Discriminator Augmented Model-Based Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

By planning through a learned dynamics model, model-based reinforcement learning (MBRL) offers the prospect of good performance with little environment interaction. However, it is common in practice for the learned model to be inaccurate, impairing planning and leading to poor performance. This paper aims to improve planning with an importance sampling framework that accounts and corrects for discrepancy between the true and learned dynamics. This framework also motivates an alternative objective for fitting the dynamics model: to minimize the variance of value estimation during planning. We derive and implement this objective, which encourages better prediction on trajectories with larger returns. We observe empirically that our approach improves the performance of current MBRL algorithms on two stochastic control problems, and provide a theoretical basis for our method.

1 Introduction

Model free reinforcement learning methods have achieved good performance on a number of complex tasks, but usually require a large amount of data collected through environment interaction. Model-based reinforcement learning (MBRL) can potentially reduce these data requirements by fitting a model of the environment dynamics on a small dataset, then planning through the learned dynamics to produce a good policy. However, inaccurate models can severely impair planning performance. Recent techniques attempt to avoid compounding model error by restricting the number of model unrolling steps, but this creates a trade-off between planning performance and sample efficiency. More importantly, such approaches assume that the learned model mostly matches the true (one timestep) dynamics, and only address compounding error that arises over many timesteps. In this paper we show that there are a range of environments where the learned model can be inaccurate even on short time horizons (e.g., one timestep), so merely mitigating compounding error is insufficient. Given the evidence that even deep neural network models can struggle to learn complex, high dimensional distributions, we expect inaccurate learned dynamics to be an increasingly important issue as we apply MBRL to harder and more realistic environments.

Consider a learned dynamics model as a generative model from which we sample transitions. Even if the model is inaccurate, some samples may be better than others. Our approach trains a discriminative model to assess the quality of sampled transitions during planning, and upweight or downweight value estimates computed from high and low quality samples, respectively. In section we also show that this method is a form of likelihood-free importance sampling that, assuming the discriminator is an optimal classifier, produces unbiased value estimates for planning. Since our discriminator can correct model error during planning, we are no longer restricted to fitting the dynamics using a maximum likelihood objective. Instead, we can learn biased dynamics models with advantageous properties, such as reduced value estimation variance during planning. We derive and implement an
Algorithm 1: \textsc{EstimateValue} produces an estimated return $R$ of some action sequence using a learned model $q$. Since $q$ may be biased with respect to the true dynamics, it uses a discriminator $D$ to correct the estimate as described in Sec. 5.1.

In this work, we look at some of the fundamental components of model-based reinforcement learning. In Section 3.1, we discuss an often ignored discrepancy caused by sampling trajectories from learned dynamics models and discuss how this bias can be corrected using importance sampling. Then, in Section 3.2, we discuss how interpreting the model-based RL in an importance sampling framework suggests a novel objective for learning models for environment dynamics.
3.1 Correcting the Sampling Bias in Model-Based RL

Consider our optimization objective $J(\pi) = \mathbb{E}_{p(\tau)}[R(\tau)]$. Using the approximate dynamics $q$ introduces a bias in our estimation of $J(\pi)$. Model-based RL methods generally use $\hat{J}_q(\pi) = \mathbb{E}_{q(\tau)}[R(\tau)]$ as a proxy for $J(\pi)$, where the trajectories are sampled from $q(\tau)$ instead of $p(\tau)$ without accounting for the resulting bias. In this work, we mitigate this bias by using the importance sampling framework:

$$\mathbb{E}_{p(\tau)}[R(\tau)] = \mathbb{E}_{q(\tau)} \left[ \frac{p(\tau)}{q(\tau)} R(\tau) \right] = \mathbb{E}_{q(\tau)} \left[ w(\tau) R(\tau) \right]$$

where we have introduced the importance sampling correction $w(\tau)$ to correct for the bias introduced by sampling trajectories from $q(\tau)$. Simplifying $w(\tau)$, we get

$$w(\tau) = \prod_{t=1}^{T-1} \frac{p(s_{t+1} | s_t, a_t)}{q(s_{t+1} | s_t, a_t)} = \prod_{t=1}^{T-1} w(s_{t+1}, a_t, s_t)$$

By exploiting the MDP’s temporal structure, we can obtain an improved importance sampling estimator that weights the per-timestep rewards instead of the trajectory return. Comparing to Eq. 2 we see that this estimator multiplies the reward at time $t$ by a weight that only depends on transitions leading up to it, and not transitions that occur after it:

$$\mathbb{E}_{p(\tau)}[R(\tau)] = \sum_{t=1}^{T} \mathbb{E}_q \left[ \left( \prod_{m=1}^{t} w(s_m, a_m, s_{m+1}) \right) r(s_t, a_t) \right]$$

