Sample Complexity of Reinforcement Learning using Linearly Combined Model Ensembles

Aditya Modi[1] **Nan Jiang**[2] **Ambuj Tewari**[1] **Satinder Singh**[1]
[1]University of Michigan Ann Arbor [2]University of Illinois at Urbana-Champaign

Abstract

Reinforcement learning (RL) methods have been shown to be capable of learning intelligent behavior in rich domains. However, this has largely been done in simulated domains without adequate focus on the process of building the simulator. In this paper, we consider a setting where we have access to an ensemble of pre-trained and possibly inaccurate simulators (models). We approximate the real environment using a state-dependent linear combination of the ensemble, where the coefficients are determined by the given state features and some unknown parameters. Our proposed algorithm provably learns a near-optimal policy with a sample complexity polynomial in the number of unknown parameters, and incurs no dependence on the size of the state (or action) space. As an extension, we also consider the more challenging problem of model selection, where the state features are unknown and can be chosen from a large candidate set. We provide exponential lower bounds that illustrate the fundamental hardness of this problem, and develop a provably efficient algorithm under additional natural assumptions.

1 INTRODUCTION

Reinforcement learning methods with deep neural networks as function approximators have recently demonstrated prominent success in solving complex and observations rich tasks like games (Mnih et al., 2015; Silver et al., 2016), simulated control problems (Todorov et al., 2012; Lillicrap et al., 2015; Mordatch et al., 2016) and a range of robotics tasks (Christiano et al., 2016; Tobin et al., 2017). A common aspect in most of these success stories is the use of simulation. Arguably, given a simulator of the real environment, it is possible to use RL to learn a near-optimal policy from (usually a large amount of) simulation data. If the simulator is highly accurate, the learned policy should also perform well in the real environment.

Apart from some cases where the true environment and the simulator coincide (e.g., in game playing) or a nearly perfect simulator can be created from the law of physics (e.g., in simple control problems), in general we will need to construct the simulator using data from the real environment, making the overall approach an instance of *model-based RL*. As the algorithms for learning from simulated experience mature (which is what the RL community has mostly focused on), the bottleneck has shifted to the creation of a good simulator. *How can we learn a good model of the world from interaction experiences?*

A popular approach for meeting this challenge is to learn using a wide variety of simulators, which imparts robustness and adaptivity to the learned policies. Recent works have demonstrated the benefits of using such an ensemble of models, which can be used to either transfer policies from simulated to real-world domains, or to simply learn robust policies (Andrychowicz et al., 2018; Tobin et al., 2017; Rajeswaran et al., 2017). Borrowing the motivation from these empirical works, we notice that the process of learning a simulator inherently includes various choices like inductive biases, data collection policy, design aspects etc. As such, instead of relying on a sole approximate model for learning in simulation, interpolating between models obtained from different sources can provide better approximation of the real environment. Previous works like Buckman et al. (2018); Lee et al. (2019); Kurutach et al. (2018) have also demonstrated the effectiveness of using an ensemble of models for decreasing modelling error or its effect thereof during learning.

In this paper, we consider building an approximate model of the real environment from interaction data

Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

using a set (or *ensemble*) of possibly inaccurate models, which we will refer to as the *base models*. The simplest way to combine the base models is to take a weighted combination, but such an approach is rather limited. For instance, each base model might be accurate in certain regions of the state space, in which case it is natural to consider a state-dependent mixture. We consider the problem of learning in such a setting, where one has to identify an appropriate combination of the base models through real-world interactions, so that the induced policy performs well in the real environment. The data collected through interaction with the real world can be a precious resource and, therefore, we need the learning procedure to be sample-efficient. Our main result is an algorithm that enjoys polynomial sample complexity guarantees, where the polynomial has no dependence on the size of the state and action spaces. We also study a more challenging setting where the featurization of states for learning the combination is unknown and has to be discovered from a large set of candidate features.

Outline. We formally set up the problem and notation in Section 2, and discuss related work in Section 3. The main algorithm is introduced in Section 4, together with its sample complexity guarantees. We then proceed to the feature selection problem in Section 5 and conclude in Section 6.

2 SETTING AND NOTATION

We consider episodic Markov decision processes (MDP). An MDP M is specified by a tuple $(\mathcal{S}, \mathcal{A}, P, R, H, P_1)$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. P denotes the transition kernel describing the system dynamics $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ and R is the per-timestep reward function $R : \mathcal{S} \times \mathcal{A} \to \Delta([0, 1])$. The agent interacts with the environment for a fixed number of timesteps, H, which determines the horizon of the problem. The initial state distribution is P_1. The agent's goal is to find a policy $\pi : \mathcal{S} \times [H] \to \mathcal{A}$ which maximizes the value of the policy:

$$v_M^\pi := \mathbb{E}_{s\sim P_1}[V_{M,1}^\pi(s)]$$

where the value function at step h is defined as:

$$V_{M,h}^\pi(s) = \mathbb{E}\Big[\sum_{h'=h}^{H} r_{h'} \;\Big|\; s_h = s, a_{h:H} \sim \pi, s_{h:H} \sim M\Big]$$

Here we use "$s_{h:H} \sim M$" to imply that the sequence of states are generated according to the dynamics of M. A policy is said to be optimal for M if it maximizes the value v_M^π. We denote such a policy as π_M and its value as v_M. We use M^* to denote the model of the true environment, and use π^* and v^* as shorthand for π_{M^*} and v_{M^*}, respectively.

