PC-MLP: Model-based Reinforcement Learning with Policy Cover Guided Exploration

Yuda Song [1] **Wen Sun** [2]

Abstract

Model-based Reinforcement Learning (RL) is a popular learning paradigm due to its potential sample efficiency compared to model-free RL. However, existing empirical model-based RL approaches lack the ability to explore. This work studies a computationally and statistically efficient model-based algorithm for both Kernelized Nonlinear Regulators (KNR) and linear Markov Decision Processes (MDPs). For both models, our algorithm guarantees polynomial sample complexity and only uses access to a planning oracle. Experimentally, we first demonstrate the flexibility and the efficacy of our algorithm on a set of exploration challenging control tasks where existing empirical model-based RL approaches completely fail. We then show that our approach retains excellent performance even in common dense reward control benchmarks that do not require heavy exploration.

1. Introduction

Model-based Reinforcement Learning (MBRL) has played a central role in Reinforcement Learning for decades and has achieved great empirical performance on tasks such as robotics (Deisenroth & Rasmussen, 2011; Levine & Abbeel, 2014) and video games (Kaiser et al., 2019). However, most existing empirical model-based RL approaches lack the ability to perform strategic exploration. Thus they usually can not guarantee any global performance.

In this work, we consider learning to control a nonlinear dynamical system. Specifically, we focus on systems that can be modeled via Reproducing Kernel Hilbert Space (RKHS). Following (Kakade et al., 2020), we name such models *Kernelized Nonlinear Regulators* (KRNs). Such model has been extensively used in the robotics community in the last two decades due to its flexibility to capture popular models such as linear dynamics (LQRs), piece-wise hybrid linear system, nonlinear models that can be modeled by higher-order polynomials, and systems that can be captured by Gaussian Processes (GPs) (e.g., (Ko et al., 2007; Deisenroth & Rasmussen, 2011; Bansal et al., 2017; Fisac et al., 2018; Umlauft et al., 2018; Mania et al., 2020)). The fact that KNRs have been widely used in real-world robotics and control problems proves that KNR is capable to model real-world dynamics. Thus it motivates the development of algorithms that have global performance guarantees, and are also provably sample and computation efficient for KNRs.

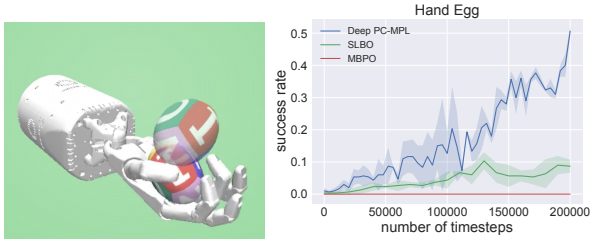

Figure 1. Example of the exploration ability of PC-MLP in the HandEgg Experiment. The environment involves complex dynamics and sparse reward. Left: the HandEgg environment. Right: PC-MLP can explore strategically and thus learns much faster than other MBRL (SLBO and MBPO) baselines which rely on random exploration.

(Kakade et al., 2020) initiated an information theoretical analysis for KNRs and provided an algorithm (LC3) that achieves near-optimal regret. However, the proposed LC3 algorithm relies on an optimistic planning oracle (see Sec. 3 for the detailed definition of an optimistic planning oracle) which is unfortunately not computationally efficient. Thus LC3 algorithm cannot be easily implemented using off-shelf computational tools developed from the planning and control community (e.g., highly efficient planning oracles). While its regret analysis is novel and tight, the computation inefficiency dramatically limits the practical usage of LC3. In this work, we develop a Model-based algorithm named *PC-MLP*, standing for Model Learning and Planning with

[*]Equal contribution [1]Machine Learning Department, Carnegie Mellon University, Pittsburgh, USA [2]Department of Computer Science, Cornell University, Ithaca , USA. Correspondence to: Yuda Song <yudas@andrew.cmu.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

Policy Cover for Exploration, that is provably sample efficient (i.e., polynomial in all relevant parameters), and is also planning-oracle efficient, i.e., it only requires access to a classic planning oracle rather than an optimistic planning oracle. Thus PC-MLP allows one to leverage existing off-shelf efficient planning oracles from the motion planning and control community, which provide excellent flexibility when deploying the algorithm to real-world control problems.

From RL side, several new linear MDP models (Yang & Wang, 2019; Jin et al., 2019; Modi et al., 2020; Zhou et al., 2020) recently have gained a lot of interest in the theoretical RL community, though unlikely KNRs, these linear MDP models haven't been demonstrated to be applicable in real world problems. While our focus in this work is on KNRs due to their proven applicability to real-world robotics problems, we nevertheless also analyze our algorithm directly on linear MDPs. Note that linear MDPs are different from KNRs as linear MDPs cannot capture simple linear dynamical systems such as LQR.

Our contributions in this work are twofold. First, theoretically, we provide a single algorithm framework PC-MLP that is provably sample efficient, and computation-wise is planning-oracle efficient (i.e., no more optimistic planning) in both KNRs and Linear MDPs (Yang & Wang, 2019), simultaneously. Our algorithm is modular and simple. It maintains a policy cover (i.e., an ensemble that contains all previous learned policies) and learns a model from the traces of the policies in the cover. Policy cover avoids catastrophic forgetting issue when fitting the model (i.e., during model fitting, the latest model overfits to the current policies' traces). The algorithm also uses a simple reward bonus scheme that is motivated from classic linear bandit algorithms (Dani et al., 2008) and also recent RL algorithms that work beyond tabular settings (Agarwal et al., 2020a). Unlike count-based reward bonus, our bonus works provably on continuous state and actions space. Empirically, we develop a practical version of PC-MLP—Deep PC-MLP, which uses deep neural network for model fitting, and random features (either Random fourier features (Rahimi & Recht, 2008) or random network based features (Burda et al., 2019)) for reward bonus design. We evaluate Deep PC-MLP extensively on common continuous control benchmarks including both sparse reward environments (e.g., sparse reward hand manipulation shown in Fig. 1) and dense reward environments. Our algorithm achieves excellent performance in both exploration challenging control tasks and the classic dense reward control tasks.

This paper is structured as follows: in section 2 we provide additional related works. Section 3 introduces the KNR and linear MDPs settings, notations and basic assumptions. In section 4 we describe our algorithm framework, PC-MLP, while in section 5 we provide the main results on the sample complexity of PC-MLP in both linear MDPs and in KNR. We then describe a practical implementation of PC-MLP using deep neural networks in section 6. Sections 7 includes a comprehensive empirical evaluation of the practical implemenation.

