Control-Oriented Model-Based Reinforcement Learning with Implicit Differentiation

Evgenii Nikishin,[1] **Romina Abachi,**[2] **Rishabh Agarwal,**[1 3] **Pierre-Luc Bacon**[1 4]

[1]Mila, Université de Montréal, [2]Vector Institute, University of Toronto
[3]Google Research, Brain Team, [4]Facebook CIFAR AI Chair
evgenii.nikishin@mila.quebec

Abstract

The shortcomings of maximum likelihood estimation in the context of model-based reinforcement learning have been highlighted by an increasing number of papers. When the model class is misspecified or has a limited representational capacity, model parameters with high likelihood might not necessarily result in high performance of the agent on a downstream control task. To alleviate this problem, we propose an end-to-end approach for model learning which directly optimizes the expected returns using implicit differentiation. We treat a value function that satisfies the Bellman optimality operator induced by the model as an implicit function of model parameters and show how to differentiate the function. We provide theoretical and empirical evidence highlighting the benefits of our approach in the model misspecification regime compared to likelihood-based methods.

1 Introduction

The conceptual separation between model learning and policy optimization is the basis for much of the work on model-based reinforcement learning (MBRL) (Sutton 1991; Boots, Siddiqi, and Gordon 2011). A standard MBRL agent first estimates the transition parameters and the reward function of a Markov Decision Process and then uses the approximate model for planning (Theil 1957; Sato, Abe, and Takeda 1988). If the estimated model perfectly captures the actual system, the resulting policies are not affected by the model approximation error. However, if the model is imperfect, the inaccuracies can lead to nuanced effects on the policy performance (Abbad and Filar 1992). Several works (Skelton 1989; Joseph et al. 2013; Lambert et al. 2020) have pointed out on *the objective mismatch* in MBRL and demonstrated that optimization of model likelihood might be unrelated to optimization of the returns achieved by the agent that uses the model. For example, accurately predicting individual pixels of the next state (Kaiser et al. 2019) might be neither easy nor necessary for decision making. Motivated by these observations, our paper studies control-oriented model learning that takes into account how the model is used by the agent.

While much of such work has focused on robust or uncertainty-based methods (Xu and Mannor 2010; Chua et al. 2018), we propose an algorithm for learning a model that directly optimizes the expected return using implicit differentiation (Christianson 1994). Specifically, we assume that there exists *an implicit function* that takes the model as input and outputs a value function that is a fixed point of the Bellman optimality operator induced by the model. We then calculate the derivatives of the optimal value function with respect to the model parameters using the implicit function theorem (IFT), allowing us to form a differentiable computational graph from model parameters to the sum of rewards. In reference to (Rust 1988; Bacon et al. 2019), we call our control-oriented method *optimal model design* (OMD).

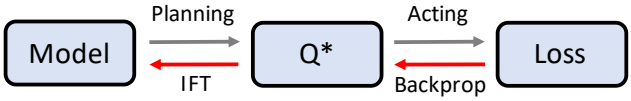

Figure 1: Illustration of the *Optimal Model Design* approach: we treat the optimal Q-function as an implicit function of the model and calculate the gradient with respect to the model parameters via the implicit function theorem as described in Section 4.1.

Our contributions can be summarized as follows:

- We propose OMD, an end-to-end MBRL method that optimizes expected returns directly.
- We characterize the set of OMD models in the tabular case and derive an approximation bound on the optimal Q-function that is tighter than a likelihood-based one.
- We propose a series of approximations to scale our approach to non-tabular environments.
- We show that OMD outperforms likelihood-based MBRL agents under the model misspecification in both tabular and non-tabular settings. This finding suggests that our method should be preferred when we cannot approximate the true model accurately.
- We empirically demonstrate that models obtained by OMD can have lower likelihood than a random model yet generate useful targets for updating the value function. This finding suggests that likelihood optimization might be an unnecessary step for MBRL.

2 Related work

Learning control-oriented models. Earlier work in optimal control and econometrics (Skelton 1989; Rust 1988) studied the relation between the model approximation error and the control performance and noted that true parameter identification could be suboptimal when the model class is limited. Joseph et al. (2013) were one of the first to address the objective mismatch (Lambert et al. 2020) and proposed an algorithm for training models that maximize the expected returns using zero-order optimization.

Several papers have proposed model learning approaches that optimize other return-aware objectives. Farahmand, Barreto, and Nikovski (2017) train a model to minimize the difference between values of the real next states and the next states predicted by the dynamics. Abachi, Ghavamzadeh, and Farahmand (2020) use the norm of the difference between policy gradients as the model objective. D'Oro et al. (2020) use a weighted maximum likelihood objective where the weights are chosen to minimize the difference between the true policy gradient and the policy gradient in the MDP induced by the model. Schrittwieser et al. (2019) use tree search and train models for image-based states by encoding them into a latent space and predicting rewards, a policy, and values without reconstruction.

The idea of differentiable planning has also been investigated. Amos et al. (2018) learn a model via differentiating the KKT conditions in the LQR setting (Dorato, Cerone, and Abdallah 1994). Tamar et al. (2016) uses a differentiable approximation of the value iteration algorithm to learn a planner. Amos and Yarats (2020) optimize the parameters of a distribution in Cross-Entropy Method (Rubinstein 1997) using a differentiable approximation of Top-K operation.

Several works have theoretically studied the control-oriented model learning. Ayoub et al. (2020) derive regret bounds for models used to predict values. Grimm et al. (2020) introduce the principle of value equivalence for MBRL defining two models to be equivalent if they induce the same Bellman operator.

Our work is closely related to the above papers and proposes to learn models by directly optimizing the sum of rewards in an end-to-end manner via gradient-based methods.

Implicit function theorem. Implicit differentiation has been applied for a variety of bi-level optimization problems. Lorraine, Vicol, and Duvenaud (2020) treat weights of a neural network as an implicit function of hyperparameters and use IFT to optimize the hyperparameters. Rajeswaran et al. (2019) study meta-learning and apply IFT to compute the outer loop gradient without the need to differentiate through the inner loop iterations. Instead of treating a neural network as a sequence of layers that transform an input, Bai, Kolter, and Koltun (2019) propose an implicit layer that corresponds to an infinite depth neural network and find a fixed point of the layer via the IFT. Our method also solves a bi-level problem: in the inner loop, we train an action-value function compatible with the model; in the outer loop, we maximize the expected returns with respect to the model.

