Combining Parametric and Nonparametric Models for Off-Policy Evaluation

Omer Gottesman [1] **Yao Liu** [2] **Scott Sussex** [1] **Emma Brunskill** [2] **Finale Doshi-Velez** [1]

Abstract

We consider a model-based approach to perform batch off-policy evaluation in reinforcement learning. Our method takes a mixture-of-experts approach to combine parametric and nonparametric models of the environment such that the final value estimate has the least expected error. We do so by first estimating the local accuracy of each model and then using a planner to select which model to use at every time step as to minimize the return error estimate along entire trajectories. Across a variety of domains, our mixture-based approach outperforms the individual models alone as well as state-of-the-art importance sampling-based estimators.

1. Introduction

In the context of reinforcement learning (RL), off-policy evaluation (OPE) refers to the task of evaluating how good a given *evaluation policy* is, using data collected under a different *behavior policy*. This is in contrast with the much simpler problem of on-policy evaluation, where the behavior and evaluation policies are identical, and the value of the policy can be estimated simply by taking the average rewards accumulated over the observed trajectories. The most common set of approaches to OPE derive from importance sampling (e.g. Precup (2000); Jiang & Li (2016)); while unbiased, they tend to have prohibitively high variance unless a very large amount of data is available.

To reduce variance, another set of approaches first use the data collected under behavior policy to learn a parametric model to approximate the environment's dynamics, and then use that model to simulate trajectories under the evaluation policy (e.g. Chow et al. (2015)). Unfortunately, poor model specification can lead to poor generalization and model bias even if the amount of data is infinite. Fonteneau et al. (2013) circumvent this issue by simulating trajectories under the evaluation policy via stitching together actual transitions observed in the data. This nonparametric approach stays closer to the data, and is thus less likely to suffer from generalization errors; unlike the parametric approach, it will be consistent in the limit of infinite data. However, in a finite batch, the required transitions to use this method may not be available in the observed data if there is a large discrepancy between the behavior and evaluation policy.

Our main contribution is to note that the parametric and nonparametric approaches above have complementary strengths: the nonparametric approach can be very accurate where data are abundant, while the parametric approach can often generalize better in situations which are not frequently observed. We therefore propose a mixture-of-experts (MoE) approach for generating trajectories which switches between sampling transitions from a parametric and nonpaprametric model. We treat the OPE as a planning problem: at every transition, we choose the model—parametric or nonparametric—to minimize the overall value estimate error. We derive estimators of local errors for each model, and across a variety of domains, demonstrate that our approach produces more accurate value estimates than a myopic strategy (that does not optimize for error in the long-term), modeling using either parametric or nonparamtric approaches alone, as well as state-of-the-art importance sampling methods.

2. Background and Notation

We denote a Markov Decision Process (MDP) by $\langle \mathcal{X}, \mathcal{A}, \gamma, f_t, f_r, p_0, \rangle$, where $\mathcal{X}$, $\mathcal{A}$ and γ are the state space, action space, and reward discount, respectively. For this work, we assume the state transition and rewards are deterministic functions of the current state and action such that $x_{t+1} = f_t(x_t, a_t)$ and $r_t = f_r(x_t, a_t)$. The MoE algorithm we present in this paper can be applied to the stochastic case as well, but the model error estimators we develop and use only apply to the deterministic case. $p_0(x)$ denotes the initial states distribution.

A history is a sequence $H^{(i)} := (x_0^{(i)}, a_0^{(i)}, r_0^{(i)}, ..., x_T^{(i)})$ where $x_0 \sim p_0$ and the actions are chosen according to a policy π such that $a_t \sim \pi(a_t|x_t)$. Finally, let $\Delta(x, x')$ be a distance metric over the space $\mathcal{X}$. Throughout this

[1]Harvard University [2]Stanford University. Correspondence to: Omer Gottesman <gottesman@fas.harvard.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

paper we use the Euclidean distance as the metric over the state space, but discuss how the choice of the metric could impact the performance of the algorithm.

The value of a policy is the expected sum of discounted rewards collected by following the policy, $v^\pi := \mathrm{E}[g_T | a_t \sim \pi]$, where we defined the total return of a history as $g_T := \sum_{t=0}^{T} \gamma^t r_t$. In off-policy evaluation, our goal is to estimate the value of an *evaluation* policy, π_e, using data collected using a different *behavior* policy π_b.

3. Related Work

One common approach to OPE is to perform evaluation using importance sampling (IS) (e.g. Precup (2000); Jiang & Li (2016); Thomas & Brunskill (2016)), where the value of the evaluation policy is estimated as a weighted average of the returns of individual trajectories, properly weighted to account for the discrepancy between the evaluation and behavior policy. This is in contrast with the nonparametric approach used in Fonteneau et al. (2010; 2013) where the observed data is used to simulate trajectories.

Another approach to OPE first builds parametric models of the environment given the batch data. The value of the evaluation policy is estimated by simulating trajectories according to the built model (e.g. Chow et al. (2015); Hanna et al. (2017); Paduraru (2012); Liu et al. (2018)). With this approach, care must be taken to minimize the bias of the models due to the lack of counterfactual data which may be important for predicting the dynamics under the evaluation policy (Johansson et al., 2016; Shalit et al., 2017; Liu et al., 2018). Our approach mitigates these concerns by only using the parametric model when there exist no similar transitions in the data.