2. Related Works

Below we discuss additional related works. On the theoretical side of KNRs, (Mania et al., 2020) studied KNRs from a system identification perspective. Their approach relies on a reachability assumption and a Lipschitz assumption on state-action features. Our work does not require any of the two assumptions. The high-level intuition is that if there is a subspace that is not reachable, it does not matter in terms of policy optimization as no policy can reach that subspace to collect rewards. When specializing in LQR, there are a lot of prior works studying sample complexity of learning in LQRs (Abbasi-Yadkori & Szepesvári, 2011; Dean et al., 2018; Mania et al., 2019; Cohen et al., 2019; Simchowitz & Foster, 2020). Optimal planning oracle in LQR exists and has a closed-form solution of the optimal control policy.

On the RL side, in both theory and in practice, model-based approaches are often considered to be sample efficient (Deisenroth & Rasmussen, 2011; Levine & Abbeel, 2014; Chua et al., 2018; Sun et al., 2018; Kurutach et al., 2018; Nagabandi et al., 2018; Luo et al., 2018; Ross & Bagnell, 2012; Sun et al., 2019; Osband & Van Roy, 2014; Ayoub et al., 2020; Lu & Van Roy, 2019). Many existing empirical MBRL algorithms lack the ability to perform exploration and thus cannot automatically adapt to exploration challenging tasks. Existing theoretical works either do not apply to KNRs directly or rely on optimistic planning oracles or a Thompson sampling approach which only guarantees a regret bound in the Bayesian setting.

3. PRELIMINARIES

We consider episodic finite horizon MDPs $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, H, r, P^\star, s_0\}$ where s_0 is a fixed initial state, [1] $\mathcal{S}$ and $\mathcal{A}$ are continuous state and action space, $H \in \mathbb{N}^+$ is the horizon, $r : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ is the reward function, and $P^\star : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ is the Markovian transition. In our learning setting, we assume reward r is known but transition $P^\star$ is unknown. The learner is equipped with a policy class $\Pi \subset \mathcal{S} \mapsto \Delta(\mathcal{A})$, and the goal is to search for an optimal policy $\pi^\star \in \Pi$, such that $\pi^\star \in \operatorname{argmax}_{\pi\in\Pi} J(\pi; r, P^\star)$, where $J(\pi; r, P) :=$

[1] Our approach generalizes to a fixed initial state distribution. We use fixed initial state to emphasize the need for exploration.

$\mathbb{E}\left[\sum_{h=0}^{H-1} r(s_h, a_h) | a_h \sim \pi(\cdot|s_h), s_{h+1} \sim P(\cdot|s_h, a_h)\right]$ is the expected total reward of π under a transition $P \in \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ and reward $r : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$. In additional to policy class Π, a model-based learner is also equipped with a model class $\mathcal{P} \subset \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$. Throughout the paper, we assume realizability in model class:

Assumption 1 (Model Realizability)**.** *We assume $\mathcal{P}$ is rich enough such that $P^\star \in \mathcal{P}$.*

We require the sample complexity of our learning algorithm to depend only on the complexity measures of the model class $\mathcal{P}$.

The goal is to find an ϵ-near optimal policy from Π, i.e., a policy $\hat{\pi}$ such that $J(\hat{\pi}; r, P^\star) \geq \max_{\pi \in \Pi} J(\pi; r, P^\star) - \epsilon$, with high probability, using number of samples scaling polynomially with respect to all relevant parameters including the complexity of the model class $\mathcal{P}$. With the above setup, we discuss two specific examples.

Kernelized Nonlinear Regulators For a KNR, we denote the transition as $s' = W^\star \phi(s, a) + \epsilon$ where $\mathcal{S} \subset \mathbb{R}^{d_s}$, $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, $\|W^\star\|_F \leq F \in \mathbb{R}^+$, and $\phi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^d$ is potentially a nonlinear mapping. In other words, $P^\star(\cdot|s, a) = \mathcal{N}(W^\star \phi(s, a), \sigma^2 I)$. Here ϕ is known, σ is known, but $W^\star$ is unknown to the learner. When ϕ corresponds to some kernel feature mapping, $W^\star \phi(s, a)$ falls into the corresponding Reproducing Kernel Hilbert Space (RKHS). In this case, the model class $\mathcal{P}$ consists of transitions parameterized by W with $\|W\|_F \leq F$. We can denote $\mathcal{W} = \{W : \|W\|_F \leq F\}$. It is clear that $W^\star \in \mathcal{W}$. Prior work LC3 relies on *an optimistic planning oracle*, i.e., $\max_{W \in \text{Ball}} \max_{\pi \in \Pi} J(\pi; r, W)$ with Ball $\subset \mathcal{W}$.

Linear MDPs We consider a specific linear MDP model $P^\star(s'|s, a) = \langle \mu^\star(s'), \phi(s, a) \rangle, \forall s, a, s'$ with $\phi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^d$, where we assume $\mu^\star \in \Upsilon \subset \mathcal{S} \mapsto \mathbb{R}^d$. Here $\mu^\star$ is unknown but ϕ is known. Recently (Agarwal et al., 2020b) show that such linear model has a latent representation interpretation. For this model we assume Υ is finite and $\mu^\star \in \Upsilon$. For norm bound, we assume $\|\mu \cdot \nu\|_2 \leq \sqrt{d}$ for any $\nu \in \mathbb{R}^{|\mathcal{S}|}$ with that $\|\nu\|_\infty \leq 1$. The statistical complexity of the model class is the log of the cardinality of Υ, i.e., $\ln |\Upsilon|$. This model does not directly capture the linear MDP proposed by (Jin et al., 2019), but captures the version from (Yang & Wang, 2019) due to its finite degree of freedom in the linear model's parameterization.[2] The algorithm from (Yang & Wang, 2019) also relies on optimistic planning with value iteration to be the specific planning oracle.

Note that two models, linear MDPs and KNRs, are different: one does not generalize the other. While linear MDP generalizes the usual tabular MDPs, the Gaussian noise makes KNRs are unable to generalize tabular MDPs directly. However KNRs generalizes polynomial dynamical systems (i.e., linear system $s' = As + Ba + \epsilon$) while linear MDPs cannot.