3 Preliminaries

Reinforcement Learning (RL) (Sutton and Barto 2018) methods follow the Markov Decision Process (MDP) formalism. An MDP is defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \gamma, p, r, \rho_0)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $p(s'|s,a)$ is a transition probability distribution (often called *dynamics*), $r(s,a)$ is a reward function, $\gamma \in [0,1)$ is a discount factor, and $\rho_0(s)$ is an initial state distribution. The pair (p, r) is jointly called *the true model*. The goal of an agent is to learn a policy $\pi(a|s)$ that maximizes the expected discounted sum of rewards $J(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$. The performance of the agent following the policy π can also be quantified using the action value function $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a\right]$.

Model-based RL methods typically train a model (p_θ, r_θ) and use it for policy or value learning. Traditional methods based on Dyna (Sutton 1991) rely on maximum likelihood estimation (MLE) of model parameters θ. For example, if the true model is assumed to be Gaussian with a parameterized mean (f_θ, r_θ) and a fixed variance, maximizing the likelihood is equivalent to minimizing the mean squared error of the prediction, namely, to solving

$$\begin{aligned} &\min_\theta \mathbb{E}_{s,a,s'}\left[\|f_\theta(s,a) - s'\|^2\right], \\ &\min_\theta \mathbb{E}_{s,a,r}\left[(r_\theta(s,a) - r)^2\right]. \end{aligned} \tag{1}$$

4 Optimal Model Design for Tabular MDPs

Consider a modification of the original RL problem statement, which was first proposed by Rust (1988) and revisited by Bacon et al. (2019). In addition to maximizing the expected returns J, we introduce a constraint forcing the action value function Q to satisfy the Bellman equation induced by the model. The optimization problem becomes

$$\begin{aligned} &\max_{Q,\theta} J(\pi_Q) \\ &\quad \text{s.t. } Q(s,a) = B^\theta Q(s,a) \ \ \forall s \in \mathcal{S}, a \in \mathcal{A}, \\ &\text{where } \ \pi_Q(a|s) = \frac{\exp Q(s,a)}{\sum_{a'} \exp Q(s,a')}. \end{aligned} \tag{2}$$

B^θ here is *the soft Bellman optimality operator* with respect to the model parameters θ:

$$B^\theta Q(s,a) \triangleq r_\theta(s,a) + \gamma \mathbb{E}_{p_\theta(s'|s,a)} \log \sum_{a'} \exp Q(s',a'). \tag{3}$$

We choose the soft Bellman operator with $\log \sum_{a'} \exp Q(s',a')$ over the "hard" version with $\max_{a'} Q(s',a')$ because of the differentiability of log-sum-exp. We also use a temperature α in softmax and log-sum-exp but omit it from the expressions for simplicity. Note that finding a fixed point of the soft Bellman optimality operator corresponds to solving the MaxEnt RL formulation (Ziebart et al. 2008), but for a sufficiently small value of α, the difference is negligible.[1]

[1] More details about the soft Bellman operator and MaxEnt RL could be found in (Levine 2018).

Suppose there exists an *implicit* function $\varphi(\theta) = Q^*$ that takes as input a model and outputs a Q-function that satisfies the constraint in (2). The sequence of transformations from the model parameters to the agent's performance can be described then using the following graph:

$$\theta \xrightarrow{\varphi} Q^* \xrightarrow{\text{exp}} \pi_{Q^*} \xrightarrow{\text{act}} J. \tag{4}$$

In Section 4.1, we show how $\frac{\partial \varphi(\theta)}{\partial \theta}$ can be calculated using the implicit function theorem (IFT). Since $\frac{\partial J(\pi)}{\partial \pi}$ can be calculated using the policy gradient theorem (Sutton et al. 1999), we can apply automatic differentiation to calculate the gradient with respect to θ:

$$\frac{\partial J(\theta)}{\partial \theta} = \underbrace{\frac{\partial J(\pi)}{\partial \pi}}_{\text{PG}} \cdot \underbrace{\frac{\partial \pi(Q^*)}{\partial Q^*}}_{\text{softmax}} \cdot \underbrace{\frac{\partial \varphi(\theta)}{\partial \theta}}_{\text{IFT}}. \tag{5}$$

Given the expression for the gradient of J with respect to θ, we use an appropriate optimization method to train the model. We call the approach *optimal model design* (OMD). Note that Dyna-based methods also train the Q-function to satisfy the constraint in (2) while using the likelihood as the objective for the model (Rajeswaran, Mordatch, and Kumar 2020). In contrast, we train θ to *directly optimize* the returns.

The optimization problem (2) suggests that OMD is a policy-based method (Sutton et al. 1999). However, we can make it a value-based approach (Watkins and Dayan 1992) by replacing the objective $J(\pi_Q)$ with the Bellman error:

$$\begin{aligned} &\min_{Q,\theta} L^{\text{true}}(Q) \triangleq \sum_{s,a} \left(Q(s,a) - BQ(s,a)\right)^2, \\ &\text{s.t. } Q(s,a) = B^\theta Q(s,a) \;\; \forall s \in \mathcal{S}, a \in \mathcal{A}, \end{aligned} \tag{6}$$

where B, similarly to B^θ, is the soft Bellman operator but induced by *the true reward r and dynamics p*. We discuss the relation between the models obtained by solving problems (2) and (6) in Section 5.1.

While the constraint $Q(s,a) = B^\theta Q(s,a)$ has to be satisfied for all state-action pairs limiting the approach to tabular MDPs, we show an extention to the function approximation case in Section 6.

4.1 Implicit Differentiation

In this subsection, we state the implicit function theorem used to calculate $\frac{\partial \varphi(\theta)}{\partial \theta}$. The IFT is a well-known result and a proof can be found, for example, in (Krantz and Parks 2012).

