# Reinforcement Learning with Misspecified Model Classes

Joshua Joseph, Alborz Geramifard, John W. Roberts, Jonathan P. How and Nicholas Roy
Massachusetts Institute of Technology
77 Massachusetts Ave., Cambridge, MA 02139 USA
{jmjoseph, agf, jwr, jhow, nickroy}@mit.edu

***Abstract*— Real-world robots commonly have to act in complex, poorly understood environments where the true world dynamics are unknown. To compensate for the unknown world dynamics, we often provide a class of models to a learner so it may select a model, typically using a minimum prediction error metric over a set of training data. Often in real-world domains the model class is unable to capture the true dynamics, due to either limited domain knowledge or a desire to use a small model. In these cases we call the model class *misspecified*, and an unfortunate consequence of misspecification is that even with unlimited data and computation there is no guarantee the model with minimum prediction error leads to the best performing policy. In this work, our approach improves upon the standard maximum likelihood model selection metric by explicitly selecting the model which achieves the highest expected reward, rather than the most likely model. We present an algorithm for which the highest performing model from the model class is guaranteed to be found given unlimited data and computation. Empirically, we demonstrate that our algorithm is often superior to the maximum likelihood learner in a batch learning setting for two common RL benchmark problems and a third real-world system, the hydrodynamic cart-pole, a domain whose complex dynamics cannot be known exactly.**

## I. Introduction

Controlling a system with complex, unknown dynamics (*e.g.*, the interaction of a robot with a fluid) is a common and difficult problem in real-world domains. The hydrodynamic cart-pole, shown in Figure 1 is such an example, constructed to capture many of the fundamental challenges associated with fluid interaction (described in greater detail in Section V-C). The system is composed of a thin flat plate (the pole) placed in a flowing channel of water with a linear actuator (the cart) attached to the pole's trailing edge via a pin joint. For this system, the objective is to learn a policy that stabilizes the pole pointing into the water current.

Reinforcement learning (RL) [1] is a framework for sequential decision making under uncertainty with an unknown dynamics model that can be used to learn a policy for problems such as the hydrodynamic cart-pole. Classical RL techniques tend to use powerful representations with many parameters that translates into high sample complexity. Even with the use of adaptive representations, modern model-free RL techniques tend to require many samples to achieve reasonable policies [2]. For real-world systems, collecting large data sets can be expensive, dangerous, or even impossible [3], so developing techniques with low data needs is paramount.

Model-based (MB) RL techniques have been shown to be more sample-efficient than model-free RL methods [4], [5], [6], [7], making them a natural choice for tackling real-world robotics problems [8]. Since the model of the system is often not fully known, practitioners often rely on domain knowledge to choose a dynamics model class that reasonably trades off sample complexity for expressive power. For real-world systems such as the hydrodynamic cart-pole, this trade-off means the chosen representation will often be *misspecified* (*i.e.,* the approximate representation cannot exactly capture the true dynamics). Such models can introduce *representational bias*, the difference between the performances of the true optimal policy and the best policy found based on the misspecified model class. Furthermore the learning algorithm using such model classes can introduce *learning bias*, which is the the difference between the performances of the best possible policy in the model class and the policy that is produced by the learner [9].

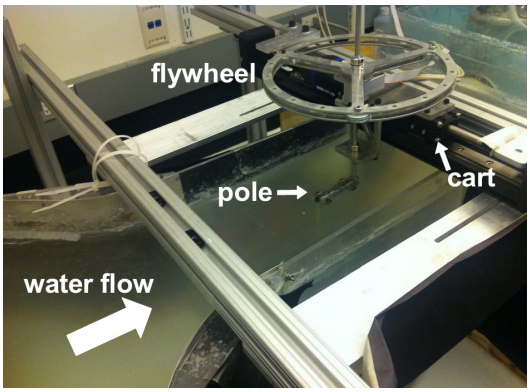



**Figure 1**. The hydrodynamic cart-pole system with the pole pointing upstream into the water current.

The focus of this paper is on identifying the cause of learning bias of MB RL and presenting an algorithm to overcome it. For a given problem, typical MB RL algorithms choose a representation from the class of potential representations by minimizing a form of error measured on the training data (*e.g.*, maximum likelihood). Unfortunately, when no representation in the chosen class can capture the true representation, the fit by the standard minimum error metrics (described in more detail in Section II) does not necessarily result in the highest possible performing policy. In other words, by optimizing a quantity other than the performance (*e.g.,* prediction error), the standard approaches introduce learning bias.

To address the issue of learning bias, Section III introduces Reward Based Model Search (RBMS), a batch MB RL algorithm that estimates the performance of models in

the model class and explicitly searches for the model that achieves the highest performance. Given infinitely dense data and unlimited computation, Section IV shows that the asymptotic learning bias of RBMS for *any* representation class is zero. Moreover, for limited data and computation RBMS performs provably no worse than the minimum error solution in expectation. Section V empirically demonstrates that RBMS often results in a large performance increase over minimum error on two common RL benchmark problems and on the real-world hydrodynamic cart-pole system. We then discuss related work and conclusions in Sections VI and VII.

