Self-Correcting Models for Model-Based Reinforcement Learning

Erik Talvitie
Department of Mathematics and Computer Science
Franklin & Marshall College
Lancaster, PA 17604-3003
erik.talvitie@fandm.edu

Abstract

When an agent cannot represent a perfectly accurate model of its environment's dynamics, model-based reinforcement learning (MBRL) can fail catastrophically. Planning involves composing the predictions of the model; when flawed predictions are composed, even minor errors can compound and render the model useless for planning. Hallucinated Replay (Talvitie 2014) trains the model to "correct" itself when it produces errors, substantially improving MBRL with flawed models. This paper theoretically analyzes this approach, illuminates settings in which it is likely to be effective or ineffective, and presents a novel error bound, showing that a model's ability to self-correct is more tightly related to MBRL performance than one-step prediction error. These results inspire an MBRL algorithm for deterministic MDPs with performance guarantees that are robust to model class limitations.

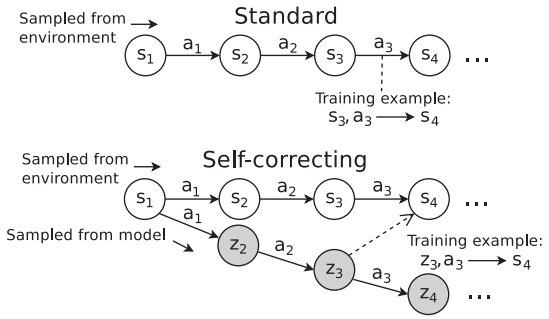

Figure 1: Top: training models to predict environment states from environment states. Bottom: training models to predict environment states from states sampled from the model.

1 Introduction

In model-based reinforcement learning (MBRL) the agent learns a predictive model of its environment and uses it to make decisions. The overall MBRL approach is intuitively appealing and there are many anticipated benefits to learning a model, most notably sample efficiency (Szita and Szepesvári 2010). Despite this, with few exceptions (e.g. Abbeel, Quigley, and Ng 2006), model-free methods have been far more successful in large-scale problems. Even as model-learning methods demonstrate increasing prediction accuracy in high-dimensional domains (e.g. Bellemare, Veness, and Talvitie 2014, Oh et al. 2015) this rarely corresponds to improvements in control performance.

One key reason for this disparity is that model-free methods are generally robust to representational limitations that prevent convergence to optimal behavior. In contrast, when the model representation is insufficient to perfectly capture the environment's dynamics (even in seemingly innocuous ways), or when the planner produces suboptimal plans, MBRL methods can fail catastrophically. If the benefits of MBRL are to be gained in large-scale problems, it is vital to understand how MBRL can be effective even when the model and planner are fundamentally flawed.

Recently there has been growing awareness that the standard measure of model quality, one-step prediction accuracy, is an inadequate proxy for MBRL performance. For instance, Sorg, Lewis, and Singh (2010) and Joseph et al. (2013) both pointed out that the most accurate model by this measure is not necessarily the best for planning. Both proposed optimizing model parameters for control performance using policy gradient methods. Though appealing in its directness, this approach arguably discards some of the benefits of learning a model in the first place.

Talvitie (2014) pointed out that one-step prediction accuracy does not account for how the model behaves when composed with itself and introduced the Hallucinated Replay meta-algorithm to address this. As illustrated in Figure 1, this approach rolls out the model and environment in parallel, training the model to predict the correct environment state (s_4) even when its input is an incorrect sampled state (z_3). This effectively causes the model to "self-correct" its rollouts. Hallucinated Replay was shown to enable meaningful planning with flawed models in examples where the standard approach failed. However, it offers no theoretical guarantees. Venkatraman, Hebert, and Bagnell (2015) and Oh et al. (2015) used similar approaches to improve models' long-range predictions, though not in the MBRL setting.

This paper presents novel error bounds that reveal the theoretical principles that underlie the empirical success of Hallucinated Replay. It presents negative results that identify settings where hallucinated training would be ineffective (Section 2.3) and identifies a case where it yields a

tighter performance bound than standard training (Section 2.4). This result allows the derivation of a novel MBRL algorithm with theoretical performance guarantees that are robust to model class limitations (Section 3). The analysis also highlights a previously underexplored practical concern with this approach, which is examined empirically (Section 4.1).

1.1 Notation and background

We focus on *Markov decision processes* (MDP). The environment's initial state s_1 is drawn from a distribution μ. At each step t the environment is in a state s_t. The agent selects an action a_t which causes the environment to transition to a new state sampled from the transition distribution: $s_{t+1} \sim P_{s_t}^{a_t}$. The environment also emits a reward, $R(s_t, a_t)$. For simplicity, assume that the reward function is known and is bounded within $[0, M]$.

A *policy* π specifies a way to behave in the MDP. Let $\pi(a \mid s) = \pi_s(a)$ be the probability that π chooses action a in state s. For a sequence of actions $a_{1:t}$ let $P(s' \mid s, a_{1:t}) = P_s^{a_{1:t}}(s')$ be the probability of reaching s' by starting in s and taking the actions in the sequence. For any state s, action a, and policy π, let $D^t_{s,a,\pi}$ be the state-action distribution obtained after t steps, starting with state s and action a and thereafter following policy π. For a state action distribution ξ, let $D^t_{\xi,\pi} = \mathbf{E}_{(s,a)\sim\xi} D^t_{s,a,\pi}$. For a state distribution μ let $D^t_{\mu,\pi} = \mathbf{E}_{s\sim\mu, a\sim\pi_s} D^t_{s,a,\pi}$. For some discount factor $\gamma \in [0, 1)$, let $D_{\mu,\pi} = (1-\gamma)\sum_{t=1}^{\infty} \gamma^{t-1} D^t_{\mu,\pi}$ be the infinite-horizon *discounted* state-action distribution under policy π.