In our setting, the agent is given access to a set of K base MDPs $\{M_1, \ldots, M_K\}$. They share the same $\mathcal{S}, \mathcal{A}, H, P_1$, and only differ in P and R. In addition, a feature map $\phi : \mathcal{S} \times \mathcal{A} \to \Delta_{d-1}$ is given which maps state-action pairs to d-dimensional real vectors. Given these two objects, we consider the class of all models which can be obtained from the following state-dependent linear combination of the base models:

Definition 2.1 (Linear Combination of Base Models)**.** *For given model ensemble $\{M_1, \ldots, M_K\}$ and the feature map $\phi : \mathcal{S} \times \mathcal{A} \to \Delta_{d-1}$, we consider models parametrized by W with the following transition and reward functions:*

$$P^W(\cdot|s, a) = \sum_{k=1}^{K} [W\phi(s, a)]_k P^k(\cdot|s, a),$$
$$R^W(\cdot|s, a) = \sum_{k=1}^{K} [W\phi(s, a)]_k R^k(\cdot|s, a).$$

We will use $M(W)$ to denote such a model for any parameter $W \in \mathcal{W}$ with $\mathcal{W}_0 \equiv \{W \in [0, 1]^{K\times d} : \sum_{i=1}^{K} W_{ij} = 1$ for all $j \in [d]\}$.

For now, let's assume that there exists some W^* such that $M^* = M(W^*)$, i.e., the true environment can be captured by our model class; we will relax this assumption shortly.

To develop intuition, consider a simplified scenario where $d = 1$ and $\phi(s, a) \equiv 1$. In this case, the matrix W becomes a $K \times 1$ stochastic vector, and the true environment is approximated by a linear combination of the base models.

Example 2.2 (Global convex combination of models)**.** *If the base models are combined using a set of constant weights $w \in \Delta_{K-1}$, then this is a special case of Definition 2.1 where $d = 1$ and each state's feature vector is $\phi(s, a) \equiv 1$.*

In the more general case of $d > 1$, we allow the combination weights to be a linear transformation of the features, which are $W\phi(s, a)$, and hence obtain more flexibility in choosing different combination weights in different regions of the state-action space. A special case of this more general setting is when ϕ corresponds to a partition of the state-action space into multiple groups, and the linear combination coefficients are constant within each group.

Example 2.3 (State space partition)**.** *Let $\mathcal{S} \times \mathcal{A} = \bigcup_{i\in[d]} \mathcal{X}_i$ be a partition (i.e., $\{\mathcal{X}_i\}$ are disjoint). Let $\phi_i(s, a) = \mathbf{1}[(s, a) \in \mathcal{X}_i]$ for all $i \in [d]$ where $\mathbf{1}[\cdot]$ is the indicator function. This ϕ satisfies the condition that*

$\phi(s,a) \in \Delta_{d-1}$, and when combined with a set of base models, forms a special case of Definition 2.1.

Goal. We consider the popular PAC learning objective: with probability at least $1-\delta$, the algorithm should output a policy π with value $v^{\pi}_{M^*} \geq v^* - \epsilon$ by collecting $\text{poly}(d, K, H, 1/\epsilon, \log(1/\delta))$ episodes of data. Importantly, here the sample complexity is not allowed to depend on $|\mathcal{S}|$ or $|\mathcal{A}|$. However, the assumption that M^* lies the class of linear models can be limiting and, therefore, we will allow some approximation error in our setting as follows:

$$\theta := \min_{W \in \mathcal{W}} \sup_{(s,a) \in \mathcal{S} \times \mathcal{A}} \left\| P^*(\cdot|s,a) - P^W(\cdot|s,a) \right\|_1 + \left\| R^*(\cdot|s,a) - R^W(\cdot|s,a) \right\|_1 \quad (1)$$

We denote the optimal parameter attaining this value by W^*. The case of $\theta = 0$ represents the *realizable* setting where $M^* = M(W^*)$ for some $W^* \in \mathcal{W}$. When $\theta \neq 0$, we cannot guarantee returning a policy with the value close to v^*, and will have to pay an additional penalty term proportional to the approximation error θ, as is standard in RL theory.

Further Notations Let π_W be a shorthand for $\pi_{M(W)}$, the optimal policy in $M(W)$. When referring to value functions and state-action distributions, we will use the superscript to specify the policy and use the subscript to specify the MDP in which the policy is evaluated. For example, we will use $V^W_{W',h}$ to denote the value of π_W (the optimal policy for model $M(W)$) when evaluated in model $M(W')$ starting from timestep h. The term $d^W_{W',h}$ denotes the state-action distribution induced by policy π_W at timestep h in the MDP $M(W')$. Furthermore, we will write $V^W_{M^*,h}$ and $d^W_{M^*,h}$ when the evaluation environment is M^*. For conciseness, $V_{W,h}$ and $Q_{W,h}$ will denote the optimal (state- and Q-) value functions in model $M(W)$ at step h (e.g., $V_{W,h}(s) \equiv V^W_{W,h}(s)$). The expected return of a policy π in model $M(W)$ is defined as:

$$v^{\pi}_W = \mathbb{E}_{s \sim P_1}[V^{\pi}_{M(W),1}(s)]. \quad (2)$$

We assume that the total reward $\sum_{h=1}^{H} r_h$ lies in $[0,1]$ almost surely in all MDPs of interest and under all policies. Further, whenever used, any value function at step $H+1$ (e.g., $V^{\pi}_{W,H+1}$) evaluates to 0 for any policy and any model.

