# OPTIMISTIC EXPLORATION WITH LEARNED FEATURES PROVABLY SOLVES MARKOV DECISION PROCESSES WITH NEURAL DYNAMICS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Incorporated with the recent advances in deep learning, deep reinforcement learning (DRL) has achieved tremendous success in empirical study. However, analyzing DRL is still challenging due to the complexity of the neural network class. In this paper, we address such a challenge by analyzing the Markov decision process (MDP) with neural dynamics, which covers several existing models as special cases, including the kernelized nonlinear regulator (KNR) model and the linear MDP. We propose a novel algorithm that designs exploration incentives via learnable representations of the dynamics model by embedding the neural dynamics into a kernel space induced by the system noise. We further establish an upper bound on the sample complexity of the algorithm, which demonstrates the sample efficiency of the algorithm. We highlight that, unlike previous analyses of RL algorithms with function approximation, our bound on the sample complexity does not depend on the Eluder dimension of the neural network class, which is known to be exponentially large (Dong et al., 2021).

**Keywords:** Reinforcement Learning, Neural Network, Representation Learning.

## 1 INTRODUCTION

Reinforcement learning (RL) aims to accomplish sequential decision-making in an uncertain environment via iteratively interacting with the environment (see Sutton et al. (1998)). Equipped with modern function approximators such as deep neural networks, deep RL algorithms achieve tremendous empirical successes (Mnih et al., 2015; Silver et al., 2017; Hafner et al., 2019).

Despite its empirical successes, the theoretical understanding of deep RL is relatively underdeveloped. There are several recent works (Abbasi-Yadkori et al., 2019; Wang et al., 2019; Fan et al., 2020) that analyze RL algorithms with neural network parameterization, including policy iteration (PI) (Lagoudakis & Parr, 2003), policy gradient (PG) (Williams, 1992) and deep Q-learning (Mnih et al., 2013). However, those works depend on restrictive assumptions that either the agent has access to a simulator or the MDPs have bounded concentrability coefficients, which in fact imply that the state space is already well-explored. Another line of research (Jiang et al., 2017; Jin et al., 2020; Cai et al., 2019; Du et al., 2021) further removes such assumptions by conducting provably efficient exploration in RL. Such a direction of research typically hinges on a low-rank MDP assumption. Thus, those works either assume that the MDP is linear in the known feature or propose computational-inefficient algorithms, limiting the ability to explore the environment with neural network parameterization. To explore the environment with neural network parameterization, a recent line of work (Wang et al., 2020; Jin et al., 2021a) analyzes the use of general function approximators in RL, covering neural network parameterization as a special case. Such analyses typically depend on the Eluder dimension (Russo & Van Roy, 2013), which unfortunately can be exponentially large even for a simple neural network class (Dong et al., 2021) and thus makes the results statistically inefficient for neural network parameterization. Therefore, we raise the following question:

*Can we design RL algorithms that can conduct provably efficient exploration in structured environments with neural network parameterization?*

Specifically, Our goal is to **develop computational-efficient algorithms whose sample efficiency does not depend on the Eluder dimension of neural networks** for structured environments with neural network parameterization. Our key insight is that, when the transition dynamics is captured by an energy-based model, we leverage the spectral decomposition of the kernel such that the challenge of distribution shift is characterized by the effective dimension of the kernel. To illustrate this insight, we propose a new model called *MDPs with neural dynamics*, which allows neural network parameterization and captures various MDP models proposed in previous works, including the KNR model (Kakade et al., 2020) and the linear MDP model (Jin et al., 2020). We then propose an algorithm, namely, *Exploration with Learnable Neural Features* (ELNF), and show that ELNF is sample efficient. ELNF iteratively fits the transition dynamics and reward functions with neural networks. Upon fitting the models, ELNF conducts exploration based on upper confidence bounds (UCB) (Abbasi-Yadkori et al. (2011)), which are obtained from the feature maps that correspond to the fitted model. We remark that the bonus in ELNF can be efficiently computed.

