Control-Oriented Model-Based Reinforcement Learning with Implicit Differentiation

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

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) (Grefenstette et al., 1990; Sutton, 1991; Lin, 1992; Boots et al., 2011; Chua et al., 2018; Hafner et al., 2019; Janner et al., 2019; Kaiser et al., 2019). 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; Kurano, 1972; Mandl, 1974; Georgin, 1978; Borkar & Varaiya, 1979; Hernández-Lerma & Marcus, 1985; Manfred, 1987; Sato et al., 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 & Filar, 1992; Manfred, 1987). 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.

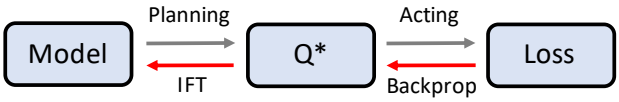

Figure 1. Illustration of the *Optimal Model Design* approach: we treat the optimal Q-function as an implicit function of the model and calculate the gradient with respect to the model parameters via the implicit function theorem as described in Section 4.1.

While much of the work in control-oriented model learning has focused on robust or uncertainty-based methods (Ludwig & Walters, 1982; Nilim & Ghaoui, 2005; Iyengar, 2005; Xu & Mannor, 2010; Yu et al., 2020), we propose an algorithm for learning a model that directly optimizes the expected return using implicit differentiation (Christianson, 1994; Griewank & Walther, 2008). 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 (Denardo, 1967) 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; Sorg et al., 2010; Bacon et al., 2019), we call our control-oriented method *optimal model design* (OMD).

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

Unsupervised Reinforcement Learning Workshop at the 37^{th} International Conference on Machine Learning, 2021.

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 demonstrate 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 et al. (2017) train a model to minimize the difference between values of the real next states and the next states predicted by the dynamics. Abachi et al. (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 et al., 1994). Tamar et al. (2016) uses a differentiable approximation of the value iteration algorithm to learn a planner. Amos & Yarats (2020) optimize the parameters of a sampling 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 but 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 et al. (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 et al. (2019) propose an implicit layer that corresponds to an infinite depth neural network and find a fixed point of the layer via IFT. Our method also solves a bi-level problem: in the inner loop, we train an action-value function compatible with the model, while in the outer loop we update the model parameters towards maximizing the expected returns.

3. Preliminaries

Reinforcement Learning (RL) (Sutton & 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 algorithms 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 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

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

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]

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:

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

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 model parameters θ (Rajeswaran et al., 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 turn it into a value-based approach (Watkins & 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 $\frac{\partial \varphi(\theta)}{\partial \theta}$.

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,*

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

We provide a proof in Appendix A. 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 $\frac{\partial \varphi(\theta)}{\partial \theta}$. We provide an implementation of a custom vector-Jacobian product for the implicit function φ in Appendix B allowing to use φ as a block in a differentiable computational graph. We provide an implementation of a custom vector-jacobian product for the implicit function φ in Appendix B allowing to use φ as a block in a differentiable computational graph.

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

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 given π_{Q^*} in a closed form (Sutton & 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

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

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 decision making. Figure 2 reflects the problems

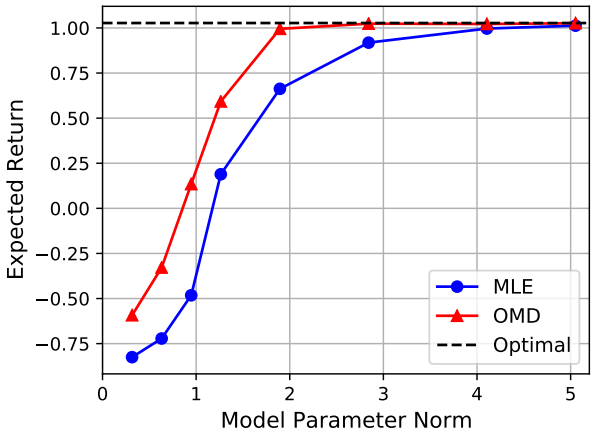

Figure 2. Expected returns for the tabular MDP under the model class misspecification. The OMD model optimizes the expected returns directly, while the MLE agent minimizes the KL divergence for model learning. OMD outperforms MLE when the model representational capacity is limited.

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. The 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. *If we let the soft Bellman operator* (3) *temperature $\alpha \to 0$ and let θ be* any *model parameters from*

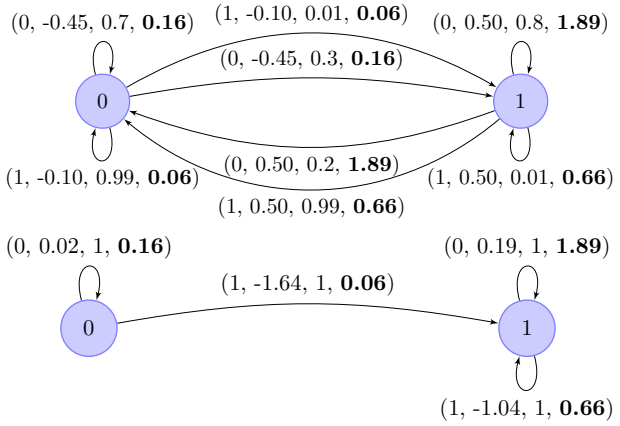

Figure 3. Two different MDPs with the same optimal Q-function (a fixed point of the induced Bellman operator). Circles represent states, tuples are organized as (action, reward, transition probability, optimal Q value). Top: the original MDP taken from (Dadashi et al., 2019). Bottom: an MDP with a trained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

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:

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

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 et al., 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.