Theorem 1. (Cauchy, Implicit Function) *Let $f : \Theta \times \mathcal{W} \to \mathcal{W}$ be a continuously differentiable function and $(\tilde{\theta}, \tilde{w})$ be a point satisfying $f(\tilde{\theta}, \tilde{w}) = \mathbf{0}$. If the Jacobian $\frac{\partial f(\tilde{\theta}, \tilde{w})}{\partial w}$ is invertible, then there exists an open set $U \subseteq \Theta$ containing $\tilde{\theta}$ and a unique continuously differentiable function φ such that $\varphi(\tilde{\theta}) = \tilde{w}$ and $f(\theta, \varphi(\theta)) = \mathbf{0}$ for all $\theta \in U$. Moreover,*

$$\frac{\partial \varphi(\theta)}{\partial \theta} = -\left(\frac{\partial f(\theta, w^*)}{\partial w}\right)^{-1} \cdot \frac{\partial f(\theta, w^*)}{\partial \theta}\Big|_{w^* = \varphi(\theta)}. \tag{7}$$

Note that (7) requires only a final point w^* satisfying the constraint and does not require knowledge about φ itself. Hence, φ can be any black-box function outputting w^*.

The gradient of the scalar objective J or L^{true} is calculated using (7). To use backpropagation, we only need to define the product of a vector and $\frac{\partial \varphi(\theta)}{\partial \theta}$. Overall, the IFT allows using φ as a block in a differentiable computational graph.

4.2 Benefits under Model Misspecification

In the previous subsection, we showed how to use implicit differentiation for training a model that aims to maximize the expected returns. In this subsection, we demonstrate that such a control-oriented model is preferable over a likelihood-based in the setting where the true model is not representable by a chosen parametric class.

Let $r_\theta \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ and $p_\theta(s'|s,a) \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|\times|\mathcal{A}|}$ be a parametric model, where parameters in p_θ denote the corresponding logits and each parameter in r_θ is a reward for a state-action pair. We consider a set of parameters $\{\theta : \|\theta\| \leq \kappa\}$ with the *bounded norm* and use κ as a measure of the model misspecification. By decreasing the bound of the norm of θ, we get a more misspecified model class. To isolate the model learning aspect, we consider the exact RL setting without sampling. We take a 2 state, 2 action MDP shown in Figure 3 with a discount factor $\gamma = 0.9$ and a uniform initial distribution ρ_0. For every θ, a function φ outputs the corresponding Q^* via performing the fixed point iteration until convergence. Q^* is transformed into the policy π_{Q^*} via softmax with the temperature $\alpha = 0.01$. We then calculate $J(\pi_{Q^*})$ in a closed form (Sutton and Barto 2018).

For OMD, we obtain the gradient of J with respect to θ using the expression (5). We then apply the projected gradient ascent where after each step we make a projection on a space of bounded parameters via

$$\theta = \begin{cases} \frac{\kappa}{\|\theta\|}\theta & \text{if } \|\theta\| > \kappa, \\ \theta & \text{if } \|\theta\| \leq \kappa \end{cases}. \tag{8}$$

Finding an MLE solution corresponds to minimizing the average KL divergence

$$\overline{\text{D}_{\text{KL}}}(p||p_\theta) = \frac{1}{|\mathcal{S}| \cdot |\mathcal{A}|} \sum_{s,a,s'} p(s'|s,a) \log \frac{p(s'|s,a)}{p_\theta(s'|s,a)}$$

for optimizing the dynamics p_θ and minimizing the squared error for the reward r_θ. We similarly perform the projected gradient descent and call the agent MLE (even though we do not use the samples for estimation).

The resulting J as a function of the norm bound κ is shown in Figure 2. When the true model is not representable by a chosen class, OMD learns a model that uses its representational capacity for helping the agent to maximize the expected returns, while the MLE agent tries to predict the next states and rewards accurately while discarding the true objective function the agent seeks to optimize.

In MDPs with high-dimensional state spaces (Bellemare et al. 2013; Beattie et al. 2016) where the underlying dynamics are complex, having a model that will accurately predict the next observation might be expensive and unnecessary for

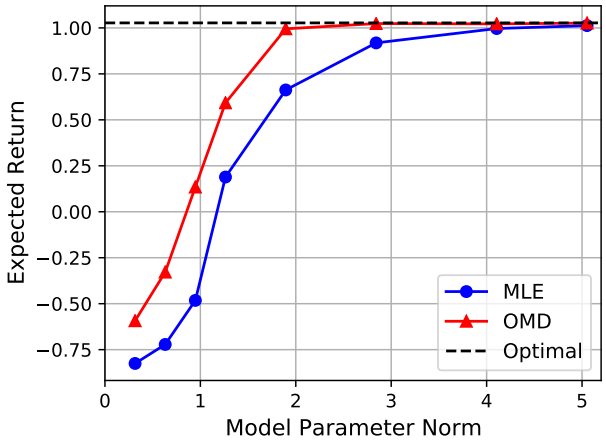

Figure 2: Expected returns for the tabular MDP under the model misspecification. The OMD model optimizes the expected returns directly, while the MLE agent minimizes the KL divergence for model learning. OMD outperforms MLE when the model representational capacity is limited.

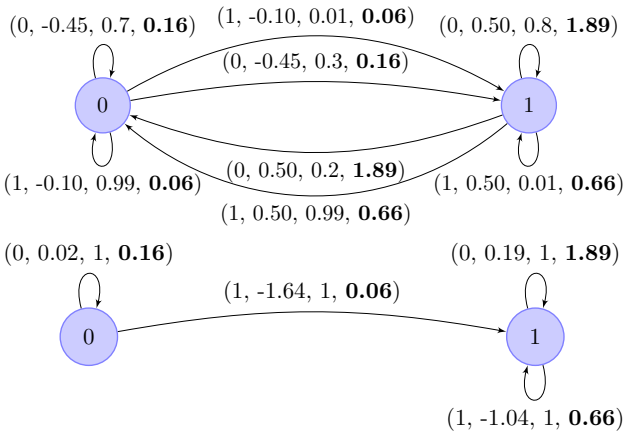

Figure 3: Two different MDPs with the same optimal Q-function (a fixed point of the induced Bellman operator). Circles represent states, tuples are organized as (action, reward, transition probability, optimal Q value). Top: the original MDP taken from (Dadashi et al. 2019). Bottom: an MDP with a trained OMD model.

decision making. Figure 2 reflects the problems an MLE-based model will face for such environments and provides evidence for using control-oriented models that leverage the available capacity of the model effectively.