The T-step *state-action value* of a policy, $Q^\pi_T(s, a)$ represents the expected discounted sum of rewards obtained by taking action a in state s and executing π for an additional $T-1$ steps: $Q^\pi_T(s,a) = \sum_{t=1}^{T} \gamma^{t-1} \mathbf{E}_{(s',a')\sim D^t_{s,a,\pi}} R(s', a')$. Let the T-step *state value* $V^\pi_T(s) = \mathbf{E}_{a\sim\pi_s}[Q^\pi_T(s,a)]$. For infinite horizons we write $Q^\pi = Q^\pi_\infty$, and $V^\pi = V^\pi_\infty$. The agent's goal will be to learn a policy π that maximizes $\mathbf{E}_{s\sim\mu}[V^\pi(s)]$.

The MBRL approach is to learn a model $\hat{P}$, approximating P, and then to use the model to produce a policy via a planning algorithm. We let $\hat{D}$, $\hat{Q}$, and $\hat{V}$ represent the corresponding quantities using the learned model. Let $\mathcal{C}$ represent the *model class*, the set of models the learning algorithm could possibly produce. Critically, in this paper, it is not assumed that $\mathcal{C}$ contains a perfectly accurate model.

2 Bounding value error

We consider an MBRL architecture that uses the simple one-ply Monte Carlo planning algorithm (one-ply MC), which has its roots in the "rollout algorithm" (Tesauro and Galperin 1996). For every state-action pair (s, a), the planner executes N T-step "rollouts" in $\hat{P}$, starting at s, taking action a, and then following a *rollout policy* ρ. Let $\bar{Q}(s,a)$ be the average discounted return of the rollouts. For large N, $\bar{Q}$ will closely approximate $\hat{Q}^\rho_T$ (Kakade 2003). The agent will select its actions greedily with respect to $\bar{Q}$. Talvitie (2015) bounds the performance of one-ply MC in terms of model quality.

For a policy π and state-action distribution ξ, let $\epsilon^{\xi,\pi,T}_{val} = \mathbf{E}_{(s,a)\sim\xi}\left[|Q^\pi_T(s,a) - \hat{Q}^\pi_T(s,a)|\right]$ be the error in the T-step state-action values the model assigns to the policy under the given distribution. Then the following result can be straightforwardly adapted from one provided by Talvitie (2015).

Lemma 1. *Let $\bar{Q}$ be the state-action value function returned by applying one-ply Monte Carlo to the model $\hat{P}$ with rollout policy ρ and rollout depth T. Let $\hat{\pi}$ be greedy w.r.t. $\bar{Q}$. For any policy π and state-distribution μ,*

$$\mathop{\mathbf{E}}_{s\sim\mu}\left[V^\pi(s) - V^{\hat{\pi}}(s)\right] \leq \frac{4}{1-\gamma}\epsilon^{\xi,\rho,T}_{val} + \epsilon_{mc},$$

where we let $\xi(s,a) = \frac{1}{2}D_{\mu,\hat{\pi}}(s,a) + \frac{1}{4}D_{\mu,\pi}(s,a) + \frac{1}{4}\left((1-\gamma)\mu(s)\hat{\pi}_s(a) + \gamma\sum_{z,b} D_{\mu,\pi}(z,b)P^b_z(s)\hat{\pi}_s(a)\right)$ and $\epsilon_{mc} = \frac{4}{1-\gamma}\|\bar{Q} - \hat{Q}^\rho_T\|_\infty + \frac{4}{1-\gamma}\|\hat{Q}^\rho_T - \hat{Q}^\rho\|_\infty + \frac{2}{1-\gamma}\|BV^\rho - V^\rho\|_\infty$ (here B is the Bellman operator).

The ϵ_{mc} term represents error due to limitations of the planning algorithm: error due to the sample average $\bar{Q}$, the limited rollout depth T, and the sub-optimality of ρ. The $\epsilon^{\xi,\rho,T}_{val}$ term represents error due to the model parameters. The key factor in the model's usefulness for planning is the accuracy of the value it assigns to the rollout policy in state-actions visited by π and $\hat{\pi}$. Our goal in the next sections is to bound $\epsilon^{\xi,\rho,T}_{val}$ in terms of measures of model accuracy, ultimately deriving insight into how to train models that will be effective for MBRL. Proofs may be found in the appendix.

2.1 One-step prediction error

Intuitively, the value of a policy should be accurate if the model is accurate in states that the policy would visit. We can adapt a bound from Ross and Bagnell (2012).

Lemma 2. *For any policy π and state-action distribution ξ,*

$$\epsilon^{\xi,\pi,T}_{val} \leq \frac{M}{1-\gamma}\sum_{t=1}^{T-1}(\gamma^t - \gamma^T) \mathop{\mathbf{E}}_{(s,a)\sim D^t_{\xi,\pi}}\left[\|P^a_s - \hat{P}^a_s\|_1\right].$$

Combining Lemmas 1 and 2 yields an overall bound on control performance in terms of the model's prediction error. This result matches common MBRL practice; it recommends minimizing the model's one-step prediction error. It acknowledges that the model may be imperfect by allowing it to have one-step error in unimportant (i.e. unvisited) states. However, if limitations of the model class prevent the model from achieving low error in important states, this bound can be quite loose, as the following example illustrates.

