LEVERAGING VALUE-AWARENESS FOR ONLINE AND OFFLINE MODEL-BASED REINFORCEMENT LEARNING

A Thesis Proposal
Presented to
The Academic Faculty

By

Nirbhay Modhe

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing
College of Computing

Georgia Institute of Technology

December 2022

LEVERAGING VALUE-AWARENESS FOR ONLINE AND OFFLINE MODEL-BASED REINFORCEMENT LEARNING

Thesis committee:

Dr. Dhruv Batra
School of Interactive Computing
Georgia Institute of Technology

Dr. Ashwin Kalyan
Allen Institute for AI

Dr. Zsolt Kira
School of Interactive Computing
Georgia Institute of Technology

Dr. Gaurav Sukhatme
University of Southern California

Dr. Mark Riedl
School of Interactive Computing
Georgia Institute of Technology

Date approved: November 29th, 2022

ACKNOWLEDGMENTS

This thesis is dedicated to all those who have supported and shaped me throughout my life.

I am grateful to my advisor, Dhruv, for his guidance and unwavering support, for always bringing out the best in me, for always reminding me of the importance of taking a moment to reflect and be in touch with the bigger picture. I would also like to thank Devi, who has played a big role in developing my work ethic and inquisitive attitude.

I am lucky to have been mentored and supported by exceptional peers at Georgia Tech – Ashwin and Rama, who have constantly reminded me of the importance of one's cultural roots and ancestral knowledge, who have never let the lack of sleep interfere with any moment of enjoyment or growth. Karan, Samyak, Prithvi, Ram, Sameer, and all of my labmates not mentioned here, it has been an absolute pleasure to have met all of you. Thank you for sharing countless moments of joy, laughter, and deep dives into (occasionally) important topics.

To my committee, thank you for all of your invaluable feedback that has helped shape this thesis.

To my undergraduate mentors and teachers Piyush Rai, Raghunath Tewari, and Surender Baswana, thank you for instilling in me a passion for computer science that constantly drives me to pursue elegant solutions to challenging problems. Your teachings have and will continue to be the source of my confidence for approaching seemingly impossible problems with an open mind. To my undergraduate friends Alok, Naman, Divyanshu, Palak, Neha, and Pramod – thank you for always being there and supporting me.

To Shubhangi, my partner in life, my source of joy in times of comfort and hardship alike, thank you for being my reason to cherish every moment in life.

It is easy to take one's parents for granted when they are always there in the background supporting every little effort throughout one's lives. While I am guilty of this in the past,

I have especially grown to realize just how lucky and grateful I am for the unconditional love, support, and blessings of my Mom and Dad. I am also grateful to my brother Vinod, for fostering my interest in electronics throughout my childhood.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Model-based Reinforcement Learning (RL) lies at the intersection of *planning* and *learning* for sequential decision making. Value-awareness in model learning has recently emerged as a means to imbue task or reward information into the objective of model learning, in order for the model to leverage specificity of a task. While finding success in theory as being superior to maximum likelihood estimation in the context of (online) model-based RL, value-awareness has remained impractical for most non-trivial tasks.

This thesis aims to bridge the gap in theory and practice by applying the principle of value-awareness to two settings – the online RL setting and offline RL setting. First, within online RL, this thesis revisits value-aware model learning from the perspective of minimizing performance difference, obtaining a novel value-aware model learning objective as a direct upper bound of it. Then, this thesis investigates and remedies the issue of stale value estimates that has so far been holding back the practicality of value-aware model learning. Using the proposed remedy, performance improvements are presented over maximum-likelihood based baselines and existing value-aware objectives, in several continuous control tasks, while also enabling existing value-aware objectives to become performant.

In the offline RL context, this thesis takes a step back from model learning and applies value-awareness towards better data augmentation. Such data augmentation, when applied to model-based offline RL algorithms, allows for leveraging unseen states with low epistemic uncertainty that have previously not been reachable within the assumptions and limitations of model-based offline RL. Value-aware state augmentations are found to enable better performance on offline RL benchmarks compared to existing baselines and non-value-aware alternatives.

CHAPTER 1
INTRODUCTION AND BACKGROUND

Reinforcement Learning (RL), with its recent success stories in super-human game performance [8, 9, 10] and intersections with deep learning [11, 12], has emerged as a promising learning paradigm with applications now spread across almost every field of science – in computer science in the form of robotics [13], recommender systems [14], natural language processing [15]; in physics for simulating complex phenomena [16]; and in chemistry and medicine for drug discovery [17]. From the advent of the two most successful approaches for sequential decision making in Markov Decision Processes (MDPs) – *planning* and *reinforcement learning*, model-based RL has emerged as field at the intersection of these two paradigms [18].

The idea of utilizing a model for planning is well-established with roots in optimal control theory, more specifically model predictive control [19]. With the advent of deep neural networks for better function approximation, the paradigm of learning to model MDPs has gained popularity, with most recent advances learning deep neural network models [12] in conjunction with the objective of approximating the optimal control policy. While several such methods have focused on how to better utilize a learned parametrized model [6, 7, 20], the choice of objective for model learning – specifically the learning of a dynamics model for predicting state transitions – has largely been overlooked.

Maximum-likelihood estimation (MLE) is the prevalent choice for the model learning objective for learning a "correct" transition model of the underlying MDP. By definition, this objective is independent of the task or reward function – such independence has not received much scrutiny as model learning has been treated as an auxiliary objective to behavior optimization and planning [1]. Recently, this assumption has been questioned by the

[1] Note that model learning is indirectly influenced by reward functions in practice as the behavior policy's

paradigm of value-aware model learning [21, 5, 22, 23], that have emphasized the injection of reward or task specific information into model learning. This independence of model learning from task or reward information is part of the broader paradigm of task-agnostic methods that aim to study generalization to different evaluation tasks (instantiated by different reward functions), which necessitates such independence [24, 25]. Such methods typically do not assume changes in transition dynamcics across training and evaluation. Value-awareness in contrast falls under paradigm of task-specific or task-dependent learning that does not change the task or reward specification across training and evaluation but attempts to provide performance difference guarantees when the environment transition dynamics are changed. However, note that assuming a fixed task or reward definition does not imply that the reward function is known, it is still learned from data and the approximate reward function may not match the true reward function. While this thesis initially (in Chapter 2) studies performance difference guarantees for changes in both reward and transition dynamics, the latter part (Chapters 3 and 4) is closely aligned with value-awareness and it's paradigm of task-dependence given a fixed reward function.

The investigations of this thesis are in three parts, described as follows.

1. **Performance difference across environments.** We first study the generalization gap of training and evaluating on different environments, where performance difference bounds are presented that are a function of the difference in reward and transition dynamics, in two settings – Value Iteration and Fitted Q Iteration. A model-advantage based model performance difference lemma is presented that admits an upper bound objective for use in model-based RL in the next part. Finally, preliminary results also show the applicability of model performance difference as a metric for ranking candidate models given a target environment, without the need for an optimal policy in the target environment.

visited state distribution is influenced by the reward or task, which indirectly biases model learning to only observe data generated by this behavior policy.

2. **Value-awareness in Online Model-based RL.** This is the standard setting for model-based RL with online-environment interactions allowed throughout training. In this setting, we revisit value-aware model learning [5, 22] by deriving a novel value-aware model learning objective from bounding the absolute model performance difference of two models given a fixed policy, using the model performance difference lemma presented earlier. We propose an alternating optimization strategy for jointly training the model and value function approximate given a fixed policy and demonstrate that this leads to significant performance improvements for all value-aware model learning objectives on several challenging continuous control tasks.

3. **Value-awareness in Offline Model-based RL** This is the offline reinforcement learning setting [26] where an offline dataset of behavior trajectories (comprising states, actions and rewards) is available for training with no further online environment interactions throughout training. In this setting, value-aware objectives for model learning do not work as their approximation requires on-policy data. We propose a means of leveraging task or reward specificity by augmenting each batch from the offline dataset with unseen states obtained by perturbations along the value-gradient direction in the state space. Useful perturbed states are obtained by uncertainty filtering to keep those states whose uncertainty is not too high (within the generalization boundary) and not too low (too close to seen data to be useful). Performance improvements are measured in comparison to COMBO [1], a strong model-based offline RL baseline.

1.1 Thesis Statement

The central statement of this thesis is as follows.

Value-awareness can be leveraged for improved performance in both online and offline reinforcement learning. In the online setting, value-awareness in the model learning objective improves model-based RL performance over maximum-likelihood

based objectives. In the offline setting, value-awareness in data augmentation allows for improved Q-value estimation in a model-based offline RL algorithm.

Specifically, the thesis aims to validate the following claims

- **Online Model-based RL:** Value-aware model learning, despite its limited practical instantiations so far in challenging continuous control MBRL benchmarks [27], can be made performant by selection of an appropriate value-aware objective and by carefully addressing moving value targets in practical implementations with parametrized models and values.

- **Offline Model-based RL:** Model-based offline RL algorithms make use of a learned dynamics and reward model to inform Q-value and policy learning beyond the observed offline data. Current approaches do not make use of all possible states where model predictions generalize well i.e. have low epistemic uncertainty. We claim that value-aware augmentation of unseen states improves performance over a baseline model-based offline RL algorithm and is a better strategy than random direction based (non-value-aware) augmentation.

1.2 Outline

Chapter 2 lays the foundation for a novel value-aware model learning objective by first taking a step back and deriving the model performance difference of a policy across two different MDPs using model-advantage, the model equivalent of the well-known policy advantage function. It presents its roots in the generalization gap and extends the optimality gap bounds of Value Iteration and Fitted Q Iteration into generalization gap bounds. Finally, it presents preliminary results indicating the accuracy of proposed model advantage approximations in predicting relative distance of a set of candidate MDPs (which represent MDPs induced by an 'approximated' dynamics model) w.r.t. a target MDP (which represents the 'true' dynamics).

While Chapter 2 presented generalization gap bounds given any two environments (for training and evaluation respectively), Chapters 3 and 4 assume the model-based RL setting in which the learned model is treated as the training environment and true environment is also the evaluation environment.

Chapter 3 introduces a novel value-aware model learning objective by means of upper bounding the absolute model performance difference introduced in Chapter 2. It then identifies and proposes a remedy for the issue of stale value estimates in the practical implementation of value-aware model learning that has been so far holding back value-aware model learning in being practically performant. It then compares performance of the novel as well as existing value-aware model learning objectives in several continuous control tasks.

Chapter 4 changes gears by shifting from the online RL setting assumed so far in previous chapters into the offline RL setting. It identifies the limits of unseen state visitation in offline RL algorithms and proposes a novel strategy to find unseen states with low epistemic uncertainty in order to improve Q-learning and consequently, offline learning performance. A Q-value gradient based state augmentation strategy is investigated and found to improve performance over several baselines and ablations in an offline RL benchmark.

CHAPTER 2

MODEL-ADVANTAGE AND GENERALIZATION GAP

2.1 Introduction

In this chapter, we approach the model learning task from the perspective that the learned model will ultimately be applied to perform policy evaluation. We take a step back from the problem of learning a model and study the general scenario of two MDPs that differ only in their transition dynamics and reward distributions. Specifically, we derive an upper bound on the performance difference of a fixed policy across two such MDPs – which we refer to as *model performance difference* – and provide ways to approximate this upper bound in practice. We then relate model performance difference to the generalization gap and optimality gap in RL and study the effect of using a sub-optimal policy in practice for ranking multiple candidate models given a true model. The need to upper bound model performance difference is inspired by works that attempt to quantify the *generalization gap* in reinforcement learning [28]. Later in Chapter 3, we delve into the specific setting where one MDP represents the MDP induced by a learned dynamics (and reward) model and the other MDP represents the unknown true MDP with the true transition dynamics model.

In order to study the model performance difference of a policy across two MDPs and how it can be upper bounded, we introduce *model-advantage* [29, 30] – a quantity similar to the well-known advantage function in RL. The standard advantage function – which we refer to as policy-advantage – evaluates the *advantage* of playing a particular action as opposed to the action of a reference policy. Similarly, we define model-advantage as the advantage of transitioning to a state as opposed to transitioning according to an MDP M, while acting according to some policy. Specifically, we are interested in the expected advantage of transitioning according to one MDP with respect to another one as reference,

which allows us to evaluate the effectiveness of using one model in lieu of the other – much like how policy-advantage helps compare two different policies.

After introducing model-advantage and its place in the model performance difference lemma (Lemma 4), this work [29] studies the role of model performance difference in the generalization gap in reinforcement learning [28] and the optimality gap between an optimal and approximated policy in Section 2.5.1.

2.2 Related Work

Generalization in RL. Studying and benchmarking generalization properties of RL agents via large-scale experiments has been the focus of many works in the recent years [31, 32, 33, 34, 35, 36, 37, 38]. Additionally, [39, 40] study generalization properties of RL agents w.r.t. changes in state representations; the latter work derives a lemma comparable to Lemma 4 for policies that are lipschitz continuous over a set of state-representations. Importantly, [28] formally define generalization gap which we adopt in this work (see Eq. (2.8)) They give formal bounds on this gap in the setting of *reparametrizable RL* while making additional assumptions like Lipschitz continuity on value functions. While we do not require any such assumptions, it is important to note that it is not possible to obtain tighter bounds without such assumptions. Going beyond evaluating generalization, other works seek to learn robust policies by drawing inspiration from techniques employed in the supervised learning literature – for *e.g.* L_2 and entropic regularization [41], data augmentation [42] and constraints like invariance or robustness to noise on the learnt state-representation [43, 44, 40].

2.3 Preliminaries

Markov Decision Processes. In this work, we consider discrete-time infinite-horizon RL problems characterized by Markov Decision Processes (MDPs) M defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{P}_0, \gamma)$. Here, $\mathcal{S}$ is the state space, $\mathcal{A}$, the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \Delta(\mathcal{S})$

the transition probabilities or *dynamics*, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [0, R_{max}]$ the reward function, $\mathcal{P}_0$ the starting state distribution and finally, γ the discount factor. The goal is to find the *optimal* policy $\pi^\star \in \Pi$ that maximizes the (discounted) total return $J : \pi \to \mathbb{R}$ *i.e.*

$$\pi^\star = \underset{\pi \in \Pi}{\operatorname{argmax}}\, J(\pi),\ J(\pi) = \mathbb{E}_{\rho_\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)\right] \tag{2.1}$$

where ρ_π is the distribution of trajectories $(s_0, a_0, s_1, \dots)$, $s_0 \sim \mathcal{P}_0$, when acting according to policy π. The state-action value or the Q-function and the value function under policy π are given by

$$Q^\pi(s,a) = \mathbb{E}_{\rho_\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \mid \pi, s_0 = s, a_0 = a\right] \quad V^\pi(s) = \mathbb{E}_{\rho_\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \mid \pi, s_0 = s\right]$$

Observe that the value function (and similarly, the Q-function) obeys the recursively defined *Bellman* equation *i.e.*

$$V^\pi(s) = \mathcal{T}^\pi V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)}\left[\mathcal{R}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)} V^\pi(s')\right]$$

where $\mathcal{T} : \mathbb{R} \to \mathbb{R}$ is the *Bellman operator*. Additionally, note that $V^\pi(s) = \mathbb{E}_{a \sim \pi} Q^\pi(s,a)$ and $J(\pi) = \mathbb{E}_{s \sim \mathcal{P}_0}[V^\pi(s)]$. A more useful version of value function (and therefore the objective in Eq. (2.1)) is obtained by defining the *future state distribution* $P^\pi_{s,t}(s') = \Pr(s_t = s' \mid \pi, s_0 = s)$ and γ-discounted stationary state distribution $d_{s,\pi}(s') = (1-\gamma)\sum_t^\infty \gamma^t P^\pi_{s,t}(s')$. Using these definitions, we can now write the value function as:

$$V^\pi(s_0) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{a_t, s_t \sim \pi P_{s_0,t}}[\mathcal{R}(s_t, a_t)] = \mathbb{E}_{a, s \sim \pi d_{s_0,\pi}}[\mathcal{R}(s,a)] \tag{2.2}$$

Advantage. Another important quantity that is computed using the value function(s) is *advantage* function defined as $\mathbb{A}^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$. Intuitively, it measures the utility of taking action a instead of continuing to act under the policy π. Importantly,

advantage is closely related to the RL objective in Eq. (2.1) through the following lemma by Kakade and Langford [45].

Lemma 1. *(Performance Difference Lemma) Given any two policies* $\pi, \pi' \in \Pi$ *we have:*

$$J(\pi) = J(\pi') + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{s_0,\pi}}\mathbb{E}_{a\sim\pi}[\mathbb{A}^{\pi'}(s,a)] \tag{2.3}$$

Owing to this lemma, many RL frameworks optimize $\mathbb{E}_{s\sim d_{s_0,\pi}}\mathbb{E}_{a\sim\pi}[\mathbb{A}^{\pi'}(s,a)]$ w.r.t. an arbitrary fixed policy π' – a surrogate objective to the ones defined in Eq. (2.1) or Eq. (3.1). For the rest of the paper, we will refer to this advantage as *policy-advantage* in order to differentiate it from *model-advantage* which we introduce next.

2.4 Model-Advantage

In this section, we formally introduce *model-advantage* and definitions associated with it. Recall that $\mathbb{A}^{\pi}(s,a)$ *i.e.* policy-advantage defined over states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$ measures the *utility* of taking action a as opposed to acting according to policy π. However, unlike policy-advantage that measures the difference in utility of taking an action, we are interested in knowing the utility difference of transitioning to a particular state, while following a single policy. Specifically, *model-advantage* denoted by $\boldsymbol{A}^{\pi}_{M}(s,s')$ compares the utility of moving to state s' and following the trajectory governed by model M as opposed to doing it from state s; both under policy π. By definition, model-advantage is local – it is a quantity dependent on a given state s' and next state s'. Comparing two models however requires a global comparison over multiple states and next states, which we will see soon in Lemma 4.

We define the model-dependent value function as follows, where we make explicit the MDP M (and hence, the transition function $\mathcal{P}_M$) used to generate trajectories.

$$V^{\pi}_{M}(s) = \mathbb{E}_{\rho^{\pi}_{M}}\left[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}_M(s_t,a_t) \mid \pi, M, s_0 = s\right]$$

Here, ρ_π is the distribution of trajectories $(s_0, a_0, s_1, \dots)$, $s_0 \sim \mathcal{P}_0$, when acting according to policy π. Now, the intuition of model-advantage [1] is given by:

$$\boldsymbol{A}^\pi_M(s, s') = \gamma \left[V^\pi_M(s') - \mathbb{E}_{s'' \sim P_M(s,\pi)} V^\pi_M(s'') \right] \tag{2.4}$$

Analogous to policy-advantage that compares different policies in the same environment, model-advantage helps compare the same policy acting in two different environments. For such a comparison, we need to look at the quantity $\mathbb{E}_{s' \sim M'} \left[\boldsymbol{A}^\pi_M(s, s') \right]$ – the expected model-advantage (evaluated at π) when the next state s' is obtained from the MDP M'. We formalize this by the following *(model) performance difference* lemma. The proof resembles the proof by [45] and is provided in Section A.1.