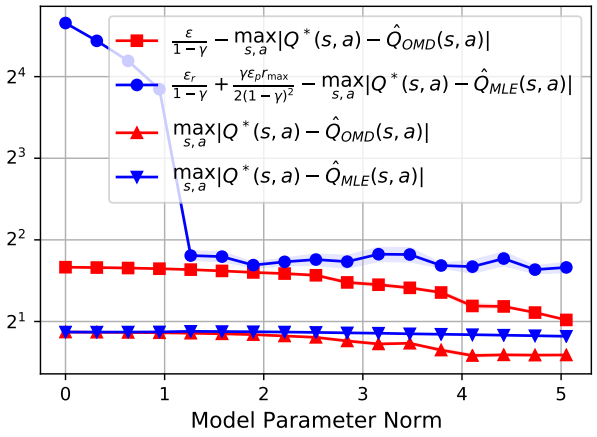

Figure 4. Q^* approximation error and tightness of the error bounds under the model misspecification. Given a limited model representation capacity, OMD agent approximates Q^* more accurately and enjoys a tighter bound.

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.*

- *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 $\max_{s,a} \left| B\hat{Q}_{\text{OMD}}(s,a) - B^{\hat{\theta}}\hat{Q}_{\text{OMD}}(s,a) \right| = \epsilon$, we have*

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

We prove the bounds in Appendix H using similar arguments as the proof of the simulation lemma (Kearns & Singh, 2002). The MLE bound has a $\frac{1}{(1-\gamma)^2}$ term making the bound loose compared to the OMD bound with only the $\frac{1}{1-\gamma}$ term. The bound suggests that OMD approximation error translates into a lower Q^* approximation error than for the MLE model. 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. We analyze the effect of the approximations and perform an ablation study in Appendix F.

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:

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

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 et al., 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 et al. (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. We investigate the phenomenon deeper and discuss possible explanations in Appendix C.

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 $L(\theta, w)$. 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.

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 (15).
until the maximum number of interactions is reached

Resulting gradient. The changes above in the objective function and the constraint yield the following optimization problem:

$$\min_{w,\theta} L^{\text{true}}(w) \quad \text{s.t.} \quad \frac{\partial L(\theta, w)}{\partial w} = \mathbf{0}. \quad (14)$$

The Q function and IFT approximations and result in the following gradient with respect to the model parameters:

$$\frac{\partial L^{\text{true}}(\theta)}{\partial \theta} \approx - \underbrace{\frac{\partial L^{\text{true}}(w^*)}{\partial w}}_{\text{grad Bellman}} \cdot \underbrace{\frac{\partial^2 L(\theta, w^*)}{\partial \theta \partial w}}_{\text{approx IFT}} \Big|_{w^* = \varphi(\theta)} \quad (15)$$

The OMD algorithm is summarised in Algorithm 1. The only difference between Dyna-based approaches and OMD (highlighted in blue) is given by the gradient used to train model parameters θ.

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.
- The parameters θ of the OMD model have low likelihood, yet the agent that acts using the Q-function trained with the model achieves near-optimal returns in the true MDP.

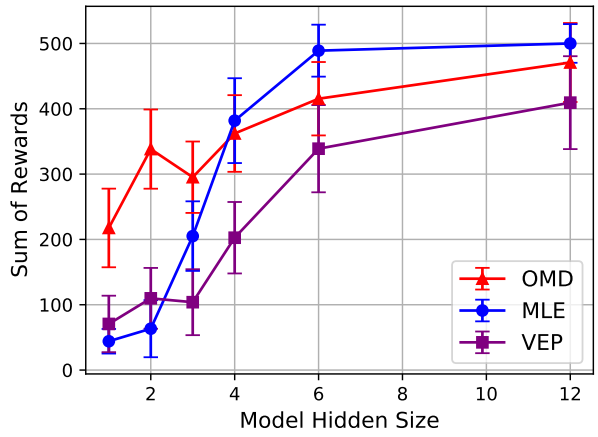

Figure 5. The final performance of the agents in CartPole for varying hidden dimensionality of the model networks. The OMD model makes useful predictions under the model misspecification. The error bar is the standard error measured over 10 random seeds.

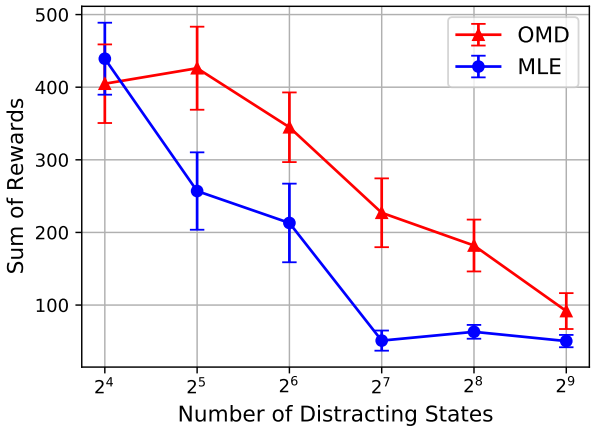

Figure 6. The performance when the state space is augmented with uninformative noise. OMD is more robust as the number of distractor increases, while VEP fails for any positive number of distractors. The error bar is the standard error measured over 10 seeds.

