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

Anonymous Author(s)
Affiliation
Address
email

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

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

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].

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. 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.

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.

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 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). 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 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:

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

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 transi-

tions. 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.

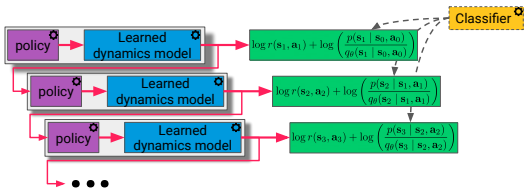


Figure 1: **Mismatched No More** is a model-based RL algorithm that learns a policy, dynamics model, and classifier. The classifier distinguishes real transitions from model transitions. The policy and dynamics are jointly optimized to sample transitions that yield high returns and look realistic, as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

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

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:

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

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 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.

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 $\pi_\theta(a_t \mid s_t)$.

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.

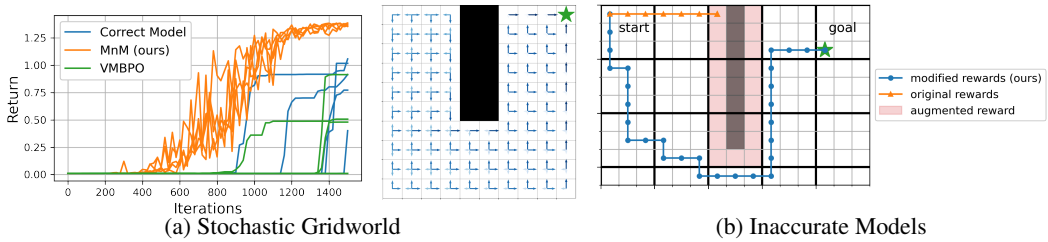
Algorithm summary. We summarize the method in Alg. 1 and provide an illustration in Fig. 1. We call the method MISMATCHED NO MORE (MnM) because the policy and model optimize the same objective, thereby resolving the objective mismatch problem noted in prior work. While the model and policy are optimized 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

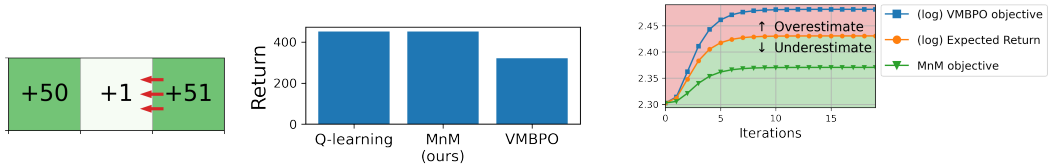
The classifier term in the augmented reward (Eq. 3) is a double edged sword. Our theoretical derivation suggests that this term should appear, and our didactic experiments demonstrate that removing this term results in suboptimal behavior. Experiments also show that this term can effectively combat errors 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. This makes sense: this classifier term is exactly the opposite 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].



(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.



(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.

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.

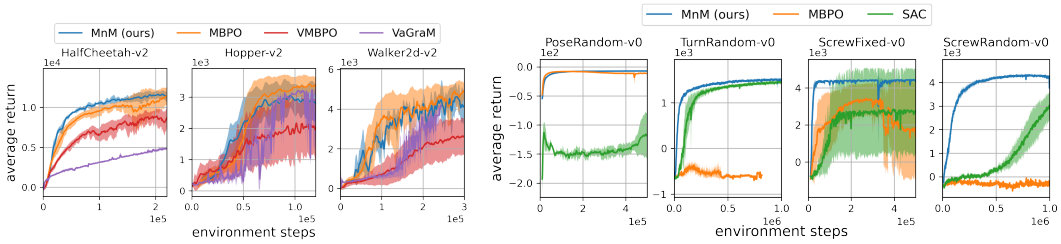
5.1 Understanding the Lower Bound and the Learned Dynamics

Our first 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



(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].

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 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.

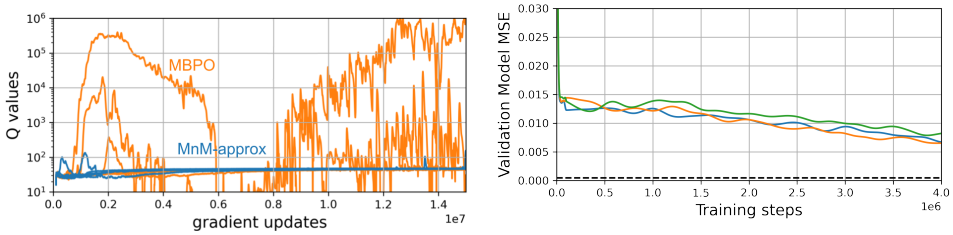
Finally, we verify Theorem 3.1 by comparing the MnM objective to the true expected return. We also compare to the objective from VMBPO, which looks similar to the MnM objective but omits the logarithmic transformation; our theory predicts that the VMBPO objective will therefore be an *upper* bound on 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.

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.

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



(a) Model exploitation

(b) MnM-approx trains stably.

Figure 6: **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.

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. 5, these alternative model learning objectives perform poorly on this task.

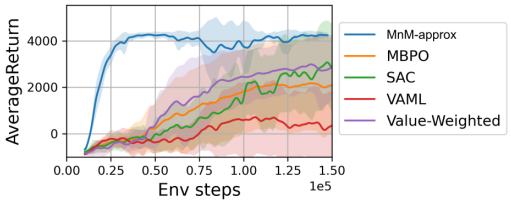


Figure 5: **Alternative model learning objectives**: Using the `DClawScrewFixed-v0` task, we compare MnM-approx and MBPO [24] to two additional model learning objectives suggested in the literature, VAML [17] and value-weighted maximum likelihood [29]. MnM-approx outperforms these alternative approaches.

To 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. 6a 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.

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. 6b, the MSE decreases stably, indicating that the adversarial nature of the MnM-approx model training does not create instabilities. In Appendix C (Fig. 7), we visualize the dynamics model learned by the deep RL implementation of MnM-approx. As in Fig. 2a, we find that the dynamics are optimistic, biasing the agent towards outcomes with high rewards.

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.

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]