As stated earlier, we only have access to the samples from the transition dynamics and not the probabilities. Thus to compute $w(\tau)$ for $\tau \sim q$, we use techniques from density ratio estimation. We setup a binary classification problem over $D : S \times A \times S \times Y \rightarrow \{0, 1\}$, where $Y^c = \{0, 1\}$. Using the transition samples from the environment and the learned dynamics, we create a dataset of transitions $\{s', a, s, y\}$, where $y = 1$ for $(s', a, s) \sim p(s' | s, a)p(s, a)$ and $y = 0$ for $(s', a, s) \sim q(s' | s, a)p(s, a)$. Effectively, we are training a discriminator to distinguish between real transitions in the environment and the transitions sampled from the learned model. Alg. 2 provides the pseudocode for training the discriminator.

To connect the discriminator to the importance weights, consider the Bayes-optimal classifier for this problem:

$$\mathcal{D}(s', a, s) = \frac{p(s' | s, a)p(s, a)}{p(s' | s, a)p(s, a) + q(s' | s, a)p(s, a)}$$

$$= \frac{p(s' | s, a)}{p(s' | s, a)/q(s' | s, a) + 1}$$

$$= \sigma \left( \log \frac{p(s' | s, a)}{q(s' | s, a)} \right) = \sigma \left( \tilde{\mathcal{D}}(s', a, s) \right)$$

$\tilde{\mathcal{D}}$ is the classifier logits, and we can recover the importance weight by exponentiating them:

$$w(s', a, s) = \frac{p(s', a, s)}{q(s', a, s)} = \exp \left( \tilde{\mathcal{D}}(s', a, s) \right)$$

For our transitions $\{s', a, s\}$, it is important to sample $s, a \sim p(s, a)$ (that is, real state-action pairs) so that the logits can be connected to the importance weights. Since, we only need samples to train the classifier, this allows us to estimate the importance weights without requiring probabilities from underlying transition dynamics. Alg. 3 provides the pseudocode for estimating the discriminator-corrected return of an action sequence, which forms the subroutine for choosing actions in sampling-based planning algorithms.

The benefit of this approach arises from the potential simplicity of the discriminative problem compared to the modelling the dynamics. The dynamics can be hard to model in the real world, as they often involve higher dimensional spaces (like RGB images) and can be multi-modal due to partial observability. The modelling and optimization challenges can often result in an incorrectly learned model, an example of which is shown in Fig 3. On the other hand, discrimination can be
an easier problem as fake and real samples can be well separated, especially in higher-dimensional
spaces. This can improve the value estimation by down-weighting samples which do not fit with the
real world samples. If the learned dynamics are indeed correct, the true and fake samples will be
indistinguishable and the importance sampling weights will be close to unity, reducing the problem
to conventional model-based RL.

### 3.2 Minimum Variance Dynamics Model Fitting

The above discussion applies to any learned model, including the conventional approach of learning
by maximizing the log-likelihood of transitions collected in the environment. The policy objective
under $q$, that is $J_q(\pi) = \frac{1}{N} \sum_{i=1}^{N} w(\tau_i) R(\tau_i)$, is an unbiased estimator of $J(\pi) \forall q$ as long as the
support of $p$ is a subset of support of $q$. However, from an optimization perspective, it is desirable
to have an estimator with the minimum variance. Sampling based planning algorithms generate
candidate action sequences and rank them according to their value function. Minimizing the variance
of the value estimate of an action sequence increases the likelihood of the action sequences getting
ranked correctly, and thus, the best action sequence getting chosen. Thus, it is favorable to reduce the
variance of our estimate $\hat{J}_q$. Assuming that the $R(\tau) \geq 0$, it can be shown that

$$q^*(\tau) = \frac{R(\tau) p(\tau)}{E_p(\tau) [R(\tau)]}$$

(9)

Here, $q^*$ denotes the sampling distribution which minimizes the variance of $\hat{J}_q$. While the denominator
for the optimal sampling distribution is intractable, the important distinction here is that trajectories
should be sampled in accordance to their density under the true trajectory distribution and the return
accumulated by the trajectory. Using the $q^*$ as the target distribution, we can setup our minimization
objective for model-based learning to be $\text{KL}(q^*(\tau) || q(\tau))$. As we show in Appendix A.1, the
gradient with respect to $q$ is:

$$\nabla_q \text{KL}(q^*(\tau) || q(\tau)) \propto - \sum_{t=1}^{T-1} E_p(\tau) [R(\tau) \nabla_q \log q(s_{t+1} | s_t, a_t)]$$

(10)

Intuitively, this corresponds to upweighting the gradient for trajectories with higher return, forcing
the dynamics model to fit better to transitions with higher return. In particular, this highlights that
the conventional approach of learning dynamics does not represent the optimal approach in the
importance sampling framework. The modified model training algorithm is described in Alg. 4 in the
appendix.