Despite the differences in two models, we present a single model-based algorithmic framework that takes $\mathcal{P}$ and Π as inputs, and outputs an ϵ near-optimal policy π in sample complexity scaling polynomially with respect to $H, 1/\epsilon, d$ and the complexity of $\mathcal{P}$, with polynomial number of calls to a planning oracle OP(Π, r, P).

Assumption 2 (Planning Oracle)**.** *Given reward r, transition P, and Π, we assume access to a planning oracle: OP$(\Pi, r, P) := \text{argmax}_{\pi \in \Pi} J(\pi; r, P)$.*

We treat OP as a black-box planning oracle and we do not place any restrictions on the form of the oracle. It could be a trajectory optimization based motion planner and controller (Ratliff et al., 2009; Todorov & Li, 2005; Sun et al., 2016) or it could be an asymptotically optimal sampling-based planner (Williams et al., 2017a; Karaman & Frazzoli, 2011). Note that OP(Π, r, P) is a computation oracle which does not use any real-world samples.

The algorithm framework simultaneously applies to both KNR and Linear MDPs, with $\mathcal{P} = \left\{P(\cdot|s, a) : \mathcal{N}(W\phi(s, a), \sigma^2 I), \forall s, a \mid \|W\|_F \leq F\right\}$ and $\mathcal{P} = \{P(\cdot|s, a) : \mu\phi(s, a), \forall s, a \mid \mu \in \Upsilon\}$ as model class inputs, respectively,

Additional Notations We denote $d_h^\pi \in \Delta(\mathcal{S} \times \mathcal{A})$ as the state-action distribution induced by policy π at time step h, and $d^\pi = \sum_{h=0}^{H-1} d_h^\pi / H$ as the average state-action distribution from π.

4. Algorithm Framework

In this section, we introduce our algorithmic framework. Alg. 1 summarizes the algorithm PC-MLP standing for **M**odel **L**earning and **P**lanning using **P**olicy **C**over for Exploration.

In high-level, the algorithm maintains a policy cover $\boldsymbol{\pi}_n = \{\pi_1, \ldots, \pi_n\}$ that contains all previously learned policies. In each episode, the model learning procedure is a maximum likelihood estimation on training data collected by $\boldsymbol{\pi}_n$. Sampling from $\boldsymbol{\pi}_n$ can be implemented by first sampling $i \in [1, \ldots, n]$ uniformly random, and then sample (s, a) from d^{π_i}. The model $\widehat{P}_n$ trained this way (via the classic Maximum Likelihood Estimation) can predict well under state-action pairs covered by $\boldsymbol{\pi}_n$. This forces the

[2] In (Yang & Wang, 2019), $\mu^\star(s) = M^\star \psi(s')$ where $M^\star \in \mathbb{R}^{d_\phi \times d_\psi}$ has bounded norm and ψ is known to the learner. Hence standard covering argument can show that $\ln |\Upsilon|$ is equivalent to the covering dimension of the linear model parameterization.

Algorithm 1 The PC-MLP Framework

Require: MDP $\mathcal{M}$, inputs $(N, K, \lambda, M, \mathcal{P}, c)$

Initialize π_1
Set policy-cover $\boldsymbol{\pi}_1 = \{\pi_1\}$
for $n = 1 \to N$ **do**
  Draw K samples $\{s_i, a_i\} \sim d_{\pi_n}$
  Set $\widehat{\Sigma}_{\pi_n} = \sum_{i=1}^{K} \phi(s_i, a_i)\phi(s_i, a_i)^\top / K$
  Set covariance matrix $\widehat{\Sigma}_n = \sum_{i=1}^{n} \Sigma_{\pi_i} + \lambda I$
  Model Learning (MLE) with data from policy-cover $\boldsymbol{\pi}_n$ and denote $\widehat{P}_n$ as an approximate optimizer of the following optimization program:

$$\max_{P \in \mathcal{P}} \sum_{i=1}^{M} \ln P(s_i' | s_i, a_i) \tag{1}$$

  where $\{s_i, a_i\} \sim d_{\boldsymbol{\pi}_n}, s' \sim P^\star_{s_i, a_i}, \forall i \in [M]$
  Set reward bonus $\widehat{b}_n$ as in Eq. 2
  Plan $\pi_{n+1} = \text{OP}\left(\Pi, r + \widehat{b}_n, \widehat{P}_n, \right)$
  Update *policy-cover*: $\boldsymbol{\pi}_{n+1} = \boldsymbol{\pi}_n \oplus \{\pi_{n+1}\}$
end for

learned model to achieve good prediction performance under the state-action pairs that are covered by the current policy cover $\boldsymbol{\pi}_n$.

For state-action pairs that do not well covered by $\boldsymbol{\pi}_n$, we design reward bonus. Specifically, we form the (unnormalized and regularized) empirical covariance matrix of the policy cover $\boldsymbol{\pi}_n$, i.e., $\widehat{\Sigma}_n = \sum_{i=1}^{n} \Sigma_{\pi_i} + \lambda I$. Intuitively, any state-action pair (s, a), whose feature $\phi(s, a)$ falls into the subspace corresponding to the eigenvectors of $\widehat{\Sigma}_n$ with small eigenvalues, is considered as novel. Concretely, we design reward bonus as:

$$\widehat{b}_n(s, a) = \min\left\{ 2c\sqrt{\phi(s, a)^\top \widehat{\Sigma}_n^{-1} \phi(s, a)}, H \right\} \tag{2}$$

with c being a parameter. Note we truncate it at H just because we know that the total reward is always upper bounded by H. To gain an intuition of the reward bonus, we can think about the tabular MDP as a special case here. To model tabular MDPs, one can design $\phi(s, a) \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ as a one-hot vector that encodes state-action pair (s, a). In this case, $\widehat{\Sigma}_n$ is a diagonal matrix and a diagonal entry approximates the probability of $\boldsymbol{\pi}_n$ visiting the corresponding state-action pair.

With reward bonus to encourage further exploration at novel state-action pairs, we invoke the planning oracle $\text{OP}(\Pi, r + \widehat{b}_n, \widehat{P}_n)$ to search for the best policy from Π that optimizes the combination $r + \widehat{b}_n$ under the empirical model $\widehat{P}_n$. The intuition here is that the optimal policy π_{n+1} returned by the planning oracle can visit novel state-action pairs due to reward bonus. Thus the new policy cover $\boldsymbol{\pi}_{n+1}$ expands the coverage of $\boldsymbol{\pi}_n$, which leads to exploration.