3 RELATED WORK

MDPs with low-rank transition matrices Yang and Wang (2019b,a); Jin et al. (2019) have recently considered structured MDPs whose transition matrices admit low-rank factorization, and the left matrix in the factorization are known to the learner as state-action features (corresponding to our ϕ). Their environmental assumption is a special case of ours, where the transition dynamics of each base model $P^k(\cdot|s,a)$ is *independent* of s and a, i.e., each base MDP can be fully specified by a single density distribution over $\mathcal{S}$. This special case enjoys many nice properties, such as the value function of any policy is also linear in state-action features, and the linear value-function class is closed under the Bellman update operators, which are heavily exploited in their algorithms and analyses. In contrast, none of these properties hold under our more general setup, yet we are still able to provide sample efficiency guarantees. That said, we do note that the special case allows these recent works to obtain stronger results: their algorithms are both statistically and computationally efficient (ours is only statistically efficient), and some of these algorithms work without knowing the K base distributions.[1] We leave the problem of utilizing this linear structure in a regret minimization framework (Jin et al., 2019) for future work.

Contextual MDPs Abbasi-Yadkori and Neu (2014); Modi et al. (2018); Modi and Tewari (2019) consider a setting similar to our Example 2.2, except that the linear combination coefficients are visible to the learner and the base models are unknown. Therefore, despite the similarity in environmental assumptions, the learning objectives and the resulting sample complexities are significantly different (e.g., their guarantees depend on $|\mathcal{S}|$ and $|\mathcal{A}|$).

Bellman rank Jiang et al. (2017) have identified a structural parameter called Bellman rank for exploration under general value-function approximation, and devised an algorithm called OLIVE whose sample complexity is polynomial in the Bellman rank. A related notion is the witness rank (the model-based analog of Bellman rank) proposed by Sun et al. (2019). While our algorithm and analyses draw inspiration from these works, our setting does not obviously yield low Bellman rank or witness rank.[2] We will also provide a more detailed comparison to Sun et al. (2019), whose algorithm is most similar to ours among the existing works, in Section 4.

[1] In our setting, not knowing the base models immediately leads to hardness of learning, as it is equivalent to learning a general MDP without any prior knowledge even when $d = K = 1$. This requires $\Omega(|\mathcal{S}||\mathcal{A}|)$ sample complexity (Azar et al., 2012), which is vacuous as we are interested in solving problems with arbitrarily large state and action spaces.

[2] In contrast, the low-rank MDPs considered by Yang and Wang (2019b,a); Jin et al. (2019) do admit low Bellman rank and low witness rank.

Mixtures/ensembles of models The closest work to our setting is the multiple model-based RL (MMRL) architecture proposed by Doya et al. (2002) where they also decompose a given domain as a convex combination of multiple models. However, instead of learning the combination coefficients for a given ensemble, their method trains the model ensemble and simultaneously learns a mixture weight for each *base model* as a function of state features. Their experiments demonstrate that each model specialized for different domains of the state space where the environment dynamics is predictable, thereby, providing a justification for using convex combination of models for simulation. Further, the idea of combining different models is inherently present in Bayesian learning methods where a posterior approximation of the real environment is iteratively refined using interaction data. For instance, Rajeswaran et al. (2017) introduce the EPOpt algorithm which uses an ensemble of simulated domains to learn robust and generalizable policies. During learning, they adapt the ensemble distribution (convex combination) over source domains using data from the target domain to progressively make it a better approximation. Similarly, Lee et al. (2019) combine a set of parameterized models by adaptively refining the mixture distribution over the latent parameter space. Here, we study a relatively simpler setting where a finite number of such base models are combined and give a frequentist sample complexity analysis for our method.

4 ALGORITHM AND MAIN RESULTS

In this section we introduce the main algorithm that learns a near-optimal policy in the aforementioned setup with a $\text{poly}(d, K, H, 1/\epsilon, \log(1/\delta))$ sample complexity. We will first give the intuition behind the algorithm, and then present the formal sample complexity guarantees. Due to space constraints, we present the complete proof in the appendix. For simplicity, we will describe the intuition for the realizable case with $\theta = 0$ ($P^* \equiv P^{W^*}$). The pseudocode (Algorithm 1) and the results are, however, stated for the general case of $\theta \neq 0$.

At a high level, our algorithm proceeds in iterations $t = 1, 2, \ldots$, and gradually refines a *version space* $\mathcal{W}_t$ of plausible parameters. Our algorithm follows an *explore-or-terminate* template and in each iteration, either chooses to explore with a carefully chosen policy or terminates with a near-optimal policy. For exploration in the t-th iteration, we collect n trajectories $\{(s_1^{(i)}, a_1^{(i)}, r_1^{(i)}, s_2^{(i)}, \ldots, s_H^{(i)}, a_H^{(i)}, r_H^{(i)})\}_{i\in[n]}$ following some exploration policy π_t (Line 7). A key component of the algorithm is to extract knowledge about W^* from these trajectories. In particular, for every h, the bag of samples $\{s_{h+1}^{(i)}\}_{i\in[n]}$ may be viewed as an unbiased draw from the following distribution

$$\frac{1}{n}\sum_{i=1}^{n} P^{W^*}\big(\cdot\,|s_h^{(i)}, a_h^{(i)}\big). \tag{7}$$

The situation for rewards is similar and will be omitted in the discussion. So in principle we could substitute W^* in Eq.(7) with any candidate W, and if the resulting distribution differs significantly from the real samples $\{s_{h+1}^{(i)}\}_{h\in[H],i\in[n]}$, we can assert that $W \neq W^*$ and eliminate W from the version space. However, the state space can be arbitrarily large in our setting, and comparing state distributions directly can be intractable. Instead, we project the state distribution in Eq.(7) using a (non-stationary) discriminator function $\{f_{t,h}\}_{h=1}^{H}$ (which will be chosen later) and consider the