Several recent works have combined IS-based estimators and model-based estimators to produce better off-policy value estimates. The most common approach uses models as part of doubly-robust methods to reduce the variance of IS-based estimators (e.g. Jiang & Li (2016); Thomas & Brunskill (2016); Farajtabar et al. (2018)). Thomas & Brunskill (2016) go further, switching from an IS-based estimate for the initial part of a trajectory to a model-based estimate for the latter part. In contrast to their work, which only switches once from data to model, our method can switch multiple times depending on which sequence of approaches will result in the most accurate value estimate.

More broadly, the general idea of switching between data and models appears in several places in the RL optimization—rather than off-policy-evaluation—literature. Monte-Carlo Tree Search (MCTS) (Browne et al., 2012; Coulom, 2006) and TD(N) (Watkins, 1989) evaluate policies by making several prediction steps into the future using data before switching to a model. More recently, Doya et al. (2002); Parbhoo et al. (2017; 2018); Peng et al. (2018) optimized trajectories with systems modeled by multiple experts. However, to our knowledge, these kinds of approaches have been used to optimize the value of a policy (often online) but not optimize off-policy evaluation error.

Algorithm 1 MoE simulator

Input: Parametric model — $(\hat{f}_{t,p}, \hat{f}_{r,p})$; Nonparametric model — $(\hat{f}_{t,np}, \hat{f}_{r,np})$; Initial state distribution estimate — $\hat{p}_0$; Number of trajectories to simulate — N_s; evaluation policy — π_e.

for $n = 1$ **to** N_s **do**
  $x_0^{(n)} \leftarrow x_0^{(n)} \sim \hat{p}_0(x)$
  for $t = 0$ **to** T **do**
    $a_t^{(n)} \leftarrow a_t^{(n)} \sim \pi_e(a | x_t^{(n)})$
    Model $\leftarrow$ ChooseModel(x_t, a_t)
    $\hat{f}_{t,MoE}, \hat{f}_{r,MoE} \leftarrow \hat{f}_{t,\text{Model}}, \hat{f}_{r,\text{Model}}$
    $x_{t+1}^{(n)} \leftarrow \hat{f}_{t,MoE}(x_t^{(n)}, a_t^{(n)})$
    $r_t^{(n)} \leftarrow \hat{f}_{r,MoE}(x_t^{(n)}, a_t^{(n)})$
  end for
  $g_T^{(n)} \leftarrow \sum_{t=0}^{T} \gamma^t r_t$
end for
return $\frac{1}{N} \sum_{n=0}^{N_s} g_T^{(n)}$

4. Method

We now introduce our mixture-of-experts (MoE) approach for choosing between parametric and nonparametric models. Algorithm 1 presents the pseudo-code for our MoE simulator. In a high-level, our approach generates N_s trajectories, using the evaluation policy to provide actions and the MoE model to provide transitions, and averages their returns. Specifically, each trajectory begins by sampling an initial state from the empirical distribution in the data. For every step of simulation, we first sample an action from the evaluation policy. Next, we choose between the two models by either greedily choosing the model with the smaller estimated transition error (Algorithm 2 in Appendix A) or using the planning method described in Section 4.1 and Algorithm 3 in Appendix A. We continue sampling the trajectory until some termination condition or maximum trajectory length is reached. The estimated value of the evaluation policy is given by the mean return collected over simulated trajectories.

A core contribution of this work is introducing a way to locally compare the transition prediction error for the parametric and nonparametric models. Accurate estimates are crucial to making sure that the sampled trajectories are as realistic as possible, which is essential to accurately esti-

mate the value of the evaluation policy. In Sections 4.2 and 4.3 we introduce and motivate these error estimators, and in Appendix E we empirically evaluate the quality of these estimators and their ability to locally select the more accurate of the two models.

4.1. Planning to optimize the policy value error bound

We first motivate why we might wish to use a planner to minimize the value estimation error, rather than simply choosing the model that is most accurate for the current state-action pair. Imagine a situation where one of our models is very accurate for a transition in the current state-action pair, but the next state lies in a region where both our models perform very poorly. In such a situation, we may be willing to accept some error in estimating the immediate transition, in order to continue simulating trajectories in regions of the state space where we have high confidence in our models.

Below, we quantify this trade-off by computing the bound on the error for the reward collected over an entire trajectory, and use a planning algorithm to select the model at each time step to minimize that bound. The derivation of the bound is closely related to the derivations in Asadi et al. (2018), with the distinction that we assume the magnitude of model error changes across different regions in the state action space, and we consider the case of deterministic transition and reward functions rather than stochastic ones. Furthermore, we consider modeling errors of both transition and reward functions, rather than only the transition function.

We first bound the state estimation error at a given time step, $\delta(t) \coloneqq \Delta(x_t, \hat{x}_t)$ where $\hat{x}_t$ is the state at time t given that the entire trajectory was simulated using the MoE model (i.e — $\hat{x}_t = \hat{f}_t(\hat{x}_{t-1}) = \hat{f}_t(\hat{f}_t(\hat{x}_{t-2})) = ...$).