Lemma 2. *(Model Performance Difference Lemma) Let M and M' be two different MDPs. Further, define $\mathcal{R}^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)}[\mathcal{R}(s, a)]$, $\mathcal{R}^\pi_\epsilon(s) = \mathcal{R}^\pi_M(s) - \mathcal{R}^\pi_{M'}(s)$ and $J(\pi) = \mathbb{E}_{s \sim \mathcal{P}_0}[V^\pi(s)]$. For any policy $\pi \in \Pi$ we have:*

$$\begin{aligned} J_M(\pi) - J_{M'}(\pi) &= \mathbb{E}_{s \sim d_{M,\pi}} \left[\mathcal{R}_\epsilon(s) \right] \\ &+ \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_{M,\pi}} \mathbb{E}_{s' \sim \mathcal{P}_M(s'|s,\pi)} \left[\boldsymbol{A}^\pi_{M'}(s, s') \right] \end{aligned} \tag{2.5}$$

Here, we use a model-dependent stationary state distribution $d_{M,\pi}(s)$ where the dynamics $\mathcal{P}_M$ are used, assuming a start state distribution $\mathcal{P}_0$. Compared to its policy counterpart the (model) performance difference lemma involves an additional reward error term $\mathbb{E}_{d_{M,\pi}} \left[\mathcal{R}_\epsilon \right]$ that vanishes when the two MDPs differ only in the transition probabilities.

We can also use the Bellman evaluation operator for policy π to obtain an equivalent for-

[1] A $Q(s, s')$ function can also be defined; however, it requires additional formalism not necessary for our exposition. See concurrent work [46] for a detailed discussion.

malization (see Section A.1):

$$J_M(\pi) = J_{M'}(\pi) + \frac{1}{1-\gamma}\mathbb{E}_{d_{M,\pi}}\underbrace{\left[\mathcal{T}_M^\pi V_M^\pi - \mathcal{T}_{M'}^\pi V_M^\pi\right]}_{\text{deviation error}} \tag{2.6}$$

The term $\mathcal{T}_M^\pi V_M^\pi - \mathcal{T}_{M'}^\pi V_M^\pi$, which we denote by $\delta_{M,M'}^\pi$ [2] represents the *deviation* in the value function when acted upon by Bellman operators corresponding to two different MDPs. This term is exactly equal to model-advantage when the reward functions of the two MDPs are the same. We can form an upper bound on the extrinsic error as:

$$\begin{aligned} J_M(\pi) - J_{M'}(\pi) &\leq \frac{1}{1-\gamma}\left\|\mathcal{T}_M^\pi V^\pi - \mathcal{T}_{M'}^\pi V^\pi\right\|_\infty \\ &\leq \frac{1}{1-\gamma}\left[\epsilon_R + \gamma\epsilon_P\left\|V^\pi\right\|_\infty\right] \end{aligned} \tag{2.7}$$

where the rewards and the dynamics themselves are individually bounded *i.e.* $\max_s \max_a |R_M(s,a) - R'_{M'}(s,a)| \leq \epsilon_R$ and $\max_s \max_a \|P_M(s,a) - P_{M'}(s,a)\|_1 \leq \epsilon_P$. Of course, note that $\|V^\pi\|_\infty$ is trivially bounded by $\frac{1}{1-\gamma}R_{\max}$, assuming rewards are bounded by $R_{\max}$.

2.5 Generalization in RL

An RL problem is characterized by an MDP M and is considered solved when a policy $\pi \in \Pi$ that maximizes the expected (discounted) return is found. However, in practice, the RL agent may then be deployed in a slightly different environment characterized by an MDP M'. Therefore, we are interested in how well an RL agent performs in an unseen environment – in other words, its ability to *generalize*.

[2] Optionally, when using the optimality operators corresponding to MDPs M and M', we drop the π and denote the deviation error as $\delta_{M,M'}(V(s))$.

2.5.1 Generalization Gap.

Let $J(\pi) := \mathbb{E}[\sum_t \mathcal{R}(s_t, a_t)]$ denote the cost function, where the stochasticity is due to the policy and transition dynamics; let $\hat{J}_n(\pi)$ denote its empirical estimate with n samples. Given an RL agent trained in MDP M with finite data, we are interested in its performance in a different MDP M'. We can formally write this *generalization gap* as:

$$\begin{aligned} \Phi &= \left|\hat{J}_n(\pi) - J'(\pi)\right| \\ &\leq \underbrace{\left|\hat{J}_n(\pi) - J(\pi)\right|}_{\text{intrinsic error}} + \underbrace{\left|J(\pi) - J'(\pi)\right|}_{\text{extrinsic error}} \end{aligned} \tag{2.8}$$

The generalization gap can be bounded with two different sources of error as indicated in Eq. (2.8). Following [28], we call them *intrinsic* error and *extrinsic* error to denote the error due to learning from finite samples and the error due to mismatch in training and deployment environments. The intrinsic error decreases, typically as $\mathcal{O}(1/\sqrt{n})$, with more samples; this is well-studied in RL literature [47, 48, 49]. The extrinsic error on the other hand is an artifact of training and deploying the RL agent in different environments and therefore, cannot be avoided.

When does π generalize? Observe that the extrinsic error is nothing but the difference in performance due to model mismatch. From Lemma 4, we know that this is equal to the model-advantange, allowing us to both estimate and bound this error term. In other words, if the model-advantage is bounded by ϵ (see Eq. (2.7)) *i.e.*

$$|J_M(\pi_M) - J_{M'}(\pi_M)| \leq \epsilon$$

we can say that π_M, the policy learnt with experiences from MDP M achieves similar performance in the target MDP M'. As model of the "test" environment is not known, a reasonable estimate of the model-advantage can be obtained with enough samples – allowing one to *predict* the extent to which the policy performs in the novel environment.

How good is π really? However, note that the above generalization gap still does not provide the complete picture. Ideally, we would like the policy π_M to have performance comparable to $\pi^*_{M'}$, the optimal policy in the target MDP M', which we quantify with the *optimality gap*.

$$\underbrace{|J_{M'}(\pi_M) - J_{M'}(\pi^*_{M'})|}_{\text{optimality gap}}$$
$$\leq \underbrace{|J_{M'}(\pi_M) - J_M(\pi_M)|}_{\text{term-I}} + \underbrace{|J_M(\pi_M) - J_{M'}(\pi^*_{M'})|}_{\text{term-II}} \tag{2.9}$$

It is easy to see that term-I is nothing but the extrinsic error in Eq. (2.8) and is related to the model-advantage (evaluated under policy π_M) through Lemma 4. Intuitively, this this term corresponds to the cost of transfering π_M learnt in the seen MDP M to the novel MDP M'.

In the rest of this section, we will bound term-II for specific instantiations of obtaining policy π_M – specifically, Value Iteration (VI) and Fitted Q-Iteration (FQI), with the former being a model-based and the latter, a model-free approach to solve MDPs.

2.5.2 Generalization with Value Iteration and Fitted Q iteration

Value Iteration. When the dynamics and the reward functions are known, Value Iteration (VI) and its variants are often employed to arrive at the optimal policy. VI is an iterative algorithm that applies the Bellman optimality operator $\mathcal{T}_M$ [3] at each step *i.e.* $V^{(n)} = \mathcal{T}_M V^{(n-1)}$. The obtained iterates converge to V^*_M , the value function of $\pi^\star_M$, asymptotically as $\mathcal{T}_M$ is a contraction in the infinity-norm. We can bound the difference in value from training on another MDP with the following theorem:

Theorem 3. *Let M, M' be two MDPs s.t. $\max_s \max_a |\mathcal{R}_M(s,a) - \mathcal{R}_{M'}(s,a)| \leq \epsilon_R$ and $\max_s \max_a \|p_M(s,a) - p_{M'}(s,a)\|_1 \leq \epsilon_P$. Let π_{n+1} be the policy obtained after n VI itera-*

[3] Assume optimality operator by default if policy is not explicitly defined

tions on MDP M *and let* $\left\|V_M^{(n+1)} - V_M^{(n)}\right\|_\infty \leq \epsilon^{(n)}$ *Then we have,*

$$\|V_{M'}^{\pi_{n+1}} - V_{M'}^{\star}\|_\infty \leq \frac{1}{1-\gamma}\left[\gamma\epsilon^{(n)} + 2\epsilon_R + \frac{2\epsilon_P R_{max}}{1-\gamma}\right]$$

We provide a similar bound when following a Fitted Q iteration (FQI) method, which is more effective when dealing with a large (or infinite) state space or unknown dynamics/reward functions. The statement and proof of the bound can be found in Section A.2.2.

2.5.3 Is My Simulator Good?

The fundamental bottleneck preventing the usage of RL to train agents in the real-world is *exploration*. As the model of the environment is not available, finding the optimal policy not only requires exploring a large search space but is also *costly*. A common strategy to avoid this issue is to learn a *coarse* policy using a simulator and then *fine-tune* it upon deployment. But how does one build the simulator in the first place? It either requires considerable domain expertise or a large number of samples from the real-world, and we must know a priori that the simulator can *express* all variations feasible in the real world. We are left with the question: *Given a set of simulators, which one is likely to "generalize" best to the real-world?*

Predicting Generalization with Model-Advantage. Recall that (model) performance difference Lemma 4 allows us to compare two models given a policy. Given M, the simulator MDP and M', the real-world MDP and π_{exp}, an expert policy for M', we can write:

$$\left|J_M(\pi_{\text{exp}}) - J_{M'}(\pi_{\text{exp}})\right| = \left|\mathbb{E}_{(s,s')\sim M}\left[\boldsymbol{A}_{M'}^{\pi_{\text{exp}}}(s,s')\right]\right|$$

To compute the advantage function $\boldsymbol{A}_{M'}^{\pi_{\text{exp}}}$, the expert policy has to be executed in the real-world. Alternately, such an "expert" policy and its corresponding value function in MDP M' can be learnt by collecting a finite set of data from the real-world – for instance, by

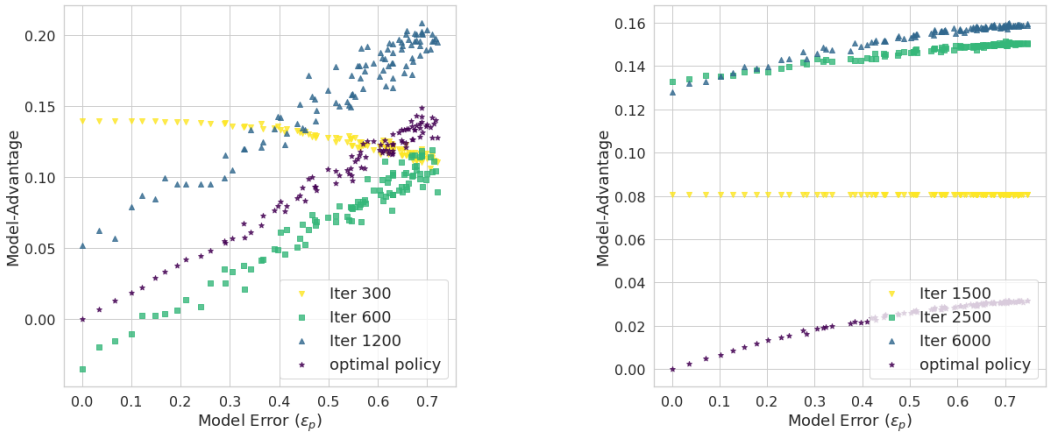


Figure 2.1: Generalization gap on sub-optimal real world policies on FrozenLake environments (FrozenLake 4x4 on the left and 8x8 on the right). Even a sub-optimal policy obtained with minimal interactions with the real-world is sufficient to use model-advantage and compare different *training* environments or simulators.

running FQI on the collected dataset [4]. After paying this one-time cost of interacting with the real-world, the model-advantage can be estimated in an inexpensive manner for every simulator with finite samples. In our experiments we will show that using an approximately optimal policy is sufficient for comparing model advantage across simulators, alleviating the need for an expert policy.

Grid World Experiments. We consider the toy environment of FrozenLake available as part of OpenAI Gym [5] to illustrate the effectiveness of the proposed model-advantage term in evaluating simulators. We treat the original setting as M, the real-world MDP and then, corrupt the transition dynamics with various levels of random noise to obtain a set of "simulators" $\{M'_i\}_{i=1}^{K}$. We then run DQN [50] that uses a single hidden-layer MLP to learn Q-values in M and obtain a sub-optimal "expert" Q-function by not running the training to completion. As can be seen from Figure 2.1, we see that even for Q-values far from optimal, the model-advantage increases with increasing modeling error ϵ_P – the same trend exhibited by the optimal Q-function in the real-world.

[4] Note that convergence to optimal value-function is guaranteed only if the distribution used to sample states is exploratory. While the "tax" of exploration cannot be waived, the hope here is that a small amount if data is sufficient to learn a sub-optimal expert.

[5] https://gym.openai.com/

2.6 Conclusion

In this work, we proposed *model-advantage* that helps compare two models, similar to policy advantage that can be used to compare two policies. We presented the theoretical connections of model performance difference with generalization in RL and the optimality gap. Further, presented toy experiments to show that even a sub-optimal policy, learnt from minimal interactions with the target environment, can help identify the training environment that facilitates maximum generalization. In the next chapter, we will employ an upper bound on the model performance difference to produce a novel task-aware model learning objective for model-based RL.

CHAPTER 3
MODEL-ADVANTAGE OPTIMIZATION FOR MODEL-BASED REINFORCEMENT LEARNING

3.1 Introduction

Chapter 2 introduced model-advantage as a quantity useful for characterizing the model performance difference of a policy across two MDPs differing in their transition dynamics and/or rewards. In this chapter, we present our recent work [51] where we switch to the more specific setting of model-based (online) reinforcement learning – where one MDP represents the learned dynamics model and the other is the true underlying MDP with unknown dynamics and reward in the reinforcement learning problem definition. In this setting, model performance difference is an important quantity that leads to a recently introduced task-aware notion of model 'correctness' – *value equivalence* [22]. The learned dynamics model is said to be value-equivalent w.r.t the true dynamics model for a fixed policy if the model performance difference for that policy is zero.

Before introducing our model learning objective, we must understand the most commonly used model learning objective of maximum likelihood estimation (MLE). MLE minimizes the KL divergence between predicted and observed next state distributions. A drawback of this approach is the issue of an objective mismatch between the model-learning objective and the ultimate purpose of using the model to find an optimal policy [52, 53]. More recent research in MBRL has focused on efforts to overcome these shortcomings – including optimizing for auxiliary objectives [54, 55, 56], augmenting model-learning with exploration strategies [7, 57], meta-learning to closely intertwine the two objectives [58] and introducing inductive biases to the model-learning objective [59]. However, these MBRL approaches still employ MLE to learn the dynamics as an intermediate step.

In this work, we revisit Value Aware Model Learning (VAML) [21, 5], an alternate objective for learning dynamics. Instead of maximizing the likelihood of the future state given the current state and action, VAML seeks to minimize the squared error of the expected value of next state predicted by the dynamics model from the value of the observed next state in data. This objective is appealing as it factors in the *utility* of the model in finding the optimal policy (through the value function) and does not require *exact* prediction of observed trajectories. It is also amenable to Dyna-style [60] deep model-based RL algorithms (e.g. [6]) where the critic in an actor-critic agent can be used for computing the value estimates required by this unique model learning objective. Value-aware model-based RL has recently found strong theoretical backing in the form of guarantees of convergence [21, 5], the value-equivalence principle [22] and use optimistic model-based RL for regret minimization [23]. The MuZero algorithm [61] is also an example of a value-aware (or 'value-equivalent') model-based approach for solving discrete action environments while leveraging Monte-Carlo tree search.

Despite the intuitive and theoretical appeal of existing value-aware model learning objectives, their utility has thus far remained under-explored beyond toy settings within the domain of continuous control. In our experiments, we find that existing value-aware objectives perform poorly with recent Dyna-style model-based RL frameworks, independently replicating recent negative results [27]. In this work, we revisit value-aware model learning from a novel perspective and bridge the gap in theory and practice in challenging continuous control applications. First, we derive an upper bound on the expected model performance difference of two MDPs or models for a fixed policy, using triangle inequality on the $L1$-norm. In contrast, prior value-aware approaches [5], though inspired by the minimization of (normed) model performance difference, do not upper bound the model performance difference with their use of the $L2$-norm, which may explain their inferior performance compared to our proposed objective in most of our continuous control tasks. Second, we call to attention the issue of stale value estimates in the naive application of

value-aware losses in the dyna-style model-based RL algorithmic framework. Upon correcting for the stale value estimates by intermittently fitting the value network during model learning, we obtain significant performance improvements on the more challenging continuous control environments. The resulting general purpose dyna-style MBRL algorithm is, to the best of our knowledge, the first known practical deployment of value-aware objectives in challenging continuous control robotic simulation environments [62].

We empirically test our proposed algorithm and novel upper bound on two recent dyna-style MBRL algorithms – SLBO [6] and MBPO [7]. We find that our algorithm successfully bridges the gap in theory and practice by reaching near-matching performance w.r.t. MLE-based baselines in most continuous control simulation tasks and outperforming them in some others. We hope that these encouraging results spur wider interest in the community leading to both adoption and further study of value-aware methods for practical model-based RL.

3.2 Related Work

MLE-based MBRL. Maximum likelihood estimation (MLE) is the the most prevalent and straight-forward objective for model learning in a model-based reinforcement learning framework [60, 63]. Broadly, MLE-based methods seek to construct a model that mimics the dynamics as accurately as possible. Unlike our proposed value-aware objective, minimizing dynamics error minimizes a looser upper bound on the model performance difference [21]. Therefore, multiple MBRL approaches that minimize various definitions of dynamics error have been proposed. For instance, [48, 64, 65] use naïve empirical frequencies. More sophisticated approaches use function approximators and minimize various statistical distances – *e.g.* KL [66], total-variation [7] or Wasserstein distances [67].

Non-MLE based MBRL. Methods that inform model learning via the value function, reward or policy have recently gained popularity [68, 69, 61, 70]. In particular, [71], [61], and [72] explore learning dynamics implicitly using the estimated value for a given state,

and using a monte-carlo tree search algorithm to plan with this learned model. However, these works learn a joint model for directly estimating future values and actions (policy) without any explicit future predictions in the state space. In contrast, we focus on the class of MBRL methods that explicitly make predictions in the state space, allowing for simple adaptations on top of of well-known MBRL frameworks e.g. Dyna-style algorithms [60].

Value-aware Model Learning. [21], [5] and [22] are the closest prior works that study the theoretical properties of value-aware objectives. They present experimental results only on toy settings (*e.g.* with ~25 states and cart-pole environments) and their algorithms do not effectively scale to challenging environments. [27] demonstrate *negative* empirical results for their practical instantiation of value aware model learning [5] with an actor-critic learner in continuous control environments such as `Pusher-v2` and `InvertedPendulum-v2`. Their algorithm employs a sparse model update, occurring only once every few policy and critic updates – different from our algorithm that builds on top of a standard Dyna-style MBRL algorithm with multiple model updates in between every sequence of policy and critic updates. Our key insight for modifying this Dyna-style algorithm takes advantage of the multiple model updates while preventing stale value estimates as a result of changing model parameters (for details, see Algorithm 2 and Approach section).

3.3 Preliminaries

Markov Decision Processes.

In this work, we consider a discrete-time infinite-horizon RL problem characterized by Markov Decision Processes (MDPs) M defined as $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{P}_0, \gamma)$. Here, $\mathcal{S}$ is the state space, $\mathcal{A}$, the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ the transition probabilities or *dynamics*, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [0, R_{max}]$ the reward function, $\mathcal{P}_0$ the starting state distribution and finally, γ the discount factor. The goal is to find the *optimal* policy $\pi^\star \in \Pi$ that maximizes the (discounted) total return $J : \pi \to \mathbb{R}$ *i.e.* $J(\pi) = \mathbb{E}_{\rho^\pi}\left[\sum_t \mathcal{R}(s_t, a_t)\right]$. where ρ^π is the distribution of trajectories $(s_0, a_0, s_1, \dots)$, $s_0 \sim \mathcal{P}_0$, when acting according to policy π.