Setup. We provide full details about the experimental setup and hyperparameters in Appendix D. We choose CartPole (Barto et al., 1983) to have controllable experiments but also include results on MuJoCo HalfCheetah (Todorov et al., 2012) with similar findings in Appendix G 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:

$$\ell_{\text{VEP}}(\theta) = \sum_{\pi \in \Pi} \sum_{V \in \mathcal{V}} \sum_{s \in \mathcal{S}} \left(B_\pi V(s) - B_\pi^\theta V(s) \right)^2, \quad (16)$$

where $B_\pi^\theta V(s) = \mathbb{E}_{a \sim \pi(a|s), s' \sim p_\theta(s'|s,a)} \left(r_\theta(s,a) + \gamma V(s') \right)$, B_π is the real model counterpart estimated from samples, 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 in isolation. 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 poor 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): accurately predicting the next observations can be infeasible because the underlying dynamics can be too involved and there might be few components that are important for taking action. 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 Figure 7. Quantitatively, the MSE of OMD models is *higher than the MSE of a randomly initialized model*, while OMD achieves higher returns than MLE. This finding suggests that the dynamics might not need to produce predictions close to the true states to be useful for planning.

Qualitatively, we visualize individual coordinates of the predictions under the model misspecification in Figure 8. We predict the immediate next states given a sequence of states and actions from an optimal trajectory. We use one of the runs of the OMD agent with the hidden size 1 that achieves optimal returns of 500 and observe that Cart Velocity predictions are nearly constant. On the other hand, an MLE model with the hidden size 1 spends its limited capacity to predict the fluctuations in Cart Velocity leading to a significant deterioration in obtained returns.

Overall, these 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.

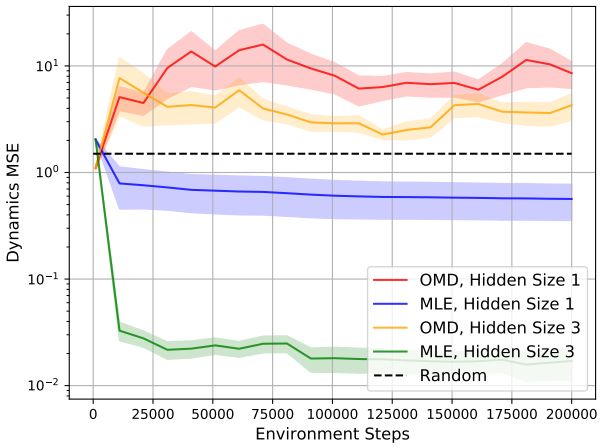

Figure 7. Mean squared error (MSE) of the next state prediction of OMD and MLE models in CartPole under model misspecification (number of hidden units 1 and 3). The shaded region is the standard error over 10 runs. The corresponding plot with returns can be found in Appendix E.

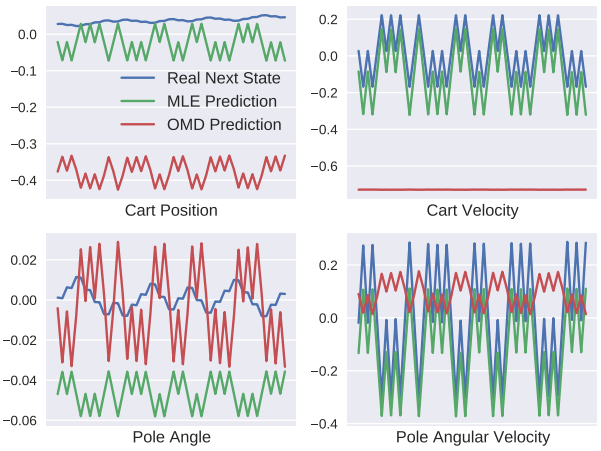

Figure 8. Next state predictions of OMD and MLE models with hidden size 1. The OMD agent discards Cart Velocity and predicts unrealistic Cart Position but achieves the optimal returns of 500.

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 (Kaiser et al., 2019; Hafner et al., 2020). Based on Figure 8, we also conjecture that OMD can learn an abstract model that can ignore parts of the original state space that are irrelevant for control. This might allow applying OMD in a zero-shot manner for transfer learning tasks where the underlying dynamics remain unchanged.

Implicit differentiation is not the only way to solve the described 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 finding a saddle point 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; 2020; Lorraine et al., 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 models for model-based reinforcement learning.

Acknowledgments

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 providing computational resources.

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

References

Abachi, R., Ghavamzadeh, M., and Farahmand, A.-m. Policy-aware model learning for policy gradient methods. *arXiv preprint arXiv:2003.00030*, 2020.

Abbad, M. and Filar, J. Perturbation and stability theory for markov control problems. *IEEE Transactions on Automatic Control*, 37(9):1415–1420, 1992. doi: 10.1109/9.159584.

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

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

Ayoub, A., Jia, Z., Szepesvari, C., Wang, M., and Yang, L. F. Model-based reinforcement learning with value-targeted regression. *arXiv preprint arXiv:2006.01107*, 2020.

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. The DeepMind JAX Ecosystem, 2020. URL http://github.com/deepmind.

Bacon, P.-L., Schäfer, F., Gehring, C., Anandkumar, A., and Brunskill, E. A lagrangian method for inverse problems in reinforcement learning. In *Optimization in RL workshop at NeurIPS 2019*, 2019.

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

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

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

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

Boots, B., Siddiqi, S. M., and Gordon, G. J. Closing the learning-planning loop with predictive state representations. *Int. J. Robotics Res.*, 30(7):954–966, 2011. doi: 10.1177/0278364911404092.

Borkar, V. and Varaiya, P. Adaptive control of markov chains, i: Finite parameter set. *IEEE Transactions on Automatic Control*, 24(6):953–957, December 1979.

Boyd, S., Boyd, S. P., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

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. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Christianson, B. Reverse accumulation and attractive fixed points. *Optimization Methods and Software*, 3(4):311–326, January 1994. doi: 10.1080/10556789408805572.

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

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

Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *arXiv preprint arXiv:1406.2572*, 2014.

Denardo, E. V. Contraction mappings in the theory underlying dynamic programming. *SIAM Review*, 9(2):165–177, April 1967. doi: 10.1137/1009030.