### 3.3 Planning and Training Loop

The previous two sections describe how to train the dynamics model, and how to train a discriminator
that corrects the sampling bias from that learned model. Further, ESTIMATEVALUE (Alg. 1) shows
how to implement discriminator-corrected value estimation. We can plug ESTIMATEVALUE as a
subroutine into most sampling based planners. Once a planner is in place, we can use it execute
actions in the real environment, collect more data, and re-train our model and discriminator. In
Appendix B, we discuss in detail the implementation of this model-based RL loop within our model
and discriminator training framework.

### 4 Experiments

In this section we evaluate our proposed approach and compare with model-based baselines. We
hypothesize that environments with multi-modal dynamics make model fitting difficult, which would
pose problems for planning. Indeed, we observe that our proposed approach improves performance
over standard model-based RL in such settings.

We devise and implement two environments with stochastic, multi-modal dynamics: IcyRoad and
Intersection. In IcyRoad the agent drives a car along an icy road, and above a threshold speed the car
has a probability of swerving off the road in either direction (hence the multi-modality) and incurring
a large negative reward. In Intersection the agent drives a car into an intersection. An oncoming car
is also entering the intersection and will turn either left or right with equal probability—the outcome is
which is unimodal. Since our environment dynamics are multimodal, it also makes sense to try fitting
we also evaluate variants that use only one aspect of our method, but not both.

Table 1: Average return for each method in the IcyRoad and Intersection environments. Columns are
variance objective without discriminator-corrected planning, is often
but it
minimum variance objective to discriminator-corrected planning does
already increases performance over the baseline. Fig. 2 depicts our analysis of what the trained model
and discriminator are doing in IcyRoad: we see that the unimodal model is struggling to fit the true
dynamics, while the discriminator is helping to correct the model error. Interestingly, adding the
maximum likelihood training, which can be harmful without discriminator-corrected planning. When
hypothesize this occurs because the minimum variance objective increases model bias versus standard
data collection in the environment. We see that in both environments,
results.

<table>
<thead>
<tr>
<th>Average return</th>
<th>IcyRoad</th>
<th>Intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unimodal</td>
<td>Multimodal</td>
</tr>
<tr>
<td>Baseline</td>
<td>0.71 ± 1.72</td>
<td>5.34 ± 0.84</td>
</tr>
<tr>
<td>DAM (minvar only)</td>
<td>0.86 ± 0.79</td>
<td>4.55 ± 3.50</td>
</tr>
<tr>
<td>DAM (disc only)</td>
<td>4.88 ± 0.50</td>
<td>3.30 ± 2.75</td>
</tr>
<tr>
<td>DAM</td>
<td>2.32 ± 1.76</td>
<td>7.94 ± 0.92</td>
</tr>
</tbody>
</table>

Table 1: Average return for each method in the IcyRoad and Intersection environments. Columns are
split by whether they use a unimodal (Gaussian) or multimodal (mixture density network) learned
dynamics model. Combining our method DAM with multimodal learned dynamics models usually
performs best among different methods we evaluated. Results show mean and standard error across 3
runs (random seeds) for each hyperparameter setting.

unknown in advance to the agent. Fig. 2 illustrates the Intersection environment, and Appendix C
both environments in more detail.

For our baseline, we fit a dynamics model using the usual maximum likelihood loss and use a
standard random shooting planner with uncorrected value estimates. In DAM we use both aspects of
our method: we train the model using the minimum variance objective (Eq. 10) and plan using the
discriminator-corrected estimator (Eq. 4). To disentangle the effects of each component of our method,
we also evaluate variants that use only one aspect of our method, but not both. DAM (disc only) only
does discriminator-corrected planning, while DAM (minvar only) only uses the minimum variance
object. Practitioners typically train models that output the parameters for a Gaussian distribution,
which is unimodal. Since our environment dynamics are multimodal, it also makes sense to try fitting
multimodal models such as mixture density networks [4]. We test each of our planning and training
methods with both a mixture density network model and a standard Gaussian model, and display both
results.

Table 1 depicts the results after 10 rounds of interleaving model (+ discriminator) training with
data collection in the environment. We see that in both environments, DAM using multimodal
dynamics models obtains the best results. DAM (minvar only), which only uses the minimum
variance objective without discriminator-corrected planning, is often worse than the baseline. We
hypothesize this occurs because the minimum variance objective increases model bias versus standard
maximum likelihood training, which can be harmful without discriminator-corrected planning. When
using unimodal models in either environment, we see that simply having a discriminator for planning
already increases performance over the baseline. Fig. 4 depicts our analysis of what the trained model
and discriminator are doing in IcyRoad: we see that the unimodal model is struggling to fit the true
dynamics, while the discriminator is helping to correct the model error. Interestingly, adding the
minimum variance objective to discriminator-corrected planning does not help with unimodal models,
but it does help with multimodal models.

