Self-Correcting Models for Model-Based Reinforcement Learning

Erik Talvitie
Department of Computer Science
Franklin & Marshall College
Lancaster, PA 17603
erik.talvitie@fandm.edu

Abstract

This work considers the problem of model-learning (system identification) for planning (control synthesis). When an agent cannot represent a perfectly accurate model of its environment's dynamics, planning based on a learned model can catastrophically fail. Planning involves composing the predictions of the model; when flawed predictions are composed, even minor errors can compound and render the model useless for making control decisions. This work presents a novel measure of model quality that accounts for the model's ability to "self correct" after making an error, thus mitigating the compounding error effect. Under some conditions this measure can be shown to be more tightly related to the quality of policies derived from the model than the standard measure of accuracy (one-step prediction error). This observation inspires an MBRL algorithm for deterministic MDPs that offers performance guarantees that are agnostic to the model class and that is empirically shown to generate good policies in the face of model class limitations that stymie the more standard approach.

Keywords: model-based reinforcement learning, model learning, system identification, model predictive control

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 which were employed in this work (http://jveness.info/software/).

1 Introduction

The model-based reinforcement learning (MBRL) approach to decision making in an unknown enviornment is to learn a predictive model of the environment's dynamics (the model learning/system identification problem) and then to use that model to make decisions (the planning/optimal control problem). In most problems of interest, limitations of the agent prevent it from learning a perfectly accurate model. In system identification this is called the *unrealizable* setting, where the true system is not contained within the class of models being considered. One might hope that the practical impact of model class limitations would be related to the severety of the limitations, but, in practice, even seemingly minor limitations can prove disastrous for planning/control sythesis.

To illustrate, consider the "Shooter" domain pictured in Figure 1a. 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 a target, 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.

The typical approach is to train the model by minimizing one-step prediction error, the error in the predictions of the next image, given the current image. That said, note that the bullseyes move back and forth across the targets. As such, the problem is second-order Markov; when the bullseye is in 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 independently, conditioned on the current image. It cannot accurately predict the bullseyes' movement, though it can predict everything else perfectly.

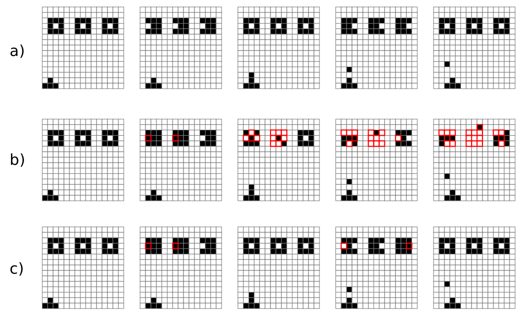

Figure 1: a) Example of Shooter dynamics. b) Propagating errors (in red) in a model optimized for one-step error. c) A self-correcting model.

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 1b. 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 one-step prediction error amongst the models in the class of factored Markov models. As anticipated, it does not correctly predict the movement of the bullseyes in the second image. Due to the resulting errors, the sampled image is unlike any the environment would generate, and thus 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. In the model a target is unlikely to persist long enough to be hit, making it useless for planning.

Note, however, that there *are* factored Markov models that are useful for planning in this problem. Consider the sample rollout pictured in Figure 1c. The model that generated this rollout makes the same one-step errors as the previous model when given an image from the environment. However, when it encounters an unreasonable sampled image it still makes reasonable predictions, effectively "self-correcting." Thus we see that models with similar one-step prediction error can vary wildly in their usefulness for planning. A better distinguisher is the accuracy of predictions far into the future.

This work presents a novel measure of model quality that captures a model's ability to "self correct" in this way. Under some conditions this measure can be proven to be more tightly related to control performance than the standard one-step prediction error. These results allow the derivation of a novel MBRL algorithm that provides strong performance guarantees that are agnostic to the model class and that is empirically robust to model class limitations.