Algorithm 1 PAC Algorithm for Linear Model Ensembles

1: **Input:** $\{M_1, \ldots M_K\}, \epsilon, \delta, \phi(\cdot,\cdot), \mathcal{W}_0$
2: **for** $t \to 1, 2, \ldots$ **do**
3: Compute *optimistic model* W_t and set π_t to π_{W_t}

$$W_t \leftarrow \arg\max_{W\in\mathcal{W}_{t-1}} V_W$$

4: Estimate the value of π_t using n_{eval} trajectories:

$$\hat{v}_t := \frac{1}{n_{\text{eval}}}\sum_{h=1}^{H} r_h^{(i)} \tag{3}$$

5: **if** $v_{W_t} - \hat{v}_t \leq 3\epsilon/4 + (3\sqrt{dK}+1)H\theta$ **then**
6: Terminate and output π_t
7: Collect n trajectories using $\pi_t : a_h \sim \pi_t(s_h)$
8: Estimate the matrix $\widehat{Z}_t$ and $\hat{y}_t$ as

$$\widehat{Z}_t := \frac{1}{n}\sum_{i=1}^{n}\sum_{h=1}^{H} \overline{V}_{t,h}\Big(s_h^{(i)}, a_h^{(i)}\Big)\,\phi\Big(s_h^{(i)}, a_h^{(i)}\Big)^\top \tag{4}$$

$$\hat{y}_t := \frac{1}{n}\sum_{i=1}^{n}\sum_{h=1}^{H} r_{h+1}^{(i)} + V_{t,h+1}\big(s_{h+1}^{(i)}\big) \tag{5}$$

9: Update the version space to $\mathcal{W}_t$ as the set:

$$\big\{W \in \mathcal{W}_{t-1} : \big|\hat{y}_t - \langle W, \widehat{Z}_t\rangle\big| \leq \tfrac{\epsilon}{12\sqrt{dK}} + H\theta\big\} \tag{6}$$

following scalar property

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{h=1}^{H}\mathbb{E}_{\substack{r\sim R^{W^*}\left(\cdot|s_h^{(i)},a_h^{(i)}\right),\\ s'\sim P^{W^*}\left(\cdot|s_h^{(i)},a_h^{(i)}\right)}}\Big[r+f_{t,h+1}(s')\Big], \quad (8)$$

which can be effectively estimated by

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{h=1}^{H}\Big(r_h^{(i)}+f_{t,h+1}\big(s_{h+1}^{(i)}\big)\Big). \quad (9)$$

Since we have projected states onto $\mathbb{R}$, Eq.(9) is the average of scalar random variables and enjoys state-space-independent concentration. Now, in order to test the validity of a parameter W in a given version space, we compare the estimate in eq. 9 with the prediction given by $M(W)$, which is:

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{h=1}^{H}\mathbb{E}_{\substack{r\sim R^{W}\left(\cdot|s_h^{(i)},a_h^{(i)}\right),\\ s'\sim P^{W}\left(\cdot|s_h^{(i)},a_h^{(i)}\right)}}\Big[r+f_{t,h+1}(s')\Big]. \quad (10)$$

As we consider a linear model class, by using linearity of expectations, Eq.(10) may also be written as:

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{h=1}^{H}\Big[W\phi(s_h^{(i)},a_h^{(i)})\Big]^{\top}\Big[\overline{V}_{t,h}(s_h^{(i)},a_h^{(i)})\Big] \quad (11)$$

$$=\Big\langle W,\frac{1}{n}\sum_{i=1}^{n}\sum_{h=1}^{H}\overline{V}_{t,h}(s_h^{(i)},a_h^{(i)})\,\phi(s_h^{(i)},a_h^{(i)})^{\top}\Big\rangle, \quad (12)$$

where $\langle A,B\rangle$ denotes $\mathrm{Tr}(A^{\top}B)$ for any two matrices A and B. In eq. 12, $\overline{V}_{t,h}$ is a function that maps (s,a) to a K dimensional vector with each entry being

$$\big[\overline{V}_{t,h}(s,a)\big]_k \coloneqq \mathbb{E}_{\substack{r\sim R^k(\cdot|s,a),\\ s'\sim P^k(\cdot|s,a)}}\Big[r+f_{t,h+1}(s')\Big]. \quad (13)$$

The intuition behind Eq.(11) is that for each fixed state-action pair $(s_h^{(i)},a_h^{(i)})$, the expectation in Eq.(8) can be computed by first taking expectation of $r+f_{t,h+1}(s')$ over the reward and transition distributions of each of the K base models—which gives $\overline{V}_h$—and then aggregating the results using the combination coefficients. Rewriting Eq.(11) as Eq.(12), we see that Eq.(8) can also be viewed as a linear measurement of W^*, where the measurement matrix is again $\frac{1}{n}\sum_{i=1}^{n}\sum_{h=1}^{H}\overline{V}_h\big(s_h^{(i)},a_h^{(i)}\big)\,\phi\big(s_h^{(i)},a_h^{(i)}\big)^{\top}$. Therefore, by estimating this measurement matrix and the outcome (Eq.(9)), we obtain an approximate linear equality constraint over $\mathcal{W}_{t-1}$ and can eliminate any candidate W that violates such constraints. By using a finite sample concentration bound over the inner product, we get a linear inequality constraint to update the version space (Eq.(6)).