Consider the "Shooter" domain introduced by Talvitie (2015), pictured in Figure 2a. The agent moves a spaceship left and right at the bottom of the screen. It can fire bullets upward, but each one has a cost (-1 reward). If a bullet hits one of the three targets, the agent receives 10 reward. Each target has a "bullseye" (the white dots). If a bullet hits the same column as a bullseye, the agent receives an additional 10 reward. Though the control problem is simple, the state/observation space is high-dimensional due to the many possible configurations of objects on the screen.

In the original Shooter the bullseyes remained still but here they move back and forth across the targets. As such, the problem is second-order Markov; when the bullseye is in

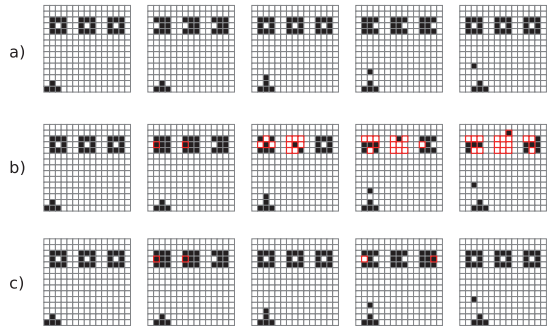

Figure 2: The Shooter game. a) Example of the real dynamics. b) Propagating errors (red outlines) in a model optimized for one-step error. c) A model optimized for multi-step error.

the center, one cannot predict its next position without knowing its previous position. The agent, however, will use a factored Markov model, predicting each pixel conditioned on the current image. It cannot accurately predict the bullseyes' movement, though it can predict everything else perfectly.

One might imagine that this limitation would be fairly minor; the agent can still obtain reward even if it cannot reliably hit the bullseyes. However, consider the sample rollout pictured in Figure 2b. Here each image is sampled from a model's one-step predictions, and is then given as input for the next predictions. This model has the lowest possible one-step prediction error. Still, as anticipated, it does not correctly predict the movement of the bullseyes in the second image. Because of the resulting errors, the sampled image is unlike any the environment would generate, and therefore unlike any the model has trained on. The model's uninformed predictions based on this unfamiliar image cause more errors in the third image, and so on. Ultimately this model assigns low probability to a target persisting more than a few steps, making it essentially useless for planning.

Note, however, that there *are* models within this model class that are useful for planning. Consider the sample rollout pictured in Figure 2c. The model that generated this rollout makes the same one-step errors as the previous model when given an environment state. However, when it encounters an unreasonable sampled state it still makes reasonable predictions, effectively "self-correcting." Talvitie (2014) presents several similar examples involving various model deficiencies. These examples illustrate the inadequacy of Lemma 2 when the model class is limited. Models with similar one-step prediction error can vary wildly in their usefulness for planning. The true distinguisher is the accuracy of predictions far into the future.

2.2 Multi-step prediction error

Since $Q^{\pi}_{T}(s,a) = \sum_{t=1}^{T} \gamma^{t-1} \mathbf{E}_{(s',a')\sim D^{t}_{s,a,\pi}} R(s',a')$, it is straightforward to bound $\epsilon^{\xi,\pi,T}_{val}$ in terms of *multi-step* error.

Lemma 3. *For any policy π and state-action distribution ξ,*

$$\epsilon^{\xi,\pi,T}_{val} \leq M \sum_{t=1}^{T} \gamma^{t-1} \mathop{\mathbf{E}}_{(s,a)\sim\xi} \left[\| D^{t}_{s,a,\pi} - \hat{D}^{t}_{s,a,\pi} \|_1 \right].$$

The bound in Lemma 2 has dependence on $\frac{1}{1-\gamma}$ because it effectively assumes the worst possible loss in value if the model samples an "incorrect" state. In contrast, Lemma 3 accounts for the model's ability to recover after an error, only penalizing it for individual incorrect transitions. Unfortunately, it is difficult to directly optimize for multi-step prediction accuracy. Nevertheless, this bound suggests that algorithms that account for a model's multi-step error will yield more robust MBRL performance.

2.3 Hallucinated one-step prediction error

We now seek to formally analyze the practice of hallucinated training, described in Section 1. Venkatraman, Hebert, and Bagnell (2015) provide some analysis but in the uncontrolled time series prediction setting. Here we focus on its impact on control performance in MBRL. As a first step, we derive a bound based on a model's ability to predict the next environment state, given a state sampled from the model's own predictions, i.e. to self-correct. For a policy π and state-action distribution ξ let $J^{t}_{\xi,\pi}$ represent the *joint* distribution over environment and model state-action pairs if π is executed in both simultaneously. Specifically, let

$$J^{t}_{\xi,\pi}(s,a,z,b) = \mathbf{E}_{(s',a')\sim\xi}[D^{t}_{s',a',\pi}(s,a)\hat{D}^{t}_{s',a',\pi}(z,b)].$$

Lemma 4. *For any policy π and state-action distribution ξ,*

$$\epsilon^{\xi,\pi,T}_{val} \leq M \sum_{t=1}^{T-1} \gamma^{t} \mathop{\mathbf{E}}_{(s,a,z,b)\sim J^{t}_{\xi,\pi}} \left[\| P^{a}_{s} - \hat{P}^{b}_{z} \|_1 \right].$$

Inspired by "Hallucinated Replay" (Talvitie 2014), we call the quantity on the right the *hallucinated one-step error*. Hallucinated one-step error is intended as a proxy for multi-step error, but having formalized it we may now see that in some cases it is a poor proxy. Note that, regardless of the policy, the multi-step and one-step error of a perfect model is 0. This is not always so for hallucinated error.