Algorithm 1: Model Based Reinforcement Learning (MBRL)

Randomly initialize policy π, model M
Initialize replay buffer $\mathcal{D} \leftarrow \varnothing$
for n_{outer} *iterations* **do**
    // model update step
    for K_{model} *updates* **do**
        $\mathcal{D} \leftarrow \mathcal{D} \bigcup \{n$ samples from true environment M' collected by $\pi\}$
        Update M' using model-learning objective on $\mathcal{D}$ // e.g. $\mathbb{E}_{\mathcal{D}}\left[KL(\hat{s}\|s)\right]$
    end
    // policy update step
    for K_{policy} *updates* **do**
        $\mathcal{D}' \leftarrow \{$Samples collected in learned model M using $\pi.\}$
        Update π using policy learning method // e.g. TRPO [73]
    end
end

The Q-function and the value function under policy π are given by $Q^{\pi}(s,a) = \mathbb{E}_{\rho^{\pi}}\left[\sum_t \mathcal{R}(s_t,a_t) \mid \pi, s_0 = s, a_0 = a\right]$ and $V^{\pi}(s) = \mathbb{E}_{\rho^{\pi}}\left[\sum_t \mathcal{R}(s_t,a_t) \mid \pi, s_0 = s\right]$ respectively. A more useful version of the value function, and therefore the RL objective itself, is obtained by defining the *future state distribution* $P^{\pi}_{s,t}(s') = \Pr(s_t = s' \mid \pi, s_0 = s)$ and γ-discounted stationary state distribution $d_{s,\pi}(s') = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P^{\pi}_{s,t}(s')$, where we drop the dependency on start state distribution when it is implicitly assumed to be known and fixed. Using these definitions, we write the value function as:

$$\begin{aligned} V^{\pi}(s_0) &= \sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{(a_t,s_t)\sim(\pi,P^{\pi}_{s_0,t})}[\mathcal{R}(s_t,a_t)] \\ &= \frac{1}{1-\gamma}\mathbb{E}_{a,s\sim\pi,d_{s_0,\pi}}[\mathcal{R}(s,a)] \end{aligned} \tag{3.1}$$

Model-Advantage and MBRL.

MBRL algorithms work by iteratively learning an approximate model and then deriving an optimal policy from this model either by planning or learning a separate policy with imagined experience. The latter case refers to the family of Dyna-style MBRL algorithms

[60] that we adopt in this work – see Algorithm 1 for a representative algorithm from this family. Model-advantage[1], proposed by [30, 29], is a key quantity that can be used to compare the utility of transitioning according to the approximate model M as opposed to the true model M'. Specifically, *model-advantage* denoted by $\boldsymbol{A}^{\pi}_{M}(s, s')$ compares the utility of moving to state s' and thereafter following the trajectory governed by model M as opposed to following M from state s itself; while acting according to policy π. The following definition in Eq. (3.2) captures this intuition. We denote model-dependent quantities with the model as subscript: transition probability distribution of M is denoted by P_M and value function as V^{π}_{M}.

$$\boldsymbol{A}^{\pi}_{M}(s, s') := \gamma \left[V^{\pi}_{M}(s') - \mathbb{E}_{s'' \sim P_M(s,\pi)} V^{\pi}_{M}(s'') \right] \tag{3.2}$$

Here, V^{π}_{M} is the model-dependent value function defined as:

$$V^{\pi}_{M}(s) = \mathbb{E}_{\rho^{\pi}_{M}} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_M(s_t, a_t) \mid \pi, M, s_0 = s \right]$$

We are now ready to restate the well-known *simulation lemma* [74] in an alternate form, such that it quantifies the model performance difference using model-advantage.

Lemma 4. *(Simulation Lemma) Let M and M' be two different MDPs. Further, define $\mathcal{R}^{\pi}_{M}(s) = \mathbb{E}_{a \sim \pi(\cdot|s)}[\mathcal{R}_M(s, a)]$ and $\mathcal{R}^{\pi}_{\delta_{M,M'}}(s) = \mathcal{R}^{\pi}_{M}(s) - \mathcal{R}^{\pi}_{M'}(s)$. For a policy $\pi \in \Pi$ we have:*

$$J_M(\pi) = J_{M'}(\pi) + \frac{1}{1-\gamma} \underbrace{\mathbb{E}_{s \sim d_{M,\pi}} \mathbb{E}_{s' \sim \mathcal{P}_M(s'|s,\pi)} \left[\boldsymbol{A}^{\pi}_{M'}(s, s') \right]}_{\text{expected model-advantage}} + \underbrace{\mathbb{E}_{s \sim d_{M,\pi}} [\mathcal{R}^{\pi}_{\delta_{M,M'}}(s)]}_{\text{reward difference}} \tag{3.3}$$

Here, we use a model-dependent stationary state distribution $d_{M,\pi}(s)$ (dropping the dependence on start state distribution) where the dynamics $\mathcal{P}_M$ are used. To simplify notation, we will write the expected model advantage term as $\mathbb{E}_{(s,s') \sim M} \left[\boldsymbol{A}^{\pi}_{M'}(s, s') \right]$ or simply

[1] Name follows policy-advantage that compares the utility of two actions [45]

$\mathbb{E}_M\left[\boldsymbol{A}^{\pi}_{M'}\right]$. A slightly different form of Lemma 4 can be obtained by explicitly indicating the model in the Bellman operator as follows.

$$\mathcal{T}^{\pi}_{M} V(s) := \mathbb{E}_{a\sim\pi}\left[\mathcal{R}_M(s,a) + \gamma \mathbb{E}_{s'\sim P_M(s'|s,a)}[V(s')]\right] \tag{3.4}$$

This leads to the following corollary that provides an alternate view of the model-advantage term (see Appendix for the proof).

Corollary 5. *Let M and M' be two different MDPs. For any policy $\pi \in \Pi$ we have:*

$$J_M(\pi) = J_{M'}(\pi) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{M,\pi}} \underbrace{\left[\mathcal{T}^{\pi}_{M} V^{\pi}_{M'}(s) - \mathcal{T}^{\pi}_{M'} V^{\pi}_{M'}(s)\right]}_{\text{deviation error}} \tag{3.5}$$

Note that the term on the right that includes the deviation error is exactly equal to model-advantage when the reward functions of the two MDPs are identical[2]. Therefore, setting aside the reward-error term in Lemma 4, model advantage can be viewed as the deviation resulting from acting according to different MDPs. Minimizing the deviation error is the basis of the objective proposed in Value-Aware Model Learning (VAML) [21, 5]. More recent work [22] shows that various MBRL methods can be thought of as minimizing the deviation error – a direct consequence of the close relationship between the deviation error and the model performance difference.

3.4 Approach: Value-Aware Objective via Model Advantage Upper Bound

In this section, we first introduce the basis of our proposed value-aware model learning objective that minimizes the performance difference of a policy in the true vs approximate model. From Eqn. Equation 3.3, this translates to optimization of expected model advantage $\mathbb{E}_M\left[\boldsymbol{A}^{\pi}_{M'}\right]$ using on-policy samples from the true environment, for which we show an empirical estimation strategy with samples from the true MDP and gradient based updates

[2] A common assumption for MBRL works proposing to learn dynamics (*e.g.* [6]). We make this assumption as well.

for a parametrized dynamics model. We then derive a novel upper bound on expected model advantage and introduce our novel general purpose algorithm for value-aware model-based RL.

3.4.1 Direct Model Advantage Optimization

For the model-learning step in MBRL, we are interested in an objective for finding model parameters ϕ corresponding to the dynamics of the approximate MDP *i.e.* $\mathcal{P}_\phi(\cdot \mid s, a)$ that eventually lead to the learning of an optimal policy in the true MDP $M^\star$. By looking at the model-advantage version of the simulation lemma (*i.e.* Lemma 4), a natural choice for a loss function is the absolute value of the expected model advantage. For brevity, we replace the expectation over $s_t \sim P^\pi_{M^\star,t}, s_{t+1} \sim P^\pi_{M^\star,t+1}$ with $\tilde{d}_t$.

$$\begin{aligned}\mathcal{L}_1(\phi) &:= \left|J_{M_\phi}(\pi) - J_{M^\star}(\pi)\right| \\ &= \left|\sum_{t=0}^{\infty}\gamma^t \mathop{\mathbb{E}}_{\tilde{d}_t}\left[\gamma V^\pi_{M_\phi}(s_{t+1}) - \gamma \mathop{\mathbb{E}}_{s''\sim\mathcal{P}^\pi_{M_\phi}(s_t,\pi)}\left[V^\pi_{M_\phi}(s'')\right]\right]\right| \end{aligned} \tag{3.6}$$

This objective can be empirically estimated via a dataset $\mathcal{D}_m$ of trajectories (state-action sequences) sampled from the true MDP $M^\star$. We omit the input of actions from π in $\mathcal{P}^\pi_{M_\phi}$.

$$\widehat{\mathcal{L}_1}(\phi) = \left|\frac{1}{m}\sum_{\substack{(s_0,a_0,\ldots,\\ a_{T-1},s_T)\epsilon\mathcal{D}_m}}\sum_{t=0}^{T-1}\gamma^t\left(V^\pi_{M_\phi}(s_{t+1}) - \mathop{\mathbb{E}}_{s'\sim P^\pi_{M_\phi}(s_t)}\left[V^\pi_{M_\phi}(s')\right]\right)\right| \tag{3.7}$$

In Eqn. Equation 3.7, the value function $V^\pi_{M_\phi}$ has a complex dependency on parameters ϕ which is hard to optimize. In practice, this value can be estimated in any Dyna-style [60] model-based RL algorithm with a parametrized value function (with parameters included in θ of policy π_θ i.e. as part of an actor-critic pair) for estimating this value. We estimate the value function without modeling its dependency on ϕ i.e. we replace $V^\pi_{M_\phi}$ with a learned value network V^{π_θ}, such that the V^{π_θ} is updated during the policy-update step of our algorithm (using imagined experience from M_ϕ) to match the true target $V^\pi_{M_\phi}$. This results in

a simple stochastic gradient update rule[3] for predictions made from $P^{\pi}_{M_\phi}(s_i)$. Finally, our empirical objective can be written as follows.

$$\widehat{\mathcal{L}_1}(\phi) = \left| \frac{1}{m} \sum_{\substack{(s_0, a_0, \ldots, \\ a_{T-1}, s_T) \in \mathcal{D}_m}} \sum_{t=0}^{T-1} \gamma^t \left(V^{\pi_\theta}(s_{t+1}) - \mathop{\mathbb{E}}_{s' \sim P^{\pi}_{M_\phi}(s_t)} \left[V^{\pi_\theta}(s') \right] \right) \right| \tag{3.8}$$

3.4.2 Model-Advantage Upper Bound

In practice, the objective in Eqn. Equation 3.8 is undesirable as it requires full length trajectory samples to compute the discounted sum and therefore, provides a sparse learning signal *i.e.* a single gradient update step from an entire trajectory. This limitation is overcome by further upper bounding Eq. (3.6) via the triangle inequality as shown below (with abbreviated notation).

$$\mathcal{L}_1(\phi) := \left| \sum_{t=0}^{\infty} \gamma^t \mathop{\mathbb{E}}_{M^\star, t} \left[\boldsymbol{A}^{\pi}_{M} \right] \right| \leq \underbrace{\sum_{t=0}^{\infty} \gamma^t \mathop{\mathbb{E}}_{M^\star, t} \left[\left| \boldsymbol{A}^{\pi}_{M} \right| \right]}_{=: \mathcal{L}^U_1(\phi)} \tag{3.9}$$

Observe that this form of the objective is now compatable with experiences *i.e.* (s, a, s', r) sampled from the true MDP $M^\star$ as opposed to ensure trajectories – thereby providing a denser learning signal despite being a proxy (upper bound) for the original objective. Later in Section 3.5.1 we demonstrate the practical benefits of the denser learning signal on discrete environments. We further make this objective amenable to minibatch training by replacing the the discounted sum over timesteps $\sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t}(\cdot)$ with the policy's discounted stationary state distribution $\mathbb{E}_{s \sim \rho_{\pi, M^\star}}(\cdot)$ – this is estimated empirically with a finite dataset of sampled experiences. Similar to Eq. (3.8), the empirical estimation version of the objec-

[3] Note that this objective can be optimized via gradient updates as long as the value function V^{π_θ} can be differentiated w.r.t. its inputs *i.e.* states, which is the case for neural networks.

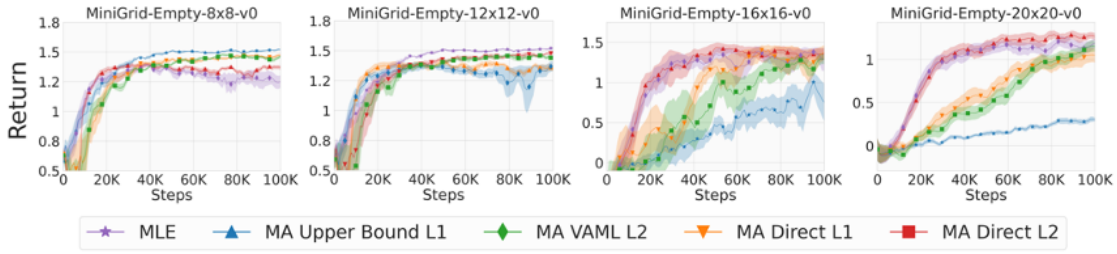


Figure 3.1: (Left) A sample of the `8x8` size gridworld environment from Gym MiniGrid [4]. (Right) Return curves over 20 random seeds on `MiniGrid Empty` environments with varying grid sizes, using 4 value aware methods and an MLE baseline. Increasing grid size negatively affects MLE performance most, while our proposed upper bound and VAML [5] are affected the least. The direct versions of L1 and L2 model-advantage based objectives (MA Direct L1 and MA Direct L2) are further slower to converge than MA Upper Bound L1 and MA VAML L2.

tive is as follows, where the summation is over trajectories $(s_t, a_t, s_{t+1}) \in \mathcal{D}_n$.

$$\widehat{\mathcal{L}_1^U}(\phi) = \frac{1}{n} \sum_{\substack{(s_t,a_t,\\ s_{t+1})\in\mathcal{D}_n}} \left(\left| V^{\pi_\theta}(s_{t+1}) - \mathop{\mathbb{E}}_{s'\sim p^{\pi}_{M_\phi}(s_t)} \left[V^{\pi_\theta}(s') \right] \right| \right) \tag{3.10}$$

Connection to VAML. Eqn. Equation 3.9 is similar to the L2 norm value-aware objective introduced in [21, 5]. In our framework, the VAML objective, $\mathcal{L}_2^{\text{VAML}}$, can be obtained by using the L2 norm in Eq. (3.9) *i.e.* $\mathcal{L}_2^{\text{VAML}} := \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{M^\star,t} \left[\left(\boldsymbol{A}_M^\pi \right)^2 \right]$. Imporantly, owing to the properties of L2 norm, $\mathcal{L}_2^{\text{VAML}}$ does *not* upper bound its corresponding L2 normed model advantage objective $\mathcal{L}_2$. We find in our experiments that $\mathcal{L}_1^U$ often has better or equal performance in conjunction with SLBO and MBPO, potentially hinting at the importance of this relationship with model-advantage.

3.4.3 General Algorithm for Value-aware Objectives

Value-aware objectives such as [5, 22] enjoy several theoretical benefits, but remain isolated from practical use beyond small, finite state toy MDPs within continuous control. We find that with a naive substitution of value-aware objectives in place of maximum likelihood (Fig. 3.4) in existing MBRL algorithms (Algorithm 2) worked well only for the easy continuous control environments, namely `Cartpole`, `Pendulum` and `Acrobot`. This

Algorithm 2: Value-Aware MBRL

Initialize $\theta = (\theta_p, \theta_v)$, the policy/value function parameters and model parameters ϕ randomly;
Initialize replay buffer $\mathcal{D}, \mathcal{D}', \mathcal{D}'' \leftarrow \varnothing$;
for K *iterations* **do**
    $\mathcal{D} \leftarrow \mathcal{D} \bigcup \{n$ samples from true environment $M^\star$ collected by $\pi_{\theta_p}\}$;
    `// model update step`
    for K_{model} *updates* **do**
        Update ϕ using value-aware model-learning objective on $\mathcal{D}$ (*e.g.* Equation 3.10) ;
        if *every* $K_{interval}$ *model updates* **then**
            `// update stale value parameters`
            $\mathcal{D}'' \leftarrow \{m$ samples collected in learned model M using $\pi_{\theta_p}\}$;
            Update θ_v to estimate discounted return with D'';
        end
    end
    `// policy update step`
    for K_{policy} *updates* **do**
        $\mathcal{D}' \leftarrow \{m$ samples collected in learned model M using $\pi_{\theta_p}\}$;
        Update θ_p, θ_v using policy learning method `// e.g. TRPO [73]`
    end
end

supports the evidence in [27] where negative results were demonstrated for their choice of simple environments – `Pusher-v2` and `InvertedPendulum-v2`, and value-aware errors alone were examined for `Hopper-v3` and `Walker2d-v3`. Next, we describe our algorithm with which we find positive results in conjunction with the SLBO algorithm [6] on `Swimmer-v1`, `Hopper-v1` and `Ant-v1` and in conjunction with the MBPO algorithm [7] on `Walker-v2` and `HalfCheetah-v2`.

Correcting Stale Value Estimation

Value-aware objectives use V^{π_θ} as a white-box estimator in place of $V^{\pi}_{M_\phi}$ when simplifying Eq. (3.7). However, dropping the dependency on ϕ in $V^{\pi}_{M_\phi}$ from Eq. Equation 3.7 to Equation 3.8 leads to an issue of stale value estimates in the default dyna-style algorithm, described as follows. For every model update in Algorithm 1, the parameter ϕ of M_ϕ is

changed and as a result, the value function term in Eq. (3.7) no longer corresponds to the same M_ϕ. This implies that for multiple consecutive model updates with a fixed V^{π_θ}, the target that V^{π_θ} is supposed to estimate has moved – making it a stale estimate.

In Algorithm 2 we remedy this issue by updating the value function intermittently (while keeping policy fixed) between model updates. Such an intermittent update is relatively cheap to perform as (i) it does not rely on any additional ground truth experience, (ii) it updates solely the value network and not the policy network, and (iii) the frequency of intermittent updates need not be very high – controlled by the new hyperparameter K_{interval} in Algorithm 2. We found that setting K_{interval} to 20 for SLBO and 5 for MBPO works well in practice. Intuitively, such intermittent value updates allow for a novel interplay in the form of a joint optimisation of model estimates and value estimates (keeping policy fixed) in conjunction with any value aware model learning objective. We hypothesize that this interplay adds stability to the optimisation of value aware objectives and verify it's role in the same with an ablation experiment (Fig. 3.4).

3.5 Experiments

In this section, we investigate our model learning objective in the context of model based reinforcement learning in two settings. First, we evaluate our algorithm in a discrete-state MDP where we optimize expected model advantage directly as well as indirectly via Eqns. Equation 3.8 , Equation 3.10, with the purpose of establishing the performance and convergence relationship among the selected value-aware objectives and a maximum-likelihood baseline in a pedagogical setting. Second, we evaluate Algorithm 2 together with our proposed and a prior value aware objective on several continuous control tasks, with two recent dyna-style MBRL algorithms – SLBO [6] and MBPO [7].