The remaining concern is to choose the exploration policy π_t and the discriminator function $\{f_{t,h}\}$ to ensure that the linear constraint induced in each iteration is significantly different from the previous ones and induces deep cuts in the version space. We guarantee this by choosing $\pi_t := \pi_{W_t}$ and $f_{t,h} := V_{W_t,h}$[3], where W_t is the *optimistic model* as computed in Line 3. That is, W_t predicts the highest optimal value among all candidate models in $\mathcal{W}_{t-1}$. Following a terminate-or-explore argument, we show that as long as π_t is suboptimal, the linear constraint induced by our choice of π_t and $\{f_{t,h}\}$ will significantly reduce the volume of the version space, and the iteration complexity can be bounded as $\mathrm{poly}(d,K)$ by an ellipsoid argument similar to that of Jiang et al. (2017). Similarly, the sample size needed in each iteration only depends polynomially on d and K and incurs no dependence on $|\mathcal{S}|$ or $|\mathcal{A}|$, as we have summarized high-dimensional objects such as $f_{t,h}$ (function over states) using low-dimensional quantities such as $\overline{V}_{t,h}$ (vector of length K).

The bound on the number of iterations and the number of samples needed per iteration leads to the following sample complexity result:

Theorem 4.1 (PAC bound for Alg. 1)**.** *In Algorithm 1, if* $n_{eval} \coloneqq \frac{32H^2}{\epsilon^2}\log\frac{4T}{\delta}$ *and* $n=\frac{1800d^2KH^2}{\epsilon^2}\log\frac{8dKT}{\delta}$ *where* $T=dK\log\frac{2\sqrt{2K}H}{\epsilon}/\log\frac{5}{3}$*, with probability at least* $1-\delta$*, the algorithm terminates after using at most*

$$\widetilde{\mathcal{O}}\Big(\frac{d^3K^2H^2}{\epsilon^2}\log\frac{dKH}{\delta}\Big) \quad (14)$$

trajectories and returns a policy π_T *with a value* $v^T \geq v^* - \epsilon - (3\sqrt{dK}+2)H\theta$*.*

By setting d and K to appropriate values, we obtain the following sample complexity bounds as corollaries:

Corollary 4.2 (Sample complexity for partitions)**.** *Since the state-action partitioning setting (Example 2.3) is subsumed by the general setup, the sample complexity is again:*

$$\widetilde{\mathcal{O}}\Big(\frac{d^3K^2H^2}{\epsilon^2}\log\frac{dKH}{\delta}\Big) \quad (15)$$

Corollary 4.3 (Sample complexity for global convex combination)**.** *When base models are combined without any dependence on state-action features (Example 2.2), the setting is special case of the general setup with* $d=1$*. Thus, the sample complexity is:*

$$\widetilde{\mathcal{O}}\Big(\frac{K^2H^2}{\epsilon^2}\log\frac{KH}{\delta}\Big) \quad (16)$$

Our algorithm, therefore, satisfies the requirement of learning a near-optimal policy without any dependence

[3]We use the simplified notation $V_{t,h}$ for $V_{W_t,h}$.

on the $|\mathcal{S}|$ or $|\mathcal{A}|$. Moreover, we can also account for the approximation error θ but also incur a cost of $(3\sqrt{dK}+1)H\theta$ in the performance guarantee of the final policy. As we use the projection of value functions through the linear model class, we do not model the complete dynamics of the environment. This leads to an additive loss of $3\sqrt{dK}H\theta$ in value in addition to the best achievable value loss of $2H\theta$ (see Corollary A.4 in the appendix).

Comparison to OLIME (Sun et al., 2019) Our Algorithm 1 shares some structural similarity with the OLIME algorithm proposed by Sun et al. (2019), but there are also several important differences. First of all, OLIME in each iteration will pick a time step and take uniformly random actions during data collection, and consequently incur polynomial dependence on $|\mathcal{A}|$ in its sample complexity. In comparison, our main data collection step (Line 7) never takes a random deviation, and we do not pay any dependence on the cardinality of the action space. Secondly, similar to how we project the transition distributions onto a discriminator function (Eq.(7) and (8)), OLIME projects the distributions onto a *static discriminator class* and uses the corresponding integral probability metric (IPM) as a measure of model misfit. In our setting, however, we find that the most efficient and elegant way to extract knowledge from data is to use a *dynamic* discriminator function, $V_{W_t,h}$, which changes from iteration to iteration and depends on the previously collected data. Such a choice of discriminator function allows us to make direct cuts on the parameter space $\mathcal{W}$, whereas OLIME can only make cuts in the value prediction space.

Computational Characteristics In each iteration, our algorithm computes the optimistic policy within the version space. Therefore, we rely on access to the following *optimistic planning oracle*:

Assumption 4.4 (Optimistic planning oracle)**.** *We assume that when given a version space $\mathcal{W}_t$, we can obtain the optimistic model through a single oracle call for $W_t = \arg\max_{W\in\mathcal{W}_t} V_W$.*

It is important to note that any version space $\mathcal{W}_t$ that we deal with is always an intersection of half-spaces induced by the linear inequality constraints. Therefore, one would hope to solve the optimistic planning problem in a computationally efficient manner given the nice geometrical form of the version space. However, even for a finite state-action space, we are not aware of any efficient solutions as the planning problem induces bilinear and non-convex constraints despite the linearity assumption. Many recently proposed algorithms also suffer from such a computational difficulty (Jiang et al., 2017; Dann et al., 2018; Sun et al., 2019).

Further, we also assume that for any given W, we can compute the optimal policy π_W and its value function: our elimination criteria in Eq. 6 uses estimates $\widehat{Z}_t$ and $\hat{y}_t$ which in turn depend on the value function. This requirement corresponds to a standard planning oracle, and aligns with the motivation of our setting, as we can delegate these computations to any learning algorithm operating in the simulated environment with the given combination coefficient. Our algorithm, instead, focuses on careful and systematic exploration to minimize the sample complexity in the real world.