Such iterative framework—iteratively fitting model and collecting new data using the optimal policy of the learned model, is widely used in practical model-based approaches (e.g., (Ross & Bagnell, 2012; Kaiser et al., 2019)), and our framework simply designs additional reward bonus for the planning oracle. There are prior works that leverage heuristic reward bonus inside tree search (Schrittwieser et al., 2019) for discrete action domains, though the reward bonus in tree search in worst case performs no better than uniform random exploration (Munos, 2014).

For KNRs and Linear MDPs models, different from the LC3 algorithm (Kakade et al., 2020)[3] and the algorithm for linear MDP from (Yang & Wang, 2019), our algorithm avoids the optimistic planning oracle and abstracts the details (e.g., optimistic value iteration) away via a black-box planning oracle. Not only optimistic planning is computationally inefficient, but it also limits the flexibility of plugging in existing efficient planning oracles developed from the motion planning and control community. As we demonstrate in our experiments, Alg. 1 has flexibility to integrate rich function approximation (e.g., deep nets for modeling transition P), and furthermore state-of-the-art model-based planning algorithms. In our experiments, we use TRPO inside the learned model as the planner. Meanwhile, we also offer another implementation using MPPI as the planner, and we perform an empirical comparison between the two planners in Appendix C.1.

We note that the idea of policy cover was previously used in PC-PG (Agarwal et al., 2020a). While PC-PG achieves PAC bounds on linear MDPs, it cannot be applied to KNRs due to the potential nonlinear Q functions in KNRs.[4]

5. Analysis

Algorithm 1 simultaneously applies to both linear MDPs and KNRs except that it takes different parameterized model class $\mathcal{P}$ as algorithmic inputs. PC-MLP achieves polynomial sample complexity in both models, with access to a planning oracle. Note prior work on KNR requires an optimistic planning oracle, which is computationally inefficient even in linear bandit.

For KNRs, regarding the MLE oracle, we can approximately optimize it via SGD. Namely, in Eq. 1, we can draw one sample at a time, and perform one step of SGD. We

[3] While (Kakade et al., 2020) experimentally rely on Thompson sampling, it is unclear if their frequentist regret bounds still hold under Thompson sampling.

[4] PC-PG crucially relies on the fact that in linear MDPs the quantity $\mathbb{E}_{s' \sim P^\star(\cdot|s,a)}[f(s')]$ is linear with respect to $\phi(s, a)$ for any function f.

perform total M steps of SGD. Due to the specific parameterization for $\mathcal{P}$ for KNRs, we can guarantee a generalization bound that the learned model parameterized by $\widehat{W}^n$, is close to $W^\star$ under $d_{\boldsymbol{\pi}_n}$,[5]

$$\mathbb{E}_{s,a\sim d_{\boldsymbol{\pi}_n}} \|\widehat{W}^n \phi(s,a) - W^\star \phi(s,a)\|_2 \leq \widetilde{O}(1/\sqrt{M}).$$

Theorem 3 (KNRs). *Fix $\epsilon \in (0, H)$ and $\delta \in (0,1)$. With probability at least $1-\delta$, PC-MLP learns a policy π such that $J(\pi; r, P^\star) \geq \max_{\pi\in\Pi} J(\pi; r, P^\star) - \epsilon$, using number of samples* $\mathrm{Poly}\left(H, 1/\epsilon, \ln(1/\delta), d, d_s, F, \frac{1}{\sigma}\right)$.

The detailed parameter setup and the polynomial dependency can be found in Theorem 15. Unlike prior work LC3 which relies on optimistic planning, here we show that KNR is planning-oracle efficient PAC learnable. We note that in their experiments, (Kakade et al., 2020) use a Thompson sampling version of LC3, though without providing a regret proof of the Thompson sampling (TS) algorithm. To the best of our knowledge, while achieving a Bayesian regret bound is possible via TS, it is unclear if TS achieves a frequentist regret bound for KNRs.

For linear MDPs result, we need a stronger assumption on the MLE oracle. Specifically, we are going to assume that we can exactly solve Eq. 1, i.e., we can find the best $P \in \mathcal{P}$ that maximizes the *training* likelihood objective, i.e.,

$$\widehat{P}_n \in \operatorname*{argmax}_{P\in\mathcal{P}} \sum_{i=1}^{M} \ln P(s_i'|s_i, a_i).$$

This poses a slightly stronger assumption on the computational MLE oracle. Such offline computational oracle has been widely assumed in RL works using general function approximation (e.g., (Ross & Bagnell, 2012; Agarwal et al., 2020b)). Prior work (Agarwal et al., 2020b) established a generalization bound for MLE in terms of the total variation distance. Specifically, we can show that under realizability assumption $P^\star \in \mathcal{P}$, we have the following generalization bound:

$$\mathbb{E}_{s,a\sim d_{\boldsymbol{\pi}_n}} \|\widehat{\mu}^n \phi(s,a) - \mu^\star \phi(s,a)\|_{tv}^2 \leq \widetilde{O}(1/M).$$

See Theorem 21 from (Agarwal et al., 2020b) for example. With this MLE maximization oracle at training time, we get the following statement for linear MDPs.

Theorem 4 (Linear MDPs Model). *Fix $\epsilon \in (0, H)$ and $\delta \in (0,1)$. With probability at least $1-\delta$, PC-MLP learns a policy π such that $J(\pi; r, P^\star) \geq \max_{\pi\in\Pi} J(\pi; r, P^\star) - \epsilon$, using number of samples* $\mathrm{Poly}\left(H, 1/\epsilon, \ln(1/\delta), d, \ln\left(|\Upsilon|\right)\right)$.

Note that the sample complexity scales polynomially with respect to $\ln\left(|\Upsilon|\right)$ rather than the cardinality $|\Upsilon|$. Note that our model generalizes the linear MDP model from (Yang & Wang, 2019) as we can use classic covering argument here and $\ln(|\Upsilon|)$ will correspond to the covering dimension of the parameter space of the linear model from (Yang & Wang, 2019). We provide detailed hyper-parameter setup and polynomials in Theorem 14, and its proofs in Appendix A.