Proposition 5. *The hallucinated one-step error of a perfect model may be non-zero.*

Proof. Consider a simple MDP with three states $\{s_0, s_h, s_t\}$ and a single action a. In the initial state s_0, a fair coin is flipped, transitioning to s_h or s_t with equal probability, where it stays forever. Consider a perfect model $\hat{P} = P$. Then $J^{1}_{s_0,a}(s_h,a,s_t,a) = P^{a}_{s_0}(s_h)P^{a}_{s_0}(s_t) = 0.25$. However, $|P^{a}_{s_h}(s_h) - P^{a}_{s_t}(s_h)| = 1 - 0 = 1$. Thus, the hallucinated one-step error of a perfect model is non-zero. □

Here the environment samples heads and the model samples tails. Given its own state, the model rightly predicts tails, but incurs error nevertheless since the environment's next state is heads. Because the model and environment dynamics are uncoupled, one cannot distinguish between model error and legitimately different stochastic outcomes. As such, the hallucinated error is misleading when the true dynamics are stochastic. This corroborates the conjecture that Hallucinated Replay may be problematic in stochastic environments (Talvitie 2014). Note that this observation applies not just to hallucinated training, but to *any* method

that attempts to improve multi-step predictions by comparing sample rollouts from the model and the environment.

While it may seem limiting to restrict our attention to deterministic environments, this is still a large, rich class of problems. For instance, Oh et al. (2015) learned models of Atari 2600 games, which are fully deterministic (Hausknecht et al. 2014); human players often perceive them as stochastic due to their complexity. Similarly, in synthetic RL domains stochasticity is often added to simulate complex, deterministic phenomena (e.g. robot wheels slipping on debris), not necessarily to capture inherently stochastic effects in the world. As in these examples, we shall assume that the environment is deterministic but complex, so a limited agent will learn an imperfect, stochastic model.

That said, even specialized to deterministic environments, the bound in Lemma 4 is loose for arbitrary policies.

Proposition 6. *The hallucinated one-step error of a perfect model may be non-zero, even in a deterministic MDP.*

Proof. Alter the coin MDP, giving the agent two actions which fully determine the coin's orientation. The original dynamics can be recovered via a stochastic policy that randomly selects s_h or s_t and then leaves the coin alone. □

Oh et al. (2015) tied action selection to the environment state only (rather than independently selecting actions in the environment and model). This prevents stochastic decoupling but may fail to train the model on state-action pairs that the policy would reach under the model's dynamics.

2.4 A Tighter Bound

In the remainder of the paper we assume that the environment is deterministic. Let $\sigma_s^{a_{1:t}}$ be the unique state that results from starting in state s and taking the action sequence $a_{1:t}$. The agent's model will still be stochastic.

Recall that our goal is to bound the value error under the one-ply MC rollout policy. Proposition 6 shows that hallucinated error gives a loose bound under arbitrary policies. We now focus on *blind policies* (Bowling et al. 2006). A blind policy depends only on the action history, i.e. $\pi(a_t \mid s_t, a_{1:t-1}) = \pi(a_t \mid a_{1:t-1})$. This class of policies ranges from stateless policies to open-loop action sequences. It includes the uniform random policy, a common rollout policy.

For any blind policy π and state-action distribution ξ, let $H^t_{\xi,\pi}$ be the distribution over environment state, model state, and action if a single action sequence is sampled from π and then executed in both the model and the environment. So,

$H^1_{\xi,\pi}(s_1, z_1, a_1) = \xi(s_1, a_1)$ when $z_1 = s_1$ (0 otherwise);

$H^2_{\xi,\pi}(s_2, z_2, a_2) = \mathbf{E}_{(s_1,a_1)\sim\xi}[\pi(a_2 \mid a_1) P^{a_1}_{s_1}(s_2) \hat{P}^{a_1}_{s_1}(z_s)]$;

and for $t > 2$, $H^t_{\xi,\pi}(s_t, z_t, a_t) =$

$$\mathbf{E}_{(s_1,a_1)\sim\xi}\left[\sum_{a_{2:t-1}} \pi(a_{2:t} \mid a_1) P^{a_{1:t-1}}_{s_1}(s_t) \hat{P}^{a_{1:t-1}}_{s_1}(z_t)\right].$$

Lemma 7. *If P is deterministic, then for any blind policy π and any state-action distribution ξ,*

$$\epsilon^{\xi,\pi,T}_{val} \leq 2M \sum_{t=1}^{T-1} \gamma^t \mathop{\mathbf{E}}_{(s,z,a)\sim H^t_{\xi,\pi}} \left[1 - \hat{P}^a_z(\sigma^a_s)\right].$$

We can also show that, in the deterministic setting, Lemma 7 gives an upper bound for multi-step error (Lemma 3) and a lower bound for one-step error (Lemma 2).