5 MODEL SELECTION WITH CANDIDATE PARTITIONS

In the previous section we showed that a near-optimal policy can be PAC-learned under our modeling assumptions, where the feature map $\phi : \mathcal{S}\times\mathcal{A}\to[0,1]$ is given along with the approximation error θ. In this section, we explore the more interesting and challenging setting where a realizable feature map ϕ is unknown, but we know that the realizable ϕ belongs to a candidate set $\{\phi_i\}_{i=1}^N$, i.e., the true environment satisfies our modeling assumption in Definition 2.1 under $\phi=\phi_{i^*}$ for some $i^*\in[N]$ with $\theta_{i^*}=0$. Note that Definition 2.1 may be satisfied by multiple ϕ_i's; for example, adding redundant features to an already realizable ϕ_{i^*} still yields a realizable feature map. In such cases, we consider ϕ_{i^*} to be the most succinct feature map among all realizable ones, i.e., the one with the lowest dimensionality. Let d_i denote the dimensionality of ϕ_i, and $d^*=d_{i^*}$.

One obvious baseline in this setup is to run Algorithm 1 with each ϕ_i and select the best policy among the returned ones. This leads to a sample complexity of roughly $\sum_{i=1}^N d_i^3$ (only the dependence on $\{d_i\}_{i=1}^N$ is considered), which can be very inefficient: When there exists j such that $d^*\ll d_j$, we pay for d_j^3 which is much greater than the sample complexity of d^*; When $\{d_i\}$ are relatively uniform, we pay a linear dependence on N, preventing us from competing with a large set of candidate feature maps.

So the key result we want to obtain is a sample complexity that scales as $(d^*)^3$, possibly with a mild multiplicative overhead dependence on d^* and/or N (e.g., $\log d^*$ and $\log N$).

Hardness Result for Unstructured $\{\phi_i\}$
Unfortunately, we show that this is impossible when $\{\phi_i\}$ is unstructured via a lower bound. In the lower bound construction, we have an exponentially large set of candidate feature maps, all of which are state space partitions. Each of the partitions has trivial dimensionalities ($d_i=2$, $K=2$), but the sample

complexity of learning is exponential, which can only be explained away as $\Omega(N)$.

Proposition 5.1. *For the aforementioned problem of learning an ϵ-optimal policy using a candidate feature set of size N, no algorithm can achieve $poly(d^*, K, H, 1/\epsilon, 1/\delta, N^{1-\alpha})$ sample complexity for any constant $0 < \alpha < 1$.*

On a separate note, besides providing formal justification for the structural assumption we will introduce later, this proposition is of independent interest as it also sheds light on the hardness of model selection with state abstractions. We discuss the further implications in Appendix B.

Proof of Proposition 5.1. We construct a linear class of MDPs with two base models M_1 and M_2 in the following way: Consider a complete tree of depth H with a branching factor of 2. The vertices forming the state space of M_1 and M_2 and the two outgoing edges in each state are the available actions. Both MDPs share the same deterministic transitions and each non-leaf node yields 0 reward. Every leaf node yields +1 reward in M_1 and 0 in M_2. Now we construct a candidate partition set $\{\phi_i\}$ of size 2^H: for ϕ_i, the i-th leaf node belongs to one equivalence class while all other leaf nodes belong to the other. (Non-leaf nodes can belong to either class as M_1 and M_2 agree on their transitions and rewards.)

Observe that the above model class contains a finite family of 2^H MDPs, each of which only has 1 rewarding leaf node. Concretely, the MDP whose i-th leaf is rewarding is exactly realized under the feature map ϕ_i, whose corresponding W^* is the identity matrix: the i-th leaf yields +1 reward as in M_1, and all other leaves yield 0 reward as in M_2. Learning in this family of 2^H MDPs is provably hard (Krishnamurthy et al., 2016), as when the rewarding leaf is chosen adversarially, the learner has no choice but to visit almost all leaf nodes to identify the rewarding leaf as long as ϵ is below a constant threshold. The proposition follows from the fact that in this setting $d^* = 2$, $K = 2$, $1/\epsilon$ is a constant, $N = 2^H$, but the sample complexity is $\Omega(2^H)$. □

This lower bound shows the necessity of introducing structural assumptions in $\{\phi_i\}$. Below, we consider a particular structure of *nested partitions* that is natural and enables sample-efficient learning. Similar assumptions have also been considered in the state abstraction literature (e.g., Jiang et al., 2015).

Nested Partitions as a Structural Assumption

Consider the case where every ϕ_i is a partition. W.l.o.g. let $d_1 \leq d_2 \leq \ldots \leq d_N$. We assume $\{\phi_i\}$ is nested, meaning that $\forall (s,a),(s',a')$,

$$\phi_i(s,a) = \phi_i(s',a') \implies \phi_j(s,a) = \phi_j(s',a'), \quad \forall i \leq j.$$

While this structural assumption almost allows us to develop sample-efficient algorithms, it is still insufficient as demonstrated by the following hardness result.