Theorems 3 and 4 indicate that the PC-MLP framework achieves an oracle-efficient PAC guarantee on KNRs and Linear MDPs simultaneously.

Algorithm 2 Deep PC-MLP

Require: MDP $\mathcal{M}$
Initialize π_1
Set replay buffer $\mathcal{D} = \emptyset$
Initialize model $\mathcal{P}_\theta$ with parameters θ
for $n = 1 \ldots$ **do**
  Draw K samples $\{s_i, a_i, s_i'\} \sim d_{\pi_n}$
  Set $\widehat{\Sigma}_{\pi_n} = \sum_{i=1}^{K} \phi(s_i, a_i)\phi(s_i, a_i)^\top / K$
  Set covariance matrix $\widehat{\Sigma}_n = \sum_{i=1}^{n} \Sigma_{\pi_i} + \lambda I$
  Add $\{s_i, a_i, s_i'\}_{i=1}^{K}$ to replay buffer $\mathcal{D}$
  Perform C steps of SGD on model P_θ (Eq. 3)
  Set reward bonus $\widehat{b}_n$ as in Eq. 2
  Denote $\pi_{n+1} :=$ TRPO $\left(r + \widehat{b}_n, P_\theta\right)$
end for

6. A Practical Algorithm: Deep PC-MLP

The PC-MLP framework and its analysis from the previous sections convey three important messages: (1) use a policy cover to ensure the learned model $\widehat{P}$ is accurate at the state-action space that is covered by all previous learned policies, (2) use the bonus to reward novel state-action pairs that are not covered by all previous policies, (3) and use a planning oracle on the combined reward to balance exploration and exploitation. PC-MLP relies on three modules: (1) MLE model fitting, (2) reward bonus design, (3) a planning oracle. In this section, we instantiate these three modules which lead to a practical implementation (Alg. 2) that is used in the experiment section.

In high level, the instantiation, *Deep PC-MLP* (Alg. 2), implements the above three modules using standard off-shelf techniques. For bonus, it uses Eq. 2 with ϕ being the value of the second from the last layer of a randomly initialized neural network, or a Random Fourier Feature (RFF) that corresponds to the RBF kernel (Rahimi & Recht, 2008). Randomly initialized network based bonus has been used in practice before in the model-free algorithm RND (Burda et al., 2019). Here we also demonstrate that such a bonus can be effective in the model-based setting. [6]

[5] The $\widetilde{O}$ notation hides absolute constants and log terms.

[6] In our setting, the random network shares the same structure

For model learning, we represent model class $\mathcal{P}$ as a class of feedforward neural networks P_θ with θ being the parameters to be optimized, that takes state-action pair as the input, and outputs the predicted next state. Under the assumption that both models and the true transition have Gaussian noise with the covariance matrix $\sigma^2 I$, negative log likelihood loss simply is reduced to standard ℓ_2 reconstruction loss. We use the standard replay buffer $\mathcal{D}$ to store all previous (s, a, s') triples. Note that since the replay buffer stores all prior experiences, it approximates the state-action coverage from the policy cover $\boldsymbol{\pi}_n$. We update the model with a few steps mini-batch SGD with the mini-batch $\{s_i, a_i, s'_i\}_{i=1}^{m+L}$ randomly sampled from the replay buffer $\mathcal{D}$. Here we use an L-step loss given its better empirical performance demonstrated in (Nagabandi et al., 2018; Luo et al., 2018):

$$\theta := \theta - \mu \sum_{i=1}^{m} \nabla_\theta \left(\sum_{l=1}^{L} \left\| (\hat{s}_{i+l} - \hat{s}_{i+l-1}) - (s_{i+l} - s_{i+l-1}) \right\|_2 \right), \tag{3}$$

where $\hat{s}_i = s_i$ and $\hat{s}_{i+l+1} = P_\theta(\hat{s}_{i+l}, a_{i+l})$. In our experiments, we use $L = 2$. [7]

For planning oracle, we use TRPO (Schulman et al., 2015) as our planner and adopt the framework in (Luo et al., 2018) where in each iteration we make multiple interchangeable updates between the model and the TRPO agent. We note that we can also leverage the state-of-art model-based planner such as Model-Predictive-Path Integral (MPPI) (Williams et al., 2017a) due to its excellent performance on challenging real-world continuous control tasks, for example, agile autonomous driving (Williams et al., 2017b). MPPI is easy to implement and for completeness, we include the pseudocode of MPPI in Appendix D.1. We also include an empirical comparison between the two planners (TRPO and MPPI) in Appendix C.1.

7. Experiments

In this section, we investigate the empirical performance of Deep PC-MPL. Our tests are mainly in two folds: we first experiment on sparse rewards experiments which are usually difficult for existing MBRL algorithms due to the demand for significant exploration. We then test our algorithm on benchmark environments with dense rewards,

as the deep dynamics model P_θ.

[7] We refer the construction of the L-step loss to Eq. (6.1) in (Luo et al., 2018). Note that such update attempts to minimize the gap between the difference between consecutive predicted states and difference between consecutive ground truth states.

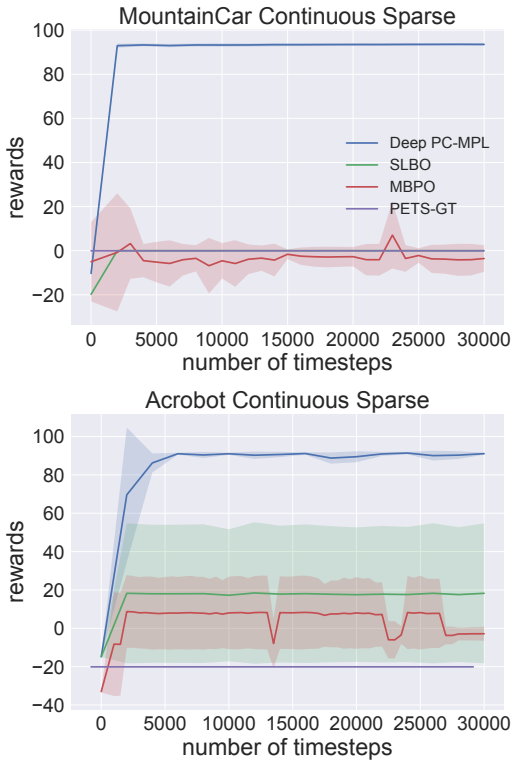

Figure 2. Episodic returns and success rates on MountainCar (top), Acrobot (bottom) environments with continuous action spaces, averaged over 4 random seeds. For the learning curve of HandEgg experiment, we refer back to Fig. 1. The solid line denotes the mean and the shaded area denotes one standard deviation. Note in the top plot the results of SLBO and PETS-GT almost overlap after the second iteration and thus the learning curve of SLBO is covered.

where exploration is not mandatory. Our results show that our practical algorithm achieves competitive results in both scenarios and a moderate amount of exploration induced by our algorithm can even boost the performance under the dense reward settings. In this section, we refer our algorithm as the version that uses ϕ from the random network and TRPO as the planner. We include all experiments and hyperparameter details in Appendix D.