## II. Minimum Error Reinforcement Learning

A Markov decision process (MDP) is a framework for sequential decision making under uncertainty [10]. A finite time MDP is defined by a tuple $(S, A, s_0, m, R, T)$, where $S$ is the set of states, $A$ is the set of actions, and $s_0$ is the starting state. The dynamics model is defined as, $m : S \times A \times S \mapsto [0, 1]$, where $m(s, a, s') = p(s'|s, a)$, the probability of transitioning to state $s'$ by taking action $a$ in state $s$. The reward function, $R : S \mapsto \mathbb{R}$, describes the reward the agent receives for being in state $s$.[1] An episode of data is a sequence $s_0, a_0, s_1, a_1, \cdots, s_{T-1}, a_{T-1}$, where $T$ is the episode length, $a_t$ is the action taken at state $s_t$, and $s_{t+1} \sim m(s_t, a_t, \cdot)$.[2] The performance, or *return*, of a policy, $\pi : S \to A$, is defined as

$$V^{\pi}(s_0) = E_{\pi}\left[\sum_{t=0}^{T-1} R(s_t) \middle| s_{\tau+1} \sim m(s_{\tau}, \pi(s_{\tau}), \cdot)\right], \quad (1)$$

where the expectation is taken over $s_1, ..., s_T$. We do not use a discount factor due to the fixed finite horizon. The objective of an MDP is to compute the optimal policy, $\pi^*$, where

$$\pi^* = \text{argmax}_{\pi} V^{\pi}(s_0). \quad (2)$$

We assume $m$ is unknown, which prevents us from calculating $V^{\pi}(s)$ using Equation 1. Instead, we must estimate $V^{\pi}(s)$ from data. Additionally, even if $m$ were known, solving Equation 2 using Equation 1 is often extremely challenging due to the difficulty of computing a closed form expression of Equation 1 even for simple MDPs. One method for estimating $V^{\pi}(s_0)$ is through Monte Carlo (MC) policy evaluation [11], where $\pi$ is executed for $N$ episodes starting from $s_0$. The estimate is computed using

$$\hat{V}^{\pi}(s_0) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T-1} R(s_t^n), \quad (3)$$

where $s_t^n$ is the agent's state at time $t$ of episode $n$ and state transitions are sampled $m$.

Given a method for evaluating Equation 1, policy search (PS) methods parameterize the set of policies and search over the parameter space to solve for the optimal policy in Equation 2 using

$$\theta^* = \text{argmax}_{\theta \in \Theta} V^{\pi_{\theta}}(s_0), \quad (4)$$

where $\theta \in \Theta$ is the policy's parameterization [12]. A naive PS technique to find $\theta^*$ is to enumerate all parameter values $\theta_1, ..., \theta_{|\Theta|}$, compute $\hat{V}^{\pi_{\theta_1}}(s_0), ..., \hat{V}^{\pi_{\theta_{|\Theta|}}}(s_0)$, and select the parameter which achieves the highest return. However, this approach is impractical for many real-world domains as the parameter space is often large or continuous, preventing $\Theta$ from being directly searched over or even enumerated. Large policy spaces can be overcome using policy gradient [13], [14], where the parameter estimate, $\hat{\theta}$, is updated using a gradient step

$$\hat{\theta}_{i+1} = \hat{\theta}_i + c\frac{\partial V^{\pi_{\hat{\theta}_i}}(s_0)}{\partial \hat{\theta}_i}, \quad (5)$$

for iteration $i$ where $c > 0$ and $\hat{\theta}_i$ is updated until convergence. Equation 5 allows us to change $\hat{\theta}$ proportional to the gradient of return, which can be estimated using Equation 3 with data generated from the policy we wish to evaluate (*i.e., on-policy* data). Therefore each $\hat{\theta}_i$ must be simulated a sufficient number of times to accurately estimate $V^{\pi_{\hat{\theta}_i}}(s_0)$.

Model-based (MB) methods solve Equation 1 by explicitly learning a dynamics model, $m(s, a, s'; \theta)$ from data, which commonly results in a significant amount of data efficiency by generalizing in the space of transitions.[3] Maximum likelihood (ML) is a common method for inferring the dynamics model which maximizes the likelihood of the model conditioned on the data, where

$$\hat{\theta} = \text{argmax}_{\theta \in \Theta} \prod_{n=1}^{N}\prod_{t=0}^{T-1} m(s_t^n, a_t^n, s_{t+1}^n; \theta). \quad (6)$$

A MB solver first uses Equation 6 to compute $\hat{\theta}$ and then uses Equations 1 and 2 to compute the policy optimal with respect to $\hat{\theta}$. The implicit assumption made by selecting a representation "closest" to the true representation (using minimum error) is that the policy which is optimal with respect to the minimum error representation will perform well in the true world. If, however, the representation for the dynamics model is not expressive enough to capture the true $m$, using Equation 6 can result in a poorly performing policy [15]. For comparison, we focus on the ML metric in this work, due to its popularity in the literature, although our results generalize to any prediction error metric (*e.g.,* minimum squared error).

## III. Reward Based Model Search

Section II discussed that minimum error techniques in RL are vulnerable to learning poor policies when the representations for the dynamics model is not expressive enough to capture the true dynamics. This section presents a novel batch MB RL approach that learns a representation which explicitly maximizes Equation 4. We treat the dynamics model class as a parameterization of the policy and search for $\hat{\theta} \in \Theta$

[1] Without loss of generality, we assume a known reward function that is only state-dependent, and a known, deterministic starting state. When the reward function is unknown, a sampled reward $r_t^n$ can be used throughout in place of $R(s_t^n)$.

[2] We focus on finite time MDPs in the interest of theoretical results.

[3] We purposely use $\theta$ for both the policy and model parameterization to illustrate that a dynamics model is an indirect policy representation.

**Algorithm 1:** MFMC Policy Return Estimation

**Input**: $\pi,\ \mathcal{D},\ s_0,\ \Delta,\ T,\ p$
**Output**: $V^{\pi}(s_0)$

**for** $n = 1$ to $p$ **do**
  $\tilde{s} \leftarrow s_0$, $e_n \leftarrow [\,]$
  **for** $t = 0$ to $T$ **do**
    $\tilde{a} \leftarrow \pi(\tilde{s})$
    $(s_t^n, a_t^n, s_{t+1}^n) \leftarrow \text{argmin}_{(s,a,s') \in \mathcal{D}} \Delta((\tilde{s}, \tilde{a}), (s, a))$
    $\mathcal{D} \leftarrow \mathcal{D} \setminus (s_t^n, a_t^n, s_{t+1}^n)$
    Append $(s_t^n, a_t^n, s_{t+1}^n)$ to $e_n$
    $\tilde{s} \leftarrow s_{t+1}^n$
**return** $V^{\pi}(s_0)$ (computed using Equation 3)

that achieves the highest return, rather than optimizing a different metric (*e.g.*, maximum likelihood). We can think of the policy as being indirectly parameterized by $\theta$ which defines the model and is updated using a policy gradient approach, such that

$$\hat{\theta} = \hat{\theta} + c\frac{\partial V^{\pi^{\theta}}(s_0)}{\partial \hat{\theta}}, \tag{7}$$

where $c > 0$.