**Contributions.** Our contribution is threefold. First, we identify a class of models that incorporates NN feature representation, which captures nonlinearity in the transition dynamics beyond the KNR and linear MDP model. We also show that our proposed setting can generalize to models in previous works (Kakade et al. (2020), Ren et al. (2021)). Second, we propose a new algorithm, namely ELNF, which tackles our proposed *MDPs with neural dynamics*. **Our algorithm is computationally efficient when we have an optimization oracle for the model estimation**. Third, we analyze the sample complexity of ELNF and show that ELNF is sample efficient. A key feature of ELNF is that **the sample complexity of ELNF depends only on the covering number of neural network classes and does not depend on the corresponding Eluder dimension.** We highlight that **our work is the first to cover arbitrary NN classes with bounded log-covering numbers.** In contrast, **previous research typically depends on the Eluder dimension (Russo & Van Roy, 2013) of the hypothesis class, which is exponentially large for simple neural network classes (Dong et al., 2021).**

### 1.1 Related Work

Our work is closely related to the line of research on provably efficient exploration in the function approximation setting (Jiang et al., 2017; Jin et al., 2020; Cai et al., 2019; Du et al., 2021; Uehara et al., 2021; Zhang et al., 2022a). Such a line of research typically hinges on MDPs with a low-rank structure. For instance, the study of linear MDPs (Jin et al., 2020; Cai et al., 2019) requires that the transition dynamics are linear in the known feature map. In contrast, the feature maps are unknown in our setting and need to be estimated. The study of low-rank MDPs (Jiang et al., 2017; Du et al., 2021; Uehara et al., 2021; Ren et al., 2022) is more closely aligned to our work in the sense that the feature map is unknown and needs to be estimated. Jiang et al. (2017) and Du et al. (2021) require optimistic planning over the confidence set of transition dynamics, which is computationally inefficient. Uehara et al. (2021) and Ren et al. (2022) propose an algorithm for low-rank MDP that is both computationally efficient and sample efficient. Nevertheless, they only consider finite hypothesis classes, and require sampling from the stationary distribution of the MDP.

Our work is also related to the study of provably efficient exploration with general function approximation (Wang et al., 2020; Jin et al., 2021a). Nevertheless, previous results typically depend on the Eluder dimension (Russo & Van Roy, 2013) of the hypothesis class, which is exponentially large for simple neural network classes (Dong et al., 2021). Yang et al. (2020) achieves sample-efficient exploration based on the overparameterized neural networks (Simsek et al., 2021) as the function approximator. However, their analysis hinges on the neural tangent kernel (NTK) and can not handle NNs beyond NTK regime. In contrast, our analysis can adapt to generic neural network classes.

Our work is also related to the analysis of model-based RL (Osband & Van Roy, 2014; Ayoub et al., 2020; Kakade et al., 2020) and representation learning (Ren et al., 2021; Nachum & Yang, 2021; Zhang et al., 2022b). The definition of our *MDPs with neural dynamics* generalizes that in Kakade et al. (2020) and Ren et al. (2021). In contrast to the KNR model in Kakade et al. (2020), we can handle the infinite neural network hypothesis class and do not require the nonlinear feature map to be known. Ren et al. (2021) require sampling from the posterior distribution of the hypothesis class, which is computational-inefficient when the hypothesis class is large. In addition, the sample complexity bound of Ren et al. (2021) depends on the Eluder dimension of the feature map class, which is exponentially large for simple neural network classes (Dong et al., 2021). In contrast, our sample complexity bound depends only on the neural network classes only through its capacity.

Our work is motivated by the complexity analysis of neural network classes. Dong et al. (2021) show that the Eluder dimension of one-layer neural network classes is exponentially large, suggesting that the previous analysis of RL algorithms based on the Eluder dimension (Russo & Van Roy, 2013) may not be applicable to neural networks.

### 1.2 Notation

For a vector $v \in \mathbb{R}^d$, we define $\|v\|_2 = (\sum_{i=1}^d v_i^2)^{1/2}$, where $v_i$ is the $i$-th element of $v$. For a real-valued function $f : \mathcal{X} \to \mathbb{R}$, we define $\|f\|_\infty = \max_{x\in\mathcal{X}} |f(x)|$. For a vector-valued function $f : \mathcal{X} \to \mathbb{R}^d$, we define $\|f\|_{\infty,2} = \max_{x\in\mathcal{X}} \|f(x)\|_2$. For a sequence of real-valued functions $r = \{r_h\}_{h=1}^H \subset \mathcal{X} \to \mathbb{R}$, we define $\|r\|_\infty = \sup_{h\in[H],x\in\mathcal{X}} |r_h(x)|$. We denote by $\mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|)$ the $\epsilon$-covering number of the function class $\mathcal{F}$ with respect to the norm $\|\cdot\|$, define $H_\infty(\mathcal{F}, \epsilon) = \log \mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_\infty)$ for a real-valued function class $\mathcal{F}$, and define $H_2(\mathcal{F}, \epsilon) = \log \mathcal{N}(\mathcal{F}, \epsilon, \|\cdot\|_{\infty,2})$ for a vector-valued function class $\mathcal{F}$. We further define $[n] = \{1, \ldots, n\}$ when $n$ is an integer. For a set $\mathcal{C}$, we denote by $\Delta(\mathcal{C})$ the set of the distributions over $\mathcal{C}$, and $\mathcal{U}(\mathcal{C})$ the uniform distribution over $\mathcal{C}$. For $g : \mathcal{X} \to \mathbb{R}$ and $\mathcal{X}_n = \{x_1, \ldots, x_n\} \subset \mathcal{X}$, we define $g[\mathcal{X}_n] = (g(x_1), \ldots, g(x_n))^\top$.