5 Theoretical Analysis

In the previous section, we have empirically demonstrated that OMD outperforms Dyna-style (Sutton 1991) MBRL agents when the model capacity is limited. This section characterizes the set of optimal solutions of OMD and compares the Q^* approximation bounds for OMD and MLE agents.

5.1 Optimal Solutions for OMD

We use the principle of value equivalence for MBRL (Grimm et al. 2020) and argue that value equivalent models are optimal solutions to (2) and (6).

Definition 1 (Optimal value equivalence)**.** *Let Q^* be an optimal action-value function for the unconstrained RL problem. Models with parameters θ and θ' are Q^*-equivalent if*

$$B^{\theta}Q^*(s,a) = B^{\theta'}Q^*(s,a) \;\; \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{9}$$

The definition is a slight modification of the value equivalence used in (Grimm et al. 2020): instead of requiring the Bellman operators to be equal for a set of value functions and policies, we require the equality for a chosen Q^* only. The subset of models that are Q^*-equivalent forms an *equivalence class* Θ_{Q^*}.

Proposition 1. *Let the soft Bellman operator* (3) *temperature $\alpha \to 0$ and let θ be* any *model parameters from the equivalence class Θ_{Q^*}. Then, (Q^*, θ) is a solution for* (2) *and* (6)*.*

This property holds by construction. The optimal Q-function maximizes the objective in the true MDP. As the log-sum-exp temperature in (3) approaches 0, we recover the "hard" target in the Bellman optimality operator:

$$\lim_{\alpha \to 0} \alpha \log \sum_{a'} \exp \frac{1}{\alpha} Q(s', a') = \max_{a'} Q(s', a'). \tag{10}$$

Thus, if we set θ to the true model, Q^* will satisfy the Bellman equation $Q^*(s,a) = B^{\theta}Q^*(s,a)$. But even though the true model belongs to the equivalence class Θ_{Q^*}, it is *not identifiable*: all models from Θ_{Q^*} are going to be indistinguishable for OMD. Seemingly undesirable at first glance, it allows OMD choosing *any* model that induces the same Bellman operator, which is beneficial under the model misspecification as shown in Section 4.2.

We provide an example of a model that is Q^*-equivalent with the true model in Figure 3. The model differs significantly, demonstrating that the equivalence class Θ_{Q^*} consists of multiple elements. Moreover, the dynamics learned by OMD are deterministic, suggesting that OMD can choose *a simpler* model that will have the same Q^* as the true model. Drawing the connection to the prior work on state abstractions (Li, Walsh, and Littman 2006), the fact that MDPs have the same optimal action values indicates that the learned models can be seen as Q^*-irrelevant with respect to a state abstraction over $\mathcal{S}$.

5.2 Approximation Bound

Our next result relates approximation errors for the optimal Q-functions under the OMD and MLE models. For simplicity, we analyze the setting with $\alpha \to 0$ and the Bellman error (6) as the objective.

Theorem 2. (Q^* approximation error) *Let Q^* be the optimal action-value function for the true MDP. Let $\hat{Q}_{\text{OMD}}$ and $\hat{Q}_{\text{MLE}}$ be the fixed points of the Bellman optimality operators for approximate OMD and MLE models respectively.*

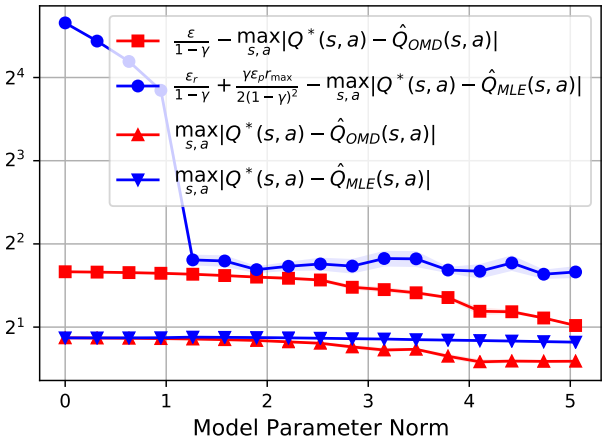

Figure 4: Q^* approximation error and tightness of the error bounds under the model misspecification. Given a limited model representation capacity, OMD agent approximates Q^* more accurately and enjoys a tighter bound.

- *If the MLE dynamics $\hat{p}$ and reward $\hat{r}$ have the bounded errors $\max_{s,a} \|p(\cdot|s,a) - \hat{p}(\cdot|s,a)\|_1 = \epsilon_p$ and $\max_{s,a} |r(s,a) - \hat{r}(s,a)| = \epsilon_r$, and the reward function is bounded $r(s,a) \in [0, r_{\max}] \quad \forall s,a$, we have*

$$\max_{s,a} \left| Q^*(s,a) - \hat{Q}_{\text{MLE}}(s,a) \right| \leq \frac{\epsilon_r}{1-\gamma} + \frac{\gamma \epsilon_p r_{\max}}{2(1-\gamma)^2};$$

- *If the Bellman optimality operator induced by the OMD model $\hat{\theta}$ has the bounded error $\max_{s,a} \left| B^{\hat{\theta}} \hat{Q}_{\text{OMD}}(s,a) - B^{\hat{\theta}} \hat{Q}_{\text{OMD}}(s,a) \right| = \epsilon$, we have*

$$\max_{s,a} \left| Q^*(s,a) - \hat{Q}_{\text{OMD}}(s,a) \right| \leq \frac{\epsilon}{1-\gamma}.$$

The proof mostly follows derivations similar to the simulation lemma (Kearns and Singh 2002). Since OMD directly optimizes ϵ, while MLE optimizes ϵ_r and ϵ_p, the bound suggests that OMD approximation error translates into a lower Q^* approximation error. Figure 4 compares empirically the errors and the tightness of the bounds for a tabular MDP where Q^* can be computed exactly. The result provides evidence that OMD indeed achieves a lower Q^* approximation error compared to an agent that seeks to estimate p and r accurately. Motivated by the theoretical findings, the next section discusses a practical version of OMD.