Section III-A outlines a method for estimating the return of a policy in continuous state spaces called Model-Free Monte Carlo (MFMC) [16]. MFMC estimates a policy's return directly from data using Equation 3 rather than from a dynamics model learned from the data. Section III-B describes a method of gradient ascent in order to maximize Equation 4. Section III-C presents our overall approach as an algorithm.

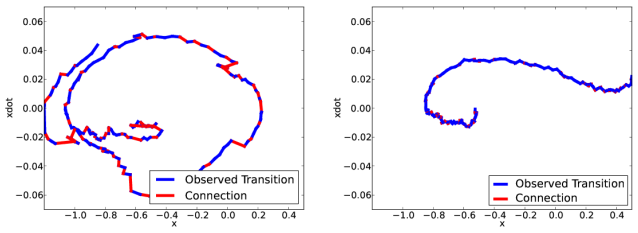
### A. Estimating a the Return of a Policy

Ideally, we would choose policies using Equations 3 and 4 with on-policy data. This results in a sample complexity at least linear in the number of evaluated policies, an impractical amount for most real-world problems. We can reduce this sample complexity by sharing data across iterations, using *off-policy* data – data generated under a policy different from the one we are evaluating. Therefore, we need an approach that allows us to perform MC-like policy evaluation from off-policy data.

In continuous state spaces, creating on-policy episodes from off-policy data is challenging because a single state is rarely visited more than once. We use Model-free Monte Carlo (MFMC) [16] for policy evaluation from off-policy data in continuous state spaces. For a policy, $\pi^{\theta}$, MFMC creates pseudo on-policy episodes from off-policy training data to compute statistics of the policy with bounded estimation error.

To construct an episode of on-policy data, MFMC uses a set of training data and a distance function. We define the data set as $\mathcal{D} = \{(s_i, a_i, s_i')\}_{i=0}^{|\mathcal{D}|}$, where $s_i' \sim m(s_i, a_i, \cdot)$. The designer-provided distance function takes the form

$$\Delta\left((s_i, a_i), (s_j, a_j)\right) = ||s_i - s_j||_S + ||a_i - a_j||_A, \tag{8}$$




(a) 5 episodes of data (b) 50 episodes of data

**Figure 2.** Phase space plots for pseudo on-policy episodes constructed for a well performing policy using the MFMC algorithm with 5 and 50 episodes of batch data.

where $i$ and $j$ are one-step transitions from $\mathcal{D}$. MFMC constructs episodes starting at $s_0$ and sequentially adds transitions such that at time $t$, when the agent is in state $\tilde{s}$, the next transition $(s_t, a_t, s_{t+1}) = \text{argmin}_{(s,a,s') \in \mathcal{D}} \Delta((\tilde{s}, \pi^{\theta}(\tilde{s})), (s, a))$. Each transition in $\mathcal{D}$ can only be used once and episodes are terminated after $T$ transitions.

MFMC also requires $p$, the number of episodes used for policy evaluation. The decision of $p$, relative to $|\mathcal{D}|$, trades off bias and variance (see [16] for analysis, bounds on bias and variance, and discussion on the trade-off). Algorithm 1 is a reproduction of MFMC from [16] for the reader's convenience.

A visualization the of episode construction procedure is shown in Figure 2 for the mountain car domain (described in Section V-A) where the agent begins near the center of the diagram and attempts to reach $x \geq 0.5$. The figure shows two phase-space plots of constructed episodes, where transitions observed in the data set are shown in blue, and are connected by red lines for illustration purposes. Episodes were generated using a well performing policy that reaches the goal. Figures 2(a) and 2(b) show two episodes constructed based on 5 and 50 episodes, respectively. Actual observed interactions are shown in blue while discontinuities in tailoring the episodes are highlighted as red segments. Notice that more data leads to an overall smoother prediction of the system evolution and the episode more accurately approximates an on-policy sequence of interactions.

### B. Policy Improvement

The second step to solving Equation 4 is the maximization over $\Theta$. As discussed in Section II, enumerating and evaluating all possible parameter values is impractical for real-world problems. Policy gradient approaches overcome this hurdle by updating the parameter value in the direction of increasing performance (Equation 7).

We take the policy gradient type approach as described in Section II, only we are updating the parameters of the dynamics model rather than a parameterization of the policy. Unfortunately, gradient ascent is known to be susceptible to local maxima. To reduce the likelihood of becoming stuck in a poor local maxima, we use random restarts in addition to including the maximum likelihood solution in the set of potential starting points (see Section IV). For this work, we chose the basic form of gradient ascent for policy

**Algorithm 2:** RBMS

**Input**: $\mathcal{D}$, $\tilde{\Theta}$, $\delta^{init}$, $\delta^{min}$
**Output**: $\pi^{\theta_{RBMS}}$

1 $V^{\pi^{\theta_{start}}}(s_0) \leftarrow -\infty$, $\theta \leftarrow \theta_{start}$, $\delta \leftarrow \delta^{init}$
2 **while** $\delta > \delta^{min}$ **do**
3  **for** each dimension of $\Theta$ **do**
4  **while** `True` **do**
5  $\{\pi^{\theta-}, \pi^{\theta}, \pi^{\theta+}\} \leftarrow$ Compute the policies for $\{\theta - \delta, \theta, \theta + \delta\}$
6  $\{V^{\pi^{\theta-}}(s_0), V^{\pi^{\theta}}(s_0), V^{\pi^{\theta+}}(s_0)\} \leftarrow$ Estimate the return of $\{\pi^{\theta-}, \pi^{\theta}, \pi^{\theta+}\}$
7  **if** $\max(V^{\pi^{\theta+}}(s_0), V^{\pi^{\theta-}}(s_0)) \leq V^{\pi^{\theta}}$ **then**
8  `Break`
9  **if** $V^{\pi^{\theta+}}(s_0) > V^{\pi^{\theta-}}(s_0)$ **then**
10  $\theta \leftarrow \theta + \delta$
11  **else**
12  $\theta \leftarrow \theta - \delta$
13  $\delta \leftarrow \delta/2$
14 **return** $\pi^{\theta}$

improvement described in Section III-C.[4]