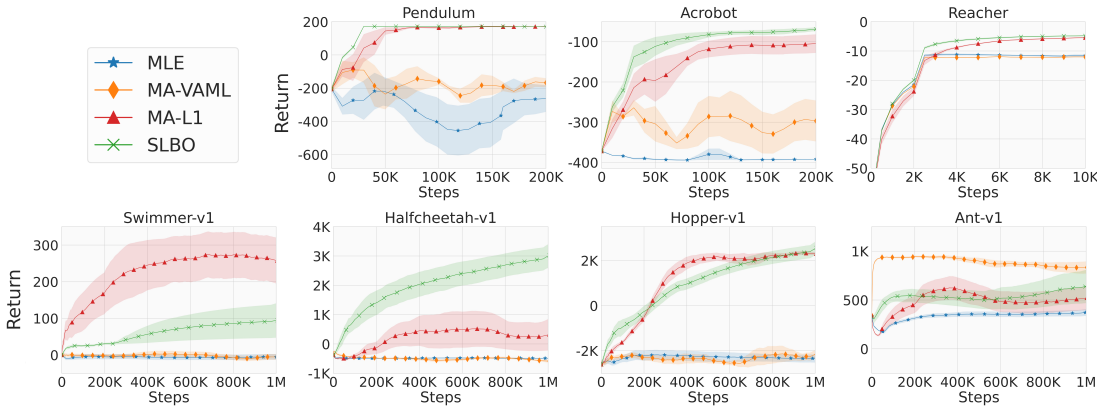


Figure 3.2: Evaluation on continuous control environments for value aware methods and baselines with SLBO [6], without tuning existing parameters, over 5 seeds. Our objective `MA-L1` achieves better return and sample efficiency in comparison to `MA-VAML` on most environments (`Ant-v1` being the exception) and in comparison to `MLE` on all environments. On the `Swimmer-v1` and `Hopper-v1` environments, we also outperform or are competitive with `SLBO`.

3.5.1 Discrete State and Control

We first establish the efficacy of value-aware objectives in a small, finite state setting, where we purportedly make transition dynamics learning hard to scale by posing it as an $|\mathcal{S}|$-way classification problem. For this experiment, we use a discrete-state episodic gridworld MDP with cardinal actions $\{$`North, South, East, West`$\}$, an $N \times N$ grid, deterministic transitions and a fixed, absorbing goal state located at the bottom right of the grid and agent spawning at the top left. A dense reward is provided for improvement in L2 distance to the goal square and an additional time-decaying reward is provided upon reaching the goal. The environment is empty except for walls along all edges. Since the values of the optimal policy are symmetric for states reflected across the major diagonal of the grid, this setting should effectively reduce the number of states in the prediction space by half ($|\mathcal{S}|/2$) for value-aware methods that find the optimal policy. We use four configurations of grid sizes: 8×8, 12×12, 16×16 and 20×20.

Methods. We denote `MA Direct L1` and `MA Direct L2` as methods that optimize $\mathcal{L}_1$ and $\mathcal{L}_2$ objectives respectively (Eq. Equation 3.8). `MA Upper Bound L1`

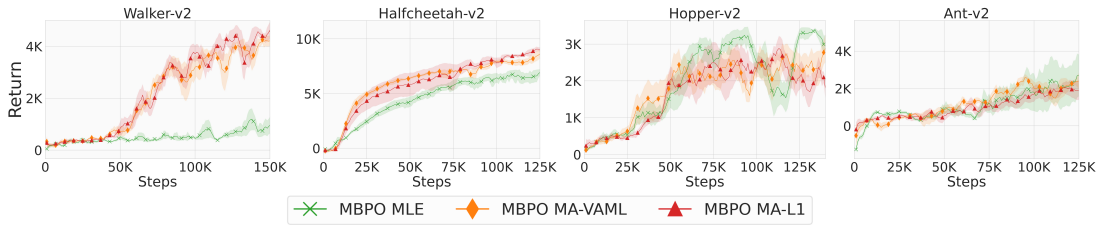


Figure 3.3: Evaluation on continuous control environments for value aware methods and baselines with MBPO [7], without tuning existing parameters, over 5 random seeds. The two value-aware objectives `MBPO MA-L1` and `MBPO MA-VAML` obtain near-matching performance with `MBPO MLE` in several environments but under-performing in others.

optimizes our proposed upper bound $\mathcal{L}_1^U$ and `MA VAML L2` optimizes $\mathcal{L}_2^{\text{VAML}}$ (IterVAML from [21]). In computing the objectives from equations Equation 3.8 and Equation 3.10, the expectation over predicted states is computed exactly as a summation over all states. `MLE` denotes the maximum likelihood baseline. For all methods, we use A2C for policy updates.

Results. Figure Figure 3.1 shows return curves for all methods and environment configurations. Return greater than 1 corresponds to reaching the goal (green square) and solving the task successfully and higher returns correspond to fewer steps taken to reach the goal. We observe that `MLE` sample efficiency decreases with increase in grid size (left to right in Fig. 3.1) and all value based methods outperform this baseline on grid sizes of `16x16` and larger. We observe that the upper bounds on expected model advantage `MA Upper Bound L1` and `MA Upper Bound L2` achieve better sample efficiency than the direct counterparts `MA L1` and `MA L2`, which is expected due to the sparser learning signal from the norm of the summation over value differences in the direct computation as opposed to sum of normed value differences in the upper bounds. In conclusion, we find that value-aware methods do outperform maximum-likelihood in discrete state settings with increasing number of states.

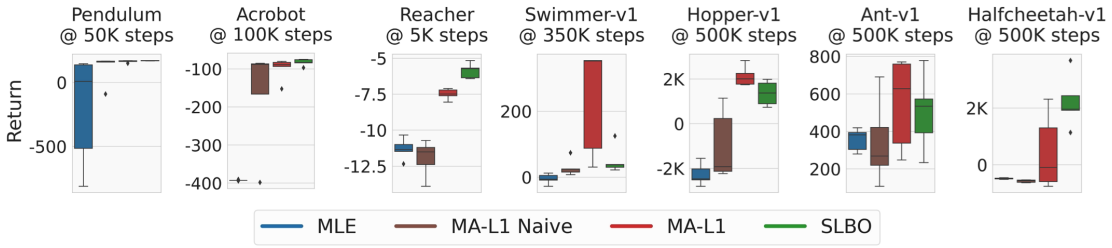


Figure 3.4: Return snapshots taken after convergence of `SLBO`, evaluated on three other variants. `MLE` corresponds to the SLBO algorithm with just an MLE model learning objective. `MA-L1 Naive` corresponds to the SLBO algorithm where the model learning is objective is replaced with a value-aware objective $\mathcal{L}_1^U$. `MA-L1` further uses Algorithm 2 for stale value estimate correction. In most environments, `MA-L1` outperforms `MA-L1 Naive` and `MLE`, indicating that the stale value estimate correction of Algorithm 2 is the reason for improved performance.

3.5.2 Continuous Control

We select two commonly adopted dyna-style MBRL algorithms – SLBO [6] and MBPO [7] as a foundation for evaluating value-aware approaches in continuous control. For SLBO, we use their open source code[4] and for MBPO, we use the open source PyTorch implementation by MBRL-Lib[5]. In both cases, we implement two modifications – (1) the option to swap out MLE with value-aware losses for model-learning and (2) the option to turn on correction of stale value estimates as per Algorithm 2.

3.5.3 Methods

For SLBO, we denote the [6]'s model-learning objective as `SLBO`. This objective uses a horizon length of 2, effectively decomposing the objective into two terms – the first term is the same as MLE, while the second term promotes smoothness (by minimizing the difference of consecutive state differences, see Eqn. 6.1 in [6] with $H = 2$). We denote an MLE-only baseline as `MLE`, which corresponds to keeping just the MLE term of this objective (dropping the smoothness). Intuitively, this should be a weaker baseline than SLBO as it represents a bare bones dyna-style MBRL algorithm. Using `MLE` as the foundation,

[4] https://github.com/facebookresearch/slbo
[5] https://github.com/facebookresearch/mbrl-lib

we obtain two value-aware variants denoted as `MA-L1` and `MA-VAML` which correspond to the empirical versions of $\mathcal{L}_1^U$ and $\mathcal{L}_2^{\text{VAML}}$ (IterVAML from [21]) as model learning objectives replacing maximum-likelihood in `MLE` while keeping everything else the same. The exact formulae of each model learning objective can be found in the Appendix. Both the value-aware variants use Algorithm 2 i.e. the proposed stale value estimate correction. In Figure 3.4, we isolate the benefits of this correction by testing the $\mathcal{L}_1^U$ objective without Algorithm 2, which we denote as `MA-L1 Naive`. We select the same MuJoCo [75] environments provided in the open source code by SLBO, shown in Figure 3.2. Additionally, we show results on two OpenAI Gym [62] environments `Pendulum` and `Acrobot`.

For MBPO, we denote the original MBPO algorithm as `MBPO MLE`. Two value-aware variants are obtained similar to SLBO, which we denote as `MBPO MA-L1` and `MBPO MA-VAML` which again correspond to the $\mathcal{L}_1^U$ and $\mathcal{L}_2^{\text{VAML}}$ respectively. We select the same environments provided in the open source code by MBPO[6], shown in Figure 3.3.

3.5.4 Results

We present return curves for SLBO variants in Figure 3.2, and MBPO variants in Figure 3.3. Among the SLBO variants, we find that our proposed objective `MA-L1` outperforms `MA-VAML` on all environments except `Ant-v1` (where `MA-VAML` performs best) but still achieves performance comparable to `SLBO`. We find a significant improvement in performance of value-aware methods over the `MLE` baseline in most environments. This is an important positive result for value-aware methods in general – they succeed in solving continuous control tasks where MLE alone fails (in all environments except `Reacher`), demonstrating that striving for learning an accurate model (with zero MLE loss) may in fact be practically inefficient or sub-optimal. We also observe that on a few environments, namely `Swimmer-v1` and `Hopper-v1`, `MA-L1` outperforms or is competitive with `SLBO` – a baseline that benefits from the smoothness regularizer in its second term, in

[6]Note that the MuJoCo environments for MBPO use the "v2" variants as opposed to the "v1" variants in SLBO.

addition to MLE.

Among the MBPO variants, we find that both value-aware variants `MBPO MA-L1` and `MBPO MA-VAML` perform similarly and are superior to `MBPO MLE` on `Walker-v2`, `HalfCheetah-v2` environments while under-performing on `Hopper-v2` and `Ant-v2`.

In Figure 3.4, using SLBO, we find that the ablation `MA-L1 Naive` is outperformed by `MA-L1` in all environments except `Pendulum` and `Acrobot`. The `MA-L1 Naive` in most cases fails to exceed the performance of the `MLE` baseline, corroborating the negative results by [27] and highlighting the importance of correcting stale value estimates in Algorithm 2.

Overall, these results demonstrate that it is possible for value-aware model learning objectives, in conjunction with Dyna-style MBRL algorithms, to not only match but also exceed performance of maximum-likelihood alone baselines (i.e. SLBO without smoothness, MBPO) as well as stronger baselines (vanilla SLBO).

3.6 Conclusion

In this chapter, the aim was to bridge the gap in theory and practice of value-aware model learning for model-based RL. We presented a novel value-aware objective inspired by bounding the model-advantage between an approximate and true model given a fixed policy, demonstrating instances of superior performance in comparison to prior value-aware objectives in conjunction with SLBO [6]. We identified the issue of stale value estimates that hamper performance of all value-aware methods in general if used as-is in the dyna-style MBRL framework. Our proposed algorithm enabled successful deployment of value-aware objectives in complex continuous control environments, representing the first positive result in the path to bringing value-aware objectives, well-known for their theoretical benefits, closer to practice.

CHAPTER 4

OFFLINE RL: VALUE-GUIDED DATA AUGMENTATION

4.1 Introduction

In Chapters 2 and 3, we have assumed that the fixed policy π, used for estimation of model-advantage and derived value-aware model learning objectives (Equation 3.6), is the source of ground truth transition data when interacting with the true environment. As a result, off-policy data cannot be used to estimate value aware model learning objectives. In this chapter, we explore a new axis of value-awareness that attempts to leverage the specificity of the task reward for better training of offline RL policies, without the need for on-policy data or interactions.

Offline RL methods rely heavily on either uncertainty estimation (in the model-based case as in MOPO [76] and MOREL [57]) or on conservative Q-learning (e.g. CQL [77]), or both (as in COMBO [1]), in order to penalize deviating from the seen state-actions in the offline dataset. This approach prevents over-estimation of Q-values on unseen state-actions while still allowing Q-value propagation to unseen states via Q-learning. As a result, some amount of deviation from seen behavior is allowed as long as the Q-value estimates for this behavior are reasonably high despite the under-estimation bias. A common design choice in model-based offline RL is to use a learned model to generate branched rollouts starting from seen states in the offline dataset [57, 1], and either penalizing the Q-value estimates of all unseen states visited by model rollouts (as in COMBO [1]), or stopping model rollouts when uncertainty estimates are high (as in MOREL [57]). However, unseen states obtained by branched model rollouts alone do not cover the set of all states where uncertainty (typically epistemic) estimates are high and can be potentially useful for effective Q-learning. Indeed, while it is common practise in offline RL to uniformly sample in the action space

to generate 'negatives' to penalize Q-values on, there are no prescriptions for explicitly searching the state space for such negatives.

This chapter introduces a first attempt at explicitly searching for unseen states where epistemic uncertainty is low, such that they may be used for improved Q-learning via propagation of Q-values to novel unseen regions of the state space not reachable by model-rollouts alone. First, we propose a simple perturb and filter strategy that takes seen states and perturbs them along a specified direction in the state space in order to obtain potentially unseen states far away from the seen data distribution and filtering them such that their epistemic uncertainty estimates are not too high (out of distribution and unknown) or too low (in-distribution but not useful). Next, we specify our direction of perturbation as the gradient of estimated Q-value w.r.t input state multiplied by positive and negative step sizes, motivated by the need to explore unseen states with higher or lower estimated Q-value. We perform offline tuning of our algorithm and compare it against COMBO [1], which we also use as a foundation to build on top of. We find that in most cases where COMBO does not fail completely, our method improves performance over COMBO and that Q-value gradient based perturbation performs better than random direction based perturbations. We also find that the performance improvement of our method depends on both the positive as well as negative direction along the Q-value gradient, with lower performance when using a single direction (positive or negative) alone. Finally, we also inspect the nearest neighbor distances of unseen states found by our method. We find that Q-value gradient based perturbations are able to produce more states farther away from the seen data distribution that also pass through the epistemic uncertainty filtration stage, as opposed to both the COMBO baseline and random direction based perturbations.

4.2 Related Work

Offline or Batch Reinforcement Learning is a long studied problem of learning optimal quantities (values or policies) from a fixed set of experiences or data, with its roots in fitted

value and Q iteration [78, 79] and has recently gained popularity due to the success of powerful function approximators such as deep neural networks in learning policies, values, rewards and dynamics [26, 76, 1, 57]. Offline RL is distinct from imitation learning as the latter assumes expert demonstrations as offline data without reward labels, whereas offline RL assumes known reward labels, allowing for use of sub-optimal or non-expert data. Offline RL datasets such as D4RL [80] and NeoRL [2] are prominent benchmarks for evaluating and comparing offline RL methods. We use environments from D4RL in this chapter for evaluating our method.

Unlabeled Data in Offline RL: Offline RL assumes that the fixed dataset consists of tuples of state, action, next state and reward. However, unlabeled data i.e. without reward labels is often plentiful and an the question of how to leverage such data to improve Offline RL with a typically smaller labeled dataset is an important question. Setting the reward of unlabeled data to zero has been shown to be a simple and effective strategy to leverage this data [81]. Intuitively, this is reasonable to expect as Q-values from reward labeled states will propagate to unlabeled states during learning of a Q-function with temporal difference.

Conservative Q-Learning (CQL): A number of works have used or adapted the method of learning conservative Q estimates by penalizing the estimated value of unseen actions while pushing up value of seen actions in the offline dataset [77]. [1] is a model-based approach that combines CQL [77] with model-based reinforcement learning by additionally penalizing the estimated values of states actions in model rollouts.

Model-based Offline RL: [1, 57] are model-based offline RL approaches that use the approximate MDP induced by a learned dynamics model network with a conservative objective. As previously mentioned, [1] uses CQL by penalizing estimated value of state-actions visited during model rollouts stating from seen states, while [57] defines their induced MDP in a conservative manner by stopping any model rollouts that fall into states with high uncertainty as measured by model ensemble disagreement. COMBO [1] expressly avoids uncertainty estimation citing its inaccuracy in some settings. However, when

we use COMBO as a foundation to build our method, we do not re-introduce uncertainty in the manner COMBO wanted to avoid – we use uncertainty to filter unseen states used for starting model rollouts as opposed to using it as a reward penalty.

Adversarial Unseen State Augmentation: [82] is a model-free online RL approach that proposes adversarial states as data augmentation for policy updates via multiple gradient steps in state space. This is similar to our approach that proposes unseen states via multiple gradient updates in the state space of the Q-value estimate. Other than the difference of objective for computing the gradient, there are also two additional differences between [82] and our approach – (1) we use both the positive and negative gradient direction for proposing unseen states and find that both are jointly more useful than one direction alone, whereas [82] is purely adversarial (minimizing their objective with gradient descent), and (2) [82] keep the sampled action fixed while minimizing their adversarial objective while we use the differentiability of the parametrized policy to sample new actions.

Relationship with Exploration and Safe RL: Safe RL [83] has a primary focus on risk-averse decisions and guarantees for safety. Our method is not an exploration method and hence, does not promote unsafe exploration by mining of unseen states, as these unseen states are mined based on uncertainty of the model itself. Further, these unseen states are not unconditionally promoted for visitation by the policy, they are only used for better Q-learning.

4.3 Preliminaries

In this section, we briefly touch upon Offline Reinforcement Learning (RL) and the baseline COMBO [1] that we build upon.

Offline RL. In Offline Reinforcement Learning, a fixed dataset of interactions $\mathcal{D} \coloneqq (s, a, r, s')$ is provided for finding the best possible policy without any environment interactions (i.e. without "online" interactions). Since we only deal with simulated environments, the policy found by an algorithm is evaluated with online evaluation.

Model-based Offline RL. COMBO [1] is a model-based offline RL algorithm that extends the concept of conservative Q-learning in CQL [77] to model-generated predictions. Algorithm 3 depicts the model-based RL algorithm used by COMBO. First, an initial model fitting stage is executed where the reward and dynamics model is trained with maximum likelihood estimation (MLE) on the offline data. Note that unlike Chapter 2, we will not be using any value-aware objectives for substituting MLE (for dynamics model learning) in this phase as we are dealing with off-policy data incompatible for estimating any value-aware objective. Second, policy and critic learning happens over a fixed number of training epochs where the data used for policy training is obtained by means of *branched rollouts*. Branched rollouts are inherited from the model-based algorithm MOPO [76] (which COMBO uses as a foundation), where a batch of initial states are sampling from the offline dataset and used to generate model-rollouts up to a fixed horizon H using actions sampled from a rollout policy $\mu(\cdot \mid s)$ (typically the same as π or sometimes a uniform over actions policy) and next states sampled from the learned dynamics model (almost always Gaussian with mean and variance predictions). Then, the policy and critic are updated using this rollout data.