Theorem 8. *If P is deterministic, then for any blind policy π and any state-action distribution ξ,*

$$\begin{aligned}
\epsilon^{\xi,\pi,T}_{val} &\leq M \sum_{t=1}^{T} \gamma^{t-1} \mathop{\mathbf{E}}_{(s,a)\sim\xi} \left[\|D^t_{s,a,\pi} - \hat{D}^t_{s,a,\pi}\|_1\right] \\
&\leq 2M \sum_{t=1}^{T-1} \gamma^t \mathop{\mathbf{E}}_{(s,z,a)\sim H^t_{\xi,\pi}} \left[1 - \hat{P}^a_z(\sigma^a_s)\right] \\
&\leq \frac{2M}{1-\gamma} \sum_{t=1}^{T-1} (\gamma^t - \gamma^T) \mathop{\mathbf{E}}_{(s,a)\sim D^t_{\xi,\pi}} \left[1 - \hat{P}^a_s(\sigma^a_s)\right].
\end{aligned}$$

Thus, with a deterministic environment and a blind rollout policy, the hallucinated one-step error of the model is more tightly related to MBRL performance than the standard one-step error. This is the theoretical reason for the empirical success of Hallucinated Replay (Talvitie 2014), which trains the model to predict the next environment state, given its own samples as input. We now exploit this fact to develop a novel MBRL algorithm that similarly uses hallucinated training to mitigate the impact of model class limitations and that offers strong theoretical guarantees.

3 Hallucinated DAgger-MC

The "Data Aggregator" (DAgger) algorithm (Ross and Bagnell 2012) was the first practically implementable MBRL algorithm with performance guarantees agnostic to the model class. It did, however, require that the planner be near optimal. DAgger-MC (Talvitie 2015) relaxed this assumption, accounting for the limitations of the planner that uses the model (one-ply MC). This section augments DAgger-MC to use hallucinated training, resulting in the Hallucinated DAgger-MC algorithm[1], or H-DAgger-MC (Algorithm 1).

In addition to assuming a particular form for the planner (one-ply MC with a blind rollout policy), H-DAgger-MC assumes that the model will be "unrolled" (similar to, e.g. Abbeel, Quigley, and Ng 2006). Rather than learning a single model $\hat{P}$, H-DAgger-MC learns a set of models $\{\hat{P}^1, \ldots, \hat{P}^{T-1}\} \subseteq \mathcal{C}$, where model $\hat{P}^i$ is responsible for predicting the outcome of step i of a rollout, given the state sampled from $\hat{P}^{i-1}$ as input. The importance of learning an unrolled model will be discussed more deeply in Section 4.1.

Much of the H-DAgger-MC algorithm is identical to DAgger-MC. The main difference lies in lines 14-18, in which ρ is executed in both the environment and the model to generate hallucinated examples. This trains the model to self-correct during rollouts. Like DAgger and DAgger-MC,

[1] "Is this a dagger which I see before me,
The handle toward my hand? Come, let me clutch thee.
I have thee not, and yet I see thee still.
Art thou not, fatal vision, sensible
To feeling as to sight? Or art thou but
A dagger of the mind, a false creation,
Proceeding from the heat-oppress'd brain?" [*Macbeth* 2.1.33–39]

Algorithm 1 Hallucinated DAgger-MC

Require: Exploration distribution ν, ONLINE-LEARNER, MC-PLANNER (blind rollout policy ρ, rollout depth T), num. iterations N, num. rollouts per iteration K

1: Get initial datasets $\mathcal{D}_1^{1:T-1}$ (maybe using ν)
2: Initialize $\hat{P}_1^{1:T-1} \leftarrow$ ONLINE-LEARNER($\mathcal{D}_1^{1:T-1}$).
3: Initialize $\hat{\pi}_1 \leftarrow$ MC-PLANNER($\hat{P}_1^{1:T-1}$).
4: **for** $n \leftarrow 2 \dots N$ **do**
5: **for** $k \leftarrow 1 \dots K$ **do**
6: With probability... ▷ Sample $(x, b) \sim \xi_n$...
7: $1/2$: Sample $(x, b) \sim D_\mu^{\hat{\pi}_{n-1}}$
8: $1/4$: Reset to $(x, b) \sim \nu$.
9: $(1-\gamma)/4$: Sample $x \sim \mu$, $b \sim \hat{\pi}_{n-1}(\cdot \mid x)$.
10: $\gamma/4$:
11: Reset to $(y, c) \sim \nu$
12: Sample $x \sim P(\cdot \mid y, c)$, $b \sim \hat{\pi}_{n-1}(\cdot \mid x)$
13: Let $s \leftarrow x$, $z \leftarrow x$, $a \leftarrow b$.
14: **for** $t \leftarrow 1 \dots T-1$ **do** ▷ Sample from H_n^t...
15: Sample $s' \sim P(\cdot \mid s, a)$.
16: Add $\langle z, a, s' \rangle$ to $\mathcal{D}_n^t$. ▷ Hallucinated training ▷ (DAgger-MC adds $\langle s, a, s' \rangle$ instead).
17: Sample $z' \sim \hat{P}_{n-1}^t(\cdot \mid z, a)$.
18: Let $s \leftarrow s'$, $z \leftarrow z'$, and sample $a \sim \rho$.
19: $\hat{P}_n^{1:T-1} \leftarrow$ ONLINE-LEARNER($\hat{P}_{n-1}^{1:T-1}$, $\mathcal{D}_n^{1:T-1}$)
20: $\hat{\pi}_n \leftarrow$ MC-PLANNER($\hat{P}_n^{1:T-1}$).
21: **return** the sequence $\hat{\pi}_{1:N}$

H-DAgger-MC requires the ability to reset to the initial state distribution μ and also the ability to reset to an "exploration distribution" ν. The exploration distribution ideally ensures that the agent will encounter states that would be visited by a good policy, otherwise no agent could promise good performance. The performance bound for H-DAgger-MC will depend in part on the quality of the selected ν.