### C. RBMS Algorithm

Our overall approach is presented in Algorithm 2. The inputs to the algorithm are the data, $\mathcal{D}$, our representation, $\Theta$, the initial gradient ascent step size, $\delta^{init}$, and the minimum gradient ascent step size, $\delta^{min}$.

RBMS begins gradient ascent by selecting a dimension of the model space to search and computes the policies for models $\theta$, $\theta + \delta$, and $\theta - \delta$ (line 5), where the parameter $\delta$ is the step size in the model space. Once the policies are computed, we use MFMC to estimate the return of each of these policies (line 6) and estimate the difference in return between the current best model $\theta$ and the models $\theta + \delta$ and $\theta - \delta$. If neither of these models yield an increase in the estimated return, we continue to the next dimension. If $\theta + \delta$ or $\theta - \delta$ are positive steps in terms of estimated return, we take the step in the steepest direction. When all dimensions have been stepped through, $\delta$ is decreased (line 13) and the process is continues until $\delta$ reaches $\delta^{min}$. The algorithm is then repeated for many random restarts, including the ML parameters.

## IV. Theoretical Analysis

As described in Section III-A, MFMC constructs a set of on-policy episodes from off-policy data and uses Equation 3 to estimate the return of $\pi^{\theta}$. This section provides the formal analysis which shows that with infinitely dense data and unlimited computation, RBMS is guaranteed to find the highest performing model from the model class (Section IV-A). We relax these assumptions in Section IV-B and guarantee that using RBMS after finding the ML solution will never result in worse performance in expectation. Section V empirically demonstrates that RBMS often results in a substantial improvement over ML techniques.

The notion of data sparsity is introduced in [16], where the $k$-sparsity of $\mathcal{D}$ is defined as

$$\alpha_k(\mathcal{D}) = \sup_{(s,a)} \Delta_k^{\mathcal{D}}(s, a), \tag{9}$$

in which $\Delta_k^{\mathcal{D}}$ is the distance of $(s, a)$'s $k$-th nearest neighbor measured by $\Delta$ over $\mathcal{D}$. Using $k$-sparsity, as defined in Equation 9, the learning bias of MCMC is bounded by

$$\left| V^{\pi^{\theta}}(s_0) - E\left[\hat{V}^{\pi^{\theta}}(s_0)\right] \right| \leq C\alpha_{pT}(\mathcal{D}), \tag{10}$$

where $k = pT$ and $C$ is a function of the Lipschitz continuity constants describing $m$, $R$, and $\pi$ [16]. Note that the bound on the estimation error is independent of $\theta$ and the expectation is taken over the data. The bound from Equation 10 provides us with the theoretical piece that both Sections IV-A and IV-B build on.

### A. Infinitely Dense Data and Unlimited Computation

Solving Equation 2 is difficult because of two complex calculations: accurately estimating $V^{\pi^{\theta}}(s_0)$ and performing the optimization over policies. Equation 10 tells us that as the density of the data increases (meaning the sparsity $\alpha_{pT}(\mathcal{D}) \to 0$), our estimate $\hat{V}^{\pi^{\theta}}(s_0) \to V^{\pi^{\theta}}(s_0)$.

With unlimited computation, we have the ability to search over the entire representation space $\Theta$ to find $\theta^* = \text{argmax}_{\theta \in \Theta} V^{\pi^{\theta}}(s_0)$.[5] This implies that with infinitely dense data and unlimited computation, RBMS is guaranteed to find $\theta^*$, the highest performing model from the model class.

### B. Limited Data and Computation

While the guarantee of finding $\theta^*$ is a nice theoretical property (Section IV-A), for real-world domains we can never expect to have infinitely dense data nor unlimited computation. Here we relax both assumptions and guarantee that using RBMS after finding the ML solution will never result in worse performance in expectation.

*1) Limited Computation:* With limited computation, we no longer have the ability to perform the maximization over $\Theta$ and instead resort to gradient ascent. Due to the non-convexity of $V^{\pi^{\theta}}(s_0)$ as a function of $\theta$, we can no longer guarantee that $\theta^*$ is found. By initializing gradient ascent with $\theta_{ML}$, the ML model, we guarantee that RBMS takes steps in the direction of improving on the ML solution. In addition to $\theta_{ML}$, we also initialize gradient ascent with many random starting locations drawn from $\Theta$ to further increase our chances of improving on the ML solution.[6]

[4] While more intelligent optimization methods [17] were considered, we found that this basic form of gradient ascent was sufficient for our optimization needs and rarely encountered difficulties with local maxima.

[5] This is trivially accomplished for discrete spaces and can be done to an arbitrary precision in continuous spaces.

[6] In the analysis, $\theta_{ML}$ can be trivially replaced by any prediction error metric (e.g., minimum squared error).