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

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

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

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

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

Gehring, C., Bacon, P.-L., and Schaefer, F. FAX: differentiating fixed point problems in JAX, 2019. Available at: https://github.com/gehring/fax.

Georgin, J. P. Estimation et controle des chaines de markov sur des espaces arbitraires. In *Lecture Notes in Mathematics*, pp. 71–113. Springer Berlin Heidelberg, 1978. doi: 10.1007/bfb0063261.

Grefenstette, J. J., Ramsey, C. L., and Schultz, A. C. Learning sequential decision rules using simulation models and competition. *Machine Learning*, 5(4):355–381, October 1990. doi: 10.1007/bf00116876.

Griewank, A. and Walther, A. *Evaluating Derivatives*. Society for Industrial and Applied Mathematics, January 2008.

Grimm, C., Barreto, A., Singh, S., and Silver, D. The value equivalence principle for model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 2020.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018a.

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

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565. PMLR, 2019.

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

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

Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., and van Zee, M. Flax: A neural network library and ecosystem for JAX, 2020. URL http://github.com/google/flax.

Hernández-Lerma, O. and Marcus, S. I. Adaptive control of discounted markov decision chains. *Journal of Optimization Theory and Applications*, 46(2):227–235, June 1985. doi: 10.1007/bf00938426.

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

Iyengar, G. N. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, May 2005. doi: 10.1287/moor.1040.0129.

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

Jiang, N. Lecture notes on statistical reinforcement learning. 2018.

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

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

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

Kalashnikov, D., Varley, J., Chebotar, Y., Swanson, B., Jonschkowski, R., Finn, C., Levine, S., and Hausman, K. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

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

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. *Jupyter Notebooks-a publishing format for reproducible computational workflows.*, volume 2016. 2016.

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

Kurano, M. Discrete-time markovian decision processes with an unknown parameter-average return criterion. *Journal of the Operations Research Society of Japan*, 15(2):67–76, 1972.

Lambert, N., Amos, B., Yadan, O., and Calandra, R. Objective mismatch in model-based reinforcement learning. *arXiv preprint arXiv:2002.04523*, 2020.

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

Li, L., Walsh, T. J., and Littman, M. L. Towards a unified theory of state abstraction for mdps. *ISAIM*, 4:5, 2006.

Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3-4):293–321, May 1992. doi: 10.1007/bf00992699.

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

Ludwig, D. and Walters, C. Optimal harvesting with imprecise parameter estimates. *Ecological Modelling*, 14(3-4): 273–292, January 1982. doi: 10.1016/0304-3800(82) 90023-0.

Mandl, P. Estimation and control in markov chains. *Advances in Applied Probability*, 6(1):40–60, March 1974. doi: 10.2307/1426206.

Manfred, S. Estimation and control in discounted stochastic dynamic programming. *Stochastics*, 20(1):51–71, January 1987. doi: 10.1080/17442508708833435.

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

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. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

Nilim, A. and Ghaoui, L. E. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, October 2005. doi: 10.1287/opre.1050.0216.

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

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

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

Rajeswaran, A., Mordatch, I., and Kumar, V. A game theoretic framework for model based reinforcement learning. *arXiv preprint arXiv:2004.07804*, 2020.

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

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

Sagun, L., Evci, U., Guney, V. U., Dauphin, Y., and Bottou, L. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Sato, M., Abe, K., and Takeda, H. 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, 1988. doi: 10. 1109/21.21595.

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

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

Sorg, J., Lewis, R. L., and Singh, S. Reward design via online gradient ascent. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems*, volume 23, pp. 2190–2198. Curran Associates, Inc., 2010.

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

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

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

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

Theil, H. A note on certainty equivalence in dynamic planning. *Econometrica*, 25(2):346, April 1957.

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

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

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

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

Xu, H. and Mannor, S. 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, pp. 2505–2513. Curran Associates, Inc., 2010.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems 33 pre-proceedings*, 2020.

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

A. Proof of IFT

The implicit function theorem is a well known result discussed in, for example, (Krantz & 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,*

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

Proof. By the assumption, we have

$$f(\theta, \varphi(\theta)) = \mathbf{0} \quad \forall \theta \in U.$$

Taking the total derivative of f with respect to θ, we have

$$\frac{\partial f(\theta, w^*)}{\partial \theta} + \frac{\partial f(\theta, w^*)}{\partial w}\frac{\partial \varphi(\theta)}{\partial \theta}\Big|_{w^*=\varphi(\theta)} = \mathbf{0}.$$

Rearranging the terms and using the invertibility of the Jacobian, we get

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

□

The implicit function theorem (IFT) allows differentiating outputs of black-box functions $\varphi(\theta)$ that do not have a closed (differentiable) form. For example, in Section 4.2, function φ takes as input the parameters of the model and applies fixed-point iteration until convergence to find the optimal value function for the given parameters. This contrasts implicit differentiation to other meta-learning algorithms like MAML (Finn et al., 2017) that differentiate *through* the iterations of the inner loop procedure.

