MOReL: Model-Based Offline Reinforcement Learning

Rahul Kidambi*
Cornell University, Ithaca
rkidambi@cornell.edu

Aravind Rajeswaran*
University of Washington, Seattle
Google Brain, Mountain View
aravraj@cs.washington.edu

Praneeth Netrapalli
Microsoft Research, India
praneeth@microsoft.com

Thorsten Joachims
Cornell University, Ithaca
tj@cs.cornell.edu

Abstract

In offline reinforcement learning (RL), the goal is to learn a highly rewarding policy based solely on a dataset of historical interactions with the environment. The ability to train RL policies offline would greatly expand where RL can be applied, its data efficiency, and its experimental velocity. Prior work in offline RL has been confined almost exclusively to model-free RL approaches. In this work, we present MOReL, an algorithmic framework for model-based offline RL. This framework consists of two steps: (a) learning a *pessimistic MDP* (P-MDP) using the offline dataset; (b) learning a near-optimal policy in this P-MDP. The learned P-MDP has the property that for *any* policy, the performance in the real environment is approximately lower-bounded by the performance in the P-MDP. This enables it to serve as a good surrogate for purposes of policy evaluation and learning, and overcome common pitfalls of model-based RL like model exploitation. Theoretically, we show that MOReL is minimax optimal (up to log factors) for offline RL. Through experiments, we show that MOReL matches or exceeds state-of-the-art results in widely studied offline RL benchmarks. Moreover, the modular design of MOReL enables future advances in its components (e.g., in model learning, planning etc.) to directly translate into improvements for offline RL.

1 Introduction

The fields of computer vision and NLP have seen tremendous advances by utilizing large-scale offline datasets [1, 2, 3] for training and deploying deep learning models [4, 5, 6, 7]. In contrast, reinforcement learning (RL) [8] is typically viewed as an online learning process. The RL agent iteratively collects data through interactions with the environment while learning the policy. Unfortunately, a direct embodiment of this trial and error learning is often inefficient and feasible only with a simulator [9, 10, 11]. Similar to progress in other fields of AI, the ability to learn from offline datasets may hold the key to unlocking the sample efficiency and widespread use of RL agents.

Offline RL, also known as batch RL [12], involves learning a highly rewarding policy using only a static offline dataset collected by one or more data logging (behavior) policies. Since the data has already been collected, offline RL abstracts away data collection or exploration, and allows prime

*Equal Contributions. Correspond to rkidambi@cornell.edu and aravraj@cs.washington.edu. This version of the paper extends the results presented at NeurIPS 2020 through the addition of results in the D4RL benchmark suite and by expanding the scope of Lemma 3.

Preprint. Under review. Initial version appeared on arXiv on May 12, 2020.

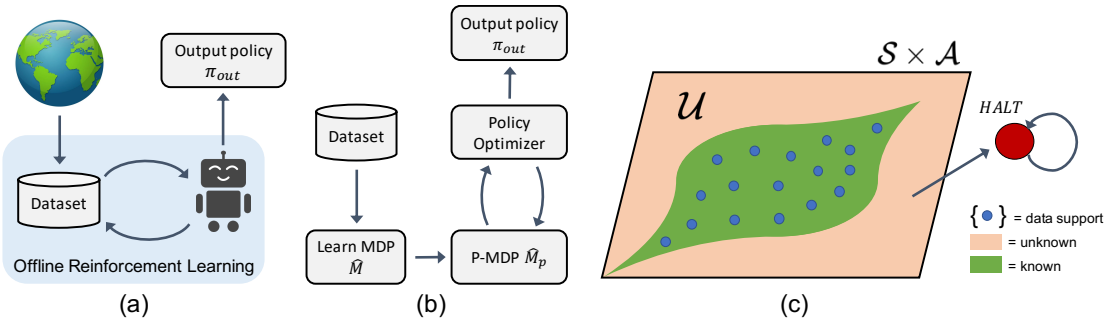

Figure 1: (a) Illustration of the offline RL paradigm. (b) Illustration of our framework, MOReL, which learns a pessimistic MDP (P-MDP) from the dataset and uses it for policy search. (c) Illustration of the P-MDP, which partitions the state-action space into known (green) and unknown (orange) regions, and also forces a transition to a low reward absorbing state (HALT) from unknown regions. Blue dots denote the support in the dataset. Function approximation and generalization allows us to learn about states not numerically identical to data support. See algorithm 1 for more details.

focus on data-driven learning of policies. This abstraction is suitable for safety sensitive applications like healthcare and industrial automation where careful oversight by a domain expert is necessary for taking exploratory actions or deploying new policies [13, 14]. Additionally, large historical datasets are readily available in domains like autonomous driving and recommendation systems, where offline RL may be used to improve upon currently deployed policies.

Due to use of static dataset, offline RL faces unique challenges. Over the course of learning, the agent has to evaluate and reason about various candidate policy updates. This *offline policy evaluation* is particularly challenging due to deviation between the state visitation distribution of the candidate policy and the logging policy. Furthermore, this difficulty is exacerbated over the course of learning as the candidate policies increasingly deviate from the logging policy. This change in distribution, as a result of policy updates, is typically called *distribution shift* and constitutes a major challenge in offline RL. Recent studies show that directly using off-policy RL algorithms with an offline dataset yields poor results due to distribution shift and function approximation errors [15, 16, 17]. To overcome this, prior works have proposed modifications like Q-network ensembles [15, 18] and regularization towards the data logging policy [19, 16, 18]. Most notably, prior work in offline RL has been confined almost exclusively to model-free methods [20, 15, 16, 19, 17, 18, 21].

Model-based RL (MBRL) presents an alternate set of approaches involving the learning of approximate dynamics models which can subsequently be used for policy search. MBRL enables the use of generic priors like smoothness and physics [22] for model learning, and a wide variety of planning algorithms [23, 24, 25, 26, 27]. As a result, MBRL algorithms have been highly sample efficient for online RL [28, 29]. However, direct use of MBRL algorithms with offline datasets can prove challenging, again due to the distribution shift issue. In particular, since the dataset may not span the entire state-action space, the learned model is unlikely to be globally accurate. As a result, planning using a learned model without any safeguards against model inaccuracy can result in *"model exploitation"* [30, 31, 29, 28], yielding poor results [32]. In this context, we study the pertinent question of how to effectively regularize and adapt model-based methods for offline RL.

Our Contributions: The principal contribution of our work is the development of MOReL (Model-based Offline Reinforcement Learning), a novel model-based framework for offline RL (see figure 1 for an overview). MOReL enjoys rigorous theoretical guarantees, enables transparent algorithm design, and offers state of the art (SOTA) results on widely studied offline RL benchmarks.

- MOReL consists of two modular steps: (a) learning a *pessimistic MDP* (P-MDP) using the offline dataset; and (b) learning a near-optimal policy for the P-MDP. For *any* policy, the performance in the true MDP (environment) is approximately lower bounded by the performance in the P-MDP, making it a suitable surrogate for purposes of policy evaluation and learning. This also guards against model exploitation, which often plagues MBRL.
- The P-MDP partitions the state space into "known" and "unknown" regions, and uses a large negative reward for unknown regions. This provides a regularizing effect during policy learning by heavily penalizing policies that visit unknown states. Such a regularization in the space of state visitations, afforded by a model-based approach, is particularly well suited for offline RL. In

contrast, model-free algorithms [16, 18] are forced to regularize the policies directly towards the data logging policy, which can be overly conservative.
- Theoretically, we establish upper bounds for the sub-optimality of a policy learned with MOReL, and a lower-bound for the sub-optimality of a policy learnable by *any* offline RL algorithm. We find that these bounds match upto log factors, suggesting that MOReL is nearly minimax optimal.
- We evaluate MOReL on standard benchmark tasks used for offline RL. MOReL obtains SOTA results in 12 out of 20 environment-dataset configurations, and performs competitively in the rest. In contrast, the best prior algorithm [18] obtains SOTA results in only 5 (out of 20) configurations.

In addition, this version of the paper extends the results presented at NeurIPS 2020 through addition of results in the D4RL benchmark suite and by expanding the scope of Lemma 3.

2 Related Work

Offline RL dates to at least the work of Lange et al. [12], and has applications in healthcare [33, 34, 35], recommendation systems [36, 37, 38, 39], dialogue systems [40, 19, 41], and autonomous driving [42]. Algorithms for offline RL typically fall under three categories. The first approach utilizes **importance sampling** and is popular in contextual bandits [43, 36, 37]. For full offline RL, Liu et al. [44] perform planning with learned importance weights [45, 46, 47] while using a notion of pessimism for regularization. However, Liu et al. [44] don't explicitly consider generalization and their guarantees become degenerate if the logging policy does not span the support of the optimal policy. In contrast, our approach accounts for generalization, leads to stronger theoretical guarantees, and obtains SOTA results on challenging offline RL benchmarks. The second, and perhaps most popular approach is based on **approximate dynamic programming (ADP)**. Recent works have proposed modification to standard ADP algorithms [48, 49, 50, 51] towards stabilizing Bellman targets with ensembles [17, 15, 19] and regularizing the learned policy towards the data logging policy [15, 16, 18]. ADP-based offline RL has also be studied theoretically [26, 52]. However, these works again don't study the impact of support mismatch between logging policy and optimal policy. Finally, **model-based RL** has been explored only sparsely for offline RL in literature [32, 53] (see appendix for details). The work of Ross and Bagnell [32] considered a straightforward approach of learning a model from offline data, followed by planning. They showed that this can have arbitrarily large sub-optimality. In contrast, our work develops a new framework utilizing the notion of pessimism, and shows both theoretically and experimentally that MBRL can be highly effective for offline RL. Concurrent to our work, Yu et al. [54] also study a model-based approach to offline RL.

A cornerstone of MOReL is the P-MDP which partitions the state space into known and unknown regions. Such a hard partitioning was considered in early works like E^3 [55], R-MAX [56], and metric-E^3 [57], but was not used to encourage pessimism. Similar ideas have been explored in related settings like online RL [58, 59] and imitation learning [60]. Our work differs in its focus on offline RL, where we show the P-MDP construction plays a crucial role. Moreover, direct practical instantiations of E^3 and metric-E^3 with function approximation have remained elusive.

3 Problem Formulation

A **Markov Decision Process (MDP)** is represented by $\mathcal{M} = \{S, A, r, P, \rho_0, \gamma\}$, where, S is the state-space, A is the action-space, $r : S \times A \to [-R_{\max}, R_{\max}]$ is the reward function, $P : S \times A \times S \to \mathbb{R}_+$ is the transition kernel, ρ_0 is the initial state distribution, and γ the discount factor. A policy defines a mapping from states to a probability distribution over actions, $\pi : S \times A \to \mathbb{R}_+$. The goal is to obtain a policy that maximizes expected performance with states sampled according to ρ_0, i.e.:

$$\max_{\pi} \; J_{\rho_0}(\pi, \mathcal{M}) := \mathbb{E}_{s\sim\rho_0}\left[V^\pi(s, \mathcal{M})\right], \text{ where, } V^\pi(s, \mathcal{M}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s\right]. \quad (1)$$

To avoid notation clutter, we suppress the dependence on ρ_0 when understood from context, i.e. $J(\pi, \mathcal{M}) \equiv J_{\rho_0}(\pi, \mathcal{M})$. We denote the optimal policy using $\pi^* := \arg\max_\pi J_{\rho_0}(\pi, \mathcal{M})$. Typically, a class of parameterized policies $\pi_\theta \in \Pi(\Theta)$ are considered, and the parameters θ are optimized.