*2) Limited Data:* The consequence of limited data is that $\hat{V}^{\pi^\theta}(s_0)$ can be an inaccurate estimate of $V^{\pi^\theta}(s_0)$ and cause gradient steps to decrease $V^{\pi^\theta}(s_0)$. Hence, one could imagine using the bound from Equation 10 and only step from $\theta_i$ to $\theta_{i+1}$ if $\hat{V}^{\pi^{\theta_{i+1}}}(s_0) \geq \hat{V}^{\pi^{\theta}_i}(s_0) + 2C\alpha_{pT}$. This would ensure that the step from $\theta_i$ to $\theta_{i+1}$ result in $V^{\pi^{\theta_{i+1}}}(s_0) \geq V^{\pi^{\theta_i}}(s_0)$ and consequently that $V^{\pi^{\theta_{RBMS}}}(s_0) \geq V^{\pi^{\theta_{ML}}}(s_0)$, where $\theta_{RBMS}$ is the model selected by RBMS. Unfortunately, this bound only holds in expectation over the data. For future work, we plan develop a probabilistic bound based on Hoeffding's inequality for MFMC to ensure that gradient steps increase $V^{\pi^\theta}(s_0)$ with high probability.

## V. Empirical Results

The experiments described in this section highlight the performance improvement of Reward Based Model Search (RBMS) over maximum likelihood (ML) learners on the RL benchmark problems of mountain car and cart-pole and the real-world hydrodynamic cart-pole system. The two benchmark domains allow for easier understanding and more extensive evaluation to study how quickly the performance of ML and RBMS degrade as the model classes become increasingly misspecified. To study the different types of misspecification, Section V-A investigates both sharp misspecification (unmodeled discontinuities in the true dynamics) and irrelevant data (unhelpful data from a region of our state-space that well performing policies will not visit). Section V-B investigates unmodeled noise and smooth misspecification (*i.e.,* a gradual change in misspecification as the agent moves through the state-space).

We compare RBMS and ML with misspecified model classes to a large tabular Markov model fit using ML.We show the resulting performance versus the amount of training data to highlight the sample complexity advantage of RBMS using small model classes over tabular representations, despite being misspecified. For all model classes, policies were found by first finely discretizing the continuous model and performing value iteration using the same discretization for the value function [10]. RBMS was run with $p = 10$ and, unless otherwise specified, used the distance function

$$\Delta\left((s,a)),(s',a')\right) = \begin{cases} \sum_{d=1}^{D} \frac{|s^d - s'^d|}{s^d_{max} - s^d_{min}} & \text{if } a = a' \\ \infty & \text{otherwise} \end{cases},$$

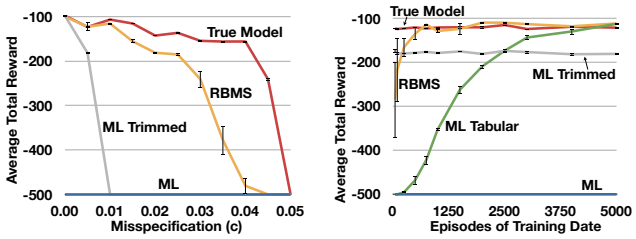
where $s^d$ is the $d$-th dimension.

### A. Mountain Car

Mountain car [18] is a standard RL benchmark simulation where the agent starts in a valley between two hills and takes actions to reach the top of the right hill and receives -1 reward for every time step the agent is not at the goal. Episodes end when the agent reaches the goal or after 500 steps. The standard mountain car dynamics are

$$x_{t+1} = x_t + \dot{x}_t, \quad \dot{x}_{t+1} = \dot{x}_t + a + \theta_1 \cos(\theta_2 x)$$

with action $a \in \{-0.001, 0.001\}$. A uniform discretization of $x$ and $\dot{x}$ with a $500 \times 250$ grid was used to represent value functions. Note the differing numbers of cells in each dimension were chosen to keep the representation small, because it is also used as the dynamics representation for the large tabular model approach. This ensured the large tabular model was made as competitive as possible in terms of data requirements.




(a) Performance vs Misspecification (b) Performance vs Data Size

**Figure 3**. Performance versus misspecification (a) and performance versus data size for $c = 0.005$ (b) for RBMS with the standard mountain car model (yellow), ML with the standard mountain car model (blue), ML with the standard mountain car model with irrelevant data removed (gray), ML with the large tabular Markov model (green), and the true model (red). The error bars show one standard error.

To study how the two approaches' performances change as their model class becomes increasingly misspecified, we modified the standard dynamics in three ways. First, we added stochasticity to the car's starting location by uniformly sampling from the interval $[-\pi/6 - 0.1, -\pi/6 + 0.1]$. Second, we simulated a rock at $x = 0.25$, such that when the car hits the rock its velocity ($\dot{x}$) decreased by $c$. Therefore, increasing $c$ corresponds to the standard car dynamics model class becoming increasingly misspecified. Third, we included a second valley to the left of the standard valley with significantly different dynamics (the goal is still on the right hill of the right valley). For the right valley we used $\theta_1^{right} = -0.0025$, $\theta_2^{right} = 3$ for the dynamics, as is commonly done in the literature [10], and $\theta_1^{left} = -0.01$, $\theta_2^{left} = 3$ for the left valley. For this experiment ML and RBMS were given the standard mountain car dynamics model, parameterized by $\theta_1$ and $\theta_2$, with both hills sharing the same parameters. The training data was generated from uniformly random policies.

Figure 3(a) shows the results of ML (blue, gray) and RBMS (yellow) as their model class becomes increasingly misspecified (with increasing $c$) for 2500 episodes of training data. For reference, the return of the planner with the true model[7] is shown in red. The blue line shows that the standard ML approach never performs well due to the data from the left hill biasing its model estimate. While it may seem straightforward to remove irrelevant data before fitting a model (*e.g.*, data from the left valley), in general for difficult problems (*e.g.*, the hydrodynamic cart-pole), identifying regions of irrelevant data can be extremely difficult. Shown in gray is a demonstration that even if we were able to trim irrelevant data before using ML, other unmodeled effects (a small influence of the rock) still cause ML to perform poorly.

In contrast, RBMS is able to learn models which allow it to reach the goal for a large range of misspecification both from the increasing effect of the rock and irrelevant data.

[7]Note that the planner first discretizes the continuous model to find a policy, so the policy plotted for the "true model" is actual a finely discretized representation of the true continuous model.