B. Implicit Differentiation in JAX

We provide an implementation of implicit differentiation in JAX (Babuschkin et al., 2020; Heek et al., 2020) by adapting the code from the library for finding fixed points (Gehring et al., 2019). The implementation requires *a solver* that takes parameters θ and an initial w_0 as input and outputs $\varphi(\theta) = w^*$ such that $f(\theta, w^*) = 0$. Then, we define a custom vector-Jacobian product that allows using $\varphi(\theta)$ as a block in a differentiable computational graph. We highlight the importance of having an implementation *without* explicitly forming the matrices in (7) which is crucial for large-scale applications. The implementation allows using both the version with the inverse Jacobian term as well as with the identity approximation as discussed in Section 6.

```
import jax.numpy as jnp
from jax import custom_vjp, vjp
from functools import partial
from jax.scipy.sparse.linalg import cg

@partial(custom_vjp, nondiff_argnums=(0, 3))
def root_solve(f, w0, p, solver):
  return solver(f, w0, p)

def fwd(f, w0, p, solver):
  sol = root_solve(f, w0, p, solver)
  return sol, (sol, p)

def rev(f, solver, res, g):
  sol, p = res
  _, dp_vjp = vjp(lambda y: f(y, sol), p)
  if USE_IDENTITY_INVERSE:
    vdp = dp_vjp(-g)[0]
  else:
    _, dsol_vjp = vjp(lambda w: f(p, w), sol)
    vdsoli = cg(lambda v: dsol_vjp(v)[0], g)
    vdp = dp_vjp(-vdsoli[0])[0]
  return jnp.zeros_like(sol), vdp

root_solve.defvjp(fwd, rev)
sol = root_solve(f, w0, p, solver)
# solver returns sol: f(p, sol) = 0
# sol is differentiable w.r.t. p
```

C. Sensitivity to IFT Approximations

The IFT provides a way to calculate the derivatives of a black-box function φ. The expression (7) is valid under the assumptions that

1. the inner loop solution w^* satisfies the equation $f(\theta, w^*) = 0$ exactly;
2. the Jacobian term $\frac{\partial f(\theta, w^*)}{\partial w}$ is inverted accurately.

Ensuring both of the conditions can be challenging for large-scale applications. This appendix analyzes the sensitivity to the conditions in a controlled manner for the 2-state MDP from Figure 3, while Appendix F studies the sensitivity for the function approximation case. At every outer loop iteration, we calculate the exact w^*, add gaussian noise with standard deviation σ, and observe the effect on the expected returns J after the convergence of θ. Figure 9 demonstrates the results for the exact outer loop gradient $\frac{\partial \varphi(\theta)}{\partial \theta}$ as well as the gradient using the identity approximation of the inverse Jacobian term. Surprisingly, we did not observe significant benefits of using the Jacobian $\frac{\partial f(\theta, w^*)}{\partial w}$. We conjecture that the inverse Jacobian acts like a preconditioner (Boyd et al., 2004) on $\frac{\partial \varphi(\theta)}{\partial \theta}$ and the preconditioner is useful in our setting only near the exact inner loop solutions w^*. We leave the theoretical investigation of the IFT sensitivity as future work and refer the reader to Lorraine et al. (2020) for a discussion about approximations of the Jacobian term.

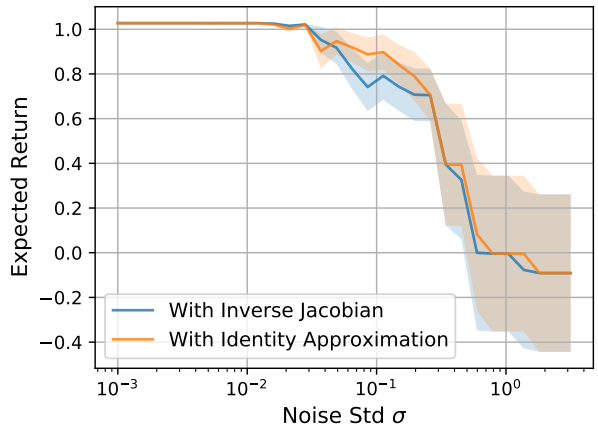

Figure 9. The expected returns as a function of the inner loop solutions noise magnitude σ. OMD with the true and the approximate $\frac{\varphi(\theta)}{\theta}$ has the same resulting returns as σ increases. The shaded region is the standard error over 10 runs.

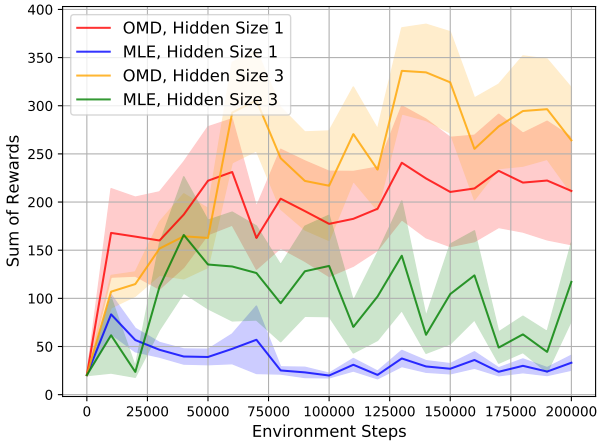

Figure 10. The evaluation returns of OMD and MLE agents in CartPole under the model misspecification (number of hidden units 1 and 3). OMD returns are larger than the returns of MLE even though MLE predicts the next states more accurately. The shaded region is the standard error over 10 runs.

D. Experimental Details

In Section 7, we use CartPole (Barto et al., 1983), an environment with 2 actions, 4-dimensional continuous state space, and optimal returns of 500. We train the agents for 200000 environment steps. The temperature α is 0.01. We sample from the replay buffer with a mini-batch size of 256. The discount factor γ is 0.99. At each time step during training, the agent chooses a random action with a probability of 0.1 for exploration. We have a separate copy of the environment where we evaluate the agent and take the average over 10 runs to estimate the returns. We run each experiment using 10 random seeds.