Lemma 1 *Let $\varepsilon_t(t)$ be the transition estimation error bound for the chosen model at time-step t,*

$$\varepsilon_t(t) \geq \Delta(\hat{x}_{t+1}, f_t(\hat{x}_t, a_t)) \tag{1}$$

The state error at time-step t is:

$$\delta(t) \coloneqq \Delta(x_t, \hat{x}_t) \leq \sum_{t'=0}^{t-1} (L_t)^{t'} \varepsilon_t(t - t' - 1) \tag{2}$$

where L_t is the Lipschitz constant of the transition function, f_t.

Proof. The proof of Lemma 1 is presented in Appendix B.

Next we compute the bound on the total return error for a particular trajectory.

Theorem 1 *Let $\varepsilon_r(t)$ be the reward estimation error bound for the chosen model at time-step t*

$$\varepsilon_r(t) \geq |f_r(\hat{x}_t, a_t) - \hat{f}_r(\hat{x}_t, a_t)| \tag{3}$$

The total return error for a trajectory is bounded by:

$$\begin{aligned} \delta_g &\coloneqq |g_T - \hat{g}_T| \\ &\leq \sum_{t=0}^{T} \gamma^t [L_r \delta(t) + \varepsilon_r(t)] \\ &\leq L_r \sum_{t=0}^{T} \gamma^t \sum_{t'=0}^{t-1} (L_t)^{t'} \varepsilon_t(t - t' - 1) + \sum_{t=0}^{T} \gamma^t \varepsilon_r(t), \end{aligned} \tag{4}$$

where L_r is the Lipschitz constant of the reward function, f_r.

Proof. From the triangle inequality we have that the bound on the reward for a given time step is

$$\begin{aligned} |r_t - \hat{r}_t| &\leq |f_r(x_t, a_t) - f_r(\hat{x}_t, a_t)| \\ &\quad + |\hat{f}_r(\hat{x}_t, a_t) - f_r(\hat{x}_t, a_t)| \\ &\leq L_r \delta(t) + \varepsilon_r(t). \end{aligned} \tag{5}$$

The proof is completed by summing over all time steps and substituting in Lemma 1. We note that for $L_t > 1$ the return error bound grows exponentially with the planning horizon, reflecting the compounding error phenomenon which is common to planning with imperfect models.

In order to simulate trajectories which minimize this bound we use a Monte-Carlo Tree Search algorithm (MCTS) (Coulom, 2006; Browne et al., 2012) to select the model to simulate the next transition with at each time step. Pseudo-code for the MCTS implemetation of the model selection algorithm is presented in Algorithm 3 in Appendix A. We emphasize that the domain over which the MCTS algorithm plans is *not* the same domain the RL agent operates on. For the MCTS algorithm, a "state" is a state-action pair from the RL domain, possible "actions" are a choice between the parametric and nonparametric model, and the return is the right hand side of the bound above.

In practice, the one-step transition and reward errors, ε_t and ε_r in Equation 4, are unknown. In the next sections we will introduce methods for estimating upper bounds for ε_t and ε_r for both the parametric and nonparametric model.

4.2. Estimating the nonparametric model error

The nonparametric model chooses transitions from transitions that have been actually observed in the data. Specifically, given a state x and action $a \sim \pi_e(a|x)$ the nonparametric model predicts as the next state and reward the corresponding features for the transition whose starting state, x_t^*, is closest to x, and whose action, a_t^*, equals a.

The transition prediction error for the nonparametric model is thus

$$\varepsilon_{\text{t,np}} = \Delta(f_t(x,a), x^*_{t+1}) = \Delta(f_t(x,a), f_t(x^*_t, a)). \quad (6)$$

which can be bounded using the Lipschitz constant of the transition function L_t:

$$\varepsilon_{\text{t,np}} \leq L_t \Delta(x, x^*_t). \quad (7)$$

The Lipschitz constant L_t can be estimated by computing

$$\hat{L}_t = \max_{i \neq j} \frac{\Delta(x^{(i)}_{t'+1}, x^{(j)}_{t''+1})}{\Delta(x^{(i)}_{t'}, x^{(j)}_{t''})}. \quad (8)$$

over all pairs of states $x^{(i)}, x^{(j)}$ in the data (Wood & Zhang, 1996).

However, in practice we expect that this estimate will be too conservative, as we wish to estimate the error locally and for a specific action. For a more realistic estimate of the nonparametric error, we can use Equation (8), but only consider transition pairs from within a given neighborhood of radius C. (We will use the same radius for estimating the parametric error in Section 4.3; we will describe how to choose C in Section 4.4.) Our final estimate will then be

$$\varepsilon_{\text{t,np}} \approx \hat{L}_t \Delta(x, x^*_t). \quad (9)$$

where $\hat{L}_t$ is estimated by using transitions starting within C of x in Equation 8.