Additionally, we can see that when the rock had little effect ($c \leq .005$), RBMS performed as well as the true model. Eventually, the influence of the rock is too significant for any policy to escape the valley, as shown by the red line eventually dropping to -500.

Figure 3(b) shows the results of ML (blue, gray), RBMS (yellow), the large tabular Markov model (green) as the amount of training data increases for fixed $c = 0.005$. For reference, the return of the planner with the true model is shown in red. This figure illustrates that less data is required to use a small model, as opposed to a large tabular Markov model with many parameters. Although the large tabular model does eventually achieve the performance of RBMS and the true model, such large data requirements are often prohibitive for real-world problems. Our approach, on the other hand, performs well using only 500 episodes, an order of magnitude reduction in the amount of training data compared to ML Tabular.

### B. Cart-pole

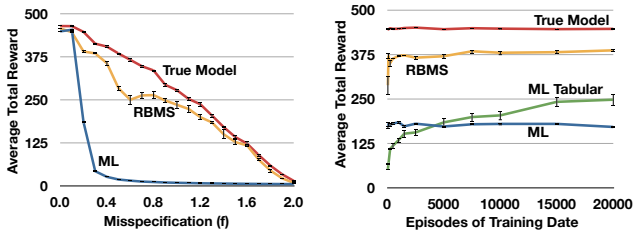
The cart-pole system [19] is composed of a cart that moves along a single axis with a pendulum (the pole) attached by a pin joint. We focus on the task of stabilizing the pole in an upright position against gravity.

The cart-pole deterministically starts at $x = \psi = \dot{x} = \dot{\psi} = 0$, where $x$ is the position of the cart, $\psi$ is the angle of the pole, and the action space $A = \{-5, 5\}$. An episode ends when the pole falls over (defined as $|\psi| > \pi/15$) or after 500 steps. The reward function is $1 - |\theta|/(\pi/15)$. The training data was generated from a uniformly random policy.

The dynamics were parameterized by three quantities: mass of the cart ($m_c$), mass of the pole ($m_p$), and the length of the pole ($l$) [20]. For the true dynamics, we chose $m_c = m_p = l = 1$ and introduced downward wind on the system which applied a force, $f$, in the direction of gravity (*i.e.,* increasing force as $\theta$ moves away from zero) causing $\theta$ to be displaced. We also added to zero mean noise on $\psi$ with standard deviation of $0.01$.

Similarly to the experiment described in V-A, the purpose of this experiment is to understand how the approaches' performance changes as the model class becomes increasingly misspecified. For ML and RBMS, we used the standard cart-pole model class, parameterized by $m_c$, $m_p$, and $l$ and represented the value function using of a $50 \times 30 \times 30$ grid over $\psi$, $\dot{x}$, and $\dot{\psi}$.

Figure 4(a) shows the results of ML (blue) and RBMS (yellow) as their standard cart-pole dynamics model class becomes increasingly misspecified (with increasing $f$) for 10,000 episodes of training data. For reference, the return of the true model is shown in red. The figure shows that the ML approach has difficulty coping with the model misspecification and for $f \geq 0.3$ the approach never learns a stabilizing policy. In contrast, RBMS is able to learn models resulting in policies that stabilized the pole for a much larger range of $f$. Eventually, the influence of the wind is too significant for any policy to stabilize the pole, as shown by the red line eventually dropping to near zero.




(a) Performance vs Misspecification (b) Performance vs Data Size

**Figure 4**. Performance versus misspecification (a) and performance versus data size for $f = 0.2$ (b) for RBMS with the standard cart-pole model (yellow), ML with the standard cart-pole model (blue), ML with the large tabular Markov model (green), and the true model (red). The error bars show one standard error.

Figure 4(b) shows the results of ML (blue), RBMS (yellow), the large tabular Markov model (green) as the amount of training data increases for a fixed $f = 0.2$. Again, for reference, the return of the true model is shown in red. While it is not shown in the figure, the large tabular Markov model does eventually achieve a level of performance equal to that of the true model after approximately 50,000 episodes. This further illustrates that less data is required to use a small model, as opposed to a large tabular Markov model with many parameters. Our approach, on the other hand, achieves a high level of return after 500 episodes, two orders of magnitude reduction in the amount of training data compared to the ML Tabular, but may never achieve performance equal to that of the true model due to the limited representational power of the misspecified model.

### C. Hydrodynamic Cart-pole

The hydrodynamic cart-pole, shown in Figure 1, is a real-world, experimental, fluid analog of the cart-pole system. It is composed of a thin flat plate (the pole) attached at its trailing edge via a pin joint to a linear actuator (the cart) that is operated at 50 Hz. The flat plate (6.5 cm wide by 20 cm tall) is submerged in a channel (22 cm wide) of flowing water, with the flat plate placed at the "upright position." In this position, the water attempts to turn the wing downstream via a "weather vane" effect, making the upright position passively unstable and the downright position passively stable.[8] The system was operated at a Reynolds number of 15,000, considered to be an intermediate flow. As opposed to large and small Reynolds number flows, intermediate flows are particularly challenging due to the difficulty of modeling and simulating them.

Learning a policy which stabilizes the pole at the upright is difficult because the coupling between the motion of the cart and the motion of the pole is dictated by the complex dynamics of the fluid, which itself has many states and is highly nonlinear. Additionally, the flywheel mounted to the pole adds extra inertia to the system and there is a free surface which can add secondary effects when the pole moves violently through the water. Success on this system

[8]Due to fluid effects, the downright position is not precisely stable as vortex shedding from the wing will cause it to oscillate slightly, but these are small deviations compared to those considered in the experiments in this paper.