Proposition 5.2. *Fixing $K = 2$, there exist base models M_1 and M_2 and* nested *state space partitions ϕ_1 and ϕ_2, such that it is information-theoretically impossible for any algorithm to obtain $poly(d^*, H, K, 1/\epsilon, 1/\delta)$ sample complexity when an adversary chooses an MDP that satisfies our environmental assumption (Definition 2.1) under either ϕ_1 or ϕ_2.*

Proof. We will again use an exponential tree style construction to prove the lower bound. Specifically, we construct two MDPs M and M' which are obtained by combining two base MDPs M_1 and M_2 using two different partitions ϕ_1 and ϕ_2. The specification of M_1 and M_2 is exactly the same as in the proof of Proposition 5.1. We choose ϕ_1 to be a partition of size $d_1 = 1$, where all nodes are grouped together. ϕ_2 has size $d_2 = 2^H$, where each leaf node belongs to a separate group. (As before, which group the inner nodes belong to does not matter.) ϕ_1 and ϕ_2 are obviously nested. We construct M that is realizable under ϕ_2 by randomly choosing a leaf and setting the weights for the convex combination as $(1/2 + 2\epsilon, 1/2 - 2\epsilon)$ for that leaf; for all other leafs, the weights are $(1/2, 1/2)$. This is equivalent to randomly choosing M from a set of 2^H MDPs, each of which has only one *good* leaf node yielding a random reward drawn from $\text{Ber}(1/2 + 2\epsilon)$ instead of $\text{Ber}(1/2)$. In contrast, M' is such that all leaf nodes yield $\text{Ber}(1/2)$ reward, which is realizable under ϕ_1 with weights $(1/2, 1/2)$.

Observe that M and M' are exactly the same as the constructions in the proof of the multi-armed bandit lower bound by (Auer et al., 2002) (the number of arms is 2^H), where it has been shown that distinguishing between M and M' takes $\Omega(2^H/\epsilon^2)$ samples. Now assume towards contradiction that there exists an algorithm that achieves $\text{poly}(d^*, H, K, 1/\epsilon, 1/\delta)$ complexity; let f be the specific polynomial in its guarantee. After $f(1, H, 2, 1/\epsilon, 1/\delta)$ trajectories are collected, the algorithm must stop if the true environment is M' to honor the sample complexity guarantee (since $d^* = 1$, $K = 2$), and proceed to collect more trajectories if M is the true environment (since $d^* = 2^H$). Making this decision essentially requires distinguishing between M' and M using $f(1, H, 2, 1/\epsilon, 1/\delta) = \text{poly}(H)$ trajectories, which contradicts the known hardness result from Auer et al. (2002). This proves the statement. □

Essentially, the lower bound creates a situation where

$d_1 \ll d_2$, and the nature may adversarially choose a model such that either ϕ_1 or ϕ_2 is realizable. If ϕ_1 is realizable, the learner is only allowed a small sample budget and cannot fully explore with ϕ_2, and if ϕ_1 is not realizable the learner must do the opposite. The information-theoretic lower bound shows that it is fundamentally hard to distinguish between the two situations: Once the learner explores with ϕ_1, she cannot decide whether she should stop or move on to ϕ_2 without collecting a large amount of data.

This hardness result motivates our last assumption in this section, that the learner knows the value of $v^\star$ (a scalar) as side information. This way, the learner can compare the value of the returned policy in each round to $v^\star$ and effectively decide when to stop. This naturally leads to our Algorithm 2 that uses a doubling scheme over $\{d_i\}$, with the following sample complexity guarantee.

Algorithm 2 Model Selection with Nested Partitions

Input:$\{\phi_1, \phi_2, \ldots, \phi_N\}$, $\{M_1, \ldots, M_K\}$, ϵ, δ, $v^\star$.
$i \to 0$
while `True` **do**
  Choose ϕ_i such that d_i is the largest among $\{d_j : d_j \le 2^i\}$.
  Run Algorithm 1 on Φ_i with $\epsilon_i = \frac{\epsilon}{2}$ and $\delta_i = \frac{\delta}{2N}$.
  Terminate the sub-routine if $t > d_i K \log \frac{2\sqrt{2K}H}{\epsilon} / \log \frac{5}{3}$.
  Let π_i be the returned policy (if any). Let $\hat{v}_i$ be the estimated return of π_i using $n_{\text{eval}} = \frac{9}{2\epsilon^2} \log \frac{2N}{\delta}$ Monte-Carlo trajectories.
  if $\hat{v}_i \ge v^* - \frac{2\epsilon}{3}$ **then**
    Terminate with output π_i.

Theorem 5.3. *When Algorithm 2 is run with the input v^*, with probability at least $1-\delta$, it returns a near-optimal policy π with $v^\pi \ge v^* - \epsilon$ using at most $\tilde{\mathcal{O}}\Big(\frac{d^{*3}K^2H^2}{\epsilon^2} \log d^* \log \frac{d^*KHN}{\delta}\Big)$ samples.*

Proof. In Algorithm 2, for each partition i, we run Algorithm 1 until termination or until the sample budget is exhausted. By union bound it is easy to verify that with probability at least $1-\delta$, all calls to Algorithm 1 will succeed and the Monte-Carlo estimation of the returned policies will be $(\epsilon/3)$-accurate, and we will only consider this success event in the rest of the proof. When the partition under consideration is realizable, we get $v^{\pi_i}_{M^*} \ge v^* - \epsilon/3$, therefore

$$\hat{v}^i \ge v^{\pi_i} - \tfrac{\epsilon}{3} \ge v^* - \tfrac{2\epsilon}{3},$$

so the algorithm will terminate after considering a realizable ϕ_i. Similarly, whenever the algorithm terminates, we have $v^{\pi_i} \ge v^* - \epsilon$. This is because

$$v^{\pi_i} \ge \hat{v}^i - \tfrac{\epsilon}{3} \ge v^* - \epsilon,$$

where the last inequality holds thanks to the termination condition of Algorithm 2, which relies on knowledge of $v^\star$. The total number of iterations of the algorithm is at most $\mathcal{O}(\log d^*)$. Therefore, by taking a union bound over all possible iterations, the sample complexity is

$$\sum_{i=1}^{J} \widetilde{\mathcal{O}}\Big(\frac{d_i^3 K^2 H^2}{\epsilon^2}\Big) \le \widetilde{\mathcal{O}}\Big(\frac{d^{*3}K^2H^2}{\epsilon^2} \log d^*\Big). \qquad \square$$