We can similarly estimate the reward error as

$$\varepsilon_{\text{r,np}} \approx \hat{L}_r \Delta(x, x^*_t) \quad (10)$$

$$\hat{L}_r = \max_{i \neq j} \frac{|r^{(i)}_{t'} - r^{(j)}_{t''}|}{\Delta(x^{(i)}_{t'}, x^{(j)}_{t''})}. \quad (11)$$

4.3. Estimating the parametric model error

The parametric model error we wish to estimate is

$$\varepsilon_{\text{t,p}} = \Delta(f_t(x,a), \hat{f}_t(x,a)), \quad (12)$$

where $\hat{f}_t(x,a)$ is the parametric model prediction for the next state given (x,a). We estimate the value of ε_{p} as the maximum error over all transitions whose initial state is within distance C of the state of interest x and whose action is a:

$$\hat{\varepsilon}_{\text{t,p}} = \max \Delta\left(\hat{f}_t(x^{(i)}_{t'}, a), x^{(i)}_{t'+1}\right). \quad (13)$$

We use the same neighborhood radius for estimating $\hat{\varepsilon}_{\text{p}}$ as we do for estimating $\hat{L}$ for the non-parametric error bound introduced in Section 4.2.

Similarly we can estimate the parametric reward error as

$$\hat{\varepsilon}_{\text{r,p}} = \max |\hat{f}_r(x^{(i)}_{t'}, a) - r^{(i)}_{t'}|. \quad (14)$$

4.4. Choosing C

It remains to choose the radius C. A large choice of C may be too conservative, smoothing out variation in the data, whereas a small C will result in high variance estimates as there will be few pairs near the desired point x. Here, we discuss one natural choice for specifying the distance radius C: Let $\hat{\varepsilon}^g_{\text{p}}$ and $\hat{L}^g_t$ be the global average parametric dynamics model error and Lipschitz constant for the transition function respectively, computed using all transitions in the data. Then $C\hat{L}^g_t$ is an estimate of the nonparametric model error if the closest point is at distance C. We therefore set our radius C to be when this nonparametric model error equals the global average parametric model error:

$$C\hat{L_t}^g = \hat{\varepsilon}^g_{\text{p}} \Rightarrow C = \frac{\hat{\varepsilon}^g_{\text{p}}}{\hat{L_t}^g}, \quad (15)$$

This choice states that for any distance greater than C, the nonparametric error estimate will exceed the global average model error.

Finally, for any choice of C that is smaller than the diameter of the data set, we may encounter situations in which there exist no observed transitions within the defined neighborhood C of the current state x. Then there will be no pairs available to estimate the Lipschitz constant $\hat{L}_t$, and no way to accurately estimate the nonparametric error. In this situation, we default to the parametric model assuming that it will likely extrapolate more gracefully than the nonparametric model.

4.5. Consistency of the MCTS-MoE simulator

Our primary contribution is to develop an estimator that provides improved empirical performance in limited data settings (which often necessitates model based off policy evaluation approaches). However consistency, the property of converging asymptotically to the correct true value, is a highly desirable property for an estimator and provides a nice reassurance of fundamental soundness. Our MoE algorithm for estimating the value v^{π_e} of the desired evaluation policy π_e is consistent under some assumptions.

Theorem 2 *Under the assumptions 1, 2, 3 in Appendix C, assuming planning error $\epsilon_{planning} = o(1)$, the MoE simulator with MCTS model selection is a consistent estimator of policy value of π_e.*

Due to space limitations, the proof and details on the assumptions are deferred to the Appendix C. Furthermore, in Appendix D we provide proof that the assumption of planning error going to zero is not required for consistency if the model error for the reward is also included in the greedy MoE model selection.

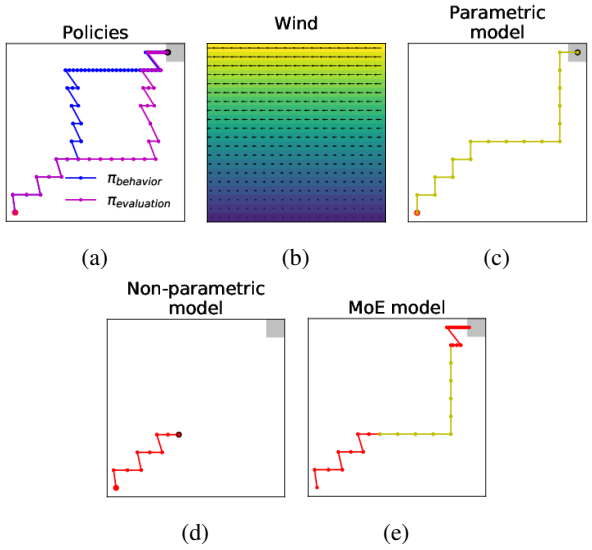


Figure 1. **Demonstration in a continuous 2D map.** The MoE model switches from using the non-parametric to the parametric model in regions where no transitions are observed.

Table 1. Value estimates in 2D motivating example

v^{π_e}	v^{π_b}	$\hat{v}^{\pi_e}_{M_\text{p}}$	$\hat{v}^{\pi_e}_{M_\text{np}}$	$\hat{v}^{\pi_e}_{M_\text{MoE}}$
-40	-53	-18	$-\infty$	**-31**

5. Demonstrations in Synthetic Domains

5.1. Motivating example for mixtures

In this section we illustrate the advantages of of using mixtures in a very simple setting of a myopic planner. We consider a two-dimensional navigation domain. The agent's state is represented by a $x \in \mathbb{R}^2$ coordinate in space. The action space is discrete with 4 actions: up, down, left, right. The transition to the next state is given by

$$x_{t+1} = x_t + \Delta_{ss} \cdot a_t + w(x_t) \tag{16}$$

where Δ_{ss} is a constant which determines the step size, a_t is the chosen action represented as a unit vector in $\mathbb{R}^2$, and $w(x_t)$ is a state dependent "wind" that pushes the agent away from the expected direction.

