theta(\tau \mid H = H)} q_\theta(\tau \mid H = H) \frac{p(s_g \mid s_H, a_H)}{q_\theta(s_g \mid s_H, a_H)} q_\theta(s_g \mid s_H, a_H) d\tau dH - \log(1-\gamma) \\
&\geq \iint p(H) q_\theta(\tau \mid H = H) \left(\log p(\tau \mid H = H) - \log q_\theta(\tau \mid H = H) + \log p(s_g \mid s_H, a_H) - \log q_\theta(s_g \mid s_H, a_H)\right) d\tau dH - \log(1-\gamma) \\
&= \iint p(H) q_\theta(\tau \mid H = \infty) \left(\sum_{t=0}^{H} \log p(s_{t+1} \mid s_t, a_t) - \log q_\theta(s_{t+1} \mid s_t, a_t)\right) + \log p(s_g \mid s_H, a_H) - \log q_\theta(s_g \mid s_H, a_H) d\tau dH - \log(1-\gamma) \\
&= \int q_\theta(\tau) \int p(H) \left(\sum_{t=0}^{H} \log p(s_{t+1} \mid s_t, a_t) - \log q_\theta(s_{t+1} \mid s_t, a_t)\right) + \log p(s_g \mid s_H, a_H) - \log q_\theta(s_g \mid s_H, a_H) dH d\tau - \log(1-\gamma) \\
&= \int q_\theta(\tau) \sum_{t=0}^{\infty} \gamma^t \left(\log p(s_{t+1} \mid s_t, a_t) - \log q_\theta(s_{t+1} \mid s_t, a_t) + (1-\gamma)(\log p(s_g \mid s_t, a_t) - \log q_\theta(s_g \mid s_t, a_t)\right) d\tau - \log(1-\gamma)) \\
&= \mathbb{E}_{q_\theta(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t \left(\log p(s_{t+1} \mid s_t, a_t) - \log q_\theta(s_{t+1} \mid s_t, a_t) + (1-\gamma)(\log p(s_g \mid s_t, a_t) - \log q_\theta(s_g \mid s_t, a_t) - \log(1-\gamma)\right)\right].
\end{aligned}
$$

□

Similar to the more complex lower bound presented in Eq. 10, this lower bound on goal-reaching can be modified (by learning a discount factor) to become a tight lower bound. The resulting objective would resemble a model-based version of the algorithm from Rudner et al. [39].

B.6 Derivation of Model Objective (Eq. 9)

Our lower bound depends on entirely trajectories sampled from the learned dynamics. In this section, we show how the same objective can be expressed as an expectation of transitions. This expression is easier to optimize, as it does not require backpropagating gradients through time. We start by writing our lower bound, conditioned on a current state s_t.

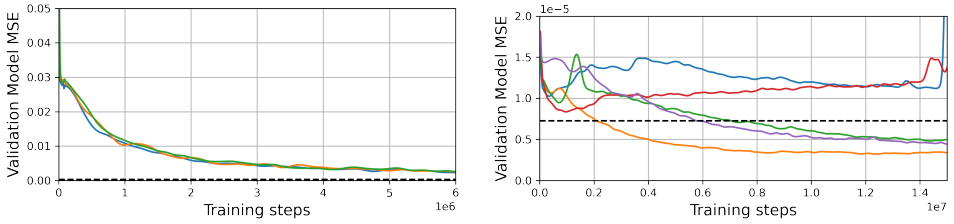
$$
\begin{aligned}
&\mathbb{E}_{\substack{\pi(a_t|s_t), \\ q_\theta(s_{t+1|s_t,a_t})}} \left[\sum_{t'=t}^{\infty} \gamma^{t'-t} \tilde{r}(s_{t'}, a_{t'}) \mid s_t\right] \\
&\qquad = \mathbb{E}_{\substack{\pi(a_t|s_t), \\ q_\theta(s_{t+1|s_t,a_t})}} \left[\tilde{r}(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}) \mid s_t\right] \\
&\qquad \overset{(a)}{=} \mathbb{E}_{\substack{\pi(a_t|s_t), \\ q_\theta(s_{t+1|s_t,a_t})}} \left[(1-\gamma)\log r(s_t, a_t) + \log \frac{C_\phi(s_t, a_t, s_{t+1})}{1 - C_\phi(s_t, a_t, s_{t+1})} - (1-\gamma)\log(1-\gamma) + \gamma V(s_{t+1}) \mid s_t\right]
\end{aligned}
$$