The conservative Q-value update used by COMBO is as follows. Here, d_f is an interpolation defined by [1] as $d_f(s,a) := fd(s,a) + (1-f)d^{\mu}_{M_\phi}(s,a)$; where d is the empirical sampling distribution from the offline dataset $\mathcal{D}$.

$$
\begin{aligned}
\hat{Q}^{k+1} &\leftarrow \operatorname*{argmin}_{Q} \mathcal{L}_{\text{TD}} + \beta \cdot \mathcal{L}_{\text{CQL}} \\
\mathcal{L}_{\text{TD}} &:= \mathop{\mathbb{E}}_{(s,a,s')\sim d_f} \left[\left(\left(r(s,a) + \gamma \mathop{\mathbb{E}}_{a'\sim\hat{\pi}^k(a'|s')} \left[\hat{Q}^k(s',a') \right] \right) - Q(s,a) \right)^2 \right] \\
\mathcal{L}_{\text{CQL}} &:= \mathop{\mathbb{E}}_{s\sim d_{M_\phi},\pi, a\sim\pi} [Q(s,a)] - \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}} [Q(s,a)]
\end{aligned} \tag{4.1}
$$

Two important points are to be noted of the COMBO algorithm which will be useful

Algorithm 3: Model-based Offline RL with COMBO [1]

Input: Offline dataset $\mathcal{D}$, parametrized policy and critic π_θ, Q_θ
Input: Parametrized dynamics model $\mathcal{P}_\phi$
Input: Rollout policy $\mu(\cdot \mid s)$, horizon $H \in \mathbb{N}$

Split D into train, val splits and optimize $\mathcal{P}_\phi$ until convergence on val set;
for $i = 1$ *to* n_{epochs} **do**
    $\mathcal{D}_0 \leftarrow$ Sample a batch of start states from $\mathcal{D}$;
    `// Branched model rollouts`
    $\hat{\mathcal{D}} \leftarrow$ Collect model rollouts with $\mu, \mathcal{P}_\phi$ up to H starting from states in $\mathcal{D}_0$;
    Fit Q_θ with a conservative objective using both $\mathcal{D}$ and $\hat{\mathcal{D}}$;
    Update policy π_θ using estimated Q_θ values on $\hat{\mathcal{D}}$;
end

later: (1) Branched model rollouts are used starting from seen states as shown in Algorithm 3, and (2) Conservative Q-value estimation is used for all rollout data (i.e. Q-values are pushed down for data produced by the model). This implies that while Q-value propagation does occur for unseen states found by model rollouts, they are down-weighted proportional to the coefficient β in the above equation ((Equation 4.1)).

4.4 Approach

In this section, we first motivate our method by highlighting the need for augmenting the model rollout buffer with unseen states. Next, we introduce a simple propose and filter strategy for finding unseen states – a proposal stage where a seen state is perturbed and a filter stage where states having estimated uncertainty too low or too high are discarded. We then propose two perturbation functions for obtaining unseen state proposals – one that perturbs randomly uniformly in any direction and a second that perturbs along the positive and negative direction of the Q-value gradient w.r.t states.

Limitations of Model Rollouts. In offline-RL, branched model-rollouts (Algorithm 3) serve an important role in optimizing the behavior policy. Beginning with a batch of seen states, branched model rollouts (using a learned dynamics and reward model) visit potentially unseen states up to a horizon H away from seen states. Using the seen and unseen

Algorithm 4: PnF-Qgrad : Value-aware unseen state augmentation

Input: Input state batch $\mathcal{D}_{\text{batch}} \subseteq \mathcal{D}$, parametrized Q-network Q_θ, policy network π_θ
Input: State-dependent uncertainty estimation function $\xi : \mathcal{S} \to \mathbb{R}$
Input: Number of steps of perturbation n_{steps}, maximum step size δ_{max}
Input: Desired number of augmented states n_{augment} to fill $\mathcal{D}_{\text{augment}}$

Initialize $\mathcal{D}_{\text{augment}} \leftarrow \varnothing$ and $U_0 := \{\xi(s) : s \in \mathcal{D}_{\text{batch}}\}$;
Initialize u_{lower}, u_{upper} as 0.25-quantile and 0.75-quantile of U_0 respectively;
while $|\mathcal{D}_{augment}| < n_{augment}$ **do**
    $\overline{\mathcal{D}}_0 \leftarrow \{s : (s, a, r, s') \in \mathcal{D}_{\text{batch}}\}$;
    Sample $\eta_s \sim U(-\delta_{\text{max}}, \delta_{\text{max}}) \forall s \in \overline{\mathcal{D}}_0$;
    for i *from* 1 *to* n_{steps} **do**
        $\overline{\mathcal{D}}_i \leftarrow \{\overline{s} : \overline{s} \leftarrow s + \eta_s \cdot \nabla_s Q_\theta(s, \pi_\theta(s))\}$;
    end
    $\overline{\mathcal{D}}_{\text{all}} \leftarrow \bigcup_{i=1}^{n_{\text{steps}}} \overline{\mathcal{D}}_i$;
    $\overline{\mathcal{D}}_{\text{filtered}} \leftarrow \{s : \xi(s) > u_{\text{lower}}$ and $\xi(s) < u_{\text{upper}}\ \forall \boldsymbol{s} \in \overline{\mathcal{D}}_{\text{all}}\}$;
    $\mathcal{D}_{\text{augment}} \leftarrow \mathcal{D}_{\text{augment}} \bigcup \overline{\mathcal{D}}_{\text{filtered}}$;
    If $|\mathcal{D}_{\text{augment}}| > n_{\text{augment}}$, subsample uniformly to keep n_{augment} elements;
end

states, the Q-function is optimized by propagation of Q-values using temporal difference and conservative Q-learning objectives, and the policy is optimized to greedily pick the highest Q-valued action at each state. While keeping the learned model, policy and offline dataset fixed, the propagation of accurate Q-values is solely dependent on the unseen states visited during model rollouts. Ideally, to fully exploit the learned model, all states where the model has low uncertainty should be included for the Q-learning update. However, this may not be the case as there is no guarantee that the states visited by H horizon model rollouts will visit all possible states where the model has low uncertainty i.e. states where the learned dynamics and reward model generalizes.

Start State Augmentation. In order to maximize coverage of states where the learned dynamics and reward model generalize to, as measured by estimated uncertainty, we propose an augmentation strategy that uses a mixture of seen and unseen states as starting points to perform model rollouts. Given a batch of n_{start} states sampled from the offline dataset, we control the fraction of this batch to replace with unseen states as $f_{\text{augment}} \in [0, 1]$.

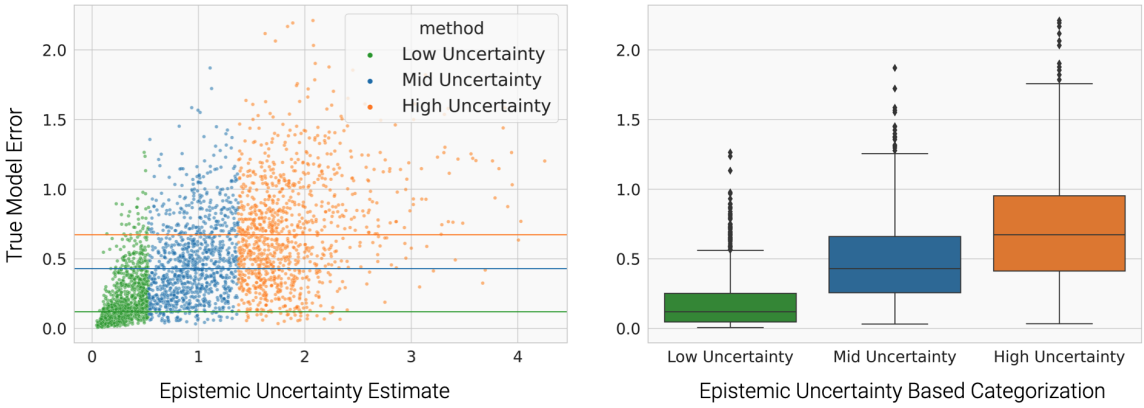


Figure 4.1: (Left) Relationship between epistemic uncertainty estimated using ensemble disagreement and true model error i.e. the mean absolute difference between model next state and reward predictions and true quantities, for the Adroit Pen manipulation task and Human demonstrations dataset. Each point corresponds to a state obtained by model rollouts from a perturbed state in Algorithm 4. Horizontal lines measure medians for respective uncertainty category. (Right) Box plot of true model error aggregated for each uncertainty category.

We find that $f_{\text{augment}} = 0.5$ works well in practice. Note that $f_{\text{augment}} = 0$ reduces to a no augmentation baseline.

Perturb and Filter. We refer to our perturb and filter method as PnF-Qgrad . Algorithm 4 details the subroutine that takes as input a batch of states sampled from the offline dataset and produces a set of potentially unseen states via a perturb and filter strategy. First, a set of candidate state proposals are generated by taking n_{steps} gradient steps using the Q-value gradient w.r.t state input, with a step size uniformly sampled from $(-\delta_{\text{max}}, \delta_{\text{max}})$ i.e. with positive as well as negative directions, and fixed across gradient steps. Next, uncertainty cut-off points $u_{\text{lower}}, u_{\text{upper}}$, computed using the 0.25 and 0.75 quantile uncertainty values of seen data, are used for filtering out states that lie outside of the range $(u_{\text{lower}}, u_{\text{upper}})$. This process is repeated until the desired number of augmented states n_{augment} have been obtained. The while loop typically runs for just one iteration in most of our hyperparameter choices.

Uncertainty Estimator. We tested 3 types of uncertainty estimators: (1) ensemble disagreement via max 2-norm deviation from mean across ensemble (used in the MOPO

re-implementation by [2]), (2) maximum 2-norm of standard deviation across ensemble from (used by MOPO [76]) and (3) standard deviation of mean predictions over ensemble. Here, the first and third estimators approximate epistemic uncertainty while the second approximates aleatoric uncertainty. We found (1) and (3) to overall work well in practise and we select (1) as our choice of (epistemic) uncertainty estimator for all of our experiments. We exclude MOREL's discrepancy based uncertainty [57], as we found it to be similar to (1) in form and practice. COMBO [1] has previously demonstrated the unreliability of MOPO's aleatoric uncertainty estimator in terms of being a predictor of true model error. In order to verify our choice of epistemic uncertainty estimator for this purpose, in Figure 4.1 we measure the relationship with true error and the effect of our quantile based cut-offs in Algorithm 4.

Ablations. In addition to using the Q-value gradient for generating state proposals, we also test an ablation of our method named PnF-Random , that substitutes the Q-value gradient in Algorithm 4, Line 7 with a unit vector sampled uniformly randomly on a hypersphere. In order to prevent random walks, we fix n_{steps} to 1 for this ablation so that the augmented states are solely a result of a single perturbation per direction. We find that the Q-gradient produces more informative state proposals after filtering than this PnF-Random , as measured by final policy performance on tested offline RL benchmarks.

4.5 Experiments

In this section, we perform empirical analyses to answer the following questions: (1) Does our `COMBO PnF-Qgrad` method lead to improvemenet in offline RL performance over a baseline? (2) Does the Q-gradient direction matter for perturbation, when compared to performance of a PnF-Random baseline? (3) Does the sign of the perturbation (i.e. ascending or descending the Q-gradient) matter for performance? and (4) What are the distance characteristics of the unseen states found with PnF augmentation as opposed to the unseen states visited by model rollouts in the baseline?

Algorithm 5: Model-based Offline RL with State Augmentation

Input: Offline dataset $\mathcal{D}$, parametrized policy and critic π_θ, Q_θ
Input: Parametrized dynamics model $\mathcal{P}_\phi$
Input: Rollout policy $\mu(\cdot \mid s)$, horizon $H \in \mathbb{N}$
Input: Fraction of start state batch to augment $f_{\text{augment}} \in [0, 1]$

Split D into train, val splits and optimize $\mathcal{P}_\phi$ until convergence on val set;
for $i = 1$ *to* n_{epochs} **do**
    $\mathcal{D}_{\text{start}} \leftarrow$ Sample a batch of n_{start} start states from $\mathcal{D}$;
    `// Augment start state batch`
    $\mathcal{D}_{\text{start}} \leftarrow$ Augment $n_{\text{augment}} := \lfloor f_{\text{augment}} \cdot n_{\text{start}} \rfloor$ states using Algorithm 4;
    `// Branched model rollouts`
    $\hat{\mathcal{D}} \leftarrow$ Collect model rollouts with $\mu, \mathcal{P}_\phi$ up to H starting from states in $\mathcal{D}_{\text{start}}$;
    Fit Q_θ with a conservative objective using both $\mathcal{D}$ and $\hat{\mathcal{D}}$;
    Update policy π_θ using estimated Q_θ values on $\hat{\mathcal{D}}$;
end

4.5.1 Performance on D4RL Environments and Datasets

We use COMBO [1] as a foundation on top of which we implement our method PnF-Qgrad . We use the open source PyTorch implementation of COMBO by [2]. Our method requires specification of the following additional hyperparameters.

1. n_{steps}: Number of gradient descent or ascent steps in Algorithm 4. We use the range $\{1, 2, 4, 8\}$ to pick the best value.

2. δ_{max}: Maximum value of step size η_s in Algorithm 4. We use the (logarithmic) range $\{1.0E-5, 5.0E-5, 1.0E-4, 5.0E-4, 1.0E-3, 5.0E-3, 1.0E-2, 5.0E-2, 1.0E-1, 0.5\}$ to pick the best value.

3. f_{augment}: Fraction of batch of start states to augment in Algorithm 5. We use the range $\{0.5, 0.9, 1.0\}$ to pick the value, with the mimunum fraction of 0.5 guaranteeing that at least half of the batch consists of augmented states.

We first compare PnF-Qrad performance to COMBO on several offline RL environments from the D4RL benchmark [80]. We refer to our method as `COMBO PnF-Qgrad`

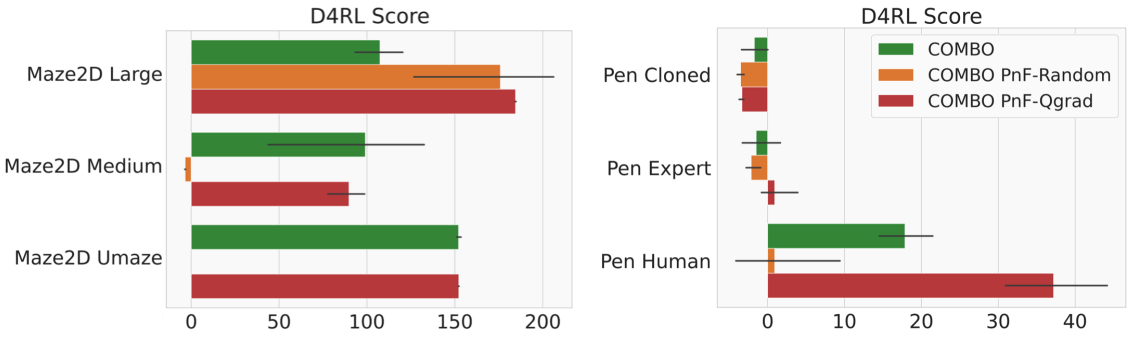


Figure 4.2: Evaluation on two sets of D4RL offline datasets, the Maze2D environment with varying maze sizes (left) and Adroit Dextrous Hand Manipulation - Pen Environment with varying dataset types (right). The Maze2D dataset contain a little under 4 million time steps. The `Pen-v1 Expert` and `Pen-v1 Cloned` datasets contain close to ~ 495000 time steps, whereas the `Pen-v1 Human` dataset contains 4950 time steps. Hyperparameters are tuned on `Maze2D-v1 Medium` for the Maze2D tasks and `Pen-v1 Human` for the Adroit Pen tasks.

to emphasize that COMBO is the base algorithm on top of which we perform unseen state augmentation. Similarly, we refer to our method with random perturbation directions as `COMBO PnF-Random` .

In D4RL, a task (e.g. `Pen-v1 Human`) is specified by a choice of environment (e.g. Adroit Pen environment) and a choice of dataset (e.g. Human demonstrations). Figure 4.2 evaluates `COMBO` , `COMBO PnF-Qgrad` and `COMBO PnF-Random` on two types of D4RL tasks – the Maze2D family of tasks that have three different environments corresponding to varying maze grid sizes (`Umaze`, `Medium`, `Large`). We find that on the `Pen-v1 Human` task consisting of 4950 time steps of human demonstrations, `COMBO PnF-Qgrad` significantly outperforms baselines. However, on the `Pen-v1 Cloned` and `Pen-v1 Expert` tasks, we find that all methods including baselines perform poorly. In the Maze2D tasks, we find an improvement over `COMBO` for both `COMBO PnF-Qgrad` and `COMBO PnF-Random` on the `Maze2D-v1 Large` task. `COMBO PnF-Random` performs poorly on the `Maze2D-v1 Medium` and `Maze2D-v1 Umaze` tasks where `COMBO` and `COMBO PnF-Qgrad` perform comparably.

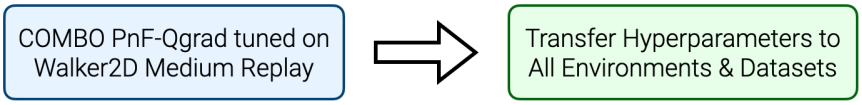
Hyperparameter Sensitivity In Table 4.1, we compare `COMBO PnF-Qgrad` ,

COMBO (our implementation using [2] and the reported performance by Yu et al. in [1]. Due to computational resource constraints, we tune hyperparameters on a single task – the Walker2D-v2 Medium-Replay task, while evaluating on all other tasks. We find that, while we do observe statistically significant improvement in performance on the same environment where hyperparameter values were tuned, we don't see a complete replication of this trend when transferring hyperameters to other tasks. We suspect that this sensitivity is a result of the varying distribution and distance between states in each dataset and environment, as these will affect the perturbation magnitude in our method PnF-Qgrad (e.g. value of $\delta_{\text{max}}, n_{\text{steps}}$).

Average Dataset Q-value In Figure 4.3, we measure the average dataset Q-value, a quantity important for not only selecting hyperparameters in an offline manner, but one which has shown to indicate online evaluation performance. COMBO [1] has shown lower average dataset Q-values lead to more conservative Q-value estimates i.e. preventing any overestimation of Q-values, and higher online evaluation performance (D4RL score). We find that throughout the 500 epochs of policy training, average dataset Q-value is significantly lower for COMBO PnF-Qgrad in comparison to COMBO. Further, in Figure 4.4, we see that this trend holds when reducing the size of the dataset, despite the performance of the two being comparable at lower dataset sizes. This suggests that conservative Q-value estimation is an important mechanism of action for our proposed method in contributing to better online evaluation performance.

In Figure 4.4, we also demonstrate the effect of reducing dataset size for the Walker2D-v2 Medium-Replay task. We reduce dataset size by selecting a contiguous sub-arrays from the front of the original dataset given a fraction (e.g. fraction 0.5 takes the first half of the dataset). While fractions of 0.5 and smaller rapidly reduce performance and make them comparable across methods, we observe that the average dataset Q-value remains significantly lower for COMBO PnF-Qgrad in comparison to COMBO.

Table 4.1: D4RL Score mean and standard error over 6 random seeds for (left to right) COMBO (reported results by [1]), the implementation of COMBO by [2] that we use and our proposed PnF-Qgrad method. PnF-Qgrad requires tuning of the hyperparameters $\delta_{\text{max}}, n_{\text{steps}}, f_{\text{augment}}$ which are sensitive to both datasets and environments. We tune these hyperparameters on the `Walker2D-v2 Medium-Replay` dataset, according to the offline tuning guidelines from [1, 3], and report online evaluation performance of the selected hyperparameters across all other environments and datasets. $*$ indicates tuned on `Walker2D-v2 Medium-Replay`.