The agent follows a policy starting from its initial state until it reaches a goal region. The reward is $r = -1$ for all non-goal states and the discount factor is 1. Thus, the value of a policy is minus the expected number of steps required to reach the goal region. In Figure 1a we show trajectories in which the agent starts at the bottom left of the domain and attempts to reach the gray area at the top right. The wind increases linearly with the y-coordinate and is directed in the negative direction of the x-coordinate (Figure 1b). Figure 1a shows trajectories generated under the behavior and evaluation policy. Because the wind is stronger near the top of the domain, the value of the evaluation policy is higher than the behavior's (Table 1).

Advantages of each individual model. As baselines we consider the performance of the parametric and nonparametric models separately. The parametric model has access to the direction and step size of each action, but does not model the wind. Because the parametric model does not take into account the wind which slows the agent down, it overestimates $\hat{v}^{\pi_e}_{M_\text{p}}$ (Table 1), as can be observed in Figure 1c. The non-parametric model does a better job of generating a realistic trajectory in the lower left region of the space where the evaluation and behavior policies are similar, but is unable to simulate a trajectory which continues past the region where the policies deviate from each other (Figure 1d). Because the nonparametric model is unable to generate trajectories which reach the goal state, it estimates $\hat{v}^{\pi_e}_{M_\text{p}}$ to be $-\infty$.

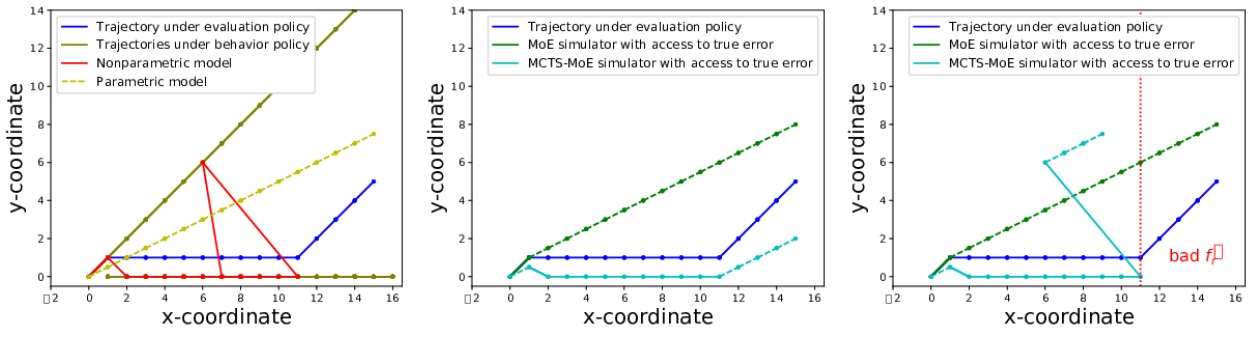
Combining the strengths of both models using the MoE model. The MoE model manages to utilize the best of both models (Figure 1e). It generates a realistic trajectory using the nonparametric model where the behavior and evaluation policies match, and switches to using the parametric model where they don't and observed transitions are not available. Note that the MoE model switches back to using the nonparametric model near the goal where the number of observed transitions is once again high. As shown in Table 1, the MoE model generates the closest estimate for v^{π_e}. Note that even for the MoE model, the estimator cannot converge to the true value of the policy, as it must use the overly optimistic parametric model where no data is collected.

5.2. Motivating examples for Planning with Mixtures

The previous example demonstrated how the MoE simulator, even with a myopic policy, can capitalize on the complementary strengths of parametric and nonparametric models. In this section we demonstrate how further accuracy can be gained by performing the model selection using planning to minimize the errors accumulated over entire trajectories.

This domain is also a 2D navigational domain where the state is defined as $x = (x_{(1)}, x_{(2)})$. There are two possible actions — "right (r)" for which $f_t(x, a = \text{“}r\text{”}) = (x_{(1)} + 1, x_{(2)})$, and "diagonal (d)" for which $f_t(x, a = \text{“}d\text{”}) = (x_{(1)} + 1, x_{(2)} + 1)$. The evaluation and behavior policies are given by

$$\pi_e(a|x) = \begin{cases} \text{“}r\text{”} & 1 \le x_{(1)} \le 11 \\ \text{“}d\text{”} & \text{otherwise} \end{cases} \tag{17}$$



(a) (b) (c)

Figure 2. **Demonstration of planning in a 2D navigation domain.** By using planning to minimize the return estimation error over entire trajectories the MCTS-MoE simulator can incur immediate transition error to decrease long term trajectory simulation error (b) and avoid simulating trajectories where the reward estimation error is too high (c).

$$\pi_b(a|x) = \begin{cases} \text{“}r\text{”} & x_{(1)} > 0 \text{ and } x_{(2)} = 0 \\ \text{“}d\text{”} & \text{otherwise} \end{cases} \tag{18}$$

Initial states for collecting observed trajectories under the behavior policy are $(0, 0)$ and $(1, 0)$, and we wish to evaluate the value of the evaluation policy given the initial state $x_0 = (0, 0)$. The trajectories under the behavior and evaluation policies are shown in Figure 2a. Also shown in Figure 2a are the trajectories generated using the nonparametric and parametric models, where the parametric model for the transitions predicts $\hat{f}_t(x, a) = f_t(x) = (x_{(1)} + 1, x_{(2)} + 0.5)$ regardless of the action (i.e. a model which is not accurate enough to distinguish between actions, but predicts the average of the transitions for each action).