6 OMD with Function Approximation

Section 4 describes optimal model design, a non-likelihood-based method for learning models in tabular MDPs. In this section, we propose several approximations to make OMD practically applicable.

Q-network. We use a neural network with parameters w to approximate the Q-values. The network is trained to minimize the Bellman error induced by the model θ:

$$\min_w L(\theta, w) \triangleq \min_w \mathbb{E}_{s,a}[Q_w(s,a) - B^\theta Q_{\bar{w}}(s,a)]^2, \quad (11)$$

Algorithm 1: Model Based RL with OMD

Input: Initial parameters w, θ, empty replay buffer $\mathcal{D}$.
repeat
  Set s to be the current state.
  Sample an action a using softmax over $Q_w(s,a)$.
  Apply a to get $r = r(s,a), s' \sim p(s'|s,a)$.
  Append (s,a,s',r) to buffer $\mathcal{D}$.
  for $i = 1$ **to** K **do**
    Sample (s,a) from buffer $\mathcal{D}$.
    Apply θ to get $r = r_\theta(s,a), s' \sim p_\theta(s'|s,a)$.
    Update Q_w parameters w to minimize $L(\theta, w)$.
  end for
  Update model parameters θ according to (14).
until the maximum number of interactions is reached

where $\bar{w}$ is a target copy of parameters w updated using exponential moving average, a standard practice to increase the stability of deep Q-learning (Mnih et al. 2015). We also use double Q-learning (Hasselt 2010; Fujimoto, Hoof, and Meger 2018) but omit it from the equations for simplicity. To estimate the expectation, we use a replay buffer (Mnih et al. 2015).

Constraint. The constraint in (2) and (6) should be satisfied for all state-action pairs making it impractical for non-tabular MDPs. We introduce an alternative but similar constraint, the first-order optimality condition for minimizing the Bellman error (11):

$$\frac{\partial L(\theta, w)}{\partial w} = \mathbf{0}. \quad (12)$$

We note that the 0 is vector-valued and has the same dimensionality as w.

Implicit differentiation. The process of training θ is bi-level: in the inner loop, we optimize the Q-function parameters to get optimal w^* corresponding to a fixed model θ; in the outer loop, we make a gradient update of θ. We make K steps of an optimization method to approximate $w^* = \varphi(\theta)$ where K is a hyperparameter and reuse the weights from the previous outer loop iterations. We follow Rajeswaran, Mordatch, and Kumar (2020) and approximate the inverse Jacobian term in $\frac{\partial \varphi(\theta)}{\partial \theta}$ with the identity matrix. Surprisingly, we did not observe benefits when using the inverse Jacobian term. The investigation of the phenomenon is beyond the scope of this work. We refer to (Rajeswaran et al. 2019; Lorraine, Vicol, and Duvenaud 2020) that also observe limited effects from the Jacobian.

Objective. We consider the problem (6) and use the Bellman error as the outer loop objective:

$$L^{\text{true}}(w) \triangleq \mathbb{E}_{s,a}[Q_w(s,a) - BQ_{\bar{w}}(s,a)]^2, \quad (13)$$

where B, again, is a soft Bellman operator induced by the true reward r and dynamics p:

$$BQ_{\bar{w}}(s,a) \triangleq r(s,a) + \gamma \mathbb{E}_{p(s'|s,a)} \log \sum_{a'} \exp Q_{\bar{w}}(s', a').$$

Note that the objective (13) is used for estimating the gradient with respect to θ *only* and w is trained to optimize

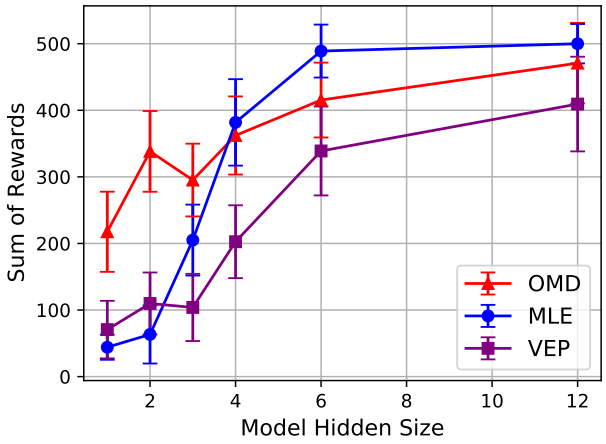

Figure 5: The final performance of the agents in CartPole for varying hidden dimensionality of the model networks. The OMD model makes useful predictions under the model misspecification. The error bar is the standard error measured over 10 random seeds.

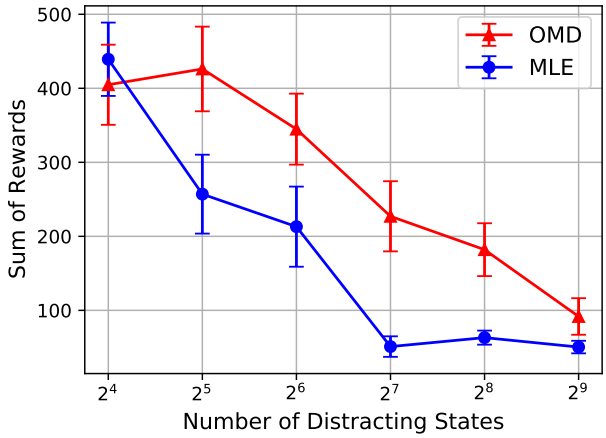

Figure 6: The performance when the state space is augmented with uninformative noise. OMD is more robust as the number of distractor increases, while VEP fails for any positive number of distractors. The standard error is measured over 10 seeds.

$L(\theta, w)$. While both L^{true} and J objectives could be used for training θ, we found that the latter requires more samples to converge. Note that optimizing the L^{true} still corresponds to maximizing the (entropy-regularized) expected returns.