1.1 Background and Notation

The analysis is focused 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 $D_{s,a,\pi}^t$ 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 $D_{\xi,\pi}^t = \mathbf{E}_{(s,a)\sim\xi}\, D_{s,a,\pi}^t$. For a state distribution μ let $D_{\mu,\pi}^t = \mathbf{E}_{s\sim\mu, a\sim\pi_s}\, D_{s,a,\pi}^t$. For some discount factor $\gamma \in [0, 1)$, let $D_{\mu,\pi} = (1-\gamma)\sum_{t=1}^{\infty} \gamma^{t-1} D_{\mu,\pi}^t$ be the infinite-horizon *discounted* state-action distribution under policy π.

The T-step *state-action value* of a policy, $Q_T^\pi(s, a)$ represents the expected discounted sum of rewards obtained by taking action a in state s and executing π for an additional $T-1$ steps: $Q_T^\pi(s, a) = \sum_{t=1}^{T} \gamma^{t-1}\, \mathbf{E}_{(s',a')\sim D_{s,a,\pi}^t}\, R(s', a')$. Let the

T-step *state value* $V_T^\pi(s) = \mathbf{E}_{a\sim\pi_s}[Q_T^\pi(s,a)]$. For infinite horizons we write $Q^\pi = Q_\infty^\pi$, and $V^\pi = V_\infty^\pi$. 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 produce a policy by apply some planning algorithm to $\hat{P}$. Let $\mathcal{C}$ represent the *model class*, the set of models the learning algorithm could possibly produce. Critically, it is not assumed that $P \in \mathcal{C}$.

For the sake of this analysis, the agent is assumed to use the simple one-ply Monte Carlo planning algorithm (one-ply MC). 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 $\hat{Q}_T^\rho$ (Kakade, 2003). The agent selects its actions greedily with respect to $\bar{Q}$.

2 Bounding Value Error

Talvitie (2015) bounds the performance of one-ply MC in terms of model quality. For a policy π and state-action distribution ξ, let $\epsilon_{val}^{\xi,\pi,T} = \mathbf{E}_{(s,a)\sim\xi}\left[|Q_T^\pi(s,a) - \hat{Q}_T^\pi(s,a)|\right]$ 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 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. $\bar{Q}$. Let $\xi(s,a) = \frac{1}{2}D_{\mu,\hat{\pi}}(s,a) + \frac{1}{4}D_{\mu,\pi}(s,a) + \frac{1}{4}\left((1-\gamma)\mu(s)\hat{\pi}_s(a) + \gamma\sum_{z,b} D_{\mu,\pi}(z,b)P_z^b(s)\hat{\pi}_s(a)\right)$. Let $\epsilon_{mc} = \frac{4}{1-\gamma}\|\bar{Q} - \hat{Q}_T^\rho\|_\infty + \frac{4}{1-\gamma}\|\hat{Q}_T^\rho - \hat{Q}^\rho\|_\infty + \frac{2}{1-\gamma}\|BV^\rho - V^\rho\|_\infty$ (here B is the Bellman operator). Then for any policy π and state-distribution μ, $\mathbf{E}_{s\sim\mu}\left[V^\pi(s) - V^{\hat{\pi}}(s)\right] \leq \frac{4}{1-\gamma}\epsilon_{val}^{\xi,\rho,T} + \epsilon_{mc}$.*

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 $\epsilon_{val}^{\xi,\rho,T}$ 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 $\hat{\pi}$. This section presents bounds on $\epsilon_{val}^{\xi,\rho,T}$ in terms of measures of model quality.

First, we can adapt a result from Ross and Bagnell (2012), bounding $\epsilon_{val}^{\xi,\rho,T}$ in terms of the one-step prediction error.

Lemma 2. *For any policy π and state-action distribution ξ, $\epsilon_{val}^{\xi,\pi,T} \leq \frac{M}{1-\gamma}\sum_{t=1}^{T-1}(\gamma^t - \gamma^T)\,\mathbf{E}_{(s,a)\sim D_{\xi,\pi}^t}\left[\|P_s^a - \hat{P}_s^a\|_1\right]$.*

Combining Lemmas 1 and 2 yields an overall bound on control performance but, as the Shooter example demonstrates, when one cannot obtain a model with low one-step prediction error, this bound can be quite loose.