We now analyze H-DAgger-MC, adapting Ross and Bagnell (2012)'s DAgger analysis. Let H_n^t be the distribution from which H-DAgger-MC samples a training triple at depth t (lines 6-13 to pick an initial state-action pair, lines 14-18 to roll out). Define the error of the model at depth t to be $\bar{\epsilon}_{prd}^t = \frac{1}{N} \sum_{n=1}^N \mathbf{E}_{(s,z,a) \sim H_n^t}[1 - \hat{P}_n^t(\sigma_s^a \mid z, a)]$.

For a policy π, let $c_\nu^\pi = \sup_{s,a} \frac{D_{\mu,\pi}(s,a)}{\nu(s,a)}$ represent the mismatch between the discounted state-action distribution under π and the exploration distribution ν. Now, consider the sequence of policies $\hat{\pi}_{1:N}$ generated by H-DAgger-MC. Let $\bar{\pi}$ be the uniform mixture over all policies in the sequence. Let $\bar{\epsilon}_{mc} = \frac{1}{N} \frac{4}{1-\gamma} \sum_{n=1}^N (\|\bar{Q}_n - \hat{Q}_{T,n}^\rho\|_\infty + \|\hat{Q}_{T,n}^\rho - \hat{Q}_n^\rho\|_\infty) + \frac{2}{1-\gamma} \|BV^\rho - V^\rho\|_\infty$ be the error induced by the choice of planning algorithm, averaged over all iterations.

Lemma 9. *In H-DAgger-MC, the policies $\hat{\pi}_{1:N}$ are such that for any policy π,*

$$\mathop{\mathbf{E}}_{s \sim \mu}\left[V^\pi(s) - V^{\bar{\pi}}(s)\right] \leq \frac{8M}{1-\gamma} c_\nu^\pi \sum_{t=1}^{T-1} \bar{\epsilon}_{prd}^t + \bar{\epsilon}_{mc}.$$

Note that this result holds for *any* comparison policy π. Thus, if $\bar{\epsilon}_{mc}$ is small and the learned models have low hallucinated one-step prediction error, then if ν is similar to the state-action distribution under *some* good policy, $\bar{\pi}$ will compare favorably to it. Like the original DAgger and DAgger-MC results, Lemma 9 has limitations. It uses the L1 loss, which is not always a practical learning objective. It also assumes that the expected loss at each iteration can be computed exactly (i.e. that there are infinitely many samples per iteration). It also applies to the average policy $\bar{\pi}$, rather than the last policy in the sequence. Ross and Bagnell (2012) discuss extensions that address more practical loss functions, finite sample bounds, and results for $\hat{\pi}_N$.

The next question is, of course, when will the learned models be accurate? Following Ross and Bagnell (2012) note that $\bar{\epsilon}_{prd}^t$ can be interpreted as the average loss of an online learner on the problem defined by the aggregated datasets at each iteration. In that case, for each horizon depth t let $\bar{\epsilon}_{mdl}^t$ be the error of the best model in $\mathcal{C}$ under the training distribution at that depth, in retrospect. Specifically, $\bar{\epsilon}_{mdl}^t = \inf_{P' \in \mathcal{C}} \frac{1}{N} \sum_{n=1}^N \mathbf{E}_{(s,z,a) \sim H_n^t}[1 - P'(\sigma_s^a \mid z, a)]$. Then the average regret for the model at depth t is $\bar{\epsilon}_{rgt}^t = \bar{\epsilon}_{prd}^t - \bar{\epsilon}_{mdl}^t$. For a no-regret online learning algorithm, $\bar{\epsilon}_{rgt}^t \to 0$ as $N \to \infty$. This gives the following bound on H-DAgger-MC's performance in terms of model regret.

Theorem 10. *In H-DAgger-MC, the policies $\hat{\pi}_{1:N}$ are such that for any policy π,*

$$\mathop{\mathbf{E}}_{s \sim \mu}\left[V^\pi(s) - V^{\bar{\pi}}(s)\right] \leq \frac{8M}{1-\gamma} c_\nu^\pi \sum_{t=1}^{T-1} (\bar{\epsilon}_{mdl}^t + \bar{\epsilon}_{rgt}^t) + \bar{\epsilon}_{mc},$$

and if the model learning algorithm is no-regret then $\bar{\epsilon}_{rgt}^t \to 0$ as $N \to \infty$ for each $1 \leq t \leq T-1$.

Theorem 10 says that if $\mathcal{C}$ contains a low-error model for each rollout depth then low error models will be learned. Then, as discussed above, if $\bar{\epsilon}_{mc}$ is small and ν visits important states, the resulting policy will yield good performance. Notably, even with hallucinated training, if $\mathcal{C}$ contains a perfect model, H-DAgger-MC will learn a perfect model.

It is important to note that this result does *not* promise that H-DAgger-MC will eventually achieve the performance of the best performing set of models in the class. The model at each rollout depth is trained to minimize prediction error given the input distribution provided by the shallower models. Note, however, that changing the parameters of a model at one depth alters the training distribution for deeper models. It is possible that better overall error could be achieved by *increasing* the prediction error at one depth in exchange for a favorable state distribution for deeper models. This effect is not taken into account by H-DAgger-MC.