Resulting gradient. The Q function and IFT approximations result in the following gradient with respect to θ:

$$\frac{\partial L^{\text{true}}(\theta)}{\partial \theta} \approx -\underbrace{\frac{\partial L^{\text{true}}(w^*)}{\partial w}}_{\text{grad Bellman}} \cdot \underbrace{\frac{\partial^2 L(\theta, w^*)}{\partial \theta \partial w}}_{\text{approx IFT}} \Big|_{w^*=\varphi(\theta)} \tag{14}$$

The OMD algorithm is summarised in Algorithm 1. The only difference between Dyna-based approaches and OMD is the model learning step highlighted in blue.

7 Experiments with Function Approximation

This section aims to test the following hypotheses:

- The OMD agent with approximations from Section 6 achieves near-optimal returns.
- The performance of OMD is better compared to MLE under the model misspecification.
- Parameters θ of the OMD model have low likelihood, yet an agent acting with the Q-function trained with the model achieves near-optimal returns in the true MDP.

Setup. We first use CartPole (Barto, Sutton, and Anderson 1983) and later include results on MuJoCo HalfCheetah (Todorov, Erez, and Tassa 2012) with similar findings further supporting our conclusions. Since OMD learns one of the Q^*-equivalent models as shown in Section 5.1, a close non-MLE baseline would be the algorithm used in the value equivalence principle (VEP) paper (Grimm et al. 2020). The VEP model minimizes the difference between the Bellman operators:

$$\ell_{\text{VEP}}(\theta) = \sum_{\pi \in \Pi} \sum_{V \in \mathcal{V}} \sum_{s \in \mathcal{S}} \left(B_\pi V(s) - B^\theta_\pi V(s)\right)^2, \tag{15}$$

where $B^\theta_\pi V(s) = \mathbb{E}_{a\sim\pi(a|s), s'\sim p_\theta(s'|s,a)} \left(r_\theta(s,a) + \gamma V(s')\right)$, B_π is the real model counterpart, and Π and $\mathcal{V}$ are predefined sets of policies and value functions.

Performance under model misspecification. We design two experiments that allow measuring the misspecification. First, we limit the model class representational capacity by controlling the number of units in hidden layers of the model. Next, we add distracting states by sampling noise from a standard gaussian and vary the number of distractors. Figure 5 and Figure 6 show the returns achieved by the agents after training in the two regimes. Note that the Q-function is updated using only the next states and rewards produced by the model, and even when the hidden dimensionality of the model is 1, the OMD model encodes useful information for taking optimal actions. Returns achieved by OMD are also more robust to the distractors indicating that the MLE focuses on predicting the parts of a state that might not be relevant for decision making. The relatively low performance of VEP suggests that learning a model to predict values for a fixed set of policies and value functions is not as effective, especially if some states are non-informative. The experiments reflect the challenges an MBRL agent will face in complex domains such as (Bellemare et al. 2013; Beattie et al. 2016; Kalashnikov et al. 2021) where accurately predicting all pixel-wise variations in next observations can be infeasible and unnecessary. Figures 5 and 6 provide evidence that using control-oriented methods would allow using the model capacity more effectively.

Likelihood of OMD model. We show the mean squared error (MSE) of OMD and MLE dynamics predictions in Table 1. The MSE of OMD models is *higher than the MSE of a randomly initialized model*, while OMD achieves higher returns than the MLE agent. This finding suggests that the dynamics might not need to produce predictions close to the true states to be useful for planning.

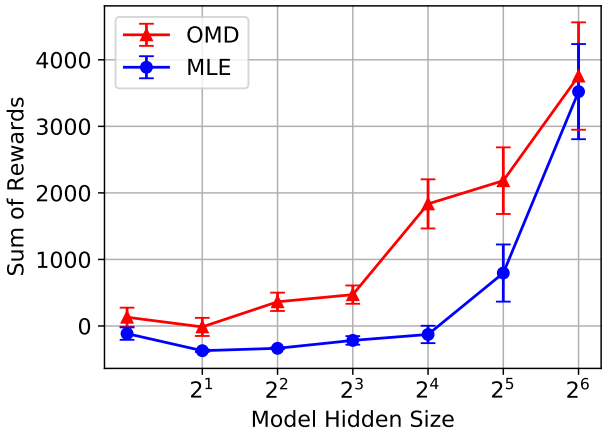

Figure 7: Returns for varying representational capacity of the OMD and MLE agents on HalfCheetah-v2. Given limited capacity, the OMD model makes more useful predictions. The std is measured over 5 runs.

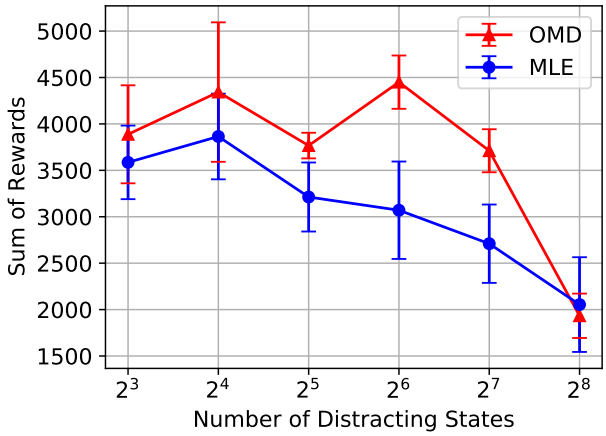

Figure 8: Returns on HalfCheetah-v2 when the state space is augmented with gaussian noise. The OMD agent is more robust to the distractors. The std is measured over 5 runs.

# hidden	1	2	3	4	6	12
MLE	0.9	0.4	10^{-2}	10^{-3}	10^{-5}	10^{-5}
OMD	8.5	5.8	4.3	17.8	11.6	5.7
Random	1.5	1.5	1.5	1.5	1.5	1.5

Table 1: MSE of the next state prediction for OMD, MLE, and random models in CartPole given varying hidden size.