2.1 Hallucinated one-step prediction error

We now seek to define a measure of a model's ability to "self correct" and then to bound the value error using that quantity. Specifically, we shall 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. For a policy π and state-action distribution ξ let $J_{\xi,\pi}^t$ represent the *joint* distribution over environment and model state-action pairs if π is executed in both simultaneously. That is, let $J_{\xi,\pi}^t(s,a,z,b) = \mathbf{E}_{(s',a')\sim\xi}[D_{s',a',\pi}^t(s,a)\hat{D}_{s',a',\pi}^t(z,b)]$.

Lemma 3. *For any policy π and state-action distribution ξ, $\epsilon_{val}^{\xi,\pi,T} \leq M\sum_{t=1}^{T-1}\gamma^t\,\mathbf{E}_{(s,a,z,b)\sim J_{\xi,\pi}^t}\left[\|P_s^a - \hat{P}_z^b\|_1\right]$.*

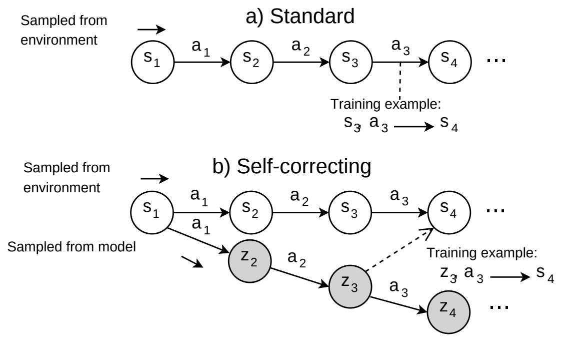

Figure 2: a) Predict next state from environment state b) Predict next state from model state

This bound suggests an alternative approach to training the model. In Figure 2a, the typical approach is illustrated: train the model to predict the next state, given the current environment state. In contrast, as illustrated in Figure 2b, this bound suggests rolling the model and the environment out in parallel, and then training the model to predict the next state (s_4), given the current, incorrect *model state* (z_3). Multiple recent results indicate that training models in this way can significantly improve long-range predictions (Venkatraman et al., 2015; Oh et al., 2015) and control performance (Talvitie, 2014; Venkatraman et al., 2016). This work adds to the theoretical understanding of those observations. Inspired by the "Hallucinated Replay" meta-algorithm (Talvitie, 2014), we call the quantity on the right the *hallucinated one-step error*.

2.2 Limitations of Hallucinated Error

Having formalized hallucinated one-step error, we can now see that Lemma 3 is loose in some cases. 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. The hallucinated one-step error of a perfect model may be non-zero if P is non-deterministic. 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 hallucinated 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.

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; human players often perceive them as stochastic due to their complexity. Similarly, 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 3 is loose for arbitrary policies. To see this, 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.

2.3 A Tighter Bound

In the remainder of the paper we assume that the environment is deterministic. Let $\sigma^{a_{1:t}}_s$ 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. The previous section argued that hallucinated error gives a loose bound under arbitrary policies. We now focus on *blind* rollout 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.

Under these assumptions we can develop an alternative form for the hallucinated error that does provide a tight bound. 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 $t > 2$, $H^t_{\xi,\pi}(s_t, z_t, a_t) = \mathbf{E}_{(s_1,a_1)\sim\xi}\left[\sum_{a_{2:t-1}} \pi(a_{2:t} \mid a_1)P^{a_{1:t-1}}_{s_1}(s_t)\hat{P}^{a_{1:t-1}}_{s_1}(z_t)\right]$.

Theorem 4. *If P is deterministic, then for any blind policy π and any state-action distribution ξ,*
$\epsilon^{\xi,\pi,T}_{val} \leq 2M\sum_{t=1}^{T-1}\gamma^t\,\mathbf{E}_{(s,z,a)\sim H^t_{\xi,\pi}}\left[1 - \hat{P}^a_z(\sigma^a_s)\right] \leq \frac{2M}{1-\gamma}\sum_{t=1}^{T-1}(\gamma^t - \gamma^T)\,\mathbf{E}_{(s,a)\sim D^t_{\xi,\pi}}\left[1 - \hat{P}^a_s(\sigma^a_s)\right].$

The far right quantity is the bound from Lemma 2, specialized to the deterministic setting. Thus, under these assumptions, the hallucinated one-step error is more tightly related to MBRL performance than the standard one-step error.