In **offline RL**, we are provided with a static dataset of interactions with the environment consisting of $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$. The data can be collected using one or more logging (or behavioral)

policies denoted by π_b. We do not assume logging policies are known in our formulation. Given $\mathcal{D}$, the goal in offline RL is to output a π_{out} with minimal sub-optimality, i.e. $J(\pi^*, \mathcal{M}) - J(\pi_{\text{out}}, \mathcal{M})$. In general, it may not be possible to learn the optimal policy with a static dataset (see section 4.1). Thus, we aim to design algorithms that would result in as low sub-optimality as possible.

Model-Based RL (MBRL) involves learning an MDP $\hat{\mathcal{M}} = \{S, A, r, \hat{P}, \hat{\rho}_0, \gamma\}$ which uses the learned transitions $\hat{P}$ instead of the true transition dynamics P. In this paper, we assume the reward function r is known and use it in $\hat{M}$. If $r(\cdot)$ is unknown, it can also be learned from data. The initial state distribution $\hat{\rho}_0$ can either be learned from the data or ρ_0 can be used if known. Analogous to $\mathcal{M}$, we use $J_{\hat{\rho}_0}(\pi, \hat{\mathcal{M}})$ or simply $J(\pi, \hat{\mathcal{M}})$ to denote performance of π in $\hat{M}$.

4 Algorithmic Framework

For ease of exposition and clarity, we first begin by presenting an idealized version of `MOReL`, for which we also establish theoretical guarantees. Subsequently, we describe a practical version of `MOReL` that we use in our experiments. Algorithm 1 presents the broad framework of `MOReL`. We now study each component of `MOReL` in greater detail.

Algorithm 1 `MOReL`: Model Based Offline Reinforcement Learning

1: **Require** Dataset $\mathcal{D}$
2: Learn approximate dynamics model $\hat{P} : S \times A \to S$ using $\mathcal{D}$.
3: Construct α-USAD, $U^\alpha : S \times A \to \{\text{TRUE}, \text{FALSE}\}$ using $\mathcal{D}$ (see Definition 1).
4: Construct the *pessimistic* MDP $\hat{\mathcal{M}}_p = \{S \cup \text{HALT}, A, r_p, \hat{P}_p, \hat{\rho}_0, \gamma\}$ (see Definition 2).
5: (OPTIONAL) Use a behavior cloning approach to estimate the behavior policy $\hat{\pi}_b$.
6: $\pi_{\text{out}} \leftarrow \text{PLANNER}(\hat{\mathcal{M}}_p, \pi_{\text{init}} = \hat{\pi}_b)$
7: **Return** π_{out}.

Learning the dynamics model: The first step involves using the offline dataset to learn an approximate dynamics model $\hat{P}(\cdot|s,a)$. This can be achived through maximum likelihood estimation or other techniques from generative and dynamics modeling [61, 62, 63]. Since the offline dataset may not span the entire state space, the learned model may not be globally accurate. So, a naïve MBRL approach that directly plans with the learned model may over-estimate rewards in unfamiliar parts of the state space, resulting in a highly sub-optimal policy [32]. We overcome this with the next step.

Unknown state-action detector (USAD): We partition the state-action space into known and unknown regions based on the accuracy of learned model as follows.

Definition 1. *(α-USAD) Given a state-action pair (s,a), define an unknown state action detector as:*

$$U^\alpha(s,a) = \begin{cases} \textit{FALSE (i.e. Known)} & \text{if } D_{TV}\left(\hat{P}(\cdot|s,a), P(\cdot|s,a)\right) \leq \alpha \textit{ can be guaranteed} \\ \textit{TRUE (i.e. Unknown)} & \textit{otherwise} \end{cases} \tag{2}$$

Here $D_{TV}\left(\hat{P}(\cdot|s,a), P(\cdot|s,a)\right)$ denotes the total variation distance between $\hat{P}(\cdot|s,a)$ and $P(\cdot|s,a)$. Intuitively, USAD provides confidence about where the learned model is accurate. It flags state-actions for which the model is guarenteed to be accurate as "known", while flagging state-actions where such a guarantee cannot be ascertained as "unknown". Note that USAD is based on the ability to guarantee the accuracy, and is not an inherent property of the model. In other words, there could be states where the model is actually accurate, but flagged as unknown due to the agent's inability to guarantee accuracy. Two factors contribute to USAD's effectiveness: (a) data availability: having sufficient data points "close" to the query; (b) quality of representations: certain representations, like those based on physics, can lead to better generalization guarantees. This suggests that larger datasets and research in representation learning can potentially enable stronger offline RL results.

Pessimistic MDP construction: We now construct a pessimistic MDP (P-MDP) using the learned model and USAD, which penalizes policies that venture into unknown parts of state-action space.

Definition 2. *The (α, κ)-pessimistic MDP is described by $\hat{\mathcal{M}}_p := \{S \cup \textit{HALT}, A, r_p, \hat{P}_p, \hat{\rho}_0, \gamma\}$. Here, S and A are states and actions in the MDP $\mathcal{M}$. HALT is an additional absorbing state we*

introduce into the state space of $\hat{\mathcal{M}}_p$. $\hat{\rho}_0$ is the initial state distribution learned from the dataset $\mathcal{D}$. γ is the discount factor (same as $\mathcal{M}$). The modified reward and transition dynamics are given by:

$$\hat{P}_p(s'|s,a) = \begin{cases} \delta(s' = \text{HALT}) & \text{if } U^\alpha(s,a) = \text{TRUE} \\ & \text{or } \ s = \text{HALT} \\ \hat{P}(s'|s,a) & \textit{otherwise} \end{cases} \qquad r_p(s,a) = \begin{cases} -\kappa & \text{if } s = \text{HALT} \\ r(s,a) & \textit{otherwise} \end{cases}$$

$\delta(s' = \text{HALT})$ is the Dirac delta function, which forces the MDP to transition to the absorbing state HALT. For unknown state-action pairs, we use a reward of $-\kappa$, while all known state-actions receive the same reward as in the environment. The P-MDP heavily punishes policies that visit unknown states, thereby providing a safeguard against distribution shift and model exploitation.

Planning: The final step in MOReL is to perform planning in the P-MDP defined above. For simplicity, we assume a planning oracle that returns an ϵ_π-sub-optimal policy in the P-MDP. A number of algorithms based on MPC [23, 64], search-based planning [65, 25], dynamic programming [49, 26], or policy optimization [27, 51, 66, 67] can be used to approximately realize this..

4.1 Theoretical Results

In order to state our results, we begin by defining the notion of hitting time.

Definition 3. *(Hitting time) Given an MDP $\mathcal{M}$, starting state distribution ρ_0, state-action pair (s,a) and a policy π, the* hitting time $T^\pi_{(s,a)}$ *is defined as the random variable denoting the first time action a is taken at state s by π on $\mathcal{M}$, and is equal to ∞ if a is never taken by π from state s. For a set of state-action pairs $\mathcal{S} \subseteq S \times A$, we define $T^\pi_{\mathcal{S}} \stackrel{def}{=} \min_{(s,a)\in\mathcal{S}} T^\pi_{(s,a)}$.*

We are now ready to present our main result with the proofs deferred to the appendix.

Theorem 1. *(Policy value with pessimism) The value of any policy π on the original MDP $\mathcal{M}$ and its $(\alpha, R_{\max})$-pessimistic MDP $\hat{\mathcal{M}}_p$ satisfies:*

$$J_{\hat{\rho_0}}(\pi, \hat{\mathcal{M}}_p) \geq J_{\rho_0}(\pi, \mathcal{M}) - \frac{2R_{max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho_0}) - \frac{2\gamma R_{max}}{(1-\gamma)^2} \cdot \alpha - \frac{2R_{max}}{1-\gamma} \cdot \mathbb{E}\left[\gamma^{T^\pi_{\mathcal{U}}}\right], \text{ and}$$

$$J_{\hat{\rho_0}}(\pi, \hat{\mathcal{M}}_p) \leq J_{\rho_0}(\pi, \mathcal{M}) + \frac{2R_{max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho_0}) + \frac{2\gamma R_{max}}{(1-\gamma)^2} \cdot \alpha,$$

where $T^\pi_{\mathcal{U}}$ denotes the hitting time of unknown states $\mathcal{U} \stackrel{def}{=} \{(s,a) : U^\alpha(s,a) = TRUE\}$ by π on $\mathcal{M}$.

Theorem 1 can be used to bound the suboptimality of output policy π_{out} of Algorithm 1.

Corollary 2. *Suppose PLANNER in Algorithm 1 returns an ϵ_π sub-optimal policy. Then, we have*

$$J_{\rho_0}(\pi^*, \mathcal{M}) - J_{\rho_0}(\pi_{out}, \mathcal{M}) \leq \epsilon_\pi + \frac{4R_{max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho_0}) + \frac{4\gamma R_{max}}{(1-\gamma)^2} \cdot \alpha + \frac{2R_{max}}{1-\gamma} \cdot \mathbb{E}\left[\gamma^{T^{\pi^*}_{\mathcal{U}}}\right].$$

Theorem 1 indicates that the difference in any policy π's value in the $(\alpha, R_{\max})$ pessimistic MDP $\hat{\mathcal{M}}_p$ and the original MDP $\mathcal{M}$ depends on: i) the total variation distance between the true and learned starting state distribution $D_{TV}(\rho_0, \hat{\rho_0})$, ii) the maximum total variation distance α between the learned model $\hat{P}(\cdot|s,a)$ and the true model $P(\cdot|s,a)$ over all *known* states i.e., $\{(s,a)|U^\alpha(s,a) = \text{FALSE}\}$ and, iii) the hitting time $T^{\pi^*}_{\mathcal{U}}$ of unknown states $\mathcal{U}$ on the original MDP $\mathcal{M}$ under the optimal policy π^*. As the dataset size increases, $D_{TV}(\rho_0, \hat{\rho_0})$ and α approach zero, indicating $\mathbb{E}\left[\gamma^{T^{\pi^*}_{\mathcal{U}}}\right]$ determines the sub-optimality in the limit. For comparison to prior work, Lemma 5 in Appendix A bounds this quantity in terms of state-action visitation distribution, which for a policy π on $\mathcal{M}$ is expressed as $d^{\pi,\mathcal{M}}(s,a) \stackrel{\text{def}}{=} (1-\gamma)\sum_{t=0}^{\infty} \gamma^t P(s_t = s, a_t = a | s_0 \sim \rho_0, \pi, \mathcal{M})$. Furthermore, we can also show that MOReL learns a policy that improves over the behavioral policy with high probability. The following lemma presents both of these results:

Lemma 3. *(Upper bound; MOReL improves over the behavioral policy) Suppose $\rho_{0,min} > 0$, $p_{min} > 0$ and $d^{\pi_b}_{min} > 0$ are the smallest non-zero elements of initial distribution ρ_0, state transition probabilities $P(\cdot|s,a)$, and discounted state probability distribution $d^{\pi_b,\mathcal{M}}(s,a)$ respectively. If the dataset $\mathcal{D}$*

consists of $n \geq \frac{C}{\left(d^{\pi_b}_{min}\right)^2} \cdot \log \frac{1}{\delta d^{\pi_b}_{min}}$ *independent trajectories sampled according to a behavior policy* π_b *with initial distribution* ρ_0*, then the output* π_{out} *of Algorithm 1 satisfies:*

$$J_{\rho_0}(\pi_b, \mathcal{M}) - J_{\rho_0}(\pi_{out}, \mathcal{M}) \leq \epsilon_\pi + \epsilon_n, \text{ and}$$

$$J_{\rho_0}(\pi^*, \mathcal{M}) - J_{\rho_0}(\pi_{out}, \mathcal{M}) \leq \epsilon_\pi + \frac{2R_{max}}{1-\gamma} \cdot \mathbb{E}\left[\gamma^{T^{\pi^*}_{\mathcal{U}}}\right] + \epsilon_n \leq \epsilon_\pi + \frac{2R_{\max}}{(1-\gamma)^2} \cdot d^{\pi^*, \mathcal{M}}(\mathcal{U}) + \epsilon_n,$$

with probability at least $1 - C\delta$*, where* C *is a large enough constant and*

$$\epsilon_n \overset{def}{=} \frac{4CR_{max}}{(1-\gamma)\rho_{0,min}} \cdot \sqrt{\frac{\log \frac{1}{\delta \rho_{0,min}}}{n}} + \frac{4C\gamma R_{max}}{(1-\gamma)^2 p_{min}} \cdot \sqrt{\frac{\log \frac{1}{\delta p_{min} d^{\pi_b}_{min}}}{d^{\pi_b}_{min} \cdot n}}$$

is an error term related to finite samples that goes to zero as $n \to \infty$*.*

The bound consists of three terms: (i) a sampling error term ϵ_n which decreases with larger dataset sizes that is typical of offline RL; (ii) an optimization error term ϵ_π that can be made small with additional compute to find the optimal policy in the learned model; and (iii) a distribution shift term that depends on the coverage of the offline dataset and overlap with the optimal policy.