Dataset type	Environment	COMBO (Yu et al.)	COMBO	COMBO PnF-Qgrad*
random	halfcheetah	38.8 ± 3.7	14.5 ± 6.4	−1.5 ± 0.5
random	hopper	17.9 ± 1.4	4.7 ± 1.8	4.6 ± 2.6
random	walker2d	7.0 ± 3.6	7.6 ± 1.4	5.1 ± 1.8
medium	halfcheetah	54.2 ± 1.5	62.6 ± 1.3	41.2 ± 4.5
medium	hopper	97.2 ± 2.2	61.0 ± 0.4	51.6 ± 15.5
medium	walker2d	81.9 ± 2.8	34.2 ± 0.0	70.4 ± 2.0
medium-replay	halfcheetah	55.1 ± 1.0	53.0 ± 6.1	41.0 ± 3.8
medium-replay	hopper	89.5 ± 1.8	41.3 ± 11.3	34.5 ± 19.1
medium-replay	**walker2d**	56.0 ± 8.6	32.1 ± 25.7	**70.9 ± 4.3**
med-expert	halfcheetah	90.0 ± 5.6	37.6 ± 0.0	86.4 ± 3.5
med-expert	hopper	111.1 ± 2.9	34.6 ± 30.4	18.7 ± 4.6
med-expert	walker2d	103.3 ± 5.6	48.3 ± 68.0	54.2 ± 25.5

4.5.2 Sign of Perturbation

In Algorithm 4, we choose the step size $\eta_s \in U(-\delta_{\text{max}}, \delta_{\text{max}})$ for a given direction $\nabla_s Q_\theta(s, \pi_\theta(s))$. In order to verify that both positive and negative step sizes for perturbation along this direction are important for good performance, we evaluate two ablations that use either a positive-only step size or a negative-only step size, using the `Pen-v1 Human` task. We refer to `COMBO PnF-Qgrad (+ve, -ve)` as our original method that samples $\eta_s \in U(-\delta_{\text{max}}, \delta_{\text{max}})$, `COMBO PnF-Qgrad (+ve-only)` as the positive-only ablation that samples $\eta_s \in U(0, \delta_{\text{max}})$ and `COMBO PnF-Qgrad (+ve-only)` as the negative-only ablation that samples $\eta_s \in U(-\delta_{\text{max}}, 0)$. Figure 4.5 highlights our findings in two settings. First, we use inherit the hyperparameter values for each ablation from the

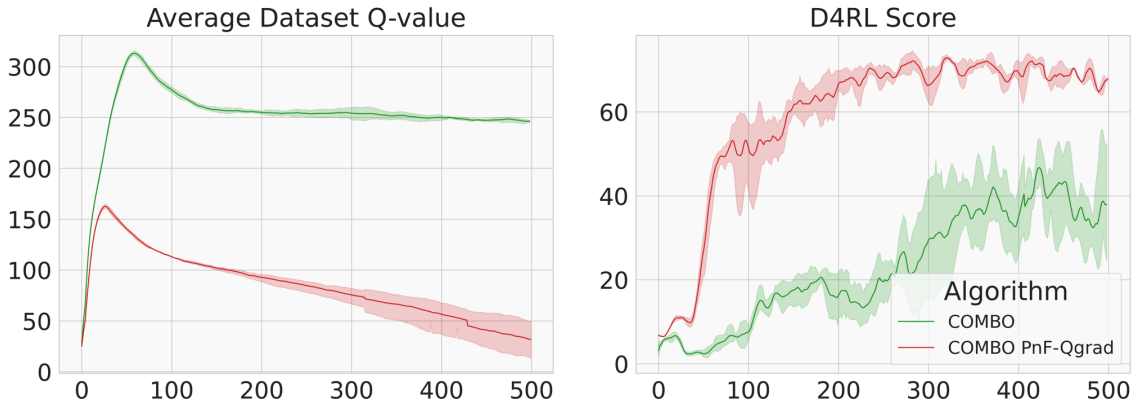


Figure 4.3: Average dataset Q-value (left) and D4RL score (right) for the `Walker2d-v2 Medium-Replay` task over 6 random seeds. The training curve is for 500 epochs of the policy update phase that occurs after an initial phase of fitting a model to the offline dataset. A consistently and significantly lower overall average dataset Q-value is obtained for `COMBO PnF-Qgrad` in comparison to the `COMBO` baseline. COMBO [1] advocates for lower average dataset Q-values as they prevent overestimation of Q-value for unseen states and actions, while also being strongly linked to higher online evaluation performance (D4RL score). We observe a similar result i.e. lower average dataset Q-values for `COMBO PnF-Qgrad` lead to better D4RL score.

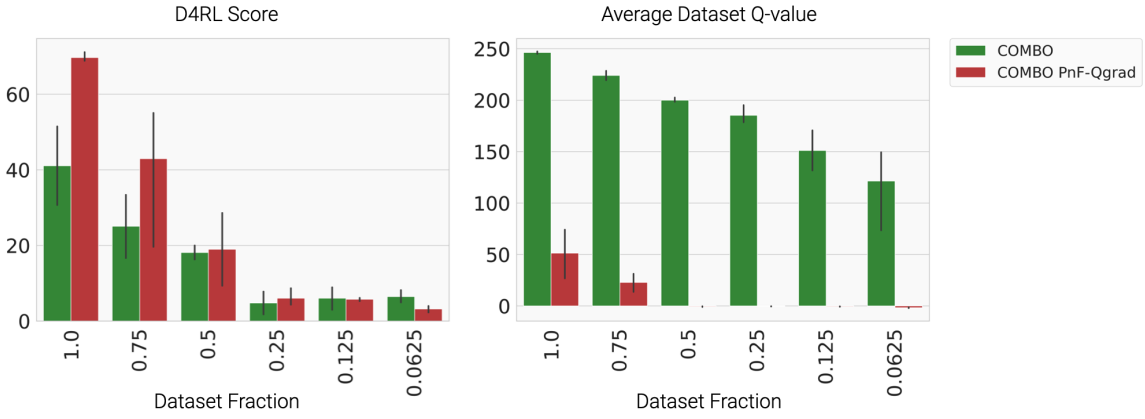


Figure 4.4: D4RL score (left) and average dataset Q-value (right) for varying dataset size fractions for the `Walker2D-v2 Medium-Replay` task. Each dataset fraction corresponds to a contiguous subarray taken from the front of the original dataset. We observe significantly lower average dataset Q-values but comparable D4RL scores for `COMBO PnF-Qgrad` versus `COMBO` as the dataset size reduces.

`COMBO PnF-Qgrad (+ve, -ve)`. In this setting, we find that performance is immediately lowered for both ablations, indicating that either both positive and negative step sizes matter, or that the ablations may need to be individually tuned. Second, we elim-

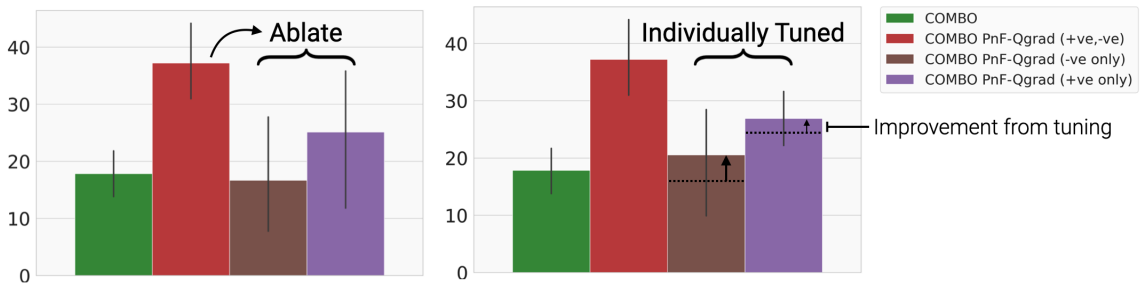


Figure 4.5: D4RL Score comparison on `Pen-v1 Human` task over 6 random seeds of ablations of PnF-Qgrad that use positive-only step size along Q-gradient (i.e. $\eta_s \sim U(0, \delta_{\text{max}})$ in Algorithm 4), and negative-only step size along Q-gradient i.e. (i.e. $\eta_s \sim U(-\delta_{\text{max}}, 0)$ in Algorithm 4). (left) Ablations for positive-only and negative-only inherit the hyperparameter values ($n_{\text{steps}}, f_{\text{augment}}, \delta_{\text{max}}$) from PnF-Qgrad (positive, negative) (red bar). (right) Each ablation is individually tuned to obtain best value of hyperparameters $n_{\text{steps}}, f_{\text{augment}}, \delta_{\text{max}}$.

inate one of the possibilities by individually tuning hyperparameter values for both ablations, where performance is slightly improved over the un-tuned methods, while still having lower performance than `COMBO PnF-Qgrad (+ve, -ve)`. As a result, we find that it is important to perturb along the positive as well as negative direction of Q-value gradient for best performance.

4.5.3 Distribution of Perturbation Distance

In order to understand the impact of our studied perturbations on the distance of augmented unseen states from seen states, we plot a histogram of nearest neighbor distances of every unseen state (w.r.t seen states in the entire offline dataset) visited by Algorithm 5 (for PnF-Qgrad and PnF-Random) and Algorithm 3 (COMBO). For each algorithm, the set of unseen states are an aggregation of all states visited during model rollouts. For Algorithm 5, we set the fraction of model rollouts that occur from unseen states, f_{augment}, to 1.0 We expect that any perturbation based unseen state augmentation has the potential to visit states in model rollouts that are far away from any nearest seen state.

Figure 4.6 plots these histograms for PnF-Qgrad , PnF-Random and COMBO on the `Pen-v1 Human` task with rollout horizon values in $\{1, 5, 10\}$. For PnF-Qgrad , we set $\delta_{\text{max}} = 0.001, n_{\text{steps}} = 4$, which are the tuned values of these hyperparameters for this task.

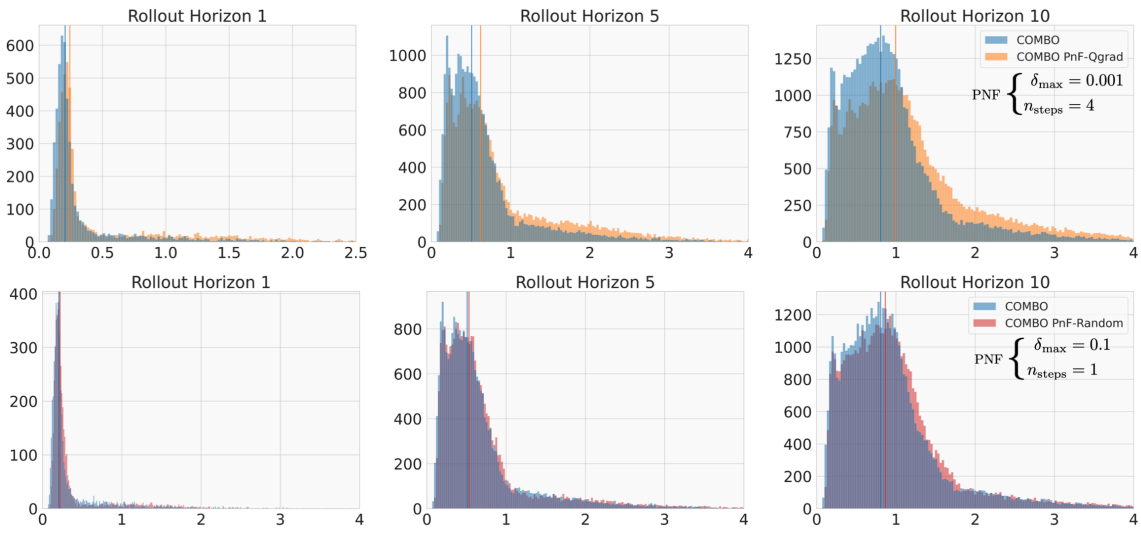


Figure 4.6: Histograms of distance of visited unseen states from seen dataset computed using L2 distance from nearest neighbor in seen dataset (X-axis) for the `Pen-v1 Human` dataset. Unseen states in the COMBO baseline are obtained by model rollouts from seen states, whereas unseen states from PnF-Qgrad and PnF-Random are obtained using model rollouts starting from augmented states using Algorithm 4 and their respective choice of perturbation directions. Rollout horizon values are (left to right) $1, 5, 10$. PnF-Qgrad uses $\delta_{\text{max}} = 0.001$, $n_{\text{steps}} = 4$ and PnF-Random uses $\delta_{\text{max}} = 0.1$, $n_{\text{steps}} = 1$, which were the best hyperparameter values selecting after tuning. Dark colored full length vertical lines correspond to median of respective distribution.

Similarly, for PnF-Random , we set $\delta_{\text{max}} = 0.1, n_{\text{steps}} = 1$ (note that for PnF-Random , n_{steps} is always set to 1 as multiple gradient steps are not necessary for a random perturbation direction). We find that despite the low value of δ_{max}, PnF-Qgrad is able to find unseen state augmentations that produce model rollout states with farther nearest neighbor L2 distance than the model rollouts states visited by no augmentation (COMBO baseline). This is in comparison to PnF-Random , which has overall distribution of distances similar to no augmentation (despite the higher δ_{max}). This indicates that PnF-Qgrad is able to find unseen states farther from seen data while still passing through uncertainty filter, whereas random perturbation are not able to do the same.

4.6 Conclusion

In this chapter, we studied how value-awareness can be applied to the domain of offline reinforcement learning (RL). In this context, we viewed value-awareness from the perspective of data or unseen state augmentation, by proposing a value-gradient based perturbation and (epistemic) uncertainty filtering algorithm that successfully found unseen states farther away from the seen data distribution that were useful for better policy learning in offline RL. In most cases where the baseline COMBO [1] algorithm had non-zero performance, we found statistically significant improvements in online evaluated policy performance. Our method does require offline tuning of hyperparameters given a dataset or task in order to be most effective. We also study ablations that highlight the importance of augmenting in both the positive and negative directions of value gradient as compared to a single direction alone. Finally, we also analyse the distribution of nearest neighbor distances of augmented states from the seen states in the offline dataset and find that value-gradient based perturbed and filtered states have a larger L2 distance than random direction based perturbed and filtered states, implying that the value-gradient can be leveraged to find unseen states farther from the seen data distribution with low epistemic uncertainty.

CHAPTER 5
CONCLUSION

This thesis investigates value-awareness for model-based online and offline reinforcement learning. The primary motivation for value-awareness is to leverage the fact that a reward specification induces a gradation of undesirable versus desirable behavior, a gradation which can inform model learning in the online RL setting (value-aware model learning), and one which can inform directions of perturbations for finding useful unseen states in the offline RL setting.

For the online setting, this thesis revisited value-aware model learning by deriving a novel value-aware objective based on upper bounding model performance difference i.e. the difference in performance of a given policy across two models. Next, this objective, along with existing value-aware objectives, were applied to practical continuous control manipulation and locomotion tasks in simulation – where the bottleneck of stale value-estimates was identified and remedied, in order to make value-aware objectives performant in these domains.

For the offline setting, this thesis studied value-gradient guided perturbations for finding unseen states with low epistemic uncertainty. Such states were shown to be useful for better policy performance in offline model-based RL algorithms.

The inductive bias of leveraging value-awareness i.e. the information in the reward specification of a task, in improving performance at that very task, is the antithesis of the paradigm of disentangling reward or task information from an environment. While the latter allows for unsupervised learning in a task independent manner and later adapting to a specific task, the former paradigm strips away anything not related to the specified task or reward in order to make learning more efficient. However, this paradigm comes with its drawbacks, primarily due to the sensitivity to reward specifications and failures

with sparse rewards. An important avenue of future research is to find a middle ground between leveraging task or reward information while also not being brittle to the intricacies of reward specification or misspecification.

Appendices

APPENDIX A

APPENDIX FOR MODEL-ADVANTAGE AND GENERALIZATION GAP

A.1 Performance Difference

A.1.1 Proof of Model Performance Difference Lemma

Restating Lemma 4:

Lemma 6. *(Model Performance Difference Lemma) Let M and M' be two different MDPs. Further, define $\mathcal{R}^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[\mathcal{R}(s,a)]$ and $\mathcal{R}^\pi_\epsilon(s) = \mathcal{R}^\pi_M(s) - \mathcal{R}^\pi_{M'}(s)$. For any policy $\pi \in \Pi$ we have:*

$$J_M(\pi) = J_{M'}(\pi) + \mathbb{E}_{s\sim d_{M,\pi}}[\mathcal{R}_\epsilon(s)] + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{M,\pi}}\mathbb{E}_{s'\sim\mathcal{P}_M(s'|s,\pi)}\left[\boldsymbol{A}^\pi_{M'}(s,s')\right]$$

Proof. Let $\mathcal{P}_0$ be the start state distribution for both MDPs, $P^\pi_{M,t}$ be the state distribution at time t, starting from $s_0 \sim \mathcal{P}_0$ in M and $d_{M,\pi}$ denote the stationary state distribution under MDP M, policy π and start state $s_0 \sim \mathcal{P}_0$. We use the following slightly modified version of the definition of value function which has a normalization of $1-\gamma$:

$$V^\pi_M(s_0) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{a_t,s_t\sim\pi P_{M,t}}[\mathcal{R}_M(s_t,a_t)]$$

Then, we have:

$$\begin{aligned}
&J_M(\pi) - J_{M'}(\pi)\\
&= \mathbb{E}_{s_0\sim\mathcal{P}_0}\left[V^\pi_M(s_0) - V^\pi_{M'}(s_0)\right]\\
&= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s_t\sim P^\pi_{M,t}}\mathbb{E}_{a_t\sim\pi(\cdot|s_t)}\left[\mathcal{R}_M(s_t,a_t)\right] - \mathbb{E}_{s_0\sim\mathcal{P}_0}\left[V^\pi_{M'}(s_0)\right]
\end{aligned}$$

$$= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{s_t\sim P^{\pi}_{M,t}}\left[\mathcal{R}^{\pi}_{M}(s_t)\right] - \mathbb{E}_{s_0\sim\mathcal{P}_0}\left[V^{\pi}_{M'}(s_0)\right]$$

$$= \sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{s_t\sim P^{\pi}_{M,t}}\left[(1-\gamma)\mathcal{R}^{\pi}_{M}(s_t) + V^{\pi}_{M'}(s_t) - V^{\pi}_{M'}(s_t)\right] - \mathbb{E}_{s_0\sim\mathcal{P}_0}\left[V^{\pi}_{M'}(s_0)\right]$$

Cancelling the first element in the summation, and shifting the series by 1 step:

$$= \sum_{t=0}^{\infty}\gamma^t \mathop{\mathbb{E}}_{\substack{s_t\sim P^{\pi}_{M,t}\\ s_{t+1}\sim P^{\pi}_{M,t+1}}}\left[(1-\gamma)\mathcal{R}^{\pi}_{M}(s_t) + \gamma V^{\pi}_{M'}(s_{t+1}) - V^{\pi}_{M'}(s_t)\right]$$

Expanding $V^{\pi}_{M'}(s_t)$ with a one-step bellman evaluation operator:

$$= \sum_{t=0}^{\infty}\gamma^t \mathop{\mathbb{E}}_{\substack{s_t\sim P^{\pi}_{M,t}\\ s_{t+1}\sim P^{\pi}_{M,t+1}}}\Bigg[(1-\gamma)\mathcal{R}^{\pi}_{M}(s_t) + \gamma V^{\pi}_{M'}(s_{t+1}) - \left((1-\gamma)\mathcal{R}^{\pi}_{M'}(s_t) + \gamma\mathbb{E}_{s''\sim\mathcal{P}^{\pi}_{M'}(s_t,\pi)}\left[V^{\pi}_{M'}(s'')\right]\right)\Bigg]$$

$$= \sum_{t=0}^{\infty}\gamma^t \mathop{\mathbb{E}}_{\substack{s_t\sim P^{\pi}_{M,t}\\ s_{t+1}\sim P^{\pi}_{M,t+1}}}\Bigg[(1-\gamma)(\mathcal{R}^{\pi}_{M}(s_t) - \mathcal{R}^{\pi}_{M'}(s_t)) + \gamma V^{\pi}_{M'}(s_{t+1}) - \gamma\mathbb{E}_{s''\sim\mathcal{P}^{\pi}_{M'}(s_t,\pi)}\left[V^{\pi}_{M'}(s'')\right]\Bigg]$$

Using definition of $\mathcal{R}^{\pi}_{\epsilon}$:

$$= \sum_{t=0}^{\infty}\gamma^t \mathop{\mathbb{E}}_{\substack{s_t\sim P^{\pi}_{M,t}\\ s_{t+1}\sim P^{\pi}_{M,t+1}}}\left[(1-\gamma)\mathcal{R}^{\pi}_{\epsilon}(s_t) + \gamma V^{\pi}_{M'}(s_{t+1}) - \gamma\mathbb{E}_{s''\sim\mathcal{P}^{\pi}_{M'}(s_t,\pi)}\left[V^{\pi}_{M'}(s'')\right]\right]$$

Using definition of $\boldsymbol{A}^{\pi}_{M'}$:

$$= \sum_{t=0}^{\infty}\gamma^t \mathop{\mathbb{E}}_{\substack{s_t\sim P^{\pi}_{M,t}\\ s_{t+1}\sim P^{\pi}_{M,t+1}}}\left[(1-\gamma)\mathcal{R}^{\pi}_{\epsilon}(s_t) + \boldsymbol{A}^{\pi}_{M'}(s_t, s_{t+1})\right]$$

$$= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{M,\pi}}\mathbb{E}_{s'\sim\mathcal{P}_M(s,\pi)}\left[\boldsymbol{A}^{\pi}_{M'}(s,s')\right] + \mathbb{E}_{s\sim d_{M,\pi}}\left[\mathcal{R}^{\pi}_{\epsilon}(s)\right]$$

□

A.1.2 Corollary: Deviation Error

The following is a useful corollary for subsequent proofs.