In *(a)*, we substituted the definition of the augmented return. For the purpose of optimizing the dynamics model, we can ignore all terms that do not depend on s_{t+1}. Removing these terms, we arrive at our model training objective (Eq. 9)

C Additional Experiments

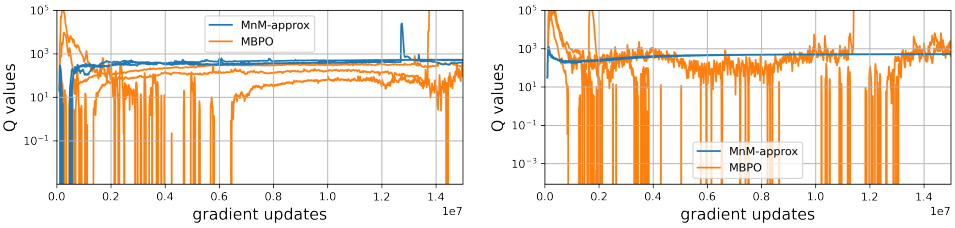
Fig. 6. We compare MnM to a number of alternative model learning methods. MBPO [24] uses a standard maximum likelihood model. We implement a version of VAML [17], which augments the maximum likelihood loss with an additional temporal difference loss; the model should predict next states that have low Bellman error. Finally, we compare to a variant of the MBPO maximum likelihood model that weights transitions based on the Q values, an idea discussed (but not actually implemented) in Lambert et al. [29]. We implement this value weighting method by computing the Q values for the current states and computing a softmax over the batch dimension to obtain per-example weights.

Fig. 8 In this experiment, we show the model MSE throughout the training of MnM-approx, similar to Fig. 7b. While MnM-approx does not optimize for MSE, the MSE is nonetheless a rough barometer for whether the model is training stably. On the `DClawScrewFixed-v0` task, we observe that $\frac{3}{3}$ seeds



(a) `DClawScrewFixed-v0` (b) `metaworld-drawer-open-v2`

Figure 8: **MnM-approx Model MSE**. Different lines show different random seeds, while the dashed horizontal line shows the minimum MSE of a maximum likelihood model (averaged across seeds).



(a) `DClawScrewFixed-v0` (b) `DClawScrewRandom-v0`

Figure 9: **MnM-approx Q-values**

all train stably, converges towards (though not quite reaching) the MSE of the maximum likelihood model. On the `metaworld-drawer-open-v2` task, the results are a bit more complicated, with $\frac{3}{5}$ seeds achieving a model MSE much lower than the maximum likelihood model, while two seeds seem to have failed to converge. Overall, these plots indicate that MnM-approx often trains stably, but some seeds can fail to converge on some environments.

Fig. 9 In this experiment, we show the Q-values throughout the training of MnM-approx, similar to Fig. 7a. In both environments, we observe that the Q-values from MnM-approx are more stable than the Q-values from MBPO. Note how the Q-values from MBPO tend to peak early during training, suggesting that they have overestimated the true returns and then correct towards a less inflated estimate of the agent's expected return

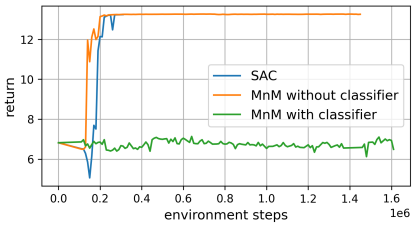


Figure 10: Adding the classifier term to MnM degrades performance on the `metaworld-drawer-open-v2` task.

Fig. 10. This is a plot from a preliminary version of MnM, when we were testing the effect of the classifier term. In experiments like these, we found that the classifier term significantly hurt performance, motivating us to not include the classifier term in the "MnM-Approx" method used in the continuous control experiments.

Fig. 11 In this experiment, we ablate the two factors of the MnM augmented reward (Eq. 3, using the stochastic gridworld from Fig. 2a. The first ablation removes the logarithm, while the second ablation removes the classifier. The results, shown in Fig. 11, indicate that the logarithm term is crucial for getting good performance, but that removing classifier term has a relative small effect, and may even boost performance by a small margin. Not to read into these results too much, the aliasing experiment in Fig. 2b has already demonstrated that the classifier term is critical for ensuring good performance in the presence of function approximation.