Results on HalfCheetah. We also test OMD and MLE agents under the model misspecification on MuJoCo HalfCheetah. As the inner optimizer, we use an implementation (Kostrikov 2021) of Soft Actor-Critic (Haarnoja et al. 2018) with the default configuration. OMD trains the model using Equation 14. The MLE agent trains the model with MSE effectively becoming the MBPO algorithm (Janner et al. 2019) without having an ensemble of models and learning the variance of the predictions. We do not include results for VEP here because Grimm et al. (2020) test it only in tabular and CartPole-like environments.

Figures 7 and 8 summarize the results. Similarly to the observations on the tabular and CartPole environments, the experiments provide evidence that OMD should be preferred in the model misspecification setup.

Overall, our findings suggest that the OMD agent achieves near-optimal returns, performs better than the MLE-based MBRL agent as well as VEP under model misspecification, and learns a model that is useful for control despite having low likelihood.

8 Discussion and Future Work

An exciting direction for future work is the extension of OMD to environments with image-based observations where model misspecification naturally arises. We expect that for complex visual domains, learning a control-oriented model should be more effective compared to model-based methods that rely on reconstruction (Hafner et al. 2020).

Implicit differentiation is not the only way to solve constrained optimization problems. Other alternatives include using Lagrangian methods as proposed for the tabular case in (Bacon et al. 2019). Since extending the approach to non-tabular MDPs would require an additional approximator for Lagrange multipliers, we conjecture that solving a saddle point problem is going to be less stable than using the IFT.

Finally, it is worth theoretically studying the sensitivity of the IFT to the approximations to the inverse Jacobian term and the inner loop solutions. Our findings, as well as findings of (Rajeswaran et al. 2019; Rajeswaran, Mordatch, and Kumar 2020; Lorraine, Vicol, and Duvenaud 2020) suggest that there is a gap between the assumptions of the IFT and its applicability in practice.

9 Conclusion

The paper proposes *optimal model design* (OMD), a method for learning control-oriented models that addresses the shortcomings of likelihood-based MBRL approaches. OMD optimizes the expected returns in an end-to-end manner and alleviates the objective mismatch of standard MBRL methods that train models using a proxy of the true RL objective. Theoretically, we characterize the set of optimal solutions to OMD and illustrate the efficacy of OMD over MLE agents for approximating optimal value functions. Empirically, we introduce approximations to apply OMD to non-tabular environments and demonstrate the improved performance of OMD in settings with limited model capacity. Perhaps surprisingly, we find that the OMD model can have low likelihood, yet the model is useful for maximizing returns. Overall, OMD sheds light on the potential of control-oriented methods for model-based reinforcement learning.

Acknowledgements

EN thanks Iurii Kemaev and Clement Gehring for invaluable help with JAX; Tristan Deleu, Gauthier Gidel, Amy Zhang, Aravind Rajeswaran, Ilya Kostrikov, Brandon Amos, and Aaron Courville for insightful discussions; Pierluca D'Oro, David Brandfonbrener, Valentin Thomas, and Timur Garipov for providing useful suggestions on the early draft of the paper; Compute Canada for computational resources.

We acknowledge the Python community (Van Rossum and Drake Jr 1995; Oliphant 2007) for developing the core set of tools that enabled this work, including JAX (Bradbury et al. 2018; Babuschkin et al. 2020), Jupyter (Kluyver et al. 2016), Matplotlib (Hunter 2007), numpy (Oliphant 2006; Van Der Walt, Colbert, and Varoquaux 2011), pandas (McKinney 2012), and SciPy (Jones, Oliphant, and Peterson 2014).

References

Abachi, R.; Ghavamzadeh, M.; and Farahmand, A.-m. 2020. Policy-Aware Model Learning for Policy Gradient Methods. *arXiv preprint arXiv:2003.00030*.

Abbad, M.; and Filar, J. 1992. Perturbation and stability theory for Markov control problems. *IEEE Transactions on Automatic Control*, 37(9): 1415–1420.

Amos, B.; Jimenez, I.; Sacks, J.; Boots, B.; and Kolter, J. Z. 2018. Differentiable mpc for end-to-end planning and control. In *Advances in Neural Information Processing Systems*, 8289–8300.

Amos, B.; and Yarats, D. 2020. The differentiable cross-entropy method. In *International Conference on Machine Learning*, 291–302. PMLR.

Ayoub, A.; Jia, Z.; Szepesvari, C.; Wang, M.; and Yang, L. F. 2020. Model-Based Reinforcement Learning with Value-Targeted Regression. *arXiv preprint arXiv:2006.01107*.

Babuschkin, I.; Baumli, K.; Bell, A.; Bhupatiraju, S.; Bruce, J.; Buchlovsky, P.; Budden, D.; Cai, T.; Clark, A.; Danihelka, I.; Fantacci, C.; Godwin, J.; Jones, C.; Hennigan, T.; Hessel, M.; Kapturowski, S.; Keck, T.; Kemaev, I.; King, M.; Martens, L.; Mikulik, V.; Norman, T.; Quan, J.; Papamakarios, G.; Ring, R.; Ruiz, F.; Sanchez, A.; Schneider, R.; Sezener, E.; Spencer, S.; Srinivasan, S.; Stokowiec, W.; and Viola, F. 2020. The DeepMind JAX Ecosystem.

Bacon, P.-L.; Schäfer, F.; Gehring, C.; Anandkumar, A.; and Brunskill, E. 2019. A Lagrangian Method for Inverse Problems in Reinforcement Learning. In *Optimization in RL workshop at NeurIPS 2019*.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2019. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, 690–701.

Barto, A. G.; Sutton, R. S.; and Anderson, C. W. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5): 834–846.

Beattie, C.; Leibo, J. Z.; Teplyashin, D.; Ward, T.; Wainwright, M.; Küttler, H.; Lefrancq, A.; Green, S.; Valdés, V.; Sadik, A.; et al. 2016. Deepmind lab. *arXiv preprint arXiv:1612.03801*.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279.

Boots, B.; Siddiqi, S. M.; and Gordon, G. J. 2011. Closing the learning-planning loop with predictive state representations. *Int. J. Robotics Res.*, 30(7): 954–966.