Corollary 7. *Let M and M' be two different MDPs. For any policy $\pi \in \Pi$ we have:*

$$J_M(\pi) = J_{M'}(\pi) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{M,\pi}}\underbrace{\left[\mathcal{T}^{\pi}_{M}V^{\pi}_{M}(s) - \mathcal{T}^{\pi}_{M'}V^{\pi}_{M}(s)\right]}_{\textit{deviation error}} \tag{A.1}$$

Proof.

$$\begin{aligned}
&J_M(\pi) - J_{M'}(\pi)\\
&= \mathbb{E}_{s_0\sim\mathcal{P}_0}\left[V^{\pi}_{M}(s_0) - V^{\pi}_{M'}(s_0)\right]\\
&= \cdots
\end{aligned}$$

Proceeding similar to the previous proof upto the following line:

$$= \sum_{t=0}^{\infty}\gamma^t \mathop{\mathbb{E}}_{\substack{s_t\sim P^{\pi}_{M,t}\\ s_{t+1}\sim P^{\pi}_{M,t+1}}}\Big[(1-\gamma)\mathcal{R}^{\pi}_{M}(s_t) + \gamma V^{\pi}_{M'}(s_{t+1}) - \Big((1-\gamma)\mathcal{R}^{\pi}_{M'}(s_t) + \gamma\mathbb{E}_{s''\sim\mathcal{P}^{\pi}_{M'}(s_t,\pi)}\left[V^{\pi}_{M'}(s'')\right]\Big)\Big]$$

Using definition of the bellman operator $\mathcal{T}^{\pi}_{M}$

$$= \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s_t\sim P^{\pi}_{M,t}}\left[\mathcal{T}^{\pi}_{M}V^{\pi}_{M'}(s_t) - \Big((1-\gamma)\mathcal{R}^{\pi}_{M'}(s_t) + \gamma\mathbb{E}_{s''\sim\mathcal{P}^{\pi}_{M'}(s_t,\pi)}\left[V^{\pi}_{M'}(s'')\right]\Big)\right]$$

Using definition of the bellman operator $\mathcal{T}^{\pi}_{M'}$

$$= \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t \sim P^{\pi}_{M,t}} \left[\mathcal{T}^{\pi}_{M} V^{\pi}_{M'}(s_t) - \mathcal{T}^{\pi}_{M'} V^{\pi}_{M'}(s_t) \right]$$
$$= \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_{\pi,M}} \left[\mathcal{T}^{\pi}_{M} V^{\pi}_{M'}(s) - \mathcal{T}^{\pi}_{M'} V^{\pi}_{M'}(s) \right]$$

□

Note that we can further upper bound the difference in values across MDPs for a policy as follows, which will be useful in subsequent proofs. We compute this bound at an arbitrary start state s_0, and it will then hold for any start state. Let $d_{M,s_0,\pi}$ be the stationary state distribution of following policy π in MDP M, starting at state s_0.

$$V^{\pi}_{M}(s_0) - V^{\pi}_{M'}(s_0)$$
$$= \cdots \tag{A.2}$$

Proceeding similar to the proof in Section A.1.1, we get the following:

$$= \frac{1}{1-\gamma} \mathbb{E}_{d_{M,s_0,\pi}} \left[\mathcal{T}^{\pi}_{M} V^{\pi}_{M'} - \mathcal{T}^{\pi}_{M'} V^{\pi}_{M'} \right]$$
$$\leq \frac{1}{1-\gamma} \left\| \mathcal{T}^{\pi}_{M} V^{\pi}_{M'} - \mathcal{T}^{\pi}_{M'} V^{\pi}_{M'} \right\|_{\infty} \tag{A.3}$$
$$\leq \frac{1}{1-\gamma} \Bigg[\left\| \mathcal{R}^{\pi}_{M'} - \mathcal{R}^{\pi}_{M} \right\|_{\infty} \tag{A.4}$$
$$+ \gamma \max_{s} \sum_{s' \in \mathcal{S}} V^{\pi}_{M'}(s') \mathbb{E}_{a \sim \pi(s)} \left[p_{M'}(s'|s,a) - p_{M}(s',a) \right] \Bigg]$$
$$\leq \frac{1}{1-\gamma} \Bigg[\epsilon_R + \gamma \left\| V^{\pi}_{M'} \right\|_{\infty} \underbrace{\max_{s} \max_{a} \left\| p_{M'}(s'|s,a) - p_{M}(s',a) \right\|_1}_{\epsilon_P} \Bigg]$$
$$\leq \frac{1}{1-\gamma} \left[\epsilon_R + \gamma \left\| V^{\pi}_{M'} \right\|_{\infty} \epsilon_P \right]$$
$$\leq \frac{1}{1-\gamma} \left[\epsilon_R + \frac{\gamma \epsilon_P R_{\max}}{1-\gamma} \right] \tag{A.5}$$

A.2 Generalization with VI & FQI

A.2.1 Value Iteration

Restating Theorem 3:

Theorem 8. *Let M, M' be two MDPs s.t. $\max_s \max_a |\mathcal{R}_M(s,a) - \mathcal{R}_{M'}(s,a)| \leq \epsilon_R$ and $\max_s \max_a \|p_M(s,a) - p_{M'}(s,a)\|_1 \leq \epsilon_P$. Let π_{n+1} be the policy obtained after n VI iterations on MDP M. Then we have,*

$$\|V^{\pi_{n+1}}_{M'} - V^{\star}_{M'}\|_\infty \leq \frac{1}{1-\gamma}\left[\gamma\epsilon^{(n)} + 2\epsilon_R + \frac{2\epsilon_P R_{max}}{1-\gamma}\right]$$

Proof. In the main paper, we derived the following results:

$$\left\|V^{\pi_{n+1}}_{M'} - V^{\pi'^{\star}}_{M'}\right\|_\infty \leq \left\|V^{\pi_{n+1}}_{M'} - V^{\pi_{n+1}}_{M}\right\|_\infty + \left\|V^{\pi_{n+1}}_{M} - V^{\pi'^{\star}}_{M'}\right\|_\infty \tag{A.6}$$

$$\left\|V^{\pi_{n+1}}_{M} - V^{\pi'^{\star}}_{M'}\right\|_\infty \leq \frac{\gamma\epsilon_n}{1-\gamma} + \frac{\delta_{n+1}\left(V^{\pi_{n+1}}_{M}\right)}{1-\gamma} \tag{A.7}$$

Now, the first term in the RHS of Eq. (A.6), we use the upper bound derived in Equation (A.5):

$$V^{\pi}_{M'} - V^{\pi}_{M} \leq \frac{1}{1-\gamma}\left[\epsilon_R + \frac{\gamma\epsilon_P R_{\max}}{1-\gamma}\right] \tag{A.8}$$

Notice that $\delta_{M,M'}$ occurs in Eq. (A.3), and using the same proof as that of Eq. (A.5), we get the following bount on $\delta_{M,M'}$ for any V.

$$\begin{aligned}\delta_{M,M'}(V) &\leq \epsilon_R + \gamma\epsilon_P \|V\|_\infty \\ &\leq \epsilon_R + \frac{\gamma\epsilon_P R_{\max}}{1-\gamma}\end{aligned} \tag{A.9}$$

Putting together Eqs. (A.7) to (A.9), we get the desired bound. □

A.2.2 Fitted-Q Iteration

Recall that $D = \{(s,a,r,s')_i\}_{i=1}^n$ is a dataset of experiences where $(s,a) \sim \mu \times U$ for some state-distribution μ and policy U, $s' \sim \mathcal{P}_M(s' \mid s,a)$ and $r \sim \mathcal{R}_M(s,a)$. FQI is an iterative algorithm that learns a function f in the model-class $\mathcal{F}$ that approximates the Q-function. At each iteration $f_k = \hat{\mathcal{T}}_M f_{k-1}$ where $\hat{\mathcal{T}}_M$ is the *empirical* Bellman operator defined as follows:

$$\hat{\mathcal{T}}_M f := \operatorname*{argmin}_{f \in \mathcal{F}} \mathcal{L}_D(f, f')$$
$$\mathcal{L}_D(f,f') := \frac{1}{|D|} \sum_{(s,a,r,s') \in D} \left(f(s,a) - r - \gamma V_{f'}(s')\right)^2$$

We define norms for functions over $\mathcal{S} \times \mathcal{A}$, similar to [84], as follows: $\|g\|_{\nu \times \pi} := \left(\mathbb{E}_{s \sim \nu, a \sim \pi}\left[|g|^2\right]\right)^{1/2}$ for $\nu \times \pi \in \Delta(\mathcal{S} \times \mathcal{A})$ and $g : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

Theorem 9. *Let M, M' be two MDPs s.t. $\max_s \max_a |\mathcal{R}_M(s,a) - \mathcal{R}_{M'}(s,a)| \leq \epsilon_R$ and $\max_s \max_a \|p_M(s,a) - p_{M'}(s,a)\|_1 \leq \epsilon_P$. Let $D = \{(s,a,r,s')_i\}_{i=1}^n$ be generated as $(s,a) \sim \mu \times U$, where μ is exploratory and U is uniform over actions, $r \sim R_M(s,a)$, $s' \sim P_M(s,a)$. Let π_{f_k} be the greedy policy w.r.t. f_k, obtained after k iterations of FQI on D. Then we have,*

$$\left\|Q^{\pi_{f_k}}_{M'} - Q^\star_{M'}\right\|_{\nu \times \pi}$$
$$\leq \frac{1}{1-\gamma}\left[\gamma^k R_{max} + O\left(\sqrt{|\mathcal{A}|\,\epsilon^{(n)}}\right) + 2\epsilon_R + \frac{2\epsilon_P R_{max}}{1-\gamma}\right]$$

Proof. Let $\hat{\pi} := \pi_{f_k}$, let v'^*, π'^* denote the optimal value function and policy in the second MDP M', and $Q^\star_{M'} := Q^{\pi'^*}_{M'}$. Let

$$
\begin{aligned}
v^{\pi'^\star}_{M'} - v^{\hat{\pi}}_{M'} &\leq \sum_{t=1}^{\infty} \gamma^{t-1} \mathbb{E}_{s \sim P^{\hat{\pi}}_{M',t}} \left[Q^\star_{M'}(s, \pi'^\star) - Q^\star_{M'}(s, \hat{\pi}) \right] \\
&\leq \sum_{h=1}^{\infty} \gamma^{h-1} \left[\left\| Q^\star_{M'} - Q^{\hat{\pi}}_{M'} \right\|_{P^{\hat{\pi}}_{M',t} \times \pi'^\star} + \left\| Q^\star_{M'} - Q^{\hat{\pi}}_{M'} \right\|_{P^{\hat{\pi}}_{M',t} \times \hat{\pi}} \right] \\
&\leq \left(\frac{2}{1-\gamma} \right) \max_{\nu \times \pi \in \Delta(\mathcal{S} \times \mathcal{A})} \left\| Q^\star_{M'} - Q^{\hat{\pi}}_{M'} \right\|_{\nu \times \pi}
\end{aligned}
$$

Now, it remains to bound $\left\| Q^{\hat{\pi}}_{M'} - Q^\star_{M'} \right\|_{\nu \times \pi}$ for any $\nu \times \pi \in \Delta(\mathcal{S} \times \mathcal{A})$. First we look at the term $\left\| Q^{\hat{\pi}}_{M} - Q^\star_{M'} \right\|_{\nu \times \pi} = \left\| f_k - Q^\star_{M'} \right\|_{\nu \times \pi}$ below. Recall that $f_k = \hat{\mathcal{T}}_M f_{k-1}$.

$$
\begin{aligned}
&\left\| f_k - Q^\star_{M'} \right\|_{\nu \times \pi} \\
&= \left\| f_k - \mathcal{T}_M f_{k-1} + \mathcal{T}_M f_{k-1} - Q^\star_{M'} \right\|_{\nu \times \pi} \\
&\leq \left\| f_k - \mathcal{T}_M f_{k-1} \right\|_{\nu \times \pi} + \left\| \mathcal{T}_M f_{k-1} - \mathcal{T}_{M'} f_{k-1} + \mathcal{T}_{M'} f_{k-1} - Q^\star_{M'} \right\|_{\nu \times \pi} \\
&\leq \sqrt{|\mathcal{A}|C} \left\| f_k - \mathcal{T}_M f_{k-1} \right\|_{\mu \times U} + \underbrace{\left\| \mathcal{T}_M f_{k-1} - \mathcal{T}_{M'} f_{k-1} \right\|_{\nu \times \pi}}_{\text{Deviation error}} + \left\| \mathcal{T}_{M'} f_{k-1} - Q^\star_{M'} \right\|_{\nu \times \pi} \\
&\leq \sqrt{2|\mathcal{A}|C\epsilon_n} + \delta^{(k-1)}_{M,M'} + \gamma \left\| V^{\pi_{f_{k-1}}}_{M} - V^\star_{M'} \right\|_{P'(\nu \times \pi)}
\end{aligned}
$$

Defining $\pi_{f,f_k}(s) := \operatorname{argmax}_{a \in \mathcal{A}} \max\{f(s,a), f_k(s,a)\}$, it is easy to verify that $\left\| V_f - V_{f_k} \right\|_\nu \leq \left\| f - f_k \right\|_{\nu \times \pi_{f,f_k}}$

$$
\begin{aligned}
&\leq \sqrt{2|\mathcal{A}|C\epsilon_n} + \delta^{(k-1)}_{M,M'} + \gamma \left\| f_{k-1} - Q^\star_{M'} \right\|_{P'(\nu \times \pi) \times \pi_{f_{k-1}, Q'^*}} \\
&\Rightarrow \left\| f_k - Q^\star_{M'} \right\|_{\nu \times \pi} \leq \frac{1-\gamma^k}{1-\gamma} \sqrt{2|\mathcal{A}|C\epsilon_n} + \sum_{i=1}^{k} \gamma^i \delta^{(i-1)}_{M,M'} + \gamma^k \left\| f_0 - Q^\star_{M'} \right\|_{P'(\nu \times \pi) \times \pi_{f_0, Q'^*}}
\end{aligned}
$$

Following [84], we denote ϵ_n as a bound on the error of using an empirical bellman operator (with a finite dataset) as opposed to the true bellman operator.

$$
\leq \frac{1-\gamma^k}{1-\gamma} \sqrt{2|\mathcal{A}|C\epsilon_n} + \sum_{i=1}^{k} \gamma^i \delta^{(i-1)}_{M,M'} + \frac{\gamma^k R_{\max}}{1-\gamma}
$$

Now, we can use this to bound $\left\|Q^{\hat{\pi}}_{M'} - Q^{\star}_{M'}\right\|_{\nu\times\pi}$ as follows, where $f_k := Q^{\hat{\pi}}_{M}$:

$$\begin{aligned}
\left\|Q^{\hat{\pi}}_{M'} - Q^{\star}_{M'}\right\|_{\nu\times\pi} &= \left\|Q^{\hat{\pi}}_{M'} - Q^{\hat{\pi}}_{M} + Q^{\hat{\pi}}_{M} - Q^{\star}_{M'}\right\|_{\nu\times\pi} \\
&\leq \left\|Q^{\hat{\pi}}_{M'} - Q^{\hat{\pi}}_{M}\right\|_{\nu\times\pi} + \left\|f_k - Q^{\star}_{M'}\right\|_{\nu\times\pi} \\
&\leq \underbrace{\left\|Q^{\hat{\pi}}_{M'} - Q^{\hat{\pi}}_{M}\right\|_{\nu\times\pi}}_{\text{Transfer Error}} + \underbrace{\frac{1-\gamma^k}{1-\gamma}\sqrt{2|\mathcal{A}|C\epsilon_n}}_{\text{FQI finite data error}} + \underbrace{\sum_{i=1}^{k}\gamma^i\delta^{(i-1)}_{M,M'}}_{\text{Deviation Error}} + \underbrace{\frac{\gamma^k R_{\max}}{1-\gamma}}_{\text{FQI finite iterations error}}
\end{aligned}$$

We can write the deviation error:

$$\begin{aligned}
&\delta^{(k-1)}_{M,M'} \\
&= \left\|\mathcal{T}_M f_{k-1} - \mathcal{T}_{M'} f_{k-1}\right\|_{\nu\times\pi} \qquad \text{(A.10)} \\
&= \mathbb{E}_{(s,a)\sim\nu\times\pi}\left[(\mathcal{T}_M f_{k-1})(s,a) - (\mathcal{T}_{M'} f_{k-1})(s,a)\right] \\
&= \mathbb{E}_{(s,a)\sim\nu\times\pi}\Big[\Big(R_M(s,a) - R_{M'}(s,a) + \gamma\left(\mathbb{E}_{s'\sim P_M(s,a)}\left[V_{f_{k-1}}(s')\right] - \mathbb{E}_{s'\sim P_{M'}(s,a)}\left[V_{f_{k-1}}(s')\right]\right)\Big)\Big] \\
&\leq \epsilon_R + \left\|\gamma\Big(\mathbb{E}_{s'\sim P_M(s,a)}\left[V_{f_{k-1}}(s')\right] - \mathbb{E}_{s'\sim P_{M'}(s,a)}\left[V_{f_{k-1}}(s')\right]\Big)\right\|_{\nu\times\pi}
\end{aligned}$$

Considering the term on the right (squared):

$$\begin{aligned}
&\left\|\gamma\left(\mathbb{E}_{s'\sim P_M(s,a)}\left[V_{f_{k-1}}(s')\right] - \mathbb{E}_{s'\sim P_{M'}(s,a)}\left[V_{f_{k-1}}(s')\right]\right)\right\|^2_{\nu\times\pi} \\
&= \mathbb{E}_{(s,a)\sim\nu\times\pi}\Bigg[\\
&\gamma^2\left(\int_{s'\in\mathcal{S}} V_{f_{k-1}}(s')[p_M(s'|s,a) - p_{M'}(s'|s,a)]ds'\right)^2\Bigg]
\end{aligned}$$