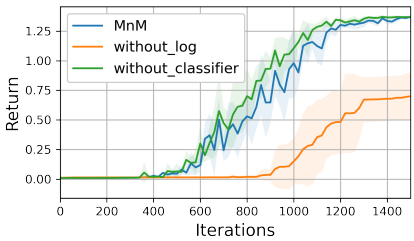


Figure 11: Stochastic gridworld ablation.

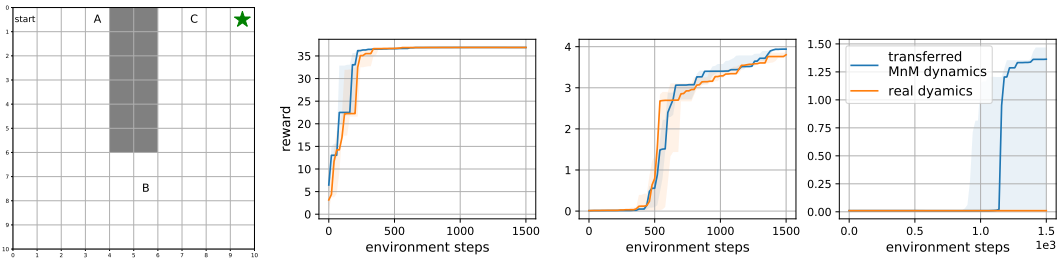


Figure 13: **Transferring the learned dynamics to a new task.** We applied Q-learning with either the true environment dynamics or with the dynamics learned by MnM on the original task (reaching the green star). The shaded region corresponds to the $[25\%, 75\%]$ region across 5 random seeds.

Fig. 12 In our experiments, we used very short model rollouts, only collecting a single transition from the model. Fig. 12 shows an ablation experiment on the `metaworld-drawer-open-v2` task where we increased the model horizon, finding that it uniformly degrades performance.

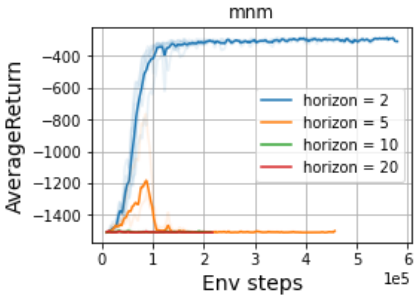


Figure 12: **Model horizon**

Fig. 13 In this experiment, we examine the transferability of MnM's learned dynamics model to new tasks. When learning a similar task, we might expect that the optimistic dynamics produced by MnM would be more useful than the true environment dynamics. However, when learning dissimilar tasks, the optimism for one task might hinder exploration towards solving the new task, and might be worse than using the true environment dynamics.

We tested these hypotheses using the stochastic gridworld from Fig. 2a. We took the dynamics learned by MnM (Fig. 2a (right)) and used it to solve new tasks, defined by placing the goal in different locations. As shown in Fig. 13 (right), tasks B and C are similar to the original task, in that they involve navigating around the wall; task A is not as similar to the original task, but is much easier.

To study the effectiveness of transferring the dynamics, we applied Q-learning to each of these tasks, either using the (incorrect) optimistic dynamics learned from the original task, or by using the true environment dynamics. We show the results in Fig. 13. On Tasks A and B, we observe little difference between Q-learning with the transferred dynamics model versus the true dynamics model. However, on task C, only by using the transferred dynamics is Q-learning able to solve this challenging task. In summary, these results suggest that the bias of the learned dynamics does not seem to hurt when learning dissimilar tasks, but can accelerate learning of challenging, similar tasks.

D Implementation Details

All experiments were run on at least three random seeds. Although we were limited by computational constraints, we find that most of our conclusions hold with $p < 0.05$, as noted in the main text.

D.1 Algorithms

Value Iteration (Fig. 2a *(right)*, Fig. 2b, Fig. 3b) For the tabular experiments that perform value iteration, we perform Polyak averaging of the policy and learned dynamics model with parameter $\tau = 0.5$. We found that the value iteration version of MnM diverged without this Polyak averaging step. Experiments were stopped when the MnM dynamics model (which depends on the value function) changed by less than 1e-6 across iterations, as measured using an L_0 norm. For VMBPO we used $\eta = 1$.