Prior results [15, 44] assume that $d^{\pi^*, \mathcal{M}}(\mathcal{U}_D) = 0$, where $\mathcal{U}_D \overset{\text{def}}{=} \{(s,a) | (s,a,r,s') \notin \mathcal{D}\} \supseteq \mathcal{U}$ is the set of state action pairs that don't occur in the offline dataset, and guarantee finding an optimal policy under this assumption. Our result significantly improves upon these in three ways: i) $\mathcal{U}_D$ is replaced by a smaller set $\mathcal{U}$, leveraging the generalization ability of learned dynamics model, ii) the sub-optimality bound is extended to the setting where full support coverage is not satisfied i.e., $d^{\pi^*, \mathcal{M}}(\mathcal{U}) > 0$, and iii) the sub-optimality bound on π_{out} is stated in terms of unknown state hitting time $T^{\pi^*}_{\mathcal{U}}$, which can be significantly better than a bound that depends only on $d^{\pi^*, \mathcal{M}}(\mathcal{U})$. To further strengthen our results, the following proposition shows that Lemma 3 is tight up to $\log$ factors.

Proposition 4. *(Lower bound) For any discount factor* $\gamma \in [0.95, 1)$*, support mismatch* $\epsilon \in \left(0, \frac{1-\gamma}{\log \frac{1}{1-\gamma}}\right]$ *and reward range* $[-R_{max}, R_{max}]$*, there is an MDP* $\mathcal{M}$*, starting state distribution* ρ_0*, optimal policy* π^* *and a dataset collection policy* π_b *such that i)* $d^{\pi^*, \mathcal{M}}(\mathcal{U}_D) \leq \epsilon$*, and ii) any policy* $\hat{\pi}$ *that is learned solely using the dataset collected with* π_b *satisfies:*

$$J_{\rho_0}(\pi^*, \mathcal{M}) - J_{\rho_0}(\hat{\pi}, \mathcal{M}) \geq \frac{R_{max}}{4(1-\gamma)^2} \cdot \frac{\epsilon}{\log \frac{1}{1-\gamma}},$$

where $\mathcal{U}_D \overset{def}{=} \{(s,a) : (s,a,r,s') \notin \mathcal{D}$ *for any* $r, s'\}$ *denotes state action pairs not in the dataset* $\mathcal{D}$*.*

We see that for $\epsilon < (1-\gamma)/(\log \frac{1}{1-\gamma})$, the lower bound obtained by Proposition 4 on the suboptimality of any offline RL algorithm matches the asymptotic (as $n \to \infty$) upper bound of Lemma 3 up to an additional log factor. For $\epsilon > (1-\gamma)/(\log \frac{1}{1-\gamma})$, Proposition 4 also implies (by choosing $\epsilon' = (1-\gamma)/(\log \frac{1}{1-\gamma}) < \epsilon$) that any offline algorithm must suffer at least constant factor suboptimality in the worst case. Finally, we note that as the size of dataset $\mathcal{D}$ increases to ∞, Theorem 1 and the optimality of PLANNER (i.e., $\epsilon_\pi = 0$) together imply that $J_{\rho_0}(\pi_{\text{out}}, \mathcal{M}) \geq J_{\rho_0}(\pi_b, \mathcal{M})$.

4.2 Practical Implementation Of MOReL

We now present a practical instantiation of MOReL (algorithm 1) utilizing a recent model-based NPG approach [28]. The principal difference is the specialization to offline RL and construction of the P-MDP using an ensemble of learned dynamics models.

Dynamics model learning: We consider Gaussian dynamics models [28], $\hat{P}(\cdot|s,a) \equiv \mathcal{N}(f_\phi(s,a), \Sigma)$, with mean $f_\phi(s,a) = s + \sigma_\Delta \text{ MLP}_\phi((s-\mu_s)/\sigma_s, (a-\mu_a)/\sigma_a)$, where $\mu_s, \sigma_s, \mu_a, \sigma_a$ are the mean and standard deviations of states/actions in $\mathcal{D}$; σ_Δ is the standard deviation of state differences, i.e. $\Delta = s' - s, (s,s') \in \mathcal{D}$; this parameterization ensures local continuity since the MLP learns only the state differences. The MLP parameters are optimized using maximum likelihood estimation with mini-batch stochastic optimization using Adam [68].

Unknown state-action detector (USAD): In order to partition the state-action space into known and unknown regions, we use uncertainty quantification [69, 70, 71, 72]. In particular, we consider

approaches that track uncertainty using the predictions of ensembles of models [69, 72]. We learn multiple models $\{f_{\phi_1}, f_{\phi_2}, \ldots\}$ where each model uses a different weight initialization and are optimized with different mini-batch sequences. Subsequently, we compute the ensemble discrepancy as $\text{disc}(s,a) = \max_{i,j} \left\| f_{\phi_i}(s,a) - f_{\phi_j}(s,a) \right\|_2$, where f_{ϕ_i} and f_{ϕ_j} are members of the ensemble. With this, we implement USAD as below, with threshold being a tunable hyperparameter.

$$U_{\text{practical}}(s,a) = \begin{cases} \text{FALSE (i.e. Known)} & \text{if } \text{disc}(s,a) \leq \text{threshold} \\ \text{TRUE (i.e. Unknown)} & \text{if } \text{disc}(s,a) > \text{threshold} \end{cases}. \tag{3}$$

5 Experiments

Through our experimental evaluation, we aim to answer the following questions:

1. **Comparison to prior work:** How does `MOReL` compare to prior SOTA offline RL algorithms [15, 16, 18] in commonly studied benchmark tasks?
2. **Quality of logging policy:** How does the quality (value) of the data logging (behavior) policy, and by extension the dataset, impact the quality of the policy learned by `MOReL`?
3. **Importance of pessimistic MDP:** How does `MOReL` compare against a naïve model-based RL approach that directly plans in a learned model without any safeguards?
4. **Transfer from pessimistic MDP to environment:** Does learning progress in the P-MDP, which we use for policy learning, effectively translate or transfer to learning progress in the environment?

To answer the above questions, we consider commonly studied benchmark tasks from OpenAI gym [73] simulated with MuJoCo [74]. Our experimental setup closely follows prior work [15, 16, 18]. The tasks considered include `Hopper-v2`, `HalfCheetah-v2`, `Ant-v2`, and `Walker2d-v2`, which are illustrated in Figure 2. We consider five different logged data-sets for each environment, totalling 20 environment-dataset combinations. Datasets are collected based on the work of Wu et al. [18], with each dataset containing the equivalent of 1 million timesteps of environment interaction. We first partially train a policy (π_p) to obtain values around 1000, 4000, 1000, and 1000 respectively for the four environments. The first exploration strategy, `Pure`, involves collecting the dataset solely using π_p. The four other datasets are collected using a combination of π_p, a *noisy* variant of π_p, and an untrained random policy. The noisy variant of π_p utilizes either epsilon-greedy or Gaussian noise, resulting in configurations `eps-1`, `eps-3`, `gauss-1`, `gauss-3` that signify various types and magnitudes of noise added to π_p. Please see appendix for additional experimental details.

We parameterize the dynamics model using 2-layer ReLU-MLPs and use an ensemble of 4 dynamics models to implement USAD as described in Section 4.2. We parameterize the policy using a 2-layer tanh-MLP, and train it using model-based NPG [28]. We evaluate the learned policies using rollouts in the (real) environment, but these rollouts are not made available to the algorithm in any way for purposes of learning. This is similar to evaluation protocols followed in prior work [18, 15, 16]. We present all our results averaged over 5 different random seeds. Note that we use the same hyperparameters for all random seeds. In contrast, the prior works whose results we compare against tune hyper-parameters separately for each random seed [18].

Comparison of `MOReL`'s performance with prior work We compare results of `MOReL` with prior SOTA algorithms like BCQ, BEAR, and all variants of BRAC. The results are summarized in Table 1.

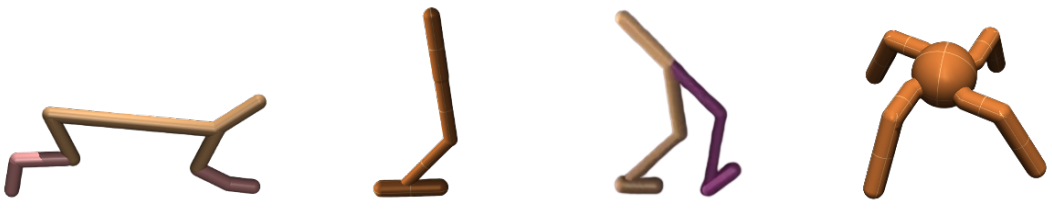

Figure 2: Illustration of the suite of tasks considered in this work. These tasks require the RL agent to learn locomotion gaits for the illustrated simulated characters.

Table 1: Results in various environment-exploration combinations. Baselines are reproduced from Wu et al. [18]. Prior work does not provide error bars. For MOReL results, error bars indicate the standard deviation across 5 different random seeds. We choose SOTA result based on the average performance.

Environment: Ant-v2					
Algorithm	BCQ [15]	BEAR [16]	BRAC [18]	Best Baseline	MOReL (Ours)
Pure	1921	2100	2839	2839	**3663±247**
Eps-1	1864	1897	2672	2672	**3305±413**
Eps-3	1504	2008	2602	2602	**3008±231**
Gauss-1	1731	2054	2667	2667	**3329±270**
Gauss-3	1887	2018	2640	2661	**3693±33**

Environment: Hopper-v2					
Algorithm	BCQ [15]	BEAR [16]	BRAC [18]	Best Baseline	MOReL (Ours)
Pure	1543	0	2291	2774	**3642±54**
Eps-1	1652	1620	2282	2360	**3724±46**
Eps-3	1632	2213	1892	2892	**3535±91**
Gauss-1	1599	1825	2255	2255	**3653±52**
Gauss-3	1590	1720	1458	2097	**3648±148**

Environment: HalfCheetah-v2					
Algorithm	BCQ [15]	BEAR [16]	BRAC [18]	Best Baseline	MOReL (Ours)
Pure	5064	5325	6207	**6209**	**6028±192**
Eps-1	5693	5435	**6307**	**6307**	5861±192
Eps-3	5588	5149	6263	**6359**	5869±139
Gauss-1	5614	5394	**6323**	**6323**	6026±74
Gauss-3	5837	5329	**6400**	**6400**	5892±128

Environment: Walker-v2					
Algorithm	BCQ [15]	BEAR [16]	BRAC [18]	Best Baseline	MOReL (Ours)
Pure	2095	2646	2694	2907	**3709±159**
Eps-1	1921	2695	3241	**3490**	2899±588
Eps-3	1953	2608	**3255**	**3255**	**3186±92**
Gauss-1	2094	2539	2893	3193	**4027±314**
Gauss-3	1734	2194	**3368**	**3368**	**2828±589**

For fairness of comparison, we reproduce results from prior work and do not run the algorithms ourselves. We provide a more expansive table with additional baseline algorithms in the appendix. Our algorithm, MOReL, achives SOTA results in 12 out of the 20 environment-dataset combinations, overlaps in error bars for 3 other combinations, and is competitive in the remaining cases. In contrast, the next best approach (a variant of BRAC) achieves SOTA results in only 5 out of 20 configurations.