Using Holder's with $1/q_1 + 1/q_2 = 1$:

$$\leq \mathbb{E}_{(s,a)\sim\nu\times\pi}\left[\gamma^2\left(\int_{s'\in\mathcal{S}}|V_{f_{k-1}}(s')|^{q_2}\,ds'\right)^{2/q_2}\left(\int_{s'\in\mathcal{S}}|p_M(s'|s,a) - p_{M'}(s'|s,a)|^{q_1}\,ds'\right)^{2/q_1}\right]$$

Setting $q_1 := 1$, $q_2 := \infty$, and using $\max_s \max_a \|p_M(s,a) - p_{M'}(s,a)\|_1 \leq \epsilon_P$

$$= \mathbb{E}_{(s,a)\sim\nu\times\pi}\left[\gamma^2\epsilon_P^2 \|V_{f_{k-1}}\|_\infty^2\right]$$
$$= \gamma^2\epsilon_P^2 \|V_{f_{k-1}}\|_\infty^2$$

Now, we get the following relation for $\delta_{M,M'}^{(k-1)}$:

$$\begin{aligned} \Rightarrow \delta_{M,M'}^{(k-1)} &\leq \epsilon_R + \gamma\epsilon_P \|V_{f_{k-1}}\|_\infty \\ &\leq \epsilon_R + \frac{\gamma\epsilon_P R_{\max}}{1-\gamma} \end{aligned} \tag{A.11}$$

Now, we derive a bound on the transfer error as follows:

$$\begin{aligned} &\left\|Q_{M'}^{\hat{\pi}} - Q_M^{\hat{\pi}}\right\|_{\nu\times\pi} \\ &\leq \left\|Q_{M'}^{\hat{\pi}} - Q_M^{\hat{\pi}}\right\|_\infty \end{aligned}$$

Using the fact that $Q_{M'}^{\hat{\pi}}$ is the fixed point of $\mathcal{T}_{M'}^{\hat{\pi}}$:

$$= \left\|\left(\mathcal{T}_{M'}^{\hat{\pi}}\right)^\infty Q_M^{\hat{\pi}} - Q_M^{\hat{\pi}}\right\|_\infty$$

Using the fact that $Q_M^{\hat{\pi}}$ is the fixed point of $\mathcal{T}_M^{\hat{\pi}}$:

$$= \left\|\left(\mathcal{T}_{M'}^{\hat{\pi}}\right)^\infty Q_M^{\hat{\pi}} - \mathcal{T}_M^{\hat{\pi}} Q_M^{\hat{\pi}}\right\|_\infty$$

Adding and subtracting, followed by triangle inequality

$$\leq \left\|\left(\mathcal{T}_{M'}^{\hat{\pi}}\right)^\infty Q_M^{\hat{\pi}} - \mathcal{T}_{M'}^{\hat{\pi}} Q_M^{\hat{\pi}}\right\|_\infty + \left\|\mathcal{T}_{M'}^{\hat{\pi}} Q_M^{\hat{\pi}} - \mathcal{T}_M^{\hat{\pi}} Q_M^{\hat{\pi}}\right\|_\infty$$

Using contraction property

$$\leq \gamma \left\| \left(\mathcal{T}^{\hat{\pi}}_{M'}\right)^{\infty} Q^{\hat{\pi}}_{M} - Q^{\hat{\pi}}_{M} \right\|_{\infty} + \underbrace{\left\| \mathcal{T}^{\hat{\pi}}_{M'} Q^{\hat{\pi}}_{M} - \mathcal{T}^{\hat{\pi}}_{M} Q^{\hat{\pi}}_{M} \right\|_{\infty}}_{\text{Deviation Error}}$$

Bounding the deviation error same way as Eq. (A.10) to Eq. (A.11):

$$\leq \gamma \left\| \left(\mathcal{T}^{\hat{\pi}}_{M'}\right)^{\infty} Q^{\hat{\pi}}_{M} - Q^{\hat{\pi}}_{M} \right\|_{\infty} + \epsilon_R + \frac{\gamma \epsilon_P R_{\max}}{1-\gamma}$$

Using the fact that $Q^{\hat{\pi}}_{M'}$ is the fixed point of $\mathcal{T}^{\hat{\pi}}_{M'}$:

$$\begin{aligned}
&\leq \gamma \left\| Q^{\hat{\pi}}_{M'} - Q^{\hat{\pi}}_{M} \right\|_{\infty} + \epsilon_R + \frac{\gamma \epsilon_P R_{\max}}{1-\gamma} \\
&\Rightarrow \left\| Q^{\hat{\pi}}_{M'} - Q^{\hat{\pi}}_{M} \right\|_{\infty} \leq \gamma \left\| Q^{\hat{\pi}}_{M'} - Q^{\hat{\pi}}_{M} \right\|_{\infty} + \epsilon_R + \frac{\gamma \epsilon_P R_{\max}}{1-\gamma} \\
&\Rightarrow \left\| Q^{\hat{\pi}}_{M'} - Q^{\hat{\pi}}_{M} \right\|_{\infty} \leq \frac{1}{1-\gamma} \left[\epsilon_R + \frac{\gamma \epsilon_P R_{\max}}{1-\gamma} \right]
\end{aligned}$$

Substituting back the deviation error, we get the desired bound.

$$\left\| Q^{\hat{\pi}}_{M'} - Q^{\star}_{M'} \right\|_{\nu \times \pi} \tag{A.12}$$

$$\begin{aligned}
&\leq \frac{1}{1-\gamma} \Bigg[\gamma^k R_{\max} + \left(1-\gamma^k\right) \sqrt{2|\mathcal{A}|C\epsilon_n} \\
&\qquad + 2 \left(\frac{\gamma \epsilon_P R_{\max}}{1-\gamma} + \epsilon_R \right) \left(1-\gamma^k\right) \Bigg]
\end{aligned} \tag{A.13}$$

$$\leq \frac{1}{1-\gamma} \left[\gamma^k R_{\max} + O\left(\sqrt{|\mathcal{A}|\, \epsilon^{(n)}} \right) + 2\epsilon_R + \frac{2\epsilon_P R_{\max}}{1-\gamma} \right]$$

□

APPENDIX B

APPENDIX FOR MODEL-ADVANTAGE OPTIMIZATION FOR MODEL-BASED REINFORCEMENT LEARNING

B.1 Hyperparameter selection and choices

Method: MA−L1	
Environment	α value
Pendulum	0.05
Acrobot	0.05
CartPole	0.05
Reacher	0.05
Ant	0.5
Swimmer	0.5
HalfCheetah	0.05
Hopper	0.05

(a)

Method: MA−VAML	
Environment	α value
Pendulum	5e-9
Acrobot	5e-9
CartPole	5e-9
Reacher	1e-10
Ant	1e-10
Swimmer	1e-10
HalfCheetah	1e-10
Hopper	1e-11

(b)

Table B.1: Choices for hyperparameters α for all value aware methods. Note that these hyperparameters were automatically selected based on highest average return over 3 random seeds at 200K time steps and over a log-scaled range of values for each method.

Our proposed objective is $\mathcal{L}_1^U$ and the VAML objective is $\mathcal{L}_2^{\text{VAML}}$. In practise, we scale each objective with a scalar α and pick the best value of α automatically for each objective and environment. In the automatic selection procedure – we run a sweep over 9 values of α in log-scale ($[5e-1, 1e-1, 5e-2, \ldots, 5e-9]$ for MA−L1 and a lower scale (shifted lower by $1e-5$) for MA−VAMl) for each method and environment with 3 random seeds and use the average return at 200K time steps for selecting the best value. See Table B.1 for all the selected hyperparameters.

B.2 GPU and Server details

Total Compute. We ran each experiment configuration such that multiple runs shared a single GPU and up to 6 CPU cores. For MuJoCo environments, 2-3 runs were sharing resources whereas for non-MuJoCo environments such as `Pendulum`, `CartPole`, etc up to 5 runs were sharing resources. Each run refers to a single random seed, and 5 random seeds were used for the final reported results for each line plotted in the return vs time step figures.

Type of Compute. We ran all experiments on an internal server with varied GPU types of either the NVIDIA Titan X or NVIDIA 2080 Ti graphics card.

REFERENCES

[1] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn, "Combo: Conservative offline model-based policy optimization," *Advances in neural information processing systems*, vol. 34, pp. 28 954–28 967, 2021.

[2] R. Qin *et al.*, "Neorl: A near real-world benchmark for offline reinforcement learning," *arXiv preprint arXiv:2102.00714*, 2021.

[3] A. Kumar, A. Singh, S. Tian, C. Finn, and S. Levine, "A workflow for offline model-free robotic reinforcement learning," *arXiv preprint arXiv:2109.10813*, 2021.

[4] M. Chevalier-Boisvert, L. Willems, and S. Pal, *Minimalistic gridworld environment for openai gym*, https://github.com/maximecb/gym-minigrid, 2018.

[5] A.-m. Farahmand, "Iterative value-aware model learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 9072–9083.

[6] Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, and T. Ma, "Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees," *arXiv preprint arXiv:1807.03858*, 2018.

[7] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," in *Advances in Neural Information Processing Systems*, 2019, pp. 12 498–12 509.

[8] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[9] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[10] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[11] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[12] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *International Conference on Machine Learning*, 2016, pp. 2829–2838.

[13] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[14] M. M. Afsar, T. Crump, and B. Far, "Reinforcement learning based recommender systems: A survey," *arXiv preprint arXiv:2101.06286*, 2021.

[15] V. Uc-Cetina, N. Navarro-Guerrero, A. Martin-Gonzalez, C. Weber, and S. Wermter, "Survey on reinforcement learning for language processing," *arXiv preprint arXiv:2104.05565*, 2021.

[16] T. Harvey and A. Lukas, "Particle physics model building with reinforcement learning," *arXiv preprint arXiv:2103.04759*, 2021.

[17] N. Stephenson *et al.*, "Survey of machine learning techniques in drug discovery," *Current drug metabolism*, vol. 20, no. 3, pp. 185–193, 2019.

[18] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based reinforcement learning: A survey," *arXiv preprint arXiv:2006.16712*, 2020.

[19] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[20] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *arXiv preprint arXiv:1805.12114*, 2018.

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

[22] C. Grimm, A. Barreto, S. Singh, and D. Silver, "The value equivalence principle for model-based reinforcement learning," *arXiv preprint arXiv:2011.03506*, 2020.

[23] A. Ayoub, Z. Jia, C. Szepesvari, M. Wang, and L. Yang, "Model-based reinforcement learning with value-targeted regression," in *International Conference on Machine Learning*, PMLR, 2020, pp. 463–474.

[24] D. Janz, J. Hron, P. Mazur, K. Hofmann, J. M. Hernández-Lobato, and S. Tschiatschek, "Successor uncertainties: Exploration and uncertainty in temporal difference learning," *Advances in Neural Information Processing Systems*, vol. 32, pp. 4507–4516, 2019.

[25] A. Barreto *et al.*, "Successor features for transfer in reinforcement learning," *Advances in neural information processing systems*, vol. 30, 2017.

[26] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[27] Â. G. Lovatto, T. P. Bueno, D. D. Mauá, and L. N. Barros, "Decision-aware model learning for actor-critic methods: When theory does not meet practice," in *Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops, 2020*, PMLR, 2020.

[28] H. Wang, S. Zheng, C. Xiong, and R. Socher, "On the generalization gap in reparameterizable reinforcement learning," in *International Conference on Machine Learning*, 2019, pp. 6648–6658.

[29] N. Modhe, H. K. Kamath, D. Batra, and A. Kalyan, "Bridging worlds in reinforcement learning with model-advantage," *LifelongML workshop at ICML2020*, 2020.

[30] A. M. Metelli, M. Mutti, and M. Restelli, "Configurable markov decision processes," in *International Conference on Machine Learning*, PMLR, 2018, pp. 3491–3500.

[31] A. Zhang, N. Ballas, and J. Pineau, "A dissection of overfitting and generalization in continuous reinforcement learning," *arXiv preprint arXiv:1806.07937*, 2018.

[32] A. Zhang, Y. Wu, and J. Pineau, "Natural environment benchmarks for reinforcement learning," *arXiv preprint arXiv:1811.06032*, 2018.

[33] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," *arXiv preprint arXiv:1812.02341*, 2018.

[34] S. Witty, J. K. Lee, E. Tosch, A. Atrey, M. Littman, and D. Jensen, "Measuring and characterizing generalization in deep reinforcement learning," *arXiv preprint arXiv:1812.02868*, 2018.

[35] E. Bengio, J. Pineau, and D. Precup, "Interference and generalization in temporal difference learning," *arXiv preprint arXiv:2003.06350*, 2020.

[36] C. Zhao, O. Siguad, F. Stulp, and T. M. Hospedales, "Investigating generalisation in continuous deep reinforcement learning," *arXiv preprint arXiv:1902.07015*, 2019.

[37] A. Irpan and X. Song, "The principle of unchanged optimality in reinforcement learning generalization," *arXiv preprint arXiv:1906.00336*, 2019.

[38] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," *arXiv preprint arXiv:1912.01588*, 2019.

[39] X. Song, Y. Jiang, Y. Du, and B. Neyshabur, "Observational overfitting in reinforcement learning," *arXiv preprint arXiv:1912.02975*, 2019.

[40] R. B. Slaoui, W. R. Clements, J. N. Foerster, and S. Toth, "Robust domain randomization for reinforcement learning," *arXiv preprint arXiv:1910.10537*, 2019.

[41] Z. Liu, X. Li, B. Kang, and T. Darrell, "Regularization matters in policy optimization," *arXiv preprint arXiv:1910.09191*, 2019.

[42] C. Ye, A. Khalifa, P. Bontrager, and J. Togelius, "Rotation, translation, and cropping for zero-shot generalization," *arXiv preprint arXiv:2001.09908*, 2020.

[43] M. Igl *et al.*, "Generalization in reinforcement learning with selective noise injection and information bottleneck," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 956–13 968.

[44] K. Lee, K. Lee, J. Shin, and H. Lee, "Network randomization: A simple technique for generalization in deep reinforcement learning," in *International Conference on Learning Representations. https://openreview. net/forum*, 2020.

[45] S. Kakade and J. Langford, "Approximately optimal approximate reinforcement learning," in *ICML*, vol. 2, 2002, pp. 267–274.

[46] A. D. Edwards *et al.*, "Estimating q (s, s') with deep deterministic dynamics gradients," *arXiv preprint arXiv:2002.09505*, 2020.

[47] S. Kakade, M. J. Kearns, and J. Langford, "Exploration in metric state spaces," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 306–312.

[48] M. G. Azar, R. Munos, and B. Kappen, "On the sample complexity of reinforcement learning with a generative model," *arXiv preprint arXiv:1206.6461*, 2012.

[49] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, "Is q-learning provably efficient?" *arXiv preprint arXiv:1807.03765*, 2018.

[50] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," Dec. 2013. arXiv: 1312.5602 [cs.LG].

[51] N. Modhe, H. Kamath, D. Batra, and A. Kalyan, "Model-advantage optimization for model-based reinforcement learning," *arXiv preprint arXiv:2106.14080*, 2021.

[52] T. Wang *et al.*, "Benchmarking model-based reinforcement learning," *arXiv preprint arXiv:1907.02057*, 2019.

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

[54] K. Lee, Y. Seo, S. Lee, H. Lee, and J. Shin, "Context-aware dynamics model for generalization in model-based reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2020, pp. 5757–5766.

[55] S. Nair, S. Savarese, and C. Finn, "Goal-aware prediction: Learning to model what matters," in *International Conference on Machine Learning*, PMLR, 2020, pp. 7207–7219.

[56] M. Tomar, A. Zhang, R. Calandra, M. E. Taylor, and J. Pineau, "Model-invariant state abstractions for model-based reinforcement learning," *arXiv preprint arXiv:2102.09850*, 2021.

[57] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims, "Morel: Model-based offline reinforcement learning," *arXiv preprint arXiv:2005.05951*, 2020.

[58] A. Nagabandi *et al.*, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," *arXiv preprint arXiv:1803.11347*, 2018.

[59] X. Lu, K. Lee, P. Abbeel, and S. Tiomkin, "Dynamics generalization via information bottleneck in deep reinforcement learning," *arXiv preprint arXiv:2008.00614*, 2020.

[60] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Machine learning proceedings 1990*, Elsevier, 1990, pp. 216–224.

[61] J. Schrittwieser *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *CoRR*, vol. abs/1911.08265, 2019. arXiv: 1911.08265.

[62] G. Brockman *et al.*, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[63] R. S. Sutton, C. Szepesvári, A. Geramifard, and M. P. Bowling, "Dyna-style planning with linear function approximation and prioritized sweeping," *arXiv preprint arXiv:1206.3285*, 2012.

[64] M. G. Azar, R. Munos, and H. J. Kappen, "Minimax pac bounds on the sample complexity of reinforcement learning with a generative model," *Machine learning*, vol. 91, no. 3, pp. 325–349, 2013.

[65] M. G. Azar, I. Osband, and R. Munos, "Minimax regret bounds for reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 263–272.

[66] S. Ross and J. A. Bagnell, "Agnostic system identification for model-based reinforcement learning," *arXiv preprint arXiv:1203.1007*, 2012.

[67] Y.-H. Wu, T.-H. Fan, P. J. Ramadge, and H. Su, "Model imitation for model-based reinforcement learning," *arXiv preprint arXiv:1909.11821*, 2019.

[68] J. Oh, S. Singh, and H. Lee, "Value prediction network," *arXiv preprint arXiv:1707.03497*, 2017.

[69] D. Silver *et al.*, "The predictron: End-to-end learning and planning," in *International Conference on Machine Learning*, PMLR, 2017, pp. 3191–3199.

[70] R. Abachi, "Policy-aware model learning for policy gradient methods," Ph.D. dissertation, University of Toronto (Canada), 2020.

[71] M. Hessel *et al.*, "Muesli: Combining improvements in policy optimization," *CoRR*, vol. abs/2104.06159, 2021. arXiv: 2104.06159.

[72] J. Schrittwieser, T. Hubert, A. Mandhane, M. Barekatain, I. Antonoglou, and D. Silver, "Online and offline reinforcement learning by planning with a learned model," *CoRR*, vol. abs/2104.06294, 2021. arXiv: 2104.06294.

[73] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[74] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Machine learning*, vol. 49, no. 2-3, pp. 209–232, 2002.

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

[76] T. Yu *et al.*, "Mopo: Model-based offline policy optimization," *arXiv preprint arXiv:2005.13239*, 2020.

[77] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[78] G. J. Gordon, "Stable function approximation in dynamic programming," in *Machine learning proceedings 1995*, Elsevier, 1995, pp. 261–268.

[79] M. Riedmiller, "Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method," in *European conference on machine learning*, Springer, 2005, pp. 317–328.

[80] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[81] T. Yu, A. Kumar, Y. Chebotar, K. Hausman, C. Finn, and S. Levine, "How to leverage unlabeled data in offline reinforcement learning," *arXiv preprint arXiv:2202.01741*, 2022.

[82] H. Zhang and Y. Guo, "Generalization of reinforcement learning with policy-aware adversarial data augmentation," *arXiv preprint arXiv:2106.15587*, 2021.

[83] S. Gu *et al.*, "A review of safe reinforcement learning: Methods, theory and applications," *arXiv preprint arXiv:2205.10330*, 2022.

[84] R. Munos and C. Szepesvári, "Finite-time bounds for fitted value iteration," *Journal of Machine Learning Research*, vol. 9, no. May, pp. 815–857, 2008.