Planning with a perfect reward model. The reward for this domain is $f_r(x, a) = f_r(x) = x_{(1)} + x_{(2)}$. In Figure 2b and 2c we demonstrate the trajectories simulated by the basic MoE simulator and MCTS-MoE simulator, where solid and dashed lines represent transitions simulated using the nonparametric and parametric models, respectively. In the example presented in Figure 2b, the parametric model predicts the true reward at a given state ($f_r = \hat{f}_r$). Because there is no error associated with predicting the reward, the only error in evaluating the evaluation policy stems from errors in predicting the states visited. In this section, we give the MCTS-MoE model access to the true errors of both the parametric and nonparametric reward as our goal is to demonstrate the effect of planning on value estimation, rather than investigate the quality of our error estimators, as we do in Appendix E.

At the first time-step the basic MoE simulator uses the nonparametric model to simulate a transition with zero error, and then uses the parametric model to simulate the following transitions as they incur smaller transition errors than the nonparametric model. However, while the error at each time-step is smaller using the parametric error, by greedily choosing the model which minimizes the immediate transition error the MoE simulator generates a trajectory which over time becomes very different from the true trajectory under the evaluation policy. In contrast, the MCTS-MoE simulator can look into the future and realize that incurring a small transition error in the first time step will lead it to the bottom region of the state space where it can generate transitions with no error by using the nonparametric model. In the first row in Table 2 we present the value estimation error for each of the models and see that the MCTS-MoE outperforms both parametric and nonparametric models, as well as the basic MoE simulator. It is worth mentioning that in this case the basic MoE model performs even worse than the individual models, since greedily choosing the more accurate model in the short term leads it to generate very unrealistic trajectories in the long run — the planning aspect of the MCTS-MoE model is designed to avoid exactly this problem.

Balancing reward and transition errors. So far we considered the case in which the parametric model for the reward is accurate, and therefore minimizing the value estimation error is equivalent to minimizing the states estimation error. Next, we consider the case where the parametric model for the reward may be inaccurate as well. In Figure 2c we plot the MCTS-MoE simulated trajectory where $\hat{f}_{r,p} = f_r$ for $x < 11$ and $\hat{f}_{r,p} = -1$ for $x \geq 11$. In this situation the reward estimation error for simulating a transition using the parametric model is so high for $x \geq 11$, that the MCTS-MoE simulator prefers to incur a large transition estimation error, which is balanced by avoiding the large reward estimation error. In the second row of Table 2 we show that in this case as well the MCTS-MoE simulator outperforms all other simulators.

Table 2. Value estimation errors in the planning toy example

	M_{p}	M_{np}	M_{MoE}	$M_{\text{MCTS-MoE}}$
Accurate $f_{r,p}$	32.5	39	39.5	**17**
Inaccurate $f_{r,p}$	46.5	39	41.5	**19.5**

6. Experimental Results

6.1. Conceptual demonstration — Acrobot

Domain details. As previously discussed, our MoE simulator will offer the biggest advantages in situations where some regions in state space which are expected to be visited by the evaluation policy are not observed under the behavior policy. In these cases the parametric model can be used to generalize and simulate the dynamics in the unobserved regions, while the nonparametric model can offer more accurate predictions in the data rich regions.

To demonstrate this property we use the Acrobot environment from the control literature (Sutton, 1996). The environment simulates two links and two joints, where the joint between the two links can be actuated. The objective of a policy is to control the actuation of the middle joint such that the end of the bottom joint rises above a certain height as quickly as possible. The reward for every time step is -1, and so the value of a policy is the average time it takes to reach the goal height. We train a near-optimal policy with an expectation time of 70 time-steps for the completion of the task. We generate 100 observed trajectories, where trajectories differ from each other due to small perturbations of the initial states. To generate a lack of observed transitions in a particular region in space, we run several experiments, and in each experiment we choose a maximal observable height, and remove from the dataset all observed transitions which start above that height.

We then train a parametric and nonparametric model on the data, and compare the performance of the two models and our greedy MoE model in predicting the expected time it would take for the desired height to be reached. The parametric model is trained as a feed-forward neural net with one layer of 64 hidden units with a $\tanh$ activation function.

Results In Figure 3 we present the RMSE of $\hat{v}^{\pi_e}$ for the different models as a function of the maximal observable height. For low maximal observable heights, the nonparametric model cannot simulate a trajectory in which the Acrobot reaches the desired height, as such transitions are not observed in the data. For such a situation the MoE model fully relies on the parametric model and matches its performance. As the maximal height is increased, the non-parametric model becomes more viable, and the MoE model combines transition predictions from both models and outperforms both individual models. For very high maximal observable heights (the goal height is 1), the nonparametric model becomes very accurate and outperforms the MoE model. This is likely due to errors in our estimate of the transition prediction error, which leads the MoE to select the parametric model in situations where the nonparametric would be more accurate.