4 Empirical Illustration

In this section we illustrate the practical impact of optimizing hallucinated error by comparing DAgger, DAgger-MC, and H-DAgger-MC in the Shooter example described in Section 2.1[2]. The experimental setup matches that of Talvi-

[2]Source code for these experiments may be found at github.com/etalvitie/hdaggermc.

tie (2015) for comparison's sake, though the qualitative comparison presented here is robust to the parameter settings.

In all cases one-ply MC was used with 50 uniformly random rollouts of depth 15 at every step. The exploration distribution was generated by following the optimal policy with $(1-\gamma)$ probability of termination at each step. The model for each pixel was learned using Context Tree Switching (Veness et al. 2012), similar to the FAC-CTW algorithm (Veness et al. 2011), and used a 7×7 neighborhood around the pixel in the previous timestep as input. Data was shared across all positions. The discount factor was $\gamma = 0.9$. In each iteration 500 training rollouts were generated and the resulting policy was evaluated in an episode of length 30. The discounted return obtained by the policy in each iteration is reported, averaged over 50 trials.

The results can be seen in Figure 3a and 3b. The shaded regions represent 95% confidence intervals for the mean performance. The benchmark lines labeled "Random" and "Perfect Model" represent the average performance of the uniform random policy and one-ply Monte Carlo using a perfect model, respectively. In Figure 3a the bullseyes move, simulating the typical practical reality that $\mathcal{C}$ does not contain a perfect model. In Figure 3b the bullseyes have fixed positions, so $\mathcal{C}$ *does* contain a perfect model.

As observed by Talvitie (2015), DAgger performs poorly in both versions, due to the suboptimal planner. DAgger-MC is able to perform well with fixed bullseyes (Figure 3b), but with moving bullseyes the model suffers from compounding errors and is not useful for planning (Figure 3a). This holds for a single model and for an "unrolled" model.

In these experiments one practically-minded alteration was made to the H-DAgger-MC algorithm. In early training the model is highly inaccurate, and thus deep rollouts produce incoherent samples. Training with these samples is counter-productive (also, the large number of distinct, nonsensical contexts renders CTS impractical). For these experiments, training rollouts in iteration i were truncated at depth $\lfloor i/10 \rfloor$. Planning rollouts in these early iterations use the models that have been trained so far and then repeatedly apply the deepest model in order to complete the rollout. Talvitie (2014), Venkatraman, Hebert, and Bagnell (2015), and Oh et al. (2015) all similarly discarded noisy examples early in training. This transient modification does not impact H-DAgger-MC's asymptotic guarantees.

In Figure 3a it is clear that H-DAgger-MC obtains a good policy despite the limitations of the model class. Hallucinated training has made MBRL possible with both a flawed model and a flawed planner while the standard approach has failed entirely. In the case that $\mathcal{C}$ contains a perfect model (rare in problems of genuine interest) H-DAgger-MC is outperformed by DAgger-MC. Despite the adjustment to training, deep models still receive noisy inputs. Theoretically the model should become perfectly accurate in the limit, though in practice it may do so very slowly.

4.1 Impact of the unrolled model

Recall that the H-DAgger-MC algorithm assumes the model will be "unrolled," with a separate model responsible for sampling each step in a rollout. This has clear practical disadvantages, but it is important theoretically. When one model is used across all time-steps, convergence to a perfect model cannot be guaranteed, even if one exists in $\mathcal{C}$.

In Figure 3c, H-DAgger-MC has been trained using a single model in Shooter with fixed bullseyes. The temporary truncation schedule described above is employed, but the training rollouts have been permanently limited to various depths. First consider the learning curve marked "Depth 15", where training rollouts are permitted to reach the maximum depth. While the rollouts are temporarily truncated the model does well, but performance degrades as longer rollouts are permitted *even though $\mathcal{C}$ contains a perfect model*!

Recall from Section 3 that changing the model parameters impacts both prediction error and the future training distribution. Furthermore, training examples generated by deep rollouts may contain highly flawed samples as inputs. Sometimes attempting to "correct" a large error (i.e. reduce prediction error) causes additional, even worse errors in the next iteration (i.e. harms the training distribution). For instance consider a hallucinated training example with the 4th screen from Figure 2b as input and the 5th screen from Figure 2a as the target. The model would effectively learn that targets can appear out of nowhere, an error that would be even harder to correct in future iterations. With a single model across timesteps, a feedback loop can emerge: the model parameters change to attempt to correct large errors, thereby causing larger errors, and so on. This feedback loop causes the observed performance crash. With an unrolled model the parameters of each sub-model cannot impact that sub-model's own training distribution, ensuring stability.

Note that none of Talvitie (2014), Venkatraman, Hebert, and Bagnell (2015), or Oh et al. (2015) used an unrolled model. As such, all of their approaches are subject to this concern. Notably, all three limited the depth of training rollouts, presumably to prevent overly noisy samples. Figure 3c shows that in this experiment, the shorter the training rollouts, the better the performance. These results show that it may be possible in practice to avoid unrolling the model by truncating training rollouts, though for now there is no performance guarantee or principled choice of rollout depth.

5 Conclusions and future work

The primary contribution of this work is a deeper theoretical understanding of how to perform effective MBRL in the face of model class limitations. Specifically we have examined a novel measure of model quality that, under some assumptions, is more tightly related to MBRL performance than standard one-step prediction error. Using this insight, we have also analyzed a MBRL algorithm that achieves good control performance despite flaws in the model and planner and provides strong theoretical performance guarantees.