Bradbury, J.; Frostig, R.; Hawkins, P.; Johnson, M. J.; Leary, C.; Maclaurin, D.; Necula, G.; Paszke, A.; VanderPlas, J.; Wanderman-Milne, S.; and Zhang, Q. 2018. JAX: composable transformations of Python+NumPy programs.

Christianson, B. 1994. Reverse accumulation and attractive fixed points. *Optimization Methods and Software*, 3(4): 311–326.

Chua, K.; Calandra, R.; McAllister, R.; and Levine, S. 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, 4754–4765.

Dadashi, R.; Taiga, A. A.; Le Roux, N.; Schuurmans, D.; and Bellemare, M. G. 2019. The value function polytope in reinforcement learning. In *International Conference on Machine Learning*, 1486–1495. PMLR.

Dorato, P.; Cerone, V.; and Abdallah, C. 1994. *Linear-quadratic control: an introduction*. Simon & Schuster, Inc.

D'Oro, P.; Metelli, A. M.; Tirinzoni, A.; Papini, M.; and Restelli, M. 2020. Gradient-aware model-based policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3801–3808.

Farahmand, A.-m.; Barreto, A.; and Nikovski, D. 2017. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, 1486–1494.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 1587–1596. PMLR.

Grimm, C.; Barreto, A.; Singh, S.; and Silver, D. 2020. The Value Equivalence Principle for Model-Based Reinforcement Learning. *Advances in Neural Information Processing Systems*, 33.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Hafner, D.; Lillicrap, T.; Norouzi, M.; and Ba, J. 2020. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*.

Hasselt, H. 2010. Double Q-learning. *Advances in neural information processing systems*, 23: 2613–2621.

Hunter, J. D. 2007. Matplotlib: A 2D graphics environment. *IEEE Annals of the History of Computing*, 9(03): 90–95.

Janner, M.; Fu, J.; Zhang, M.; and Levine, S. 2019. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 12519–12530.

Jones, E.; Oliphant, T.; and Peterson, P. 2014. *SciPy: Open source scientific tools for Python*.

Joseph, J.; Geramifard, A.; Roberts, J. W.; How, J. P.; and Roy, N. 2013. Reinforcement learning with misspecified model classes. In *2013 IEEE International Conference on Robotics and Automation*, 939–946. IEEE.

Kaiser, L.; Babaeizadeh, M.; Milos, P.; Osinski, B.; Campbell, R. H.; Czechowski, K.; Erhan, D.; Finn, C.; Kozakowski, P.; Levine, S.; et al. 2019. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*.

Kalashnikov, D.; Varley, J.; Chebotar, Y.; Swanson, B.; Jonschkowski, R.; Finn, C.; Levine, S.; and Hausman, K. 2021. MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale. *arXiv preprint arXiv:2104.08212*.

Kearns, M.; and Singh, S. 2002. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2): 209–232.

Kluyver, T.; Ragan-Kelley, B.; Pérez, F.; Granger, B. E.; Bussonnier, M.; Frederic, J.; Kelley, K.; Hamrick, J. B.; Grout, J.; Corlay, S.; et al. 2016. *Jupyter Notebooks-a publishing format for reproducible computational workflows.*, volume 2016.

Kostrikov, I. 2021. JAXRL: Implementations of Reinforcement Learning algorithms in JAX.

Krantz, S. G.; and Parks, H. R. 2012. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media.

Lambert, N.; Amos, B.; Yadan, O.; and Calandra, R. 2020. Objective Mismatch in Model-based Reinforcement Learning. *arXiv preprint arXiv:2002.04523*.

Levine, S. 2018. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*.

Li, L.; Walsh, T. J.; and Littman, M. L. 2006. Towards a Unified Theory of State Abstraction for MDPs. *ISAIM*, 4: 5.

Lorraine, J.; Vicol, P.; and Duvenaud, D. 2020. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, 1540–1552. PMLR.

McKinney, W. 2012. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.".

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Oliphant, T. E. 2006. *A guide to NumPy*, volume 1. Trelgol Publishing USA.

Oliphant, T. E. 2007. Python for scientific computing. *Computing in Science & Engineering*, 9(3): 10–20.

Rajeswaran, A.; Finn, C.; Kakade, S. M.; and Levine, S. 2019. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, 113–124.

Rajeswaran, A.; Mordatch, I.; and Kumar, V. 2020. A Game Theoretic Framework for Model Based Reinforcement Learning. *arXiv preprint arXiv:2004.07804*.

Rubinstein, R. Y. 1997. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1): 89–112.

Rust, J. 1988. Maximum likelihood estimation of discrete control processes. *SIAM journal on control and optimization*, 26(5): 1006–1024.

Sato, M.; Abe, K.; and Takeda, H. 1988. Learning control of finite Markov chains with an explicit trade-off between estimation and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(5): 677–684.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; et al. 2019. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*.

Skelton, R. 1989. Model error concepts in control design. *International Journal of Control*, 49(5): 1725–1753.

Sutton, R. S. 1991. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4): 160–163.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S.; McAllester, D.; Singh, S.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12: 1057–1063.

Tamar, A.; Wu, Y.; Thomas, G.; Levine, S.; and Abbeel, P. 2016. Value iteration networks. *arXiv preprint arXiv:1602.02867*.

Theil, H. 1957. A Note on Certainty Equivalence in Dynamic Planning. *Econometrica*, 25(2): 346.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

Van Der Walt, S.; Colbert, S. C.; and Varoquaux, G. 2011. The NumPy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2): 22–30.

Van Rossum, G.; and Drake Jr, F. L. 1995. *Python tutorial*, volume 620. Centrum voor Wiskunde en Informatica Amsterdam.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning*, 8(3-4): 279–292.

Xu, H.; and Mannor, S. 2010. Distributionally Robust Markov Decision Processes. In Lafferty, J.; Williams, C.; Shawe-Taylor, J.; Zemel, R.; and Culotta, A., eds., *Advances in Neural Information Processing Systems*, volume 23, 2505–2513. Curran Associates, Inc.

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, 1433–1438. Chicago, IL, USA.