Discussion. Model selection in online learning—especially in the context of sequential decision making—is generally considered very challenging. There has been relatively limited work in the generic setting until recently for some special cases. For instance, Foster et al. (2019) consider the model selection problem in linear contextual bandits with a sequence of nested policy classes with dimensions $d_1 < d_2 < \ldots$. They consider a similar goal of achieving sub-linear regret bounds which only scale with the optimal dimension d_{m^*}. In contrast to our result, they do not need to know the achievable value in the environment and give no-regret learning methods in the *knowledge-free* setting. However, this is not contradictory to our lower bound: Due to the extremely delayed reward signal, our construction is equivalent to a multi-armed bandit problem with 2^H arms. Our negative result (Proposition 5.2) shows a lower bound on sample complexity which is exponential in horizon, therefore eliminating the possibility of sample efficient and knowledge-free model selection in MDPs.

6 CONCLUSION

In this paper, we proposed a sample efficient model based algorithms which learns a near-optimal policy by approximating the true environment via a feature dependent convex combination of a given ensemble. Our algorithm offers a sample complexity bound which is independent of the size of the environment and only depends on the number of parameters being learnt. In addition, we also consider a model selection problem, show exponential lower bounds and then give sample efficient methods under natural assumptions. The proposed algorithm and its analysis relies on a linearity assumption and shares this aspect with existing exploration methods for rich observation MDPs. We leave the possibility of considering a richer class of convex combinations to future work. Lastly, our work also revisits the open problem of coming up with a computational and sample efficient model based learning algorithm.

Acknowledgements

This work was supported in part by a grant from the Open Philanthropy Project to the Center for Human-Compatible AI, and in part by NSF grant CAREER IIS-1452099.

References

Abbasi-Yadkori, Y. and Neu, G. (2014). Online learning in mdps with side information. *arXiv preprint arXiv:1406.6812*.

Agarwal, A., Rakhlin, A., and Bartlett, P. (2008). Matrix regularization techniques for online multitask learning. Technical Report UCB/EECS-2008-138, EECS Department, University of California, Berkeley.

Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. (2018). Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Azar, M. G., Munos, R., and Kappen, H. J. (2012). On the sample complexity of reinforcement learning with a generative model. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1707–1714. Omnipress.

Buckman, J., Hafner, D., Tucker, G., Brevdo, E., and Lee, H. (2018). Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, pages 8224–8234.

Christiano, P., Shah, Z., Mordatch, I., Schneider, J., Blackwell, T., Tobin, J., Abbeel, P., and Zaremba, W. (2016). Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*.

Dann, C., Jiang, N., Krishnamurthy, A., Agarwal, A., Langford, J., and Schapire, R. E. (2018). On oracle-efficient pac rl with rich observations. In *Advances in Neural Information Processing Systems*, pages 1422–1432.

Doya, K., Samejima, K., Katagiri, K.-i., and Kawato, M. (2002). Multiple model-based reinforcement learning. *Neural computation*, 14(6):1347–1369.

Foster, D. J., Krishnamurthy, A., and Luo, H. (2019). Model selection for contextual bandits. *arXiv preprint arXiv:1906.00531*.

Givan, R., Dean, T., and Greig, M. (2003). Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223.

Jiang, N., Krishnamurthy, A., Agarwal, A., Langford, J., and Schapire, R. E. (2017). Contextual decision processes with low bellman rank are pac-learnable. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1704–1713. JMLR. org.

Jiang, N., Kulesza, A., and Singh, S. (2015). Abstraction selection in model-based reinforcement learning. In *International Conference on Machine Learning*, pages 179–188.

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

Kearns, M. and Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232.

Krishnamurthy, A., Agarwal, A., and Langford, J. (2016). Pac reinforcement learning with rich observations. In *Advances in Neural Information Processing Systems 29*, pages 1840–1848.

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. (2018). Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*.

Lee, G., Hou, B., Mandalika, A., Lee, J., and Srinivasa, S. S. (2019). Bayesian policy optimization for model uncertainty. In *International Conference on Learning Representations*.

Li, L., Walsh, T. J., and Littman, M. L. (2006). Towards a unified theory of state abstraction for mdps. In *ISAIM*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

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

Modi, A., Jiang, N., Singh, S., and Tewari, A. (2018). Markov decision processes with continuous side information. In *Algorithmic Learning Theory*, pages 597–618.

Modi, A. and Tewari, A. (2019). Contextual markov decision processes using generalized linear models. *arXiv preprint arXiv:1903.06187*.

Mordatch, I., Mishra, N., Eppner, C., and Abbeel, P. (2016). Combining model-based policy search with online model learning for control of physical humanoids. In *2016 IEEE International Conference*

on Robotics and Automation (ICRA), pages 242–248. IEEE.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. (2017). Epopt: Learning robust neural network policies using model ensembles. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484.

Sun, W., Jiang, N., Krishnamurthy, A., Agarwal, A., and Langford, J. (2019). Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches. In Beygelzimer, A. and Hsu, D., editors, *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 2898–2933, Phoenix, USA. PMLR.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE.

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

Whitt, W. (1978). Approximations of dynamic programs, i. *Mathematics of Operations Research*, 3(3):231–243.

Yang, L. and Wang, M. (2019a). Sample-optimal parametric q-learning using linearly additive features. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6995–7004, Long Beach, California, USA. PMLR.

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