Comparison of MOReL's performance in the D4RL benchmark suite The D4RL benchmark suite [75] for offline RL was introduced in concurrent work. We also study the performance of MOReL in this benchmark suite. We find that MOReL achieves the highest (normalized) score in 5 out of 12 domains studied, while the next best algorithm (CQL) achieves the highest score in only 3 out of 12 domains. Furthermore, we observe that MOReL is often very competitive with the best performing algorithm in any given domain even if it doesn't achieve the top score. However, in many domains, MOReL significantly improves over the state of the art (e.g. hopper-medium-replay and hopper-random). To aggregate results across multiple domains, we consider the average of the normalized scores as a proxy, and observe that MOReL significantly outperforms prior algorithms.

Table 2: Results of various algorithms on the D4RL benchmark suite. Each number is the normalized score computed as (score − random policy score) / (expert policy score − random policy score). The raw score for MOReL was taken to be the average over the last 100 iterations of policy learning averaged over 3 random seeds. Results of MOPO [54] and CQL [76] are reported from their respective papers. Remaining results are taken from the D4RL benchmark suite white-paper [75].

Dataset	**Environment**	**MOReL (Ours)**	**MOPO**	**CQL**	**SAC-Off**	**BEAR**	**BRAC-p**	**BRAC-v**
random	halfcheetah	25.6	34.4	**35.4**	30.5	25.1	24.1	31.2
random	hopper	**53.6**	11.7	10.8	11.3	11.4	11	12.2
random	walker2d	**37.3**	13.6	7	4.1	7.3	-0.2	1.9
medium	halfcheetah	42.1	42.3	44.4	-4.3	41.7	43.8	**46.3**
medium	hopper	**95.4**	28.0	86.6	0.8	52.1	32.7	31.1
medium	walker2d	77.8	17.8	74.5	0.9	59.1	77.5	**81.1**
medium-replay	halfcheetah	40.2	**53.1**	46.2	-2.4	38.6	45.4	47.7
medium-replay	hopper	**93.6**	67.5	48.6	3.5	33.7	0.6	0.6
medium-replay	walker2d	**49.8**	39.0	32.6	1.9	19.2	-0.3	0.9
medium-expert	halfcheetah	53.3	**63.3**	62.4	1.8	53.4	44.2	41.9
medium-expert	hopper	108.7	23.7	**111**	1.6	96.3	1.9	0.8
medium-expert	walker2d	95.6	44.6	**98.7**	-0.1	40.1	76.9	81.6
Average	Average	**64.42**	36.58	54.85	4.13	39.83	29.80	31.44

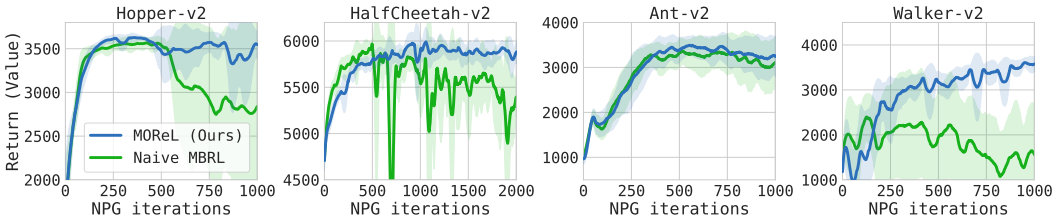

Figure 3: MOReL and Naive MBRL learning curves. The x-axis plots the number of model-based NPG iterations, while y axis plots the return (value) in the real environment. The naive MBRL algorithm is highly unstable while MOReL leads to stable and near-monotonic learning. Notice however that even naive MBRL learns a policy that performs often as well as the best model-free offline RL algorithms.

Importance of Pessimistic MDP To highlight the importance of P-MDP, we again consider the Pure-partial dataset outlined above. We compare MOReL with a naïve MBRL approach that first learns a dynamics model using the offline data, followed by running model-based NPG without any safeguards against model inaccuracy. The results are summarized in Figure 3. We observe that the naïve MBRL approach already works well, achieving results comparable to prior algorithms like BCQ and BEAR. However, MOReL clearly exhibits more stable and monotonic learning progress. This is particularly evident in Hopper-v2, HalfCheetah-v2, and Walker2d-v2, where an uncoordinated set of actions can result in the agent falling over. Furthermore, in the case of naïve MBRL, we observe that performance can quickly degrade after a few hundred steps of policy improvement, such as in case of Hopper-v2, HalfCheetah-v2 and Walker2d-v2. This suggests that the learned model is being over-exploited. In contrast, with MOReL, we observe that the learning curve is stable and nearly monotonic even after many steps of policy improvement.

Quality of logging policy Section 4.1 indicates that it is not possible for any offline RL algorithm to learn a near-optimal policy when faced with support mismatch between the dataset and optimal policy. To verify this experimentally for MOReL, we consider two datasets (of the same size) collected using the Pure strategy. The first uses a partially trained policy π_p (called Pure-partial), which is the same as the Pure dataset studied in Table 1. The second dataset is collected using an untrained random Gaussian policy (called Pure-random). Table 3 compares the results of MOReL using these two datasets. We observe that the value of policy learned with Pure-partial dataset far exceeds the value with the Pure-random dataset. Thus, the value of policy used for data logging plays a crucial role in the performance achievable with offline RL.

Table 3: Value of the policy learned by MOReL (5 random seeds) when working with a dataset collected with a random (untrained) policy (Pure-random) and a partially trained policy (Pure-partial).

Environment	Pure-random	Pure-partial
Hopper-v2	2354 ± 443	3642 ± 54
HalfCheetah-v2	2698 ± 230	6028 ± 192
Walker2d-v2	1290 ± 325	3709 ± 159
Ant-v2	1001 ± 3	3663 ± 247

Transfer from P-MDP to environment

Finally, we study how the learning progress in P-MDP relates to the progress in the environment. Our theoretical results (Theorem 1) suggest that the value of a policy in the P-MDP cannot substantially exceed the value in the environment. This makes the value in the P-MDP an approximate lower bound on the true performance, and a good surrogate for optimization. In Figure 4, we plot the value or return of the policy in the P-MDP and environment over the course of learning. Note that the policy is being learned in the P-MDP, and as a result we observe a clear monotonic learning curve for value in the P-MDP, consistent with the monotonic improvement theory of policy gradient methods [77, 78]. We observe that the value in the true environment closely correlates with the value in P-MDP. In particular, the P-MDP value never substantially exceeds the true performance, suggesting that the pessimism helps to avoid model exploitation.

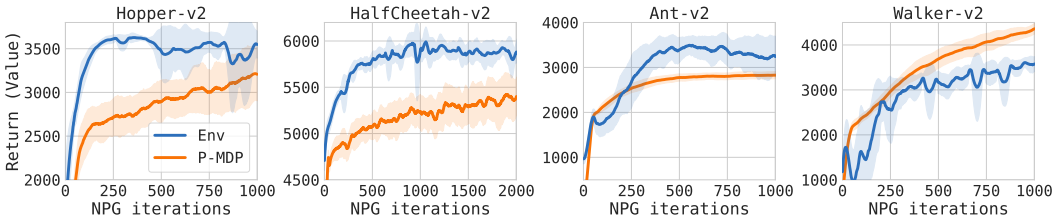

Figure 4: Learning curve using the `Pure-partial` dataset, see paper text for details. The policy is learned using the pessimistic MDP (P-MDP), and we plot the performance in both the P-MDP and the real environment over the course of learning. We observe that the performance in the P-MDP closely tracks the true performance and never substantially exceeds it, as predicted in section 4.1. This shows that the policy value in the P-MDP serves as a good surrogate for the purposes of offline policy evaluation and learning.

6 Conclusions

We introduced `MOReL`, a new model-based framework for offline RL. `MOReL` incorporates both *generalization* and *pessimism* (or conservatism). This enables `MOReL` to perform policy improvement in known states that may not directly occur in the static offline dataset, but can nevertheless be predicted using the dataset by leveraging the power of generalization. At the same time, due to the use of pessimism, `MOReL` ensures that the agent does not drift to unknown states where the agent cannot predict accurately using the static dataset.

Theoretically, we obtain bounds on the suboptimality of `MOReL` which improve over those in prior work. We further showed that this suboptimality bound cannot be improved upon by *any* offline RL algorithm in the worst case. Experimentally, we evaluated `MOReL` in the standard continuous control benchmarks in OpenAI gym and showed that it achieves state of the art results. The modular structure of `MOReL` comprising of model learning, uncertainty estimation, and model-based planning allows the use of a variety of approaches such as multi-step prediction for model learning, abstention for uncertainty estimation, or model-predictive control for action selection. In future work, we hope to explore these directions.

Acknowledgements

The authors thank Prof. Emo Todorov for generously providing the MuJoCo simulator for use in this paper. Rahul Kidambi thanks Mohammad Ghavamzadeh and Rasool Fakoor for pointers to related works and other valuable discussions/pointers about offline RL. Aravind Rajeswaran thanks Profs. Sham Kakade and Emo Todorov for valuable discussions. The authors also thank Prof. Nan Jiang and Anirudh Vemula for pointers to related work. Rahul Kidambi acknowledges funding from NSF Award CCF $-$ 1740822 and computing resources from the Cornell "Graphite" cluster. Part of this work was completed when Aravind held dual affiliations with the University of Washington and Google Brain. Aravind acknowledges financial support through the JP Morgan PhD Fellowship in AI. Thorsten Joachims acknowledges funding from NSF Award IIS $-$ 1901168. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Broader Impact

This paper studies offline RL, which allows for data driven policy learning using pre-collected datasets. The ability to train policies offline can expand the range of applications where RL can be applied as well as the sample efficiency of any downstream online learning. Since the dataset has already been collected, offline RL enables us to abstract away the exploration or data collection challenge. Safe exploration is crucial for applications like robotics and healthcare, where poorly designed exploratory actions can have harmful physical consequences. Avoiding online exploration by an autonomous agent, and working with a safely collected dataset, can have the broader impact of alleviating safety challenges in RL. That said, the impact of RL agents to the society at large is highly dependent on the

design of the reward function. If the reward function is designed by malicious actors, any RL agent, be it offline or not, can present negative consequences. Therefore, the design of reward functions requires checks, vetting, and scrutiny to ensure RL algorithms are aligned with societal norms.

References

[1] J. Deng, W. Dong, R. Socher, L. J. Li, K.Li, and L. Fei Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[2] W. Fisher, G. Doddington, and K. Goudie-Marshall. The DARPA speech recognition research database: Specification and status. In *Proceedings of the DARPA Workshop*, pages 93–100, 1986.

[3] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005, 2013.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6), 2017.

[5] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition, November 26 2012.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[7] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.

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

[9] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafał Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018.

[10] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362:1140–1144, 2018.

[11] Oriol Vinyals, Igor Babuschkin, Wojciech Marian Czarnecki, Michaël Mathieu, Andrew Joseph Dudzik, Junyoung Chung, Duck Hwan Choi, Richard W. Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pages 1–5, 2019.