7.1. Sparse Reward Environments

We first investigate how our algorithm mitigates the exploration issue faced by traditional MBRL algorithms with three continuous control and robotic environments in OpenAI Gym (Brockman et al., 2016). The first environment is the Mountain Car environment (Moore, 1990) with a continuous action space of $[-1, 1]$. Upon every timestep, a small control cost is incurred, and the learner receives no reward until they reach the goal where they receive a reward of 100 and the episode terminates. The second environment

	HalfCheetah	Hopper-ET	Walker2D-ET	Reacher	Ant
PETS-CEM	2795.3 ± 879.9	129.3 ± 36.0	-2.5 ± 6.8	-15.1 ± 1.7	1165.5 ± 226.9
SLBO	1097.7 ± 166.4	805.7 ± 142.4	207.8 ± 108.7	-4.1 ± 0.1	718.1 ± 123.3
Best MBBL	$\mathbf{3639.0 \pm 1185.8}$	926.9 ± 154.1	312.5 ± 493.4	-4.1 ± 0.1	$\mathbf{1852.1 \pm 141.0}$
Deep PC-MLP (Ours)	3599.9 ± 662.6	$\mathbf{1076.1 \pm 378.6}$	$\mathbf{1873.9 \pm 927.0}$	$\mathbf{-3.8 \pm 0.2}$	1583.1 ± 903.6
TRPO	-12 ± 85.5	237.4 ± 33.5	229.5 ± 27.1	-10.1 ± 0.6	323.3 ± 24.9
	Humanoid-ET	SlimHumanoid-ET	Swimmer	Swimmer-v0	Pendulum
PETS-CEM	110.8 ± 91.0	355.1 ± 157.1	306.3 ± 37.3	22.1 ± 25.2	167.4 ± 53.0
SLBO	1377.0 ± 150.4	776.1 ± 252.5	125.2 ± 93.2	46.1 ± 18.4	173.5 ± 2.5
Best MBBL	1377.0 ± 150.4	1084.3 ± 77.0	$\mathbf{336.3 \pm 15.8}$	$\mathbf{85.0 \pm 98.9}$	$\mathbf{177.3 \pm 1.9}$
DEEP PC-MLP (Ours)	$\mathbf{1775.5 \pm 322.2}$	$\mathbf{1521.0 \pm 232.6}$	161.8 ± 167.8	26.1 ± 21.0	174.0 ± 2.4
TRPO	289.8 ± 5.2	281.3 ± 10.9	215.7 ± 10.4	37.9 ± 2.0	166.7 ± 7.3

Table 1. Final performance for benchmark Mujoco locomotion and navigation tasks. The MBRL (top 4) algorithms are evaluated after 200k real world samples and the MFRL algorithm (TRPO) is evaluated after 1 million real world samples. All the results of the baselines (both model-based and model-free ones) are directly adopted from (Wang et al., 2019). We use the same 4 random seeds as in the benchmark paper while testing our algorithm and reporting the results. Here "ET" in the environment name denotes that the planner has access to the termination function, which is a common assumption in current MBRL algorithms (Janner et al., 2019; Rajeswaran et al., 2020). We use bold font to highlight the results of the algorithms with top 1 episodic rewards in each of the environments.

is Acrobot (Sutton, 1996; Geramifard et al., 2015), and while the original action space is discrete, here we change the action space to continuous with range $[-1, 1]$. We use the same reward scheme as in Mountain Car. The third environment is a hand manipulation task on the egg object (Plappert et al., 2018). We follow the original task design that incurs a constant penalty for not reaching the goal but we change the reward to 10 upon reaching the goal state. We remove the rotational perturbation but still keep the positional perturbation during goal sampling. This slightly relaxes the problem since our focus is on exploration issue instead of solving goal-conditioned RL problems and one algorithm is still required to learn the rotational dynamics by strategic exploration.

We observe that these three environments are very challenging to traditional MBRL algorithms: in the first two environments, because of the existence of the motor cost, one suboptimal policy is to just take actions with minimal motor costs. Thus the model will only be accurate around the initial states and never reach the goal state. For the manipulation environment, the huge state space and complex dynamics prohibit random exploration, which could require impractical numbers of samples to accurately capture the dynamics.

Here we compare with three MBRL baselines: a) PETS-GT (Chua et al., 2018) with CEM (Botev et al., 2013), and here we gives it the access to the *ground truth* dynamics. We also include baselines with moderate exploration power: b) SLBO (Luo et al., 2018) enforces entropy regularization during TRPO updates and adding Ornstein-Uhlunbeck noise while collecting samples. c) MBPO (Janner et al., 2019) uses SAC (Haarnoja et al., 2018) as the planner to encourage exploration. We plot the learning curves in Fig. 2. In the first two environments, while all the other baselines completely fail, Deep PC-MPL achieves the optimal performance within very few model updates. This indicates that with our constructed bonus, the planner has enough exploration power to reach the goal state with small numbers of samples. The results on the manipulation task also verify our hypothesis: with strategic exploration, our algorithm captures the dynamics in a reasonable number of samples and thus the planner learns to reach to the goal while planning under the accurate model. However, with the same number of samples, the other baseline (SLBO) with random exploration could barely reach the goal state with the inaccurate dynamics model due to insufficient exploration.

7.2. Dense Reward Environments

For dense reward environments, we follow the same setup as in the MBRL benchmark paper (Wang et al., 2019). We test Deep PC-MPL in 10 Mujoco (Todorov et al., 2012) locomotion and navigation environments. We report the final evaluation performance of our algorithm (after training with 200k real-world samples) in Table 1. For reference, we include the performances of PETS-CEM (Chua et al., 2018), SLBO (Luo et al., 2018), and the best MBRL algorithm in the benchmark paper for each corresponding environment (denoted as Best MBBL). We also include the evaluation result of our planner algorithm, TRPO, trained with 1 million real-world samples. Following the format in (Wang et al., 2019), we also provide the ranking of each algorithm in Table 2.