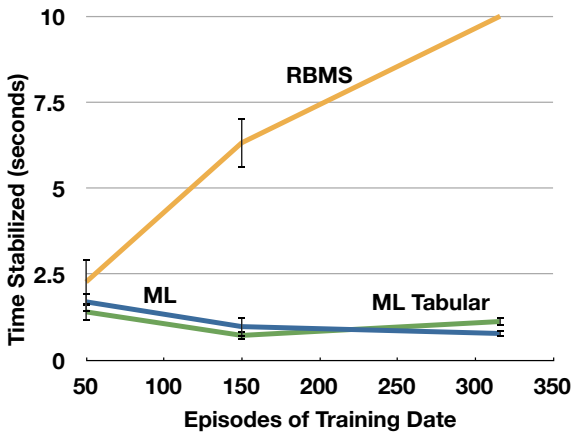



**Figure 5**. Time stabilized versus number of episodes for RBMS with the standard cart-pole model (yellow), ML with the standard cart-pole model (blue), and ML with a large tabular Markov model (green) on hydrodynamic cart-pole. The error bars represent one standard error.

will demonstrate the robustness of the technique to the many uncertainties inevitably present in actual hardware, and particularly present in fluid systems.

Our goal is to learn a controller from training data that stabilizes the pole similar to the experiment performed in Section V-B. We collected a data set of 316 episodes (approximately 45 minutes of data) from the hydrodynamic cart-pole system from a variety of hand-designed, poor controllers, none of which stabilized the pole for more than a few seconds and many of them quickly fell over. For the system, an episode was ended when the pole fell over (defined as $|\psi| > {}^{\pi}/_{15}$) or a maximum length of 500 steps (10 seconds) was reached. For our model class we used the standard cart-pole system, parameterized by $m_c$, $m_p$, $l$ as a model class of this far more complex system. To represent the value function, we discretized the state space $[x, \psi, \dot{x}, \dot{\psi}]$ into $7\times11\times11\times11$ bins with an action space $A = \{-1, -0.9, -0.8, ..., 1\}$. We used a quadratic cost function and used the distance function

$$\Delta\left((s,a)),(s',a')\right) = \frac{|x-x'|}{x_{max}-x_{min}} + \frac{|\psi-\psi'|}{\psi_{max}-\psi_{min}} + \frac{|\dot{x}-\dot{x}'|}{\dot{x}_{max}-\dot{x}_{min}} + \frac{|\dot{\psi}-\dot{\psi}'|}{\dot{\psi}_{max}-\dot{\psi}_{min}} + \frac{|a-a'|}{a_{max}-a_{min}}.$$

Figure 5 shows the performance on the real system of the learned controllers for the standard cart-pole model fit using ML (blue), the standard cart-pole model fit using RBMS (yellow), and a large tabular Markov model fit using ML (green), where each controller was run 10 times and one standard error is shown on the figure. The results show that the 45 minutes of training data was sufficient to learn a controller which stabilized the system for the maximum amount of time in all 10 trials from the small representation. Neither ML with the misspecified model nor ML with the large Markov model were able to achieve any statistically significant improvement in performance. This experiment demonstrates that despite the misspecified representation, RBMS was still able to use limited off-policy data to perform well on an extremely complex system, because it focused on finding the model that achieves the highest performance rather than minimizing the prediction error.

## VI. Related Work

The approaches most closely related to our work are Policy search (PS) techniques, which search the policy space for high performing policies using gradient techniques [21], [22]. In order to estimate the gradient, some methods obtain new samples for the new policy in order to avoid the learning bias, which can translate into high sample complexity. Existing off-policy PS methods [23] use importance sampling to reuse the data while attempting to avoid learning bias. In general, such methods are limited to discrete domains and stochastic policies. Additionally, sample efficient PS methods generally require data from high performing policies [7]. We suspect that applying RBMS in a PS setting, where the policy is directly parameterized by $\theta$, would be successful in cases where we, as designers, struggle to construct a model class that contains high performing policies learnable from limited data. In these cases we may prefer to directly parameterize the policy instead of using a dynamics model. We plan to investigate this approach and compare it to the model-based approach described here in future work.

Model Based (MB) methods aim to capture the unknown dynamics using a representation of the dynamics model. While expressive representations [24], [25] can capture nearly any type of world dynamics, they are still vulnerable to choosing models which perform arbitrarily poorly, especially from limited data due to the bias introduced by the learner. Compact representations may eliminate the sample complexity problem, yet once combined with classical learning methods (*e.g.*, maximum likelihood), they incur substantial learning bias, as shown in our empirical results. We hypothesize that we can remedy this shortcoming by applying RBMS to Bayesian nonparametric models and plan to explore that direction in future work.

Model-free Value Based (VB) techniques sidestep learning an explicit world model and directly compute a value function, although [26] showed that MB and VB RL methods using linear representations are equivalent. In the online setting, VB methods *e.g.*, [27] eliminate the learning bias by using on-policy data, yet are sample inefficient. For real-world problem, batch VB techniques [28] have been shown to be sample efficient, yet they are sensitive to the distribution of the training data[9]. Manually filtering the training data by a domain expert has been used to correct for the sampling distribution; for example, in a bicycle domain, episodes were trimmed after a certain length to expose the agent to a higher proportion of the data from the bicycle balancing, which is more likely to be seen under good policies [29]. When the representation containing the true value function is unknown, batch VB methods cannot guarantee that the highest performing value function is chosen from the representation, and generally only show convergence [10]. We believe we can overcome this limitation by using the parameterization

[9]For comparison note that the policies learned by the large tabular Markov model are equivalent to the policies LSPI [28] would learn if given the tabular representation for its value function.

$V(s;\theta)$ for RBMS to find the highest performing value function in the VB setting.

While there has been relatively little work on overcoming learning bias, there has been a great deal of work in reducing representational bias by growing the representation using nonparameteric dynamics models [30], [31], Bayesian nonparametric dynamics models [25], nonparametric value functions [32], [2], and kernel-based methods [33], [34]. The work specifically focused on reducing misspecification error generally has relied on strong assumptions about the true model [35], [36]. [9] states that we must "*pick and tailor learning methods to work with imperfect representations*," and specifically highlights meta-learning [37], and combining value-based RL and policy search [21], [22] as work headed in this direction. Despite these previous algorithms being a significant step toward the goal of reducing learning bias, for a given problem and representation, it is extremely difficult to know which method will provide the desired reduction in learning bias without implementing a variety of methods.