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 reports the results of augmenting DAgger-MC to optimize hallucinated one-step error, rather than one-step prediction error, resulting in the Hallucinated DAgger-MC algorithm, or H-DAgger-MC.

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 et al. 2006). Rather than learning a single model $\hat{P}$, H-DAgger-MC learns a set of models $\{\hat{P}^1, \dots, \hat{P}^{T-1}\} \subseteq \mathcal{C}$, where model $\hat{P}^i$ predicts 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 below.

H-DAgger-MC inherits the agnostic guarantees of DAgger-MC, and, due to Theorem 4, gives a tighter performance bound. In brief, if 1) the model-learning algorithm is no-regret, 2) the exploration policy visits states that a good policy would visit, 3) the model class contains a model for each rollout depth with low hallucinated error, and 4) one-ply MC would perform well with a perfect model, then H-DAgger-MC will yield a good policy. It is important to note, however, that this result does *not* promise that H-DAgger-MC will eventually learn 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.

3.1 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. The experimental setup matches that of Talvitie (2015) for comparison's sake, though the qualitative comparison is robust to the parameter settings.

In Figure 3a the bullseyes move so $\mathcal{C}$ does not contain a perfect model. Both DAgger and DAgger-MC fail to learn a model useful for planning while H-DAgger-MC is able to perform well. In Figure 3b the bullseyes have fixed positions, so $\mathcal{C}$ *does* contain a perfect model. DAgger still performs poorly due to the suboptimal planner. DAgger-MC outperforms H-DAgger-MC; the noisy training examples in H-DAgger-MC slow down learning. Theoretically the model should become perfectly accurate in the limit, though in practice it may do so very slowly.

Figure 3c shows results using H-DAgger-MC with a single model across timesteps (rather than an "unrolled" model) in Shooter with fixed bullseyes. Training rollouts have been truncated to various depths. We see that in this case good performance cannot be guaranteed, even though the model class 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). With a single model across timesteps, a feedback loop can emerge: the model parameters change to attempt to correct 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. That said, this experiment also shows that truncating training rollouts can mitigate the negative feedback loop, though there is no theoretical guidance for rollout depth at this time.

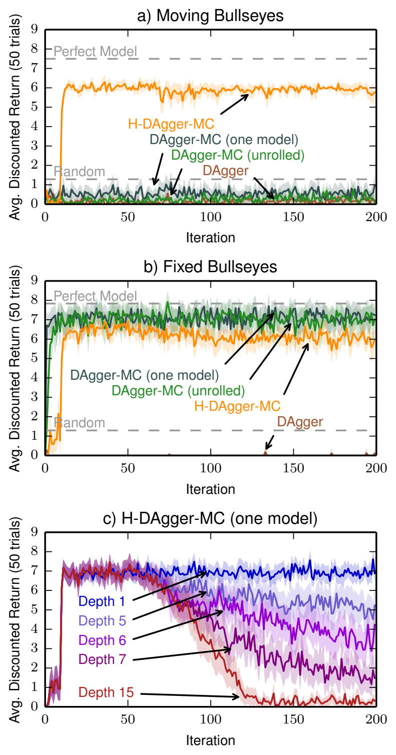

Figure 3: Empirical results with H-DAgger-MC.

4 Conclusion

In order to use learned models for planning/control in large, complex problems, it is critical to be able to learn models that are inaccurate and *also* useful for planning. This work introduces hallucinated one-step error which, under some conditions, is a better measure of a model's usefulness for planning than one-step prediction error. By optimizing the hallucinated error, the Hallucinated DAgger-MC algorithm is able to succeed even in the face of model limitations.

References

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

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

S. M. Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, University of London, 2003.

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

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

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

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

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

A. Venkatraman, R. Capobianco, L. Pinto, M. Hebert, D. Nardi, and A. J. Bagnell. Improved learning of dynamics for control. In *International Symposium on Experimental Robotics*, 2016.