[12] Sascha Lange, Thomas Gabel, and Martin A. Riedmiller. Batch reinforcement learning. In *Reinforcement Learning*, volume 12. Springer, 2012.

[13] Philip S Thomas. Safe reinforcement learning. *PhD Thesis*, 2014. URL http://scholarworks.umass.edu/dissertations_2/514.

[14] Philip S. Thomas, Bruno Castro da Silva, Andrew G. Barto, Stephen Giguere, Yuriy Brun, and Emma Brunskill. Preventing undesirable behavior of intelligent machines. *Science*, 366(6468): 999–1004, 2019.

[15] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *CoRR*, abs/1812.02900, 2018.

[16] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *CoRR*, abs/1906.00949, 2019.

[17] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. Striving for simplicity in off-policy deep reinforcement learning. *CoRR*, abs/1907.04543, 2019.

[18] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *CoRR*, arXiv:1911.11361, 2019.

[19] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Àgata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind W. Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *CoRR*, abs/1907.00456, 2019.

[20] Romain Laroche and Paul Trichelair. Safe policy improvement with baseline bootstrapping. *CoRR*, abs/1712.06924, 2017.

[21] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *CoRR*, arXiv:1912.02074, 2019.

[22] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodríguez, and Thomas A. Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *ArXiv*, abs/1903.11239, 2019.

[23] Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *ACC*, 2005.

[24] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4906–4913. IEEE, 2012.

[25] Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4:1–43, 2012.

[26] Rémi Munos and Csaba Szepesvari. Finite-time bounds for fitted value iteration. *J. Mach. Learn. Res.*, 9:815–857, 2008.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[28] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A Game Theoretic Framework for Model-Based Reinforcement Learning. In *ICML*, 2020.

[29] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *CoRR*, abs/1906.08253, 2019.

[30] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *ICLR*. OpenReview.net, 2018.

[31] Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. *ArXiv*, abs/1809.05214, 2018.

[32] Stephane Ross and Drew Bagnell. Agnostic system identification for model-based reinforcement learning. In *ICML*, 2012.

[33] Omer Gottesman, Fredrik D. Johansson, Joshua Meier, Jack Dent, Donghun Lee, Srivatsan Srinivasan, Linying Zhang, Yi Ding, David Wihl, Xuefeng Peng, Jiayu Yao, Isaac Lage, Christopher Mosch, Li-Wei H. Lehman, Matthieu Komorowski, Aldo Faisal, Leo Anthony Celi, David A. Sontag, and Finale Doshi-Velez. Evaluating reinforcement learning algorithms in observational health settings. *CoRR*, abs/1805.12298, 2018.

[34] Lu Wang, Wei Zhang 0056, Xiaofeng He, and Hongyuan Zha. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In Yike Guo and Faisal Farooq, editors, *KDD*, pages 2447–2456. ACM, 2018.

[35] Chao Yu, Guoqi Ren, and Jiming Liu 0001. Deep inverse reinforcement learning for sepsis treatment. In *ICHI*, pages 1–3. IEEE, 2019. ISBN 978-1-5386-9138-0.

[36] Alexander L. Strehl, John Langford, and Sham M. Kakade. Learning from logged implicit exploration data. *CoRR*, abs/1003.0120, 2010.

[37] Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *J. Mach. Learn. Res*, 16:1731–1755, 2015.

[38] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys*. ACM, 2016.

[39] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. Top-k off-policy correction for a reinforce recommender system. *CoRR*, abs/1812.02353, 2018.

[40] Li Zhou, Kevin Small, Oleg Rokhlenko, and Charles Elkan. End-to-end offline goal-oriented dialog policy learning via policy gradient. *CoRR*, abs/1712.02838, 2017.

[41] Nikos Karampatziakis, Sebastian Kochman, Jade Huang, Paul Mineiro, Kathy Osborne, and Weizhu Chen. Lessons from real-world reinforcement learning in a customer support bot. *CoRR*, abs/1905.02219, 2019.

[42] Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Kumar Yogamani. Deep reinforcement learning framework for autonomous driving. *CoRR*, abs/1704.02532, 2017.

[43] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms, 2010. Comment: 10 pages, 7 figures, revised from the published version at the WSDM 2011 conference.

[44] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *CoRR*, abs/1904.08473, 2019.

[45] Assaf Hallak and Shie Mannor. Consistent on-line off-policy evaluation. *CoRR*, abs/1702.07121, 2017.

[46] Carles Gelada and Marc G. Bellemare. Off-policy deep reinforcement learning by bootstrapping the covariate shift. In *AAAI*, pages 3647–3655. AAAI Press, 2019.

[47] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *CoRR*, abs/1906.04733, 2019.

[48] Chris Watkins. Learning from delayed rewards. *PhD Thesis, Cambridge University*, 1989.

[49] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[50] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

[51] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

[52] Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *ICML*, 2019.

[53] Mohammad Ghavamzadeh, Marek Petrik, and Yinlam Chow. Safe policy improvement by minimizing robust baseline regret. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2298–2306, 2016.

[54] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *CoRR*, abs/2005.13239, 2020.

[55] Michael Kearns and Satinder Singh. Near optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

[56] Ronen I. Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3:213–231, 2001.

[57] Sham M. Kakade, Michael J. Kearns, and John Langford. Exploration in metric state spaces. In *ICML*, 2003.

[58] Nan Jiang. Pac reinforcement learning with an imperfect model. In *AAAI*, pages 3334–3341, 2018.

[59] Anirudh Vemula, Yash Oza, J. Andrew Bagnell, and Maxim Likhachev. Planning and execution using inaccurate models with provable guarantees, 2020.

[60] Samuel K. Ainsworth, Matt Barnes, and Siddhartha S. Srinivasa. Mo' states mo' problems: Emergency stop mechanisms from observation. *CoRR*, abs/1912.01649, 2019.

[61] Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. Improving multi-step prediction of learned time series models. In *AAAI*, 2015.

[62] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *ArXiv*, abs/1506.03099, 2015.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[64] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos Theodorou. Information theoretic mpc for model-based reinforcement learning. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, 2017.

[65] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning, 1998.

[66] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham Kakade. Towards Generalization and Simplicity in Continuous Control. In *NIPS*, 2017.

[67] Rasool Fakoor, Pratik Chaudhari, and Alexander J. Smola. P3o: Policy-on policy-off policy optimization. *CoRR*, abs/1905.01756, 2019.

[68] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015.

[69] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *CoRR*, abs/1806.03335, 2018.

[70] Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through bayesian deep q-networks. In *ITA*, pages 1–9. IEEE, 2018.

[71] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *ICLR*. OpenReview.net, 2019.

[72] Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control. In *International Conference on Learning Representations (ICLR)*, 2019.

[73] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[74] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pages 5026–5033. IEEE, 2012.

[75] Justin Fu, Aviral Kumar, Ofir Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *ArXiv*, abs/2004.07219, 2020.

[76] Aviral Kumar, Aurick Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *ArXiv*, abs/2006.04779, 2020.

[77] Sham M. Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, 2002.

[78] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

[79] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

[80] Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477, 2018.

[81] Ofir Nachum and Bo Dai. Reinforcement learning via fenchel-rockafellar duality. *CoRR*, arXiv:2001.01866, 2020.

[82] Alekh Agarwal, Sham M. Kakade, and Lin F. Yang. On the optimality of sparse model-based planning for markov decision processes. *CoRR*, abs/1906.03804, 2019.

[83] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[84] Sham M. Kakade. A natural policy gradient. In *NIPS*, pages 1531–1538, 2001.

[85] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

[86] Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. *ArXiv*, abs/1906.08649, 2020.

[87] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, S. Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *ArXiv*, abs/1907.02057, 2019.

[88] Anusha Nagabandi, Kurt Konoglie, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. *ArXiv*, abs/1909.11652, 2019.

[89] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots. *ArXiv*, abs/1907.03613, 2019.

[90] Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *ICRA*, 2018.

[91] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *ICLR*. OpenReview.net, 2019.

A Theoretical Results: Proofs For Section 4.1

In this section, we present the proofs of our main results Theorem 1 and Proposition 4.

Proof of Theorem 1. We wish to show the following two inequalities.

$$J_{\hat{\rho_0}}(\pi, \hat{\mathcal{M}}_p) \geq J_{\rho_0}(\pi, \mathcal{M}) - \frac{2R_{\max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho_0}) - \frac{2\gamma R_{\max}}{(1-\gamma)^2} \cdot \alpha - \frac{2R_{\max}}{1-\gamma} \cdot \mathbb{E}\left[\gamma^{T^{\pi}_{\mathcal{U}}}\right], \text{ and}$$

$$J_{\hat{\rho_0}}(\pi, \hat{\mathcal{M}}_p) \leq J_{\rho_0}(\pi, \mathcal{M}) + \frac{2R_{\max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho_0}) + \frac{2\gamma R_{\max}}{(1-\gamma)^2} \cdot \alpha.$$

The proof of this theorem is inspired by the simulation lemma of [55], with some additional modifications due to pessimism, and goes through the pessimistic MDP $\mathcal{M}_p$, which is the same as $\hat{\mathcal{M}}_p$ except that the starting state distribution is ρ_0 instead of $\hat{\rho}_0$ and the transition probability from a known state-action pair (s, a) is $P(s'|s, a)$ instead of $\widehat{P}(s'|s, a)$. More concretely, $\mathcal{M}_p$ is described by $\{S \cup \text{HALT}, A, r_p, P_p, \rho_0, \gamma\}$, where HALT is an additional absorbing state we introduce similar to what we did for $\hat{\mathcal{M}}_p$. The modified reward and transition dynamics are given by:

$$P_p(s'|s,a) = \begin{cases} \delta(s' = \text{HALT}) & \text{if } U^{\alpha}(s,a) = \text{TRUE} \\ & \text{or } \ s = \text{HALT} \\ P(s'|s,a) & \text{otherwise.} \end{cases} \qquad r_p(s,a) = \begin{cases} -\kappa & \text{if } s = \text{HALT} \\ r(s,a) & \text{otherwise} \end{cases}$$

We first show that

$$J_{\hat{\rho_0}}(\pi, \hat{\mathcal{M}}_p) \geq J_{\rho_0}(\pi, \mathcal{M}_p) - \frac{2R_{\max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho_0}) - \frac{2\gamma R_{\max}}{(1-\gamma)^2} \cdot \alpha, \text{ and}$$

$$J_{\hat{\rho_0}}(\pi, \hat{\mathcal{M}}_p) \leq J_{\rho_0}(\pi, \mathcal{M}_p) + \frac{2R_{\max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho_0}) + \frac{2\gamma R_{\max}}{(1-\gamma)^2} \cdot \alpha,$$

The main idea is to couple the evolutions of any given policy on the pessimistic MDP $\mathcal{M}_p$ and the model-based pessimistic MDP $\hat{\mathcal{M}}_p$ so that $(s_{t-1}, a_{t-1}) \stackrel{\text{def}}{=} (s^{\mathcal{M}_p}_{t-1}, a^{\mathcal{M}_p}_{t-1}) = (s^{\hat{\mathcal{M}}_p}_{t-1}, a^{\hat{\mathcal{M}}_p}_{t-1})$.