We have also seen negative results indicating that hallucinated one-step error may not be an effective optimization criterion in the most general setting. This poses the open challenge of relaxing the assumptions of deterministic dynamics and blind policies, or of developing alternative approaches for improving multi-step error in more general settings. We have further observed that hallucinated training can cause stability issues, since model parameters affect

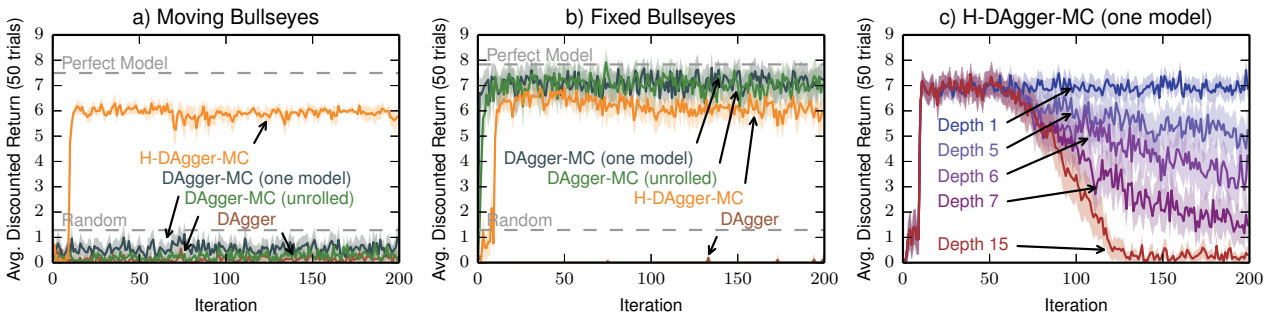

Figure 3: a,b) Comparing DAgger, DAgger-MC, and H-DAgger-MC in Shooter with moving and fixed bullseyes, respectively. c) H-DAgger-MC in Shooter with fixed bullseyes using a single model across time steps, truncating rollouts at various depths.

both prediction error and the training distribution itself. It would be valuable to develop techniques that account for both of these effects when adapting model parameters.

Specializing to the one-ply MC planning algorithm may seem restrictive, but then again, the choice of planning algorithm cannot make up for a poor model. When the model class is limited, H-DAgger-MC is likely still a good choice over DAgger, even with a more sophisticated planner. Still, it would be valuable to investigate whether these principles can be applied to more sophisticated planning algorithms.

Though this work has assumed that the reward function is known, the results presented here can be straightforwardly extended to account for reward error. However, this also raises the interesting point that sampling an "incorrect" state has little negative impact if the sampled state's rewards and transitions are similar to the "correct" state. It may be possible to exploit this to obtain still tighter bounds, and more effective guidance for model learning in MBRL architectures.

Acknowledgements

This work was supported in part by NSF grant IIS-1552533. Many thanks to Marc Bellemare whose feedback has positively influenced the work, both in substance and presentation. Thanks to Drew Bagnell and Arun Venkatraman for their valuable insights. Thanks also to Joel Veness for his freely available FAC-CTW and CTS implementations (http://jveness.info/software/).

References

Abbeel, P.; Quigley, M.; and Ng, A. Y. 2006. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 1–8.

Bellemare, M. G.; Veness, J.; and Talvitie, E. 2014. Skip context tree switching. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 1458–1466.

Bowling, M.; McCracken, P.; James, M.; Neufeld, J.; and Wilkinson, D. 2006. Learning predictive state representations using non-blind policies. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 129–136.

Hausknecht, M.; Lehman, J.; Miikkulainen, R.; and Stone, P. 2014. A neuroevolution approach to general atari game playing. *IEEE Transactions on Computational Intelligence and AI in Games* 6(4):355–366.

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 (ICRA)*, 939–946.

Kakade, S. M. 2003. *On the sample complexity of reinforcement learning*. Ph.D. Dissertation, University of London.

Oh, J.; Guo, X.; Lee, H.; Lewis, R. L.; and Singh, S. 2015. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems 28 (NIPS)*, 2845–2853.

Ross, S., and Bagnell, D. 2012. Agnostic system identification for model-based reinforcement learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 1703–1710.

Sorg, J.; Lewis, R. L.; and Singh, S. 2010. Reward design via online gradient ascent. In *Advances in Neural Information Processing Systems 23 (NIPS)*, 2190–2198.

Szita, I., and Szepesvári, C. 2010. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 1031–1038.

Talvitie, E. 2014. Model regularization for stable sample rollouts. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI)*, 780–789.

Talvitie, E. 2015. Agnostic system identification for monte carlo planning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, 2986–2992.

Tesauro, G., and Galperin, G. R. 1996. On-line policy improvement using monte-carlo search. In *Advances in Neural Information Processing Systems 9 (NIPS)*, 1068–1074.

Veness, J.; Ng, K. S.; Hutter, M.; Uther, W. T. B.; and Silver, D. 2011. A Monte-Carlo AIXI Approximation. *Journal of Artificial Intelligence Research (JAIR)* 40:95–142.

Veness, J.; Ng, K. S.; Hutter, M.; and Bowling, M. 2012. Context tree switching. In *Proceedings of the 2012 Data Compression Conference (DCC)*, 327–336. IEEE.

Venkatraman, A.; Hebert, M.; and Bagnell, J. A. 2015. Improving multi-step prediction of learned time series models. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, 3024–3030.