Q-learning (Fig. 2a) For the experiments with Q-learning (both with and without the MnM components), we performed ϵ-greedy exploration with $\epsilon = 0.5$. We used a learning rate of $1e-2$. For this task alone, we compute the MnM dynamics analytically by combining the true environment dynamics with the learned value function, allowing for clearer theoretical analysis. For fair comparison, all

Table 2: **Gradient updates per real environment step**: This parameter was separately tuned for each method and each environment.

	SAC	MBPO	MnM
HalfCheetah-v2	-	40	20
Hopper-v2	-	40	20
Walker2d-v2	-	40	20
metaworld-drawer-open-v2	40	40	40
DClawPoseRandom-v0	20	20	20
DClawTurnRandom-v0	40	40	40
DClawScrewFixed-v0	40	40	40
DClawScrewRandom-v0	40	40	40

methods receive the same amount of data, perform the same number of updates, and are evaluated using the real environment dynamics. For VMBPO we used $\eta = 1$.

SAC for continuous control tasks. We used the SAC implementation from TF-Agents [21] with the default hyperparameters.

MBPO for continuous control tasks. We implement MBPO on top of the SAC implementation from TF-Agents [21]. Unless otherwise mentioned, we take the default parameters from this implementation. We use an ensemble of 5 dynamics models, each with 4 hidden layers of size 256. The dynamics model predicts the whitened difference between the next state and the current state. That is, to obtain the prediction for the next state, the predictions are scaled by a per-coordinate variance, shifted by a per-coordinate mean, and then added to the current state. These whitened predictions are clipped to have a minimum standard deviation of 1e-5; without this, we found that the MBPO model resulted in numerical instability. The model is trained using the standard maximum likelihood objective, with all members of the ensemble being trained on the same data. To sample data from this model we perform 1-step rollouts, starting at states visited in the true dynamics. We perform one batch of rollouts in parallel using a batch size of 256. To sample the corresponding action, with probability 50% we take the action that was executed in the true dynamics; with probability 50% we sample an action from the current policy. We found that this modification slightly improves the results of MBPO. We use a batch size of 256. We have two replay buffers: the model replay buffer has size 256e3 and the replay buffer of real experience has size 1e6. At the start of training, we collect 1e4 transitions from the real environment, train the dynamics model on this experience for 1e5 batches, and only then start training the policy. We use a learning rate of 3e-4 for all components. To stabilize learning, we maintain a target dynamics model using an exponential moving average ($\tau = 0.001$), and use this target dynamics model to sample transitions for training. We update the model, policy, and value functions at the same rate we sample experience from the learned model, which is more frequently than we collect experience from the real environment (see Table 2).

MnM for Continuous Control Tasks We implement MnM on top of the SAC implementation from TF-Agents [21]. Unless otherwise mentioned, we take the default parameters from this implementation. Our model architecture is exactly the same as our MBPO implementation, and we follow the same training protocol.

Unlike MBPO, MnM also learns a classifier for distinguishing real versus model transitions. The classifier architecture is a 2 layers neural network with 1024 hidden units in each layer. We found that this large capacity was important for stable learning. We add input noise with $\sigma = 0.1$ while training the classifier. We whiten the inputs to the classifier by subtracting a coordinate-wise mean and dividing by a coordinate-wise standard deviation. When training the classifier, we take samples from both the dynamics model and the target dynamics model as negative examples, finding that this stabilizes learning somewhat. Following the suggestion of prior work [40], we use one-sided label smoothing with value 0.1, only smoothing the negative predictions and not the positive predictions.

We found that gradient penalties and spectral normalization decreased performance. We found that automatically tuning the classifier input noise also decreased performance. We found that mixup had little effect. We found that the loss would often plateau around 1e4 batches, but would eventually start decreasing again after 2e4 - 2e5 batches.

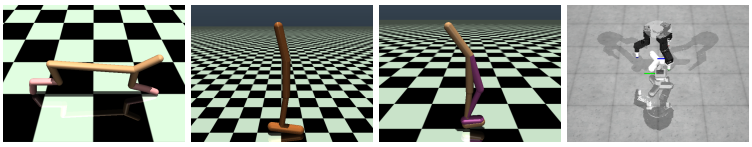


Figure 14: **Environments**: Our experiments included three tasks from OpenAI Gym and four tasks from ROBEL.