The results show that Deep PC-MPL achieves the best performances in 5 out of 10 environments, including the most challenging control environments such as Humanoid and

	PETS-CEM	SLBO
Mean Rank	5.6/11	4/11
Median Rank	6/11	5/11
	Best MBBL	Deep PC-MLP (Ours)
Mean Rank	4/11	2.4/11
Median Rank	4/11	1.5/11

Table 2. The rankings of our algorithm and other baselines in the 10 benchmark environments in Table 1. The rankings are out of 11 algorithms (10 baselines from the benchmark paper plus Deep PC-MLP). We rank the algorithms descendingly based on their mean episodic returns for each task. We then take the average position in the ranking (from 1 to 11) across the 10 tasks as the mean rank and likewise for the median rank. Note here the best mean and median rank from the benchmark paper (denoted as Best MBBL) may not be the rank of the same algorithm.

Walker. This indicates that our exploration scheme helps boost the performances even under dense reward settings. Note that our algorithm uses the same set of hyperparameters for all 10 environments.

7.3. Ablation Study on Bonus

Next, we investigate how the magnitude of the exploration bonus affects the performance of our algorithm. We control the amount of exploration by changing the bonus coefficient c defined in Eq. 2 and we show the learning curves of different coefficient choices in Fig. 3.

For sparse reward environment such as Mountain Car, lacking exploration results in a suboptimal policy that takes actions that minimize the motor cost, which resembles the behaviors of the other baseline algorithms in section 7.1. If the bonus signal is not strong enough ($c = 1$), we hypothesize that the control cost still dominates and thus it results in the same suboptimal policy with limited exploration around the initial states. For dense reward environments, existence of bonus ($c = 0.1$) outperforms situation where there is no bonus ($c = 0$). However, if the bonus signal is too strong ($c = 1$), it will focus too much on exploration while ignoring exploitation.

7.4. Ablation Study: Vanishing Bonus

According to the construction of bonus, our algorithm will assign small bonuses to state-action pairs that are already visited. Theoretically, the bonuses of all state-action pairs will eventually converge to 0 after we fully explore the environment. In this section, we investigate the trend of the bonus empirically in the sparse reward environment MountainCar. Fig. 4 shows the curve of the bonus per timestep that the planner receives during the evaluation phase in the real-world environment. Here all hyperparameters are fixed as in the settings in section 7.1. We discover that the average bonus converges to near 0 quickly, indicating that our algorithm can finish fully exploring the environment within the very first few iterations.

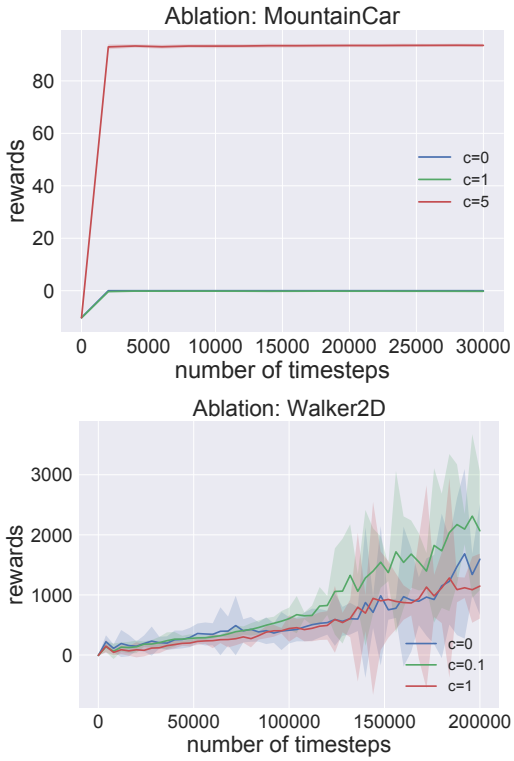

Figure 3. Ablation study: Comparison of different bonus coefficients in sparse reward environment MountainCar (top) and dense reward environment Walker2D (bottom). The results are averaged over 4 random seeds. The solid line denotes the mean and the shaded area denotes one standard deviation. In the first plot, note that the behaviors when $c = 0$ and $c = 1$ are identical thus the learning curves overlap.

8. Conclusion

In this paper, we introduce a new algorithm framework PC-MLP, which stands for Model Learning and Planning with Policy Cover for Exploration. We show that the same algorithm framework achieves polynomial sample complexity on both linear MDP and KNR models. Computation wise, the algorithm uses a reward bonus and a black-box planner that optimizes the combination of the ground truth reward and the reward bonus inside the learned models.

Our algorithm is modular and has great flexibility to integrate with modern rich function approximators such as deep neural networks. We provide a practical instantiation of PC-MLP where we use a deep neural network to model transition dynamics and uses reward bonus based on random features either from RFF or from a fully connected

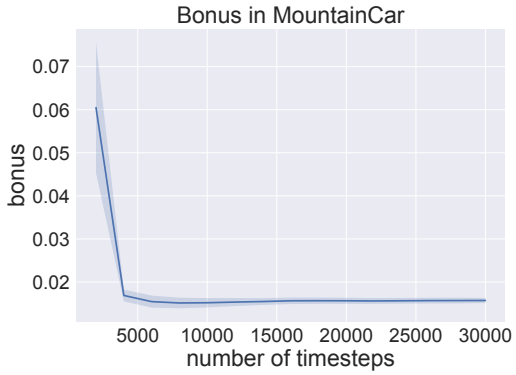

Figure 4. Decay of bonus per timestep as Deep PC-MPL fully explores the Mountain Car environment. The results are collected on the same random seeds as in section 7.1.

layer of a randomly initialized neural network. Our bonus scheme is simple and is motivated by classic linear bandit theory and recent RL theory on models beyond tabular MDPs. Extensive empirical results indicate that our algorithm works well on exploration challenging tasks including a high-dimensional manipulation task. For a common continuous control benchmark where the dense reward is available, our algorithm still provides competitive results. Further ablation study indicates that even for dense reward settings, a mild amount of exploration is helpful.