## VII. Conclusion

In this paper we showed that the standard RL approach of fitting misspecified representations by minimizing a form of prediction error (*e.g.*, maximum likelihood) does not necessarily result in maximizing return. We presented Reward Based Model Search (RBMS) as an algorithm for learning misspecified models in RL. RBMS has zero learning bias with infinitely dense data and unlimited computation. With limited data and computation, RBMS can be seen as an improvement on the typical minimum error technique, which provably, in expectation, does not decrease performance. We empirically demonstrated that RBMS results in a large performance increase over the maximum likelihood on two common RL benchmark problems and a third real-world system, the hydrodynamic cart-pole, with extremely complex dynamics that cannot be known exactly.

## Acknowledgements

This research was sponsored by Behzad Kamgar-Parsi and ONR under MURI N00014-11-1-0688 and we gratefully acknowledge their support. We also thank Russ Tedrake and the Robot Locomotion Group for their use of the hydrodynamic cart pole.

## References

[1] L. P. Kaelbling, M. L. Littman, and A. P. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*.

[2] A. Geramifard, F. Doshi, J. Redding, N. Roy, and J. How, "Online discovery of feature dependencies," in *ICML*, 2011.

[3] P. Diaconis, "Theories of data analysis: From magical thinking through classical statistics," 1985.

[4] S. Kalyanakrishnan, P. Stone, and Y. Liu, "Model-based reinforcement learning in a complex domain," in *Robot Soccer World Cup XI*, 2008.

[5] R. S. Sutton, C. Szepesvari, A. Geramifard, and M. Bowling, "Dyna-style planning with linear function approximation and prioritized sweeping," in *UAI*, 2008.

[6] H. Yao, R. Sutton, S. Bhatnagar, D. Diao, and C. Szepesvari, "Multi-step dyna planning for policy evaluation and control," in *NIPS*, 2009.

[7] J. Kober, E. Oztop, and J. Peters, "Reinforcement learning to adjust robot movements to new situations," in *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[8] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," 2007.

[9] S. Kalyanakrishnan and P. Stone, "On learning with imperfect representations," in *Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2011.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, May 1998.

[11] A. Barto and M. Duff, "Monte carlo matrix inversion and reinforcement learning," in *NIPS*, 1994.

[12] L. Peshkin, "Reinforcement learning by policy search," 2000.

[13] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Machine Learning*, 1992.

[14] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems 12*, 2000.

[15] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research*, 2001.

[16] R. Fonteneau, S. A. Murphy, L. Wehenkel, and D. Ernst, "Model-free monte carlo-like policy evaluation," *Journal of Machine Learning Research - Proceedings Track*, vol. 9, pp. 217–224, 2010.

[17] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[18] S. Singh, R. S. Sutton, and P. Kaelbling, "Reinforcement learning with replacing eligibility traces," in *Machine Learning*, 1996, pp. 123–158.

[19] M. W. Spong, "Underactuated mechanical systems," in *Control Problems in Robotics and Automation*. Springer-Verlag, 1998.

[20] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, 1983.

[21] L. Baird and A. Moore, "Gradient descent for general reinforcement learning," in *Proceedings of the 1998 conference on Advances in neural information processing systems II*. MIT Press, 1999.

[22] C. Guestrin, M. G. Lagoudakis, and R. Parr, "Coordinated reinforcement learning," in *Proceedings of the Nineteenth International Conference on Machine Learning*, ser. ICML '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 227–234.

[23] N. Meuleau, L. Peshkin, L. Kaelbling, and K. Kim, "Off-policy policy search," MIT Articical Intelligence Laboratory, Tech. Rep., 2000.

[24] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A Model-Based and Data-Efficient Approach to Policy Search," in *Proceedings of the 28th International Conference on Machine Learning*, L. Getoor and T. Scheffer, Eds., Bellevue, WA, USA, June 2011.

[25] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy, "A Bayesian Nonparametric Approach to Modeling Motion Patterns," *Autonomous Robots*, vol. 31, no. 4, pp. 383–400, 2011.

[26] R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. L. Littman, "An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning," in *ICML*, 2008.

[27] G. Rummery and M. Niranjan, "Online Q-learning using connectionist systems," *Cambridge University Engineering Department*, 1994.

[28] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning*, vol. 4, pp. 1107–1149, 2003.

[29] M. Petrik, G. Taylor, R. Parr, and S. Zilberstein, "Feature selection using regularization in approximate linear programs for Markov decision processes," in *ICML*, 2010.

[30] D. A. Vasquez Govea, T. Fraichard, and C. Laugier, "Growing Hidden Markov Models: An Incremental Tool for Learning and Predicting Human and Vehicle Motion," *IJRR*, 2009.

[31] K. Ure, A. Geramifard, G. Chowdhary, and J. How, "Adaptive Planning for Markov Decision Processes with Uncertain Transition Models via Incremental Feature Dependency Discovery," in *ECML*, 2012.

[32] S. Whiteson, M. Taylor, and P. Stone, "Adaptive tile coding for value function approximation," University of Texas at Austin, Tech. Rep.

[33] D. Ormoneit and P. Glynn, "Kernel-based reinforcement learning in average-cost problems: An application to optimal portfolio choice," in *Advances in Neural Information Processing Systems*, 2000.

[34] A. S. Barreto, D. Precup, and J. Pineau, "Reinforcement learning using kernel-based stochastic factorization," in *NIPS*, 2011.

[35] H. White, "Maximum likelihood estimation of misspecified models," *Econometrica*, vol. 50, no. 1, pp. 1–25, Jan. 1982.

[36] M. Sugiyama, "Active learning for misspecified models," in *Advances in Neural Information Processing Systems 18*, 2006.

[37] R. Vilalta and Y. Drissi, "A perspective view and survey of Meta-Learning," *Artificial Intelligence Review*, vol. 18, pp. 77–95, 2002.