We set the number of Q-function updates K equal to 1. For the results with $K = 3$ and $K = 10$, see Figure 11. We highlight that after each outer loop step, weights w are warm-started using the last iterate of the previous inner loop (instead of randomly initializing w and training from scratch). We use Adam optimizer with the learning rate 10^{-3} for updating θ and perform a hyperparameter sweep over the learning rate for w in $\{3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}\}$. We make a sweep over the moving average coefficient τ for the target network $\bar{w}$ in $\{0.005, 0.01\}$. Both of the parameters control how fast the Q-network parameters are updated relatively to the model parameters. Since the CartPole environment is non-stochastic, we use a deterministic dynamics model. All networks have two hidden layers and ReLU activations (Nair & Hinton, 2010). For both hidden layers in all networks, we set the dimensionality to 32. In the experiment with the limited model class capacity, we vary the hidden dimensionality in $\{1, 2, 3, 4, 6, 12\}$ for the dynamics and reward networks to measure how the limitation affects the agent's performance. In the experiment with the distractors, we vary the number of gaussians in $\{2^4, 2^5, 2^6, 2^7, 2^8, 2^9\}$ to measure how the uninformative state components affect the returns.

For the value equivalence principle (VEP) baseline, we have followed the experimental setup from the original paper (Grimm et al., 2020). Perhaps surprisingly, the authors find that it suffices to use a set of all deterministic state-independent policies as Π and 5 random value functions as $\mathcal{V}$ for CartPole (see Appendix A.2.3 in (Grimm et al., 2020)).

We have used CPU-only nodes of the internal cluster. Each experiment requires 10 seeds $\times$ 3 algorithms $\times$ 2 τ's $\times$ 6 hidden sizes / 6 numbers of distractors $\times$ 3 LRs resulting in 2160 total jobs.

E. Returns under Model Misspecification

In Section 7, we demonstrate that OMD models have *a higher* mean squared error (MSE) than predictions of a randomly initialized model (Figure 7). We provide the corresponding returns of the OMD and MLE agents in Figure 10. While the MLE model has more accurate predictions, OMD achieves higher returns using the next states significantly deviating from the real next states. The result suggests that *likelihood optimization may be an unnecessary step* for MBRL algorithms.

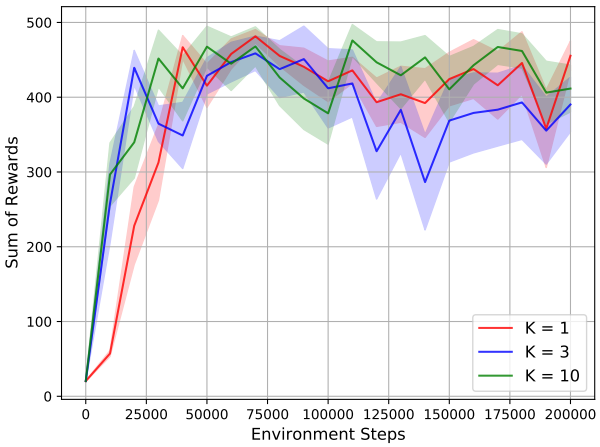

Figure 11. The evaluation returns of OMD agents for the varying number of inner loop steps K per an outer loop step. The difference between 1, 3, and 10 is insignificant. The shaded region is the standard error over 10 runs.

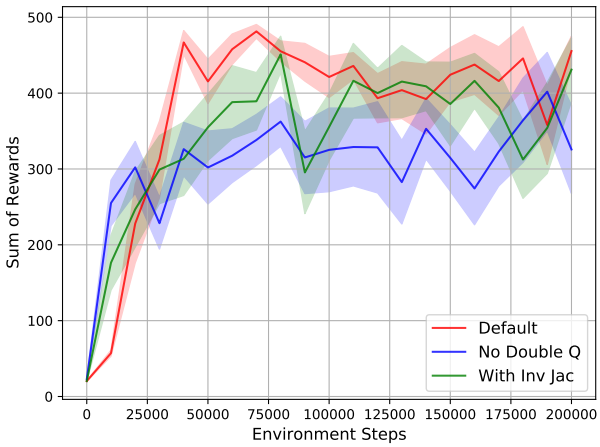

Figure 12. The evaluation returns of OMD agents for the default agent, the agent without double Q learning, and the agent without the identity approximation of the inverse jacobian. Using two Q networks increases the returns, while using the inverse jacobian does not change the performance significantly. The shaded region is the standard error over 10 runs.

F. Ablation Study

Section 6 introduces a series of approximations to scale the OMD algorithm to non-tabular environments. We analyze the effect of the approximations by varying the number of inner loop steps K, using the inverse Jacobian term, and using a single Q-function estimator (without double Q-learning).

Figure 11 shows the (undiscounted) returns for K varying in $\{1, 3, 10\}$. We did not observe significant changes in performance for different values of K for CartPole. The result suggests that as long as the Q-network update speed (which is also controlled by the target update coefficient τ and learning rates) stays aligned with the model update speed, adding more inner loop steps is not necessary.

Figure 12 demonstrates the returns without using double Q learning and without using the identity approximation of the inverse jacobian term $\left(\frac{\partial f(\theta, w^*)}{\partial w}\right)^{-1}$. We observed that Q-functions trained with OMD can be prone to overestimation of Q-values showing that double Q-learning is important for OMD. We conjecture that training a model that maximizes the returns can amplify the overestimation bias (Hasselt, 2010) caused by using (soft) maximized sampled targets.