References

Abbasi-Yadkori, Y. and Szepesvári, C. Regret bounds for the adaptive control of linear quadratic systems. In *Conference on Learning Theory*, pp. 1–26, 2011.

Agarwal, A., Henaff, M., Kakade, S., and Sun, W. Pc-pg: Policy cover directed exploration for provable policy gradient learning. *arXiv preprint arXiv:2007.08459*, 2020a.

Agarwal, A., Kakade, S., Krishnamurthy, A., and Sun, W. Flambe: Structural complexity and representation learning of low rank mdps. *arXiv preprint arXiv:2006.10814*, 2020b.

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

Bansal, S., Calandra, R., Xiao, T., Levine, S., and Tomiin, C. J. Goal-driven dynamics learning via bayesian optimization. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 5168–5173. IEEE, 2017.

Botev, Z. I., Kroese, D. P., Rubinstein, R. Y., and L'Ecuyer, P. The cross-entropy method for optimization. In *Handbook of statistics*, volume 31, pp. 35–59. Elsevier, 2013.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=H1lJJnR5Ym`.

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

Cohen, A., Koren, T., and Mansour, Y. Learning linear-quadratic regulators efficiently with only sqrtT regret. In *International Conference on Machine Learning*, pp. 1300–1309, 2019.

Dani, V., Hayes, T. P., and Kakade, S. M. Stochastic linear optimization under bandit feedback. In *COLT*, pp. 355–366, 2008.

Dean, S., Mania, H., Matni, N., Recht, B., and Tu, S. Regret bounds for robust adaptive control of the linear quadratic regulator. In *Advances in Neural Information Processing Systems*, pp. 4188–4197, 2018.

Deisenroth, M. and Rasmussen, C. E. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, pp. 465–472, 2011.

Devroye, L., Mehrabian, A., and Reddad, T. The total variation distance between high-dimensional gaussians. *arXiv preprint arXiv:1810.08693*, 2018.

Fisac, J. F., Akametalu, A. K., Zeilinger, M. N., Kaynama, S., Gillula, J., and Tomlin, C. J. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7): 2737–2752, 2018.

Geramifard, A., Dann, C., Klein, R. H., Dabney, W., and How, J. P. Rlpy: a value-function-based reinforcement learning framework for education and research. *J. Mach. Learn. Res.*, 16(1):1573–1578, 2015.

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

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*, 2019.

Jin, C., Yang, Z., Wang, Z., and Jordan, M. I. Provably efficient reinforcement learning with linear function approximation. *arXiv preprint arXiv:1907.05388*, 2019.

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

Kakade, S., Krishnamurthy, A., Lowrey, K., Ohnishi, M., and Sun, W. Information theoretic regret bounds for online nonlinear control. *arXiv preprint arXiv:2006.12466*, 2020.

Karaman, S. and Frazzoli, E. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

Ko, J., Klein, D. J., Fox, D., and Haehnel, D. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *Proceedings 2007 ieee international conference on robotics and automation*, pp. 742–747. IEEE, 2007.

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

Levine, S. and Abbeel, P. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pp. 1071–1079, 2014.

Lu, X. and Van Roy, B. Information-theoretic confidence bounds for reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2458–2466, 2019.

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

Mania, H., Tu, S., and Recht, B. Certainty equivalent control of LQR is efficient. *arXiv preprint arXiv:1902.07826*, 2019.

Mania, H., Jordan, M. I., and Recht, B. Active learning for nonlinear system identification with guarantees. *arXiv preprint arXiv:2006.10277*, 2020.

Modi, A., Jiang, N., Tewari, A., and Singh, S. Sample complexity of reinforcement learning using linearly combined model ensembles. In *International Conference on Artificial Intelligence and Statistics*, pp. 2010–2020, 2020.

Moore, A. W. Efficient memory-based learning for robot control. Technical report, 1990.

Munos, R. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. 2014.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *IEEE International Conference on Robotics and Automation*, pp. 7559–7566, 2018.

Osband, I. and Van Roy, B. Model-based reinforcement learning and the Eluder dimension. In *Advances in Neural Information Processing Systems*, pp. 1466–1474, 2014.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pp. 1177–1184, 2008.

Rajeswaran, A., Mordatch, I., and Kumar, V. A game theoretic framework for model based reinforcement learning. In *International Conference on Machine Learning*, pp. 7953–7963. PMLR, 2020.

Ratliff, N., Zucker, M., Bagnell, J. A., and Srinivasa, S. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pp. 489–494. IEEE, 2009.

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

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

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Simchowitz, M. and Foster, D. J. Naive exploration is optimal for online LQR. *arXiv preprint arXiv:2001.09576*, 2020.

Sun, W., van den Berg, J., and Alterovitz, R. Stochastic extended lqr for optimization-based motion planning under uncertainty. *IEEE Transactions on Automation Science and Engineering*, 13(2):437–447, 2016.

Sun, W., Gordon, G. J., Boots, B., and Bagnell, J. Dual policy iteration. In *Advances in Neural Information Processing Systems*, pp. 7059–7069, 2018.

Sun, W., Jiang, N., Krishnamurthy, A., Agarwal, A., and Langford, J. Model-based RL in contextual decision processes: PAC bounds and exponential improvements over model-free approaches. In *Conference on Learning Theory*, pp. 1–36, 2019.

Sutton, R. S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, pp. 1038–1044, 1996.

Todorov, E. and Li, W. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.*, pp. 300–306. IEEE, 2005.

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

Umlauft, J., Pöhler, L., and Hirche, S. An uncertainty-based control lyapunov approach for control-affine systems modeled by gaussian process. *IEEE Control Systems Letters*, 2(3):483–488, 2018.

Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E., Zhang, S., Zhang, G., Abbeel, P., and Ba, J. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

Williams, G., Aldrich, A., and Theodorou, E. A. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017a.

Williams, G., Wagener, N., Goldfain, B., Drews, P., Rehg, J. M., Boots, B., and Theodorou, E. A. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1714–1721. IEEE, 2017b.

Yang, L. F. and Wang, M. Reinforcement leaning in feature space: Matrix bandit, kernels, and regret bound. *arXiv preprint arXiv:1905.10389*, 2019.

Zhou, D., He, J., and Gu, Q. Provably efficient reinforcement learning for discounted mdps with feature mapping. *arXiv preprint arXiv:2006.13165*, 2020.