Assuming that such a coupling can be performed in the first step, since $\left\| P(s,a) - \hat{P}(s,a) \right\|_1 \leq \alpha$, this coupling can be performed at each subsequent step with probability $1 - \alpha$. The probability that the coupling is not valid at time t is at most $1 - (1-\alpha)^t$. So the total difference in the values of the policy π on the two MDPs can be upper bounded as:

$$\begin{aligned} \left| J_{\hat{\rho_0}}(\pi, \hat{\mathcal{M}}_p) - J_{\rho_0}(\pi, \mathcal{M}_p) \right| &\leq \frac{2R_{\max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho_0}) + \sum_t \gamma^t \left(1 - (1-\alpha)^t\right) \cdot 2 \cdot R_{\max} \\ &\leq \frac{2R_{\max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho_0}) + \frac{2\gamma R_{\max}}{(1-\gamma)^2} \cdot \alpha. \end{aligned}$$

We now argue that

$$J_{\rho_0}(\pi, \mathcal{M}_p) \geq J_{\rho_0}(\pi, \mathcal{M}) - \frac{2R_{\max}}{1-\gamma} \cdot \mathbb{E}\left[\gamma^{T^{\pi}_{u}}\right], \text{ and}$$

$$J_{\rho_0}(\pi, \mathcal{M}_p) \leq J_{\rho_0}(\pi, \mathcal{M}).$$

For the first part, we see that the evolution of any policy π on the pessimistic MDP $\mathcal{M}_p$, can be coupled with the evolution of π on the actual MDP $\mathcal{M}$ until π encounters an unknown state. From this point, the total rewards obtained on the pessimistic MDP $\mathcal{M}_p$ will be $\frac{-R_{\max}}{1-\gamma}$, while the maximum total reward obtained by π on $\mathcal{M}$ from that point on is $\frac{R_{\max}}{1-\gamma}$. Multiplying by the discount factor $\mathbb{E}\left[\gamma^{T^{\pi}_{u}}\right]$ proves the first part.

For the second part, consider any policy π and let it evolve on the MDP $\mathcal{M}$ as $(s, a, s'_{\mathcal{M}})$. Simulate an evolution of the same policy π on $\mathcal{M}_p$, $\left(s, a, s'_{\mathcal{M}_p}\right)$, as follows: if $(s, a) \in SA_k$, then $s'_{\mathcal{M}_p} = s'_{\mathcal{M}}$ and if $(s, a) \in \mathcal{U}$, then $s'_{\mathcal{M}_p} = \text{HALT}$. We see that the rewards obtained by π on each transition in $\mathcal{M}_p$ is less than or equal to that obtained by π on the same transition in $\mathcal{M}$. This proves the second part of the lemma. □

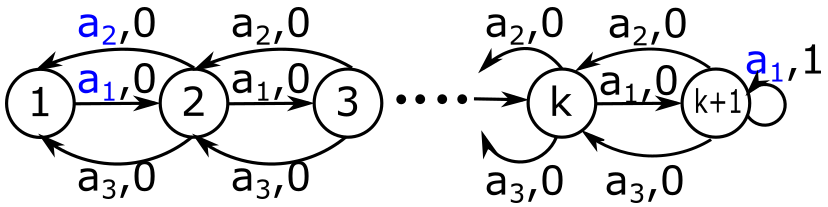

Figure 5: This example shows that the suboptimality of any offline RL algorithm is at least $\frac{R_{\max}}{4(1-\gamma)^2} \times \frac{d^{\pi^*,\mathcal{M}}(\mathcal{U}_D)}{\log \frac{1}{1-\gamma}}$ in the worst case and hence Corollary 2 is tight. The states $1, 2, \cdots, k+1$ in the MDP are depicted under the circles. The actions a_1, a_2, a_3, rewards and transitions are depicted on the arrows connecting the states. The actions taken by the behavior (i.e. the data collection) policy are depicted in blue. See Proposition 4 and its proof for more details.

Lemma 5. *(Hitting time and visitation distribution) For any set $\mathcal{S} \subseteq S \times A$, and any policy π, we have $\mathbb{E}\left[\gamma^{T^\pi_{\mathcal{S}}}\right] \leq \frac{1}{1-\gamma} \cdot d^{\pi,\mathcal{M}}(\mathcal{S})$.*

Proof of Lemma 5. The proof is rather straightforward. We have

$$\mathbb{E}\left[\gamma^{T^\pi_{\mathcal{U}}}\right] \leq \sum_{(s',a') \in \mathcal{U}} \mathbb{E}\left[\gamma^{T^\pi_{(s',a')}}\right] \leq \sum_{(s',a') \in \mathcal{U}} \sum_{t=0}^{\infty} \gamma^t P(s_t = s', a_t = a' | s_0 \sim \rho_0, \pi, \mathcal{M})$$
$$= \frac{1}{1-\gamma} \sum_{(s',a') \in \mathcal{U}} d^{\pi,\mathcal{M}}(s', a') = \frac{1}{1-\gamma} \cdot d^{\pi,\mathcal{M}}(\mathcal{U}).$$

□

Proof of Proposition 4. We consider the MDP in Figure 5, where we set $k = 10 \log \frac{1}{1-\gamma}$. The MDP has $k+1$ states, with three actions a_1, a_2 and a_3 at each state. The rewards (shown on the transition arrows) are all 0 except for the action a_1 taken in state $k+1$, in which case it is 1. Note that the rewards can be scaled by $R_{\max}$ but for simplicity, we consider the setting with $R_{\max} = 1$. It is clear that the optimal policy π^* is to take the action a_1 in all the states. The starting state distribution ρ_0 is state 1 with probability $p_0 \stackrel{\text{def}}{=} \frac{\epsilon}{(1-\gamma)\log \frac{1}{1-\gamma}}$ and state $k+1$ with probability $1 - p_0$. The actions taken by the data collection policy are shown in blue. Since the dataset consists only of (state, action, reward, next state) pairs $(1, a_1, 0, 2), (2, a_2, 0, 1)$ and $(k+1, a_1, 1, k+1)$ we see that $\mathcal{U}_D = (S \times A) \setminus \{(1, a_1), (2, a_2), (k+1, a_1)\}$ and $d^{\pi^*,\mathcal{M}}(\mathcal{U}_D) = (1-\gamma) \cdot \sum_{t=1}^{k-1} \gamma^t \cdot p_0 \leq (1-\gamma) \cdot (k-1) \cdot p_0 \leq \epsilon$ proving the first claim. Since none of the states and actions in $\mathcal{U}_D$ are seen in the dataset, after permuting the actions if necessary, the expected time taken by any policy learned from the dataset, to reach state $k+1$ starting from state 1 is at least $\exp(k/5) \geq (1-\gamma)^{-2}$. So, the value of any policy $\hat{\pi}$ learned from the dataset is at most $\frac{1-p_0}{1-\gamma} + \frac{p_0 \cdot \gamma^{(1-\gamma)^{-2}}}{1-\gamma} = \frac{1}{1-\gamma} - p_0 \cdot \frac{1-\gamma^{(1-\gamma)^{-2}}}{1-\gamma} \leq \frac{1}{1-\gamma} - \frac{3p_0}{4(1-\gamma)}$, where we used $\gamma \in [0.95, 1)$ in the last step. On the other hand, the value of π^* is at least $\frac{1-p_0}{1-\gamma} + p_0 \cdot \left(\frac{1}{1-\gamma} - k\right)$. So the suboptimality of any learned policy is at least $p_0 \cdot \left(\frac{3}{4(1-\gamma)} - k\right) = p_0 \cdot \left(\frac{3}{4(1-\gamma)} - 10 \log \frac{1}{1-\gamma}\right) \geq \frac{p_0}{4(1-\gamma)}$, where we again used $\gamma \in [0.95, 1)$ in the last step. Substituting the value of p_0 proves the proposition. □

Proof of Lemma 3. We first note that the empirical starting distribution $\hat{\rho}_0$ satisfies $D_{TV}(\rho_0, \hat{\rho}_0) \leq \frac{C}{\rho_{0,\min}} \cdot \sqrt{\frac{\log \frac{1}{\delta \rho_{0,\min}}}{n}}$, for a large enough constant C. This is because for each state s in the support of ρ_0, its empirical frequency in $\mathcal{D}$ satisfies:

$$|\hat{\rho}_0(s) - \rho_0(s)| \leq C \sqrt{\frac{\log \frac{1}{\delta \rho_{0,\min}}}{n}},$$

with probability at least $1-\delta\rho_{0,\min}$ using Chernoff's bound, where C is an absolute numerical constant. Using union bound over at most $\frac{1}{\rho_{0,\min}}$ states in the support of ρ_0, we see that with probability at least $1-\delta$, we have $D_{TV}(\rho_0, \hat{\rho}_0) \leq \frac{C}{\rho_{0,\min}} \cdot \sqrt{\frac{\log \frac{1}{\delta\rho_{0,\min}}}{n}}$.

Similarly, for any state action pair (s,a), denoting $n_{(s,a)}$ as the number of times (s,a) appears in $\mathcal{D}$, we have that:

$$\frac{n_{(s,a)}}{n} - d^{\pi_b,\mathcal{M}}(s,a) \geq -C\sqrt{\frac{\log \frac{1}{\delta d^{\pi_b}_{\min}}}{n}},$$

with probability at least $1 - d^{\pi_b}_{\min}\delta$. Again using a union bound over all state-action pairs in the support of $d^{\pi_b,\mathcal{M}}(\cdot)$, we see that:

$$n_{(s,a)} \geq d^{\pi_b}_{\min} \cdot n - C\sqrt{n \cdot \log \frac{1}{\delta d^{\pi_b}_{\min}}},$$

for every (s,a) in the support of $d^{\pi_b,\mathcal{M}}(\cdot)$ with probability at least $1-\delta$. The asssumption on the size of n then implies that $n_{(s,a)} \geq \frac{d^{\pi_b}_{\min} \cdot n}{2}$. Using a similar Chernoff bound argument, we see that $D_{TV}(P(\cdot|s,a), \hat{P}(\cdot|s,a)) \leq \frac{C}{p_{\min}} \cdot \sqrt{\frac{\log \frac{1}{\delta p_{\min} d^{\pi_b}_{\min}}}{n_{(s,a)}}}$ for every (s,a) in the support of $d^{\pi_b,\mathcal{M}}(\cdot)$ with probability at least $1-\delta$. By choosing $\alpha = \frac{C}{p_{\min}} \cdot \sqrt{\frac{\log \frac{1}{\delta p_{\min} d^{\pi_b}_{\min}}}{n_{(s,a)}}}$, we see that $\mathcal{U} \cap \text{Supp}(d^{\pi_b,\mathcal{M}}) = \emptyset$ and hence $T^{\pi_b}_{\mathcal{U}} = \infty$. By Theorem 1, we have that for any policy π, we have:

$$\begin{aligned}
J_{\rho_0}(\pi_{\text{out}}, \mathcal{M}) &\geq J_{\hat{\rho}_0}(\pi_{\text{out}}, \hat{\mathcal{M}}_p) - \frac{2R_{\max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho}_0) - \frac{2\gamma R_{\max}}{(1-\gamma)^2} \cdot \alpha \\
&\geq J_{\hat{\rho}_0}(\pi, \hat{\mathcal{M}}_p) - \epsilon_\pi - \frac{2R_{\max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho}_0) - \frac{2\gamma R_{\max}}{(1-\gamma)^2} \cdot \alpha \\
&\geq J_{\rho_0}(\pi, \mathcal{M}) - \epsilon_\pi - \frac{4R_{\max}}{1-\gamma} \cdot D_{TV}(\rho_0, \hat{\rho}_0) - \frac{4\gamma R_{\max}}{(1-\gamma)^2} \cdot \alpha - \frac{2R_{\max}}{1-\gamma} \cdot \mathbb{E}\left[\gamma^{T^\pi_{\mathcal{U}}}\right].
\end{aligned}$$

Plugging $\pi = \pi_b$ gives us the first assertion and plugging $\pi = \pi^*$ and using Lemma 5 gives us the second assertion. □

B Detailed Related Work

Our work takes a model-based approach to offline RL. We review related work pertaining to both of these domains in this section.