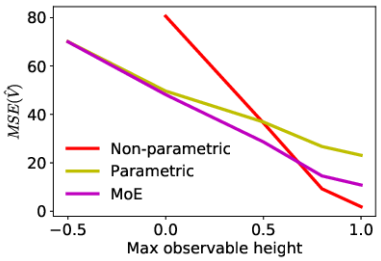
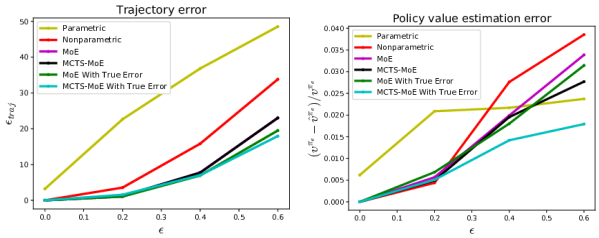
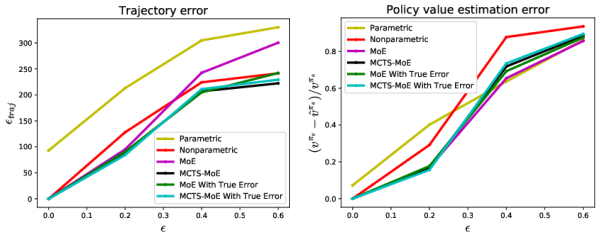


Figure 3. **Acrobot.** The MoE model relies on the parametric models in reigmes when there is not sufficient data for the nonparametric model to be reliable, and combines the advantages of both models in regimes where the models complement each other.



(a) Cancer (b) Cancer



(c) HIV (d) HIV

Figure 4. **Medical simulators.** By combining the advantages of the parametric and nonparametric models, the MoE model can outperform both individual models in terms of simulating accurate trajectories and estimating the evaluation policy value.

6.2. Medical simulators

We compare our MoE simulator with different OPE estimators for two medical simulators: one for cancer (Ribba et al., 2012) and one for HIV (Ernst et al., 2006) patients. In Appendix F we provide details of the simulators and the evaluation policies used. For both domains, we use as behavior policies ϵ-greedy policies of the evaluation policy.

Table 3. Relative value estimation RMSE for medical simulators.

	M_{p}	M_{np}	M_{MoE}	$M_{\text{MCTS-MoE}}$	IS	WDR
Cancer	0.021	0.027	0.020	**0.019**	1.0	0.22
HIV	0.65	0.88	0.64	**0.63**	1.0	0.99

We test the performance of both the greedy MoE simulator and the planning simulator. For each of the two MoE simulators we test their performance both when using the estimates for the errors derived in Sections 4.2 and 4.3, and their performance when they are given access to the true errors of each model. Access to the true error is unrealistic for real data, but it allows us to investigate the quality of our error estimates and how much the MoE simulator can potentially be improved by using better error estimates.

Metrics. For both domains we compared the MoE simulators with the parametric and nonparametric models using two metrics. The first is the difference between the trajectories simulated using the the models and the trajectory which under the true environment. We define the trajectory error as $\varepsilon_{\text{traj}} \coloneqq \sum_{t=0}^{T} \Delta(x_t, \hat{x}_t)$, where $\hat{x}_t$ is the state prediction at time t using the tested model. The second is the RMSE for the evaluation policy value estimate.

Baselines. We compare the evaluation policy value prediction of the MoE simulators with the performance of both the parametric and nonparametric models individually, as well as common importance sampling based OPE methods.

Performance of the different models. We first compared the model based estimators with importance sampling based OPE methods. IS based estimators tend to perform poorly with limited data. Indeed, the value estimation errors for all IS based methods were at least an order of magnitude larger than all model based methods. In Table 3 we present a comparison of the RMSE of the value estimation for both domains and behavior policy with $\epsilon = 0.4$ to demonstrate that for these domains IS methods perform significantly worse than model based methods (comparison to additional IS estimators is presented in Appendix F.1). This huge difference in performance is consistent across all experimental parameters we tested, and we therefore focus the rest of the results in this section on comparing model based OPE methods only.

In Figure 4 we present the trajectory simulation error ($\varepsilon_{\text{traj}}$) for the different models, and the RMSE of the estimated evaluation policy value using the different models and evaluation methods. For the cancer domain (Figures 4a and 4b), the MoE model generates more realistic trajectories (smaller trajectory error) which result in better policy value estimates. Introducing planning further improves the value estimation performance, especially when the MoE has access to the true errors.

For the HIV domain (Figures 4c-4d), the MoE simulator achieves lower trajectory errors, except for high values of ϵ in the behavior policy. We believe this is due to the inaccuracy of the model error estimation, since the MoE simulator achieves lower trajectory error when given access to the true model errors. We note, however, that the introduction of planning allows the MoE simulator to obtain low trajectory error even without access to the true model errors. For this domain, we see that for large values of ϵ, small trajectory error does not necessarily result in smaller policy evaluation error. This is because the Euclidean distance penalizes error in all state dimensions equally, while in this case one state dimension is much more relevant to the reward. In Appendix F.3, we demonstrate how choosing a domain-appropriate metric can further improve the value estimation prediction. Furthermore, in Appendix F.2 we empirically test our simulators for consistency and show that the value estimation error for both domains decreases as the number of trajectories is increased.