Figure 2: Left: Depiction of different scenarios in the Intersection environment, where the agent
controls the silver car. Both cars are entering the intersection at the same time, but the agent does
not know whether the oncoming green car will turn left (left panel) or turn right (right panel). If the
agent drives into the intersection too fast, it may collide. Right: The learned (unimodal) model and
ground truth distributions \( p(y_{t+1} | s_t, a_t) \), where \( s_t = (2, 0, 2, 0) \) and the action is to accelerate. After
training with maximum likelihood, the unimodal model erroneously places large probability mass on
unrealistic regions of the state space. The discriminator produces importance weights (black arrows,
log scale) that downweight the model’s samples in the unrealistic regions.

Figure 2: The learned (unimodal) model and
ground truth distributions \( p(y_{t+1} | s_t, a_t) \), where \( s_t = (2, 0, 2, 0) \) and the action is to accelerate. After
training with maximum likelihood, the unimodal model erroneously places large probability mass on
unrealistic regions of the state space. The discriminator produces importance weights (black arrows,
log scale) that downweight the model’s samples in the unrealistic regions.

Table 1: Average return for each method in the IcyRoad and Intersection environments. Columns are
split by whether they use a unimodal (Gaussian) or multimodal (mixture density network) learned
dynamics model. Combining our method DAM with multimodal learned dynamics models usually
performs best among different methods we evaluated. Results show mean and standard error across 3
runs (random seeds) for each hyperparameter setting.

Table 1: Average return for each method in the IcyRoad and Intersection environments. Columns are
split by whether they use a unimodal (Gaussian) or multimodal (mixture density network) learned
dynamics model. Combining our method DAM with multimodal learned dynamics models usually
performs best among different methods we evaluated. Results show mean and standard error across 3
runs (random seeds) for each hyperparameter setting.
5 Related Work

Model-based RL has a rich history in control and robotics. Classically, the methods for model-based control have assumed access to underlying dynamics \[35,22,26,24\] or make simplifying assumptions about the environment dynamics \[21,18\]. While such simplifying assumptions can yield exact or efficient controllers, the models cannot faithfully represent the underlying dynamics, especially when considering high dimensional state spaces like RGB images. Recent work has focused on combining high-capacity function approximators such as neural networks with model-based reinforcement learning \[20,27,28,6,12,32,14,29\], which employ sampling-based planning methods \[38,27,31\].

Concurrently, some recent work has integrated policy networks with deep models \[16,37\] in spirit of Dyna \[34\]. Neural networks allow more flexible representation of non-linear dynamics, allowing model-based RL to scale to complex high dimensional tasks \[17,28\]. However, despite the incredible progress, none of these prior works account for the bias introduced by using learned models for planning.

Classically, the field of robust control has looked at designing controllers for with uncertain transition models \[3,9\]. More relevantly, \[11\] incorporates bias correction for approximate transition models to learn better policies. However, they only focus on approximation error which arises due to the use of simple linear models. More recently, \[19\] discusses the mismatch which arises between the training objectives and the control objectives, and how that discrepancy can lead to biases in model learning. In contrast to these works, our work focuses on the sampling bias introduced despite using flexible function approximators like neural networks. To some extent, our work also addresses the bias discussed in \[19\], as our objective encourages the transition model to fit better to trajectories with higher return.

Prior work has considered estimating value functions using transition models \[15,11,5\]. However, our work addresses the biased values generated by the use of learned transition models. To that extent, we rely on classifier-based density ratio estimation techniques \[33\], which have been employed in generative modelling \[36,13\]. \[10\] use similar principles to correct for differences in dynamics when transferring experience between environments, where the classifier’s corrections are realized by adjusting the reward appropriately before applying model-free RL. To the best of our knowledge, ours is first work which raises and addresses the issue of sampling bias in context of model-based RL.

6 Conclusion

We introduce a framework for mitigating the impact of model error on planning in model-based RL. Our framework trains discriminators to correct value estimation bias, while also introducing a novel model training objective for minimizing value estimation variance during planning. Empirically, we find that these modifications improve agent performance in environments with stochastic and multi-modal dynamics that pose a challenge for standard model fitting techniques. By improving planning in environments that are difficult to model properly, this framework is a step towards more efficient and performant reinforcement learning in complex, real world problems.

References