B.1 Offline RL

Offline RL dates at least to the work of Lange et al. [12]. In this setting, an RL agent is provided access to a typically large offline dataset, using which it has to produce a highly rewarding policy. This has direct applications in fields like healthcare [33, 34, 35], recommendation systems [36, 37, 38, 39], dialogue systems [40, 19, 41], and autonomous driving [42]. We refer the readers to the review paper of Levine et al. [79] for an overview of potential applications. On the algorithmic front, prior work in offline RL can be broadly categorized into three groups as described below.

Importance sampling The first approach to offline RL is through importance sampling. In this approach, trajectories from the offline dataset are directly used to estimate the policy gradient, which is subsequently corrected using importance weights. This approach is particularly common in contextual bandits literature [43, 36, 37] where the importance weights are relatively easier to estimate due to the non-sequential nature of the problem. For MDPs, Liu et al. [44] present an importance sampling based off-policy policy gradient method by estimating state distribution weights [45, 46, 47]. The work of Liu et al. [44] also utilizes the notion of *pessimism* by optimizing only over a subset of states visited by the behavioral policy. They utilize importance weighted policy gradient (with estimated

importance weights) to optimize this MDP. However, their work does not naturally capture a notion of generalization over the state space. Moreover, their results require strong assumptions on the data collecting policy in the sense of ensuring support on states visited by the optimal policy. Our framework, MOReL, provides the same guarantees under identical assumptions, but we also show that the performance of MOReL degrades gracefully when these assumptions aren't satisfied.

Dynamic programming The overwhelming majority of recent algorithmic work in offline RL is through the paradigm of approximate dynamic programming. In principle, any off-policy algorithm based on Q-learning [48, 49] or actor-critic architectures [50, 80, 51] can be used with a static offline dataset. However, recent empirical studies confirm that such a direct extension leads to poor results due to the challenges of overestimation bias in generalization and distribution shift. To address overestimation bias, prior work has proposed approaches like ensembles of Q-networks [17, 15, 19]. As for distribution shift, the principle approach used is to regularize the learned policy towards the data logging policy [15, 16, 18]. Different regularization schemes, such as those based on KL-divergence and maximum mean discrepancy (MMD), have been considered in the past. Wu et al. [18] perform a comparative study of such regularization schemes and find that they all perform comparably. ADP-based offline RL has also be studied theoretically [26, 52], with Chen and Jiang [52] providing an information-theoretic lower bound on sample complexity. However, these works again don't study the impact of support mismatch between logging policy and optimal policy. Finally, a recent line of work [21, 81] focuses on obtaining provably convergent methods for minimizing the (one-step) Bellman error using Duality theory. While they show promising results in continuous control tasks in the online RL setting, their performance in the offline RL setting is yet to be studied.

Model-based RL The interplay between model-based methods and offline RL has only been sparsely explored. The work of Ross & Bagnell [32] theoretically studied the performance of MBRL in the batch setting. In particular, the algorithm they analyzed involves learning a dynamics model using the offline dataset, and subsequently planning in the learned model without any additional safeguards. Their theoretical results are largely negative for this algorithm, suggesting that in the worst case, this algorithm could have arbitrarily large sub-optimality. In addition, their sub-optimality bounds become pathologically loose when the data logging distribution does not share support with the distribution of the optimal policy. Model-based offline RL methods from a safe policy improvement perspective have also been considered [53]. In contrast to both these works, we present a novel algorithmic framework that constructs and pessimistic MDP, and show that this is crucial for better empirical results and sharper theoretical analysis.

B.2 Advances in Model-Based RL

Since our work utilizes model-based RL, we review the most directly related work in the online RL setting. Classical works in MBRL have focused extensively on tabular MDPs and linear quadratic regulartor (LQR). For tabular MDPs (in the online RL setting), the first known polynomial time algorithms were the model-based algorithms of E^3 [55] and R-MAX [56]. More recent work suggests that model-based methods are minimax optimal for tabular MDPs when equipped with a wide restart state distribution [82]. However, these works critically rely on the tabular nature of the problem. Since each table entry is typically considered to be independent, and updates to any entry to do not affect other entries, tabular MDPs do not afford any notion of generalization. The metric-E^3 [57] algorithm aims to overcome this challenge by considering an underlying metric space for state-actions that enables generalization. While this work provides a strong theoretical basis, it does not directly provide a practical algorithm that can be used with function approximation. Our work is perhaps conceptually closest to E^3 and metric-E^3 which partitions the state space into known and unknown regions. A cornerstone of MOReL is the P-MDP which partitions the state space into known and unknown regions, as in, E^3 [55] and R-MAX [56], but these constructions were not developed to encourage pessimism. However, all of these works primarily deal with the standard (online) RL setting. Our work differs in its focus on offline RL, where we show the P-MDP construction plays a crucial role. Moreover, direct practical instantiations of E^3 and metric-E^3 with function approximation have remained elusive.

In recent years, along with an explosion of interest in deep RL, MBRL has emerged as a powerful class of approaches for sample efficient learning. Modern MBRL methods (typically in the online RL setting) can support the use of flexible function approximators like neural networks, as well as generic priors like smoothness and approximate knowledge of physics [22], enabling the learning of

accurate models. Furthermore, MBRL can draw upon the rich literature on model-based planning including model predictive control (MPC) [23, 24, 64, 72], search based planning [25, 65], dynamic programming [26, 83], and policy optimization [84, 78, 66, 27, 51]. These advances in MBRL have enabled highly sample efficient learning in widely studied benchmark tasks [85, 29, 86, 87, 28], as well as in a number of challenging robotic control tasks like aggressive driving [64], dexterous hand manipulation [88, 28], and quadrupedal locomotion [89]. Among these works, the recent work of Rajeswaran et al. [28] demonstrated state of the art results with MBRL in a range of benchmark tasks, and forms the basis for our practical implementation. In particular, our model learning and policy optimization subroutines are extended from the MAL framework in Rajeswaran et al. [28]. However, our work crucially differs from it due to the pessimistic MDP construction, which we show is important for success in the offline RL setting.

C Additional Experimental Details And Setup

C.1 Environment Details And Setup

As mentioned before, following recent efforts in offline RL [15, 16, 18], we consider four continuous control tasks: `Hopper-v2`, `HalfCheetah-v2`, `Ant-v2`, `Walker2d-v2` from OpenAI gym [73] simulated with MuJoCo [74]. As normally done in MBRL literature with OpenAI gym tasks [30, 90, 91, 28], we reduce the planning horizon for the environments to 400 or 500. Similar to [91, 28], we append our state parameterization with center of mass velocity to compute the reward from observations. Mirroring realistic settings, we assume access to data collected using a partially trained (sub-optimal) policy interacting with the environment. To obtain a partially trained policy π_p [15, 16, 18], we run (online) TRPO [78] until the policy reaches a value of 1000, 4000, 1000, 1000 respectively for these environments. This policy in conjunction with exploration strategies are used to collect the datasets (see below for more details). All our results are obtained by averaging runs of five random seeds (for the planning algorithm), with the seed values being $123, 246, 369, 492, 615$. Each of our experiments are run with 1 NVidia GPU and 2 CPUs using a total of 16GB of memory.

C.2 Dynamics Model, Policy Network And Evaluation

We use 2 hidden layer MLPs with 512 (for `Hopper-v2`, `Walker2d-v2`, `Ant-v2`) or 1024 (for `HalfCheetah-v2`) ReLU activated nodes each for representing the dynamics model, use an ensemble of four such models for building the USAD, and our policy is represented with a 2 hidden layer MLP with 32 tanh activated nodes in each layer. The dynamics model is learnt using Adam [68] and the policy parameters are learnt using model-based NPG steps [28]. We set hyper-parameters and track policy learning curve by performing rollouts in the real environment; these rollouts aren't used for other purposes in the learning procedure. Similar protocols are used in prior work[15, 16, 18].

C.3 Description Of Types Of Policies

We build off the experimental setup of [18]. Towards this, we first go over some notation. Firstly, let π_b represent the behavior policy, π_r is a random policy that picks actions according to a certain probability distribution (for e.g., Gaussian π_r^g/Uniform π_r^u etc.), π_p a partially-trained policy, which one can assume is better than a random policy in value. Let $\pi_b^u(q)$ be a policy that plays random actions with probability q, and sampled actions from π_b with probability $1-q$. Let $\pi_b^g(\beta)$ be a policy that adds zero-mean Gaussian noise with standard deviation β to actions sampled from π_b. Consider a behavior policy which, for instance, can be a partially trained data logging policy π_b. We consider five different exploration strategies, each corresponding to adding different kinds of exploratory noise to π_b, as described below.

C.4 Datasets And Exploration Strategies

For each environment, we use a combination of a behavior policy π_b, a noisy behavior policy $\tilde{\pi}_b$ (see below), and a pure random stochastic process π_r to collect several datasets, following Wu et al. [18]. Each dataset contains the equivalent of 1 million timesteps of interactions with the environment. See below for detailed instructions.

($\mathcal{E}\mathbf{1}$) `Pure`: The entire dataset is collected with the data logging (behavioral) policy π_b.

($\mathcal{E}\mathbf{2}$) `Eps-1`: 40% of the dataset is collected with π_b, another 40% collected with $\pi_b^u(0.1)$, and the final 20% is collected with a random policy π_r.

($\mathcal{E}\mathbf{3}$) `Eps-3`: 40% of the dataset is collected with π_b, another 40% collected with $\pi_b^u(0.3)$, and the final 20% is collected with a random policy π_r.

($\mathcal{E}\mathbf{4}$) `Gauss-1`: 40% of the dataset is collected with π_b, another 40% collected with $\pi_b^g(0.1)$, and the final 20% is collected with a random policy π_r.

($\mathcal{E}\mathbf{5}$) `Gauss-3`: 40% of the dataset is collected with π_b, another 40% collected with $\pi_b^g(0.3)$, and the final 20% is collected with a random policy π_r.

C.5 Hyperparameter Selection

Refer to table 4 for details with regards to parameters of `MOReL`. For all environments and data collection strategies, we learn two-layer MLP based dynamics models with ReLU activations by minimizing the one-step prediction errors using Adam [68] and utilize four of these models for defining the USAD. The negative reward for defining the absorbing unknown state is set as the minimum reward in the dataset $\mathcal{D}$ offsetted by a value that is searched over $\{30, 50, 100, 200\}$.

Ascertaining unknown state-action pairs: In order to ascertain unknown state-action pairs, we compute the model disagreement as: $\text{disc}(s,a) = \max_{i \neq j} ||f_{\phi_i}(s,a) - f_{\phi_j}(s,a)||_2$, where, f_{ϕ_i} and f_{ϕ_j} are members of the ensemble of learnt dynamics model. Specifically, we compute $\text{disc}(s,a)$ over all state-action pairs that occur in the static dataset $\mathcal{D}$. Next, we can compute the mean μ_d, standard deviation σ_d and the max m_d of the disagreements evaluated for every state-action pair occuring in the dataset. Then, we utilize an upper-confidence inspired strategy by defining a threshold $\text{thresh} = \mu_d + \beta \cdot \sigma_d$. The value of beta is tuned between 0 to $\beta_{\max} = (m_d - \mu_d)/\sigma_d$ in steps of 5. For any model-based rollout encountered during planning, if the discrepancy of the state-action pair at a given timestep exceeds thresh, the rollout is truncated at this timestep and is assigned a large negative reward. We emphasize that for every environment, all hyper-parameters (except for β) is maintained at the same value across all exploration settings.

With regards to the policy and the planning algorithm, we consider a $(32, 32)$ tanh MLP optimized using normalized model-based NPG steps (see, for instance, the work of Rajeswaran et al. [28] for the model-based NPG algorithm). Parameters of model-based NPG is described in table 5.