Surprisingly, we did not observe significant benefits from using the inverse Jacobian $\left(\frac{\partial f(\theta, w^*)}{\partial w}\right)^{-1}$. We conjecture that there are two reasons explaining the phenomenon. First, the modified constraint in (14) forces the gradient of $L(\theta, w)$ to be zero, implying that the Jacobian is in fact the Hessian matrix $\frac{\partial^2 L(\theta, w^*)}{\partial w^2}$. Dauphin et al. (2014); Sagun et al. (2017) observed that Hessians of neural networks tend to be singular. Since a system of linear equations with a singular matrix has multiple solutions, it is up to a linear algebra solver to choose the solution. One of the alternatives would be a min-norm solution corresponding to the Moore-Penrose pseudoinverse and the solution might not be providing a useful inductive bias for the learning process of θ. Second, the Jacobian term could be useful only in proximity to the exact inner loop solution w^*. Since the practical algorithm performs only K inner loop steps and does not reach the exact w^*, the curvature information provided by the Jacobian might not be beneficial for training θ.

Finally, we tried to use the gradient constraint in (14) in the tabular setting. We got similar results as with the constraint on Q-values in (2) suggesting that the two constraints have similar effects on the learning process. Overall, the ablation study provides evidence that OMD is robust to the choice of the number of inner loop steps and the IFT approximations, while double Q-learning is the only important algorithmic modification.

G. Results on HalfCheetah

We provide an additional comparison of OMD and MLE agents under the model misspecification on MuJoCo HalfCheetah (Todorov et al., 2012). For both agents, the inner optimizer is Soft Actor-Critic (SAC) (Haarnoja et al., 2018a;b) with the default configuration. OMD trains the model using 15. 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

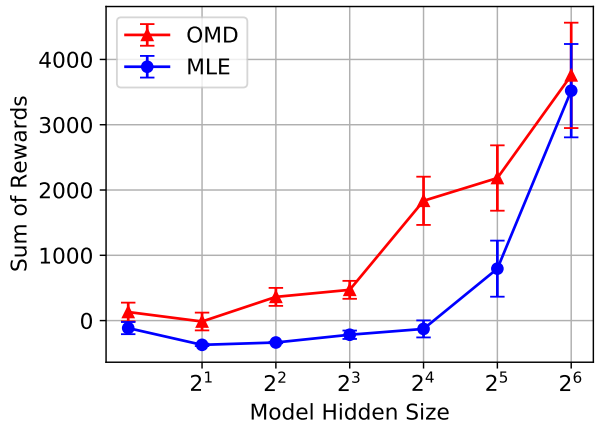

Figure 13. Returns for varying representational capacity of the OMD and MLE agents on HalfCheetah-v2. Given limited capacity, the OMD model makes more useful predictions. The std is measured over 5 runs.

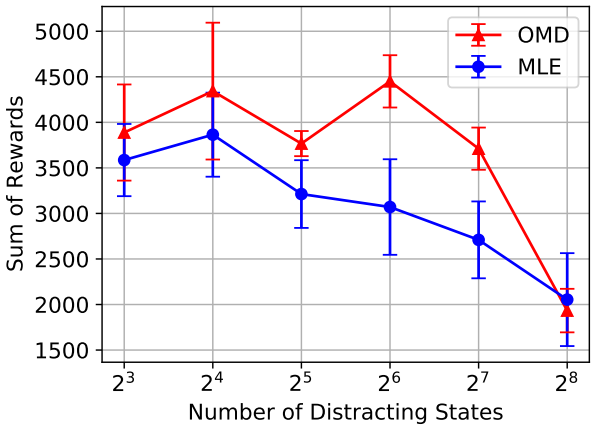

Figure 14. Returns when the state space is augmented with uninformative noise on HalfCheetah-v2. The OMD agent is more robust to the distractors. The std is measured over 5 runs.

the variance of the predictions.

We perform a hyperparameter sweep over the model learning rate in $\{10^{-4}, 3{\cdot}10^{-4}\}$, over the SAC networks learning rate in $\{10^{-4}, 3{\cdot}10^{-4}\}$, and over K in $\{1, 3\}$. Model hidden size is 64 for the experiment with distractors.

Figures 13 and 14 summarize the results. Similarly to the observations on the tabular and CartPole environments, the experiments provide evidence that OMD should be preferred over the likelihood-based agent in the model misspecification setup.

H. Proof of OMD Bounds

Section 5.2 discusses the bounds on Q^* approximation error obtained by the MLE and OMD agents. We first prove a lemma relating the error of the model approximation and the Bellman operator approximation. We then prove a theorem giving a bound on Q^* error. For simplicity, we focus on the case with the Bellman error in the true MDP (6) as the objective function and "hard" versions of the Bellman optimality operators which are obtained by taking the limit of the log-sum-exp temperature $\alpha \to 0$. Note that results for MLE hold for any agent that approximates the reward and dynamics functions, but we call the agent MLE since it is a common choice for model parameters estimation.

Notation. We denote $p(\cdot|s,a)$ and $Q(\cdot,a)$ as vectors of transition probabilities and Q-values for all states in $\mathcal{S}$. The MLE model is given by $(\hat{p}, \hat{r})$ and the corresponding Bellman optimality operator is denoted as $\hat{B}Q$. To have a distinction between OMD and MLE, we denote OMD parameters as $\hat{\theta}$ and the corresponding operator as $B^{\hat{\theta}}Q$. $\|f\|_\infty = \sup_x |f(x)|$ is the infinity norm of a function f. $\mathbf{1}$ is a vector of an appropriate size with ones as entries.