Like the MBPO model, we first collect 1e4 transitions of experience from the real environment using a random policy, then train the dynamics model and classifier for 1e5 batches, and only then start updating the policy. Because the Q values are poor at the start of training, we only add the value term to the model loss (resulting in the optimistic dynamics model) after 2e5 batches (1e5 batches of model training, then 1e5 batches of model+policy training). To further improve stability, we compute the value term in the model loss by taking the minimum over two target value functions (like TD3 [18]). We update the model, classifier policy, and value functions at the same rate we sample experience from the learned model, which is more frequently than we collect experience from the real environment (see Table 2).

D.2 Environments

This section provides details for the environments used in our experiments. We visualize the environments in Fig. 14.

Gridworld for Fig. 2a. This task is a 10×10 gridworld with obstacles shown in Fig. 2a. There are four discrete actions, corresponding to moving to the four adjacent cells. With probability 50%, the agent's action is ignored and a random action is taken instead. The agent starts in the top-left cell. The agent receives a reward of +0.001 at each time step and a reward of +10 when at the goal state. Episodes have 200 steps and we use a discount $\gamma = 0.9$.

Gridworld for Fig. 2b. This task is a 15×15 gridworld with obstacles shown in Fig. 2b. There are four discrete actions, corresponding to moving to the four adjacent cells. The dynamics are deterministic. The agent starts in the top-left cell. The reward is +1.0 at every state except the goal state, where the agent receives a reward of +100.0. We use $\gamma = 0.9$ and compute optimal policies analytically using value iteration. We implement the aliasing by averaging together the dynamics for each block of 3×3 states. Importantly, the averaging was done to the *relative* dynamics (e.g., action 1 corresponds to move right) not the *absolute* dynamics (e.g., action 1 corresponds to moving to state (3, 4)). To handle edge effects, we modified the averaged dynamics so that the agent could not exit the gridworld. We computed the classifier analytically using the true and learned dynamics models. However, the augmented rewards become infinite because the learned model assigns non-zero probability to transitions that cannot occur under the true dynamics. This is not a failure of our theory, as the optimal policy would choose to never visit these states, but it presents a challenge for optimization. We therefore added label smoothing with parameter 0.7 to the classifier.

Gridworld for Fig. 2a This task is a 10×10 gridworld with obstacles shown in Fig. 2a *(Right)*. There are four discrete actions, corresponding to moving to the four adjacent cells. With probability 90%, the agent's action is ignored and a random action is taken instead. The agent starts in the top-left cell. The rewards depend on the Manhattan distance to the goal: transitions that lead away from the goal have a reward of +0.001, transitions that do not change the distance to the goal have a reward of +1.001, and transitions that decrease the distance to the goal have a reward of +2.001. We use $\gamma = 0.5$ and compute values and returns analytically using value iteration.

Gridworld for Fig. 3b. This task used the same dynamics as Fig. 2a. The one change is that the agent receives a reward of +1.0 at each time step and a reward of +10.0 at the goal state. We estimate the more complex lower bound (Eq. 10) by also learning the discount factor. However, since we currently do not have a method for learning non-Markovian dynamics to fully optimize this lower bound, we do not expect the lower bound to become tight.

`HalfCheetah-v2, Hopper-v2, Walker2d-v2`. These tasks are taken directly from the OpenAI benchmark [7] without modification.

`metaworld-drawer-open-v2`. This task is based on the `drawer-open-v2` task from the Metaworld benchmark [50]. To increase the difficulty of this task, we remove the reward shaping term (`reward_for_caging`) and just optimize the reward for opening the drawer (`reward_for_opening`).

`DClawPoseRandom-v0`, `DClawTurnRandom-v0`, `DClawScrewFixed-v0`, `DClawScrewRandom-v0`. These tasks are taken directly from the ROBEL benchmark [1] without modification.

D.3 Differences between MnM and MnM-Approx

Both MnM and MnM-Approx learn the same dynamics classifier, and both use the same GAN-like objective (Eq. 9) to update the model. Both perform the same policy updates. The difference is the reward function used to update the Q-function:

$$\tilde{r}(s_t, a_t, s_{t+1}) = (1-\gamma)\log r(s_t, a_t) + \log\left(\frac{p(s_{t+1} \mid s_t, a_t)}{q(s_{t+1} \mid s_t, a_t)}\right) \tag{MnM}$$

$$\tilde{r}(s_t, a_t, s_{t+1}) = r(s_t, a_t). \tag{MnM-Approx}$$