7. Discussion

In this paper we demonstrated the effectiveness of a method for combining a parametric and nonparametric model for performing OPE. Our method is consistent (under mild assumptions) in the limit of infinite data, while effectively choosing sequences of imperfect models to reduce the error in the value estimate with finite data.

Our methods take advantage of techniques used in planning for off-policy evaluation. While MCTS worked well as a planner for our tasks, one can imagine substituting any future modern planner. Similarly, we found that our approach for estimating the errors of the models allowed the planner to make choices for more accurate off-policy evaluation than the baselines. That said, improving the quality of error estimates is an important direction for research. A very related question is of what metric is appropriate for a particular domain. We imagine that in many domains expert knowledge may be available; there are also interesting directions in optimizing that metric from data.

Finally, in this work we assumed that the transition and reward functions are deterministic. We emphasize that our approach can be applied to stochastic domains without modification to the planning algorithm; the only change would be defining an appropriate error estimate—for example, rather than defining the transition error as a distance between the true and simulated next state, we might define it as the distance between the state distribution under the true environment and the one predicted by the models. Producing accurate estimates in the stochastic setting is an interesting direction for future work.

Acknowledgements

FDV and OG acknowledge support from NSF RI- 1718306.

References

Asadi, K., Misra, D., and Littman, M. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 264–273, 2018.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Chow, Y., Petrik, M., and Ghavamzadeh, M. Robust policy optimization with baseline guarantees. *arXiv preprint arXiv:1506.04514*, 2015.

Coulom, R. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

Doya, K., Samejima, K., Katagiri, K.-i., and Kawato, M. Multiple model-based reinforcement learning. *Neural computation*, 14(6):1347–1369, 2002.

Ernst, D., Stan, G.-B., Goncalves, J., and Wehenkel, L. Clinical data based optimal sti strategies for hiv: a reinforcement learning approach. In *Decision and Control, 2006 45th IEEE Conference on*, pp. 667–672. IEEE, 2006.

Farajtabar, M., Chow, Y., and Ghavamzadeh, M. More robust doubly robust off-policy evaluation. In *International Conference on Machine Learning*, pp. 1446–1455, 2018.

Fonteneau, R., Murphy, S., Wehenkel, L., and Ernst, D. Model-free monte carlo-like policy evaluation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, pp. 217–224, 2010.

Fonteneau, R., Murphy, S. A., Wehenkel, L., and Ernst, D. Batch mode reinforcement learning based on the synthesis of artificial trajectories. *Annals of operations research*, 208(1):383–416, 2013.

Hanna, J. P., Stone, P., and Niekum, S. Bootstrapping with models: Confidence intervals for off-policy evaluation. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 538–546. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

Jiang, N. and Li, L. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pp. 652–661. JMLR. org, 2016.

Johansson, F., Shalit, U., and Sontag, D. Learning representations for counterfactual inference. In *International Conference on Machine Learning*, pp. 3020–3029, 2016.

Liu, Y., Gottesman, O., Raghu, A., Komorowski, M., Faisal, A. A., Doshi-Velez, F., and Brunskill, E. Representation balancing mdps for off-policy policy evaluation. In *Advances in Neural Information Processing Systems*, pp. 2644–2653, 2018.

Paduraru, C. *Off-policy evaluation in Markov decision processes*. PhD thesis, Ph. D. Dissertation. McGill University, 2012.

Parbhoo, S., Bogojeska, J., Zazzi, M., Roth, V., and Doshi-Velez, F. Combining kernel and model based learning for hiv therapy selection. *AMIA Summits on Translational Science Proceedings*, 2017:239, 2017.

Parbhoo, S., Gottesman, O., Ross, A. S., Komorowski, M., Faisal, A., Bon, I., Roth, V., and Doshi-Velez, F. Improving counterfactual reasoning with kernelised dynamic mixing models. *PloS one*, 13(11):e0205839, 2018.

Peng, X., Ding, Y., Wihl, D., Gottesman, O., Komorowski, M., Lehman, L.-w. H., Ross, A., Faisal, A., and Doshi-Velez, F. Improving sepsis treatment strategies by combining deep and kernel-based reinforcement learning. In *AMIA Annual Symposium Proceedings*, volume 2018, pp. 887. American Medical Informatics Association, 2018.

Precup, D. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, pp. 80, 2000.

Ribba, B., Kaloshi, G., Peyre, M., Ricard, D., Calvez, V., Tod, M., Bernard, B. C., Idbaih, A., Psimaras, D., Dainese, L., et al. A tumor growth inhibition model for low-grade glioma treated with chemotherapy or radiotherapy. *Clinical Cancer Research*, pp. clincanres–0084, 2012.

Shalit, U., Johansson, F. D., and Sontag, D. Estimating individual treatment effect: generalization bounds and algorithms. In *International Conference on Machine Learning*, pp. 3076–3085, 2017.

Sutton, R. S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pp. 1038–1044, 1996.

Thomas, P. and Brunskill, E. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 2139–2148, 2016.

Watkins, C. J. C. H. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

Wood, G. and Zhang, B. Estimation of the lipschitz constant of a function. *Journal of Global Optimization*, 8 (1):91–103, 1996.