Lemma 1. (Bellman operator error bound) *Let Q be an action-value function. If the dynamics $\hat{p}$ and the 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 $r(s,a) \in [0, r_{\max}]$ $\forall s, a$, we have*

$$\left\| BQ - \hat{B}Q \right\|_\infty \leq \epsilon_r + \frac{\gamma \epsilon_p r_{\max}}{2(1-\gamma)}. \tag{17}$$

Proof. Using the derivations similar to the proof of the simulation lemma (Jiang, 2018), we obtain for any (s,a)

$$\begin{aligned}
&\left| BQ(s,a) - \hat{B}Q(s,a) \right| \\
&= \left| \left(r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} Q(s',a') \right) - \right. \\
&\quad \left. \left(\hat{r}(s,a) + \gamma \sum_{s'} \hat{p}(s'|s,a) \max_{a'} Q(s',a') \right) \right| \\
&\leq |r(s,a) - \hat{r}(s,a)| + \\
&\quad \gamma \left| \sum_{s'} \left(p(s'|s,a) - \hat{p}(s'|s,a) \right) \max_{a'} Q(s',a') \right| \\
&= \epsilon_r + \qquad \because p \text{ and } \hat{p} \text{ are distributions} \\
&\quad \gamma \left| \sum_{s'} \left(p(s'|s,a) - \hat{p}(s'|s,a) \right) \left(\max_{a'} Q(s',a') - \frac{r_{\max}}{2(1-\gamma)} \right) \right|
\end{aligned}$$

$$
\begin{aligned}
&\leq \epsilon_r + \qquad \because \text{Hölder's inequality} \\
&\gamma \left\| p(\cdot|s,a) - \hat{p}(\cdot|s,a) \right\|_1 \cdot \left\| \max_{a'} Q(\cdot, a') - \frac{r_{\max}}{2(1-\gamma)} \mathbf{1} \right\|_\infty \\
&\leq \epsilon_r + \gamma \epsilon_p \left\| \max_{a'} Q(\cdot, a') - \frac{r_{\max}}{2(1-\gamma)} \mathbf{1} \right\|_\infty \\
&\leq \epsilon_r + \frac{\gamma \epsilon_p r_{\max}}{2(1-\gamma)} \qquad \because 0 \leq Q(s', a') \leq \frac{r_{\max}}{1-\gamma}.
\end{aligned}
$$

Since the inequalities hold for all state-action pairs, we can take the maximum over (s, a) and obtain

$$
\max_{s,a} \left| BQ(s,a) - \hat{B}Q(s,a) \right| \leq \epsilon_r + \frac{\gamma \epsilon_p r_{\max}}{2(1-\gamma)}.
$$

□

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.*

- *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 $\max_{s,a} \left| B\hat{Q}_{\text{OMD}}(s,a) - B^{\hat{\theta}}\hat{Q}_{\text{OMD}}(s,a) \right| = \epsilon$, we have*

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

Proof. **(OMD)** For all state-action pairs (s, a) we get

$$
\begin{aligned}
&\left| Q^*(s,a) - \hat{Q}(s,a) \right| \\
&= \left| BQ^*(s,a) - B^{\hat{\theta}}\hat{Q}(s,a) \right| \qquad \because \text{Fixed point} \\
&= \left| BQ^*(s,a) - B\hat{Q}(s,a) + B\hat{Q}(s,a) - B^{\hat{\theta}}\hat{Q}(s,a) \right| \\
&\leq \left| BQ^*(s,a) - B\hat{Q}(s,a) \right| + \left| B\hat{Q}(s,a) - B^{\hat{\theta}}\hat{Q}(s,a) \right| \\
&\leq \epsilon + \left| BQ^*(s,a) - B\hat{Q}(s,a) \right| \\
&= \epsilon + \left| \left(r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} Q^*(s', a') \right) - \right. \\
&\qquad \left. \left(r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} \hat{Q}(s', a') \right) \right| \\
&= \epsilon + \gamma \left| \sum_{s'} p(s'|s,a) \left(\max_{a'} Q^*(s', a') - \max_{a'} \hat{Q}(s', a') \right) \right| \\
&\leq \epsilon + \gamma \left\| p(\cdot|s,a) \right\|_1 \cdot \left\| \max_{a'} Q^*(\cdot, a') - \max_{a'} \hat{Q}(\cdot, a') \right\|_\infty \\
&\leq \epsilon + \gamma \left\| \max_{a'} Q^*(\cdot, a') - \max_{a'} \hat{Q}(\cdot, a') \right\|_\infty \\
&\leq \epsilon + \gamma \max_{a'} \left\| Q^*(\cdot, a') - \hat{Q}(\cdot, a') \right\|_\infty + \\
&= \epsilon + \gamma \max_{s', a'} \left| Q^*(s', a') - \hat{Q}(s', a') \right|.
\end{aligned}
$$

Taking the maximum over (s, a), we get the recursion:

$$
\begin{aligned}
&\max_{s,a} \left| Q^*(s,a) - \hat{Q}(s,a) \right| \leq \epsilon + \gamma \max_{s,a} \left| Q^*(s,a) - \hat{Q}(s,a) \right| \\
&\max_{s,a} \left| Q^*(s,a) - \hat{Q}(s,a) \right| \leq \frac{\epsilon}{(1-\gamma)}.
\end{aligned}
$$

(MLE) The proof for MLE can be obtained using the same derivations and additionally using the result of the lemma bounding the difference between the Bellman operators:

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

□

The last inequality demonstrates that the OMD bound is tighter. OMD model directly optimizes $\left| B\hat{Q}(s,a) - \hat{Q}(s,a) \right| = \left| B\hat{Q}(s,a) - B^{\hat{\theta}}\hat{Q}(s,a) \right|$, while MLE minimizes ϵ_r and ϵ_p that only upper bound $\left| B\hat{Q}(s,a) - \hat{B}\hat{Q}(s,a) \right|$ as suggested by the lemma. Hence, given the same budget of representational capacity, OMD will learn a model that is more helpful for approximating the optimal Q-function. Finally, Figure 4 empirically supports our theory showing that both the Q^* approximation error and the error-bound gap are smaller for OMD.