Table 4: Hyper-parameters for each environment for MOReL. Note that most hyper-parameters are common across domains, and the differences are primarily in reward penalty and number of fitting epochs, which are necessarily environment specific.

Environment: Ant-v2	
Parameter	Value
Dynamics Model	MLP(512, 512)
Activation	ReLU
# Training Epochs	300
Adam Stepsize	5e-4
Batch Size	256
Horizon	500
Negative Reward	$r_{\min}(\mathcal{D}) - 100$
USAD	4-dynamics models

Environment: Hopper-v2	
Parameter	Value
Dynamics Model	MLP(512, 512)
Activation	ReLU
# Training Epochs	300
Adam Stepsize	5e-4
Batch Size	256
Horizon	400
Negative Reward	$r_{\min}(\mathcal{D}) - 50$
USAD	4-dynamics models

Environment: HalfCheetah-v2	
Parameter	Value
Dynamics Model	MLP(1024,1024)
Activation	ReLU
# Training Epochs	3000
Adam Stepsize	5e-4
Batch Size	256
Horizon	500
Negative Reward	$r_{\min}(\mathcal{D}) - 200$
USAD	4-dynamics models

Environment: Walker-v2	
Parameter	Value
Dynamics Model	MLP(512, 512)
Activation	ReLU
# Training Epochs	300
Adam Stepsize	5e-4
Batch Size	256
Horizon	400
Negative Reward	$r_{\min}(\mathcal{D}) - 30$
USAD	4-dynamics models

Table 5: Hyper-parameters for model-based policy optimization. Note that most hyperparameters are common except the number of iterations and exploration noise.

Environment: Ant-v2	
Parameter	Value
Policy Net	MLP(32,32)
Non-linearity	Tanh
# updates	1000
$\log \sigma_{\text{init}}$	-0.1
$\log \sigma_{\min}$	-2.0
# trajectories for gradient	200
# Eval trajectories	25
# CG Steps/Damping	10, 1e-4

Environment: Hopper-v2	
Parameter	Value
Policy Net	MLP(32,32)
Non-linearity	Tanh
# updates	500
$\log \sigma_{\text{init}}$	-0.25
$\log \sigma_{\min}$	-2.0
# trajectories for gradient	50
# Eval trajectories	25
# CG Steps/Damping	25, 1e-4

Environment: HalfCheetah-v2	
Parameter	Value
Policy Net	MLP(32,32)
Non-linearity	Tanh
# updates	2500
$\log \sigma_{\text{init}}$	-1.0
$\log \sigma_{\min}$	-2.0
# trajectories for gradient	40
# Eval trajectories	25
# CG Steps/Damping	25, 1e-4

Environment: Walker-v2	
Parameter	Value
Policy Net	MLP(32,32)
Non-linearity	Tanh
# updates	1000
$\log \sigma_{\text{init}}$	-0.5
$\log \sigma_{\min}$	-2.0
# trajectories for gradient	100
# Eval trajectories	25
# CG Steps/Damping	25, 1e-4

Table 6: Results in the four environments and five exploration configurations.

Environment: Ant-v2			**Partially trained policy: 1241**		
Algorithm	Pure ($\mathcal{E}1$)	Eps-1 ($\mathcal{E}2$)	Eps-3 ($\mathcal{E}3$)	Gauss-1 ($\mathcal{E}4$)	Gauss-3 ($\mathcal{E}5$)
SAC [51]	0	-1109	-911	-1071	-1498
BC	1235	1300	1278	1203	1240
BCQ [15]	1921	1864	1504	1731	1887
BEAR [16]	2100	1897	2008	2054	2018
MMD_vp [18]	2839	2672	2602	2667	2640
KL_vp [18]	2514	2530	2484	2615	2661
KL_dual_vp [18]	2626	2334	2256	2404	2433
W_vp [18]	2646	2417	2409	2474	2487
MMD_pr [18]	2583	2280	2285	2477	2435
KL_pr [18]	2241	2247	2181	2263	2233
KL_dual_pr [18]	2218	1984	2144	2215	2201
W_pr [18]	2241	2186	2284	2365	2344
Best Baseline	2839	2672	2602	2667	2661
MOReL (Ours)	**3663** $\pm$**247**	**3305** $\pm$**413**	**3008** $\pm$**231**	**3329** $\pm$**270**	**3693** $\pm$**33**

Environment: Hopper-v2			**Partially trained policy: 1202**		
Algorithm	Pure ($\mathcal{E}1$)	Eps-1 ($\mathcal{E}2$)	Eps-3 ($\mathcal{E}3$)	Gauss-1 ($\mathcal{E}4$)	Gauss-3 ($\mathcal{E}5$)
SAC [51]	0.2655	661.7	701	311.2	592.6
BC	1330	129.4	828.3	221.1	284.6
BCQ [15]	1543	1652	1632	1599	1590
BEAR [16]	0	1620	2213	1825	1720
MMD_vp [18]	2291	2282	1892	2255	1458
KL_vp [18]	2774	2360	2892	1851	2066
KL_dual_vp [18]	1735	2121	2043	1770	1872
W_vp [18]	2292	2187	2178	1390	1739
MMD_pr [18]	2334	1688	1725	1666	2097
KL_pr [18]	2574	1925	2064	1688	1947
KL_dual_pr [18]	2053	1985	1719	1641	1551
W_pr [18]	2080	2089	2015	1635	2097
Best Baseline	2774	2360	2892	2255	2097
MOReL (Ours)	**3642** $\pm$**54**	**3724** $\pm$**46**	**3535** $\pm$**91**	**3653** $\pm$**52**	**3648** $\pm$**148**

Environment: Walker-v2			**Partially trained policy: 1439**		
Algorithm	Pure ($\mathcal{E}1$)	Eps-1 ($\mathcal{E}2$)	Eps-3 ($\mathcal{E}3$)	Gauss-1 ($\mathcal{E}4$)	Gauss-3 ($\mathcal{E}5$)
SAC [51]	131.7	213.5	127.1	119.3	109.3
BC	1334	1092	1263	1199	1137
BCQ [15]	2095	1921	1953	2094	1734
BEAR [16]	2646	2695	2608	2539	2194
MMD_vp [18]	2694	3241	**3255**	2893	**3368**
KL_vp [18]	2907	3175	2942	**3193**	3261
KL_dual_vp [18]	2575	**3490**	3236	3103	3333
W_vp [18]	2635	2863	2758	2856	2862
MMD_pr [18]	2670	2957	2897	2759	3004
KL_pr [18]	2744	2990	2747	2837	2981
KL_dual_pr [18]	2682	3109	3080	2357	3155
W_pr [18]	2667	3140	2928	1804	2907
Best Baseline	2907	3490	3255	3193	3368
MOReL (Ours)	**3709** $\pm$**159**	2899 $\pm$588	**3186** $\pm$**92**	**4027** $\pm$**314**	**2828** $\pm$**589**

Environment: HalfCheetah-v2			Partially trained policy: 4206		
Algorithm	`Pure` ($\mathcal{E}1$)	`Eps-1` ($\mathcal{E}2$)	`Eps-3` ($\mathcal{E}3$)	`Gauss-1` ($\mathcal{E}4$)	`Gauss-3` ($\mathcal{E}5$)
SAC [51]	5093	6174	5978	6082	6090
BC	4465	3206	3751	4084	4033
BCQ [15]	5064	5693	5588	5614	5837
BEAR [16]	5325	5435	5149	5394	5329
MMD_vp [18]	6207	**6307**	6263	**6323**	**6400**
KL_vp [18]	6104	6212	6104	6219	6206
KL_dual_vp [18]	**6209**	6087	**6359**	5972	6340
W_vp [18]	5957	6014	6001	5939	6025
MMD_pr [18]	5936	6242	6166	6200	6294
KL_pr [18]	6032	6116	6035	5969	6219
KL_dual_pr [18]	5944	6183	6207	5789	6050
W_pr [18]	5897	5923	5970	5894	6031
Best Baseline	6209	6307	6263	6323	6400
`MOReL` (Ours)	6028 $\pm$192	5861 $\pm$152	5869 $\pm$139	6026 $\pm$74	5892 $\pm$128

C.6 Ablation Study with the `Pure-partial` dataset

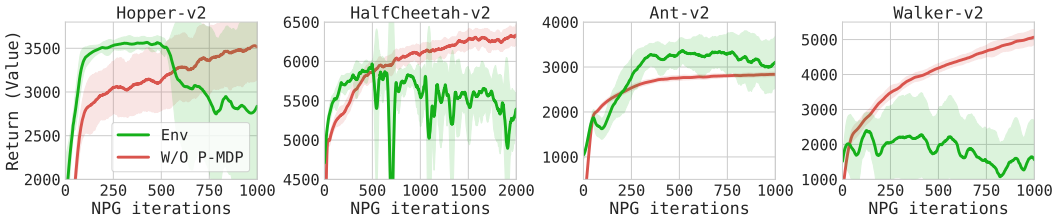

Figure 6: Learning curve using naïve MBRL with the `Pure-partial` dataset. Contrast the learning exhibited by naïve MBRL in this figure with `MOReL` in Figure 4.

C.7 Hyperparameter Guidelines and Ablations

We did not have resources to perform a thorough hyperparamter search, and largely used our intuitions to guide the choice of hyperparameters. We believe that better results are possible with hyperparameter optimization. First, we present the influence of the discrepancy threshold for differentiating known and unknown states. We first define the maximum discipancy in the dataset:

$$\text{disc}_{\mathcal{D}} = \max_{(s,a)\in\mathcal{D}} \max_{i,j} \|f_i(s,a) - f_j(s,a)\|$$

where $\mathcal{D}$ denotes offline dataset, and f_i denotes i^{th} dynamics model in the ensemble.

Table 7: Influence of discrepancy threshold on the `Hopper-v2` task. We use a penalty of 0.0 along with episode termination for visiting unknown regions in these experiments. We train all the cases for 1000 iterations, and report the average value over the last 100 iterations.

Discrepancy Threshold	Value in P-MDP	Value in true MDP
$0.1 \times \text{disc}_{\mathcal{D}}$	1315.16	2082.21
$0.2 \times \text{disc}_{\mathcal{D}}$	2479.92	3244.48
$0.5 \times \text{disc}_{\mathcal{D}}$	3074.75	3359.66
$1.0 \times \text{disc}_{\mathcal{D}}$	3543.23	3595.60
$5.0 \times \text{disc}_{\mathcal{D}}$	3245.66	3027.59
Naive-MBRL	3656.08	2809.66

Our general observations and guidelines for hyperparameters are:

1. In Table 7, we first note that $0.1 \times \text{disc}_{\mathcal{D}}$ has the most amount of pessimism and Naive-MBRL has the least/no amount of pessimism. We observe that we obtain best results in the true MDP with an intermediate level of pessimism. Having either too much pessimism or no pessimism both lead to poor results, but for very different reasons that we outline below.
2. A high degree of pessimism makes policy optimization in the P-MDP difficult. The optimization process may be slow or highly noisy. This is due to non-smoothness introduced in the dynamics and reward due to abrupt changes involving early episode terminations. If difficulty in policy optimization is observed in the P-MDP, we recommend considering reducing the degree of pessimism.
3. With a lack or low degree of pessimism, policy optimization is typically easier, but the performance in the true MDP might degrade. If it is observed that the value in the P-MDP overestimates the value in the true MDP substantially, then we recommend increasing the degree of pessimism.
4. For the tasks considered in this work, positive rewards indicate progress towards the goal. Most of the locomotion tasks involve forward velocity as the primary component of the the reward term. In these cases, we observed that the choice of reward penalty for going into unknown regions did not play a crucial role, as long as it was ≤ 0. The degree of influence of this parameter in other environments is yet to be determined, and beyond the scope of our empirical study.