## 2 Preliminary

We consider an episodic MDP $\mathcal{V}^* = (\mathcal{S}, \mathcal{A}, H, \mathcal{P}^*, r^*)$ with a state space $\mathcal{S} \in \mathbb{R}^d$, an action space $\mathcal{A}$, a horizon $H$, transition kernels $\mathcal{P}^* = \{\mathcal{P}_h^*\}_{h=1}^H$, and reward functions $r^* = \{r_h^*\}_{h=1}^H$. We assume that the reward functions are bounded and deterministic, that is, $\|r_h^*\|_\infty \in [0, 1]$ for all $h \in [H]$. We also assume that the action space is finite, that is, $|\mathcal{A}| < \infty$. The agent iteratively interacts with the environment as follows. At the beginning of each episode, the agent determines a policy $\pi = \{\pi_h\}_{h=1}^H$, where $\pi_h : \mathcal{S} \to \Delta(\mathcal{A})$ for any $h \in [H]$. Without loss of generality, we assume that the initial state is fixed to $s_{\text{init}} \in \mathcal{S}$ across all episodes. At the $h$-th step, the agent receives a state $s_h$ and takes an action $a_h$ following $a_h \sim \pi_h(\cdot \mid s_h)$. Subsequently, the agent receives a reward $r_h^*(s_h, a_h)$ and the next state following $s_{h+1} \sim \mathcal{P}_{h+1}^*(\cdot \mid s_h, a_h)$. The episode ends after the agent receives the last state $s_{H+1}$. For a given policy $\pi = \{\pi_h\}_{h=1}^H$, where $\pi_h : \mathcal{S} \to \Delta(\mathcal{A})$ for any $h \in [H]$, we define the value function $V_h^\pi$ and the $Q$-function $Q_h^\pi$ for any $h \in [H]$ as

$$
\begin{aligned}
V_h^\pi(s; r^*, \mathcal{P}^*) &= \mathbb{E}_{\pi,\mathcal{P}^*}\Big[\sum_{i=h}^H r_i^*(s_i, a_i) \,\Big|\, s_h = s\Big], \\
Q_h^\pi(s, a; r^*, \mathcal{P}^*) &= \mathbb{E}_{\pi,\mathcal{P}^*}\Big[\sum_{i=h}^H r_i^*(s_i, a_i) \,\Big|\, s_h = s, a_h = a\Big].
\end{aligned} \tag{2.1}
$$

Here the expectation $\mathbb{E}_{\pi,\mathcal{P}^*}[\cdot]$ in (2.1) is taken with respect to $s_{i+1} \sim \mathcal{P}_i^*(\cdot \mid s_i, a_i)$ and $a_i \sim \pi_i(\cdot \mid s_i)$ for $i \in \{h, h+1, \ldots, H\}$. For convenience, we define $V_{H+1}^\pi(s; r, \mathcal{P}) = 0$ for any state $s \in \mathcal{S}$, reward function $r$, transition kernel $\mathcal{P}$ and policy $\pi$. For simplicity, we define the expected total reward $J(\pi; r^*, \mathcal{P}^*)$ as $J(\pi; r^*, \mathcal{P}^*) = V_1^\pi(s_{\text{init}}; r^*, \mathcal{P}^*)$. We have the following Bellman equations for any $(s, a, h) \in \mathcal{S} \times \mathcal{A} \times [H]$ and any policy $\pi$,

$$
\begin{aligned}
V_h^\pi(s; r^*, \mathcal{P}^*) &= \big\langle Q_h^\pi(s, \cdot\,; r^*, \mathcal{P}^*), \pi_h(\cdot \mid s)\big\rangle_{\mathcal{A}}, \\
Q_h^\pi(s, a; r^*, \mathcal{P}^*, \pi) &= \mathbb{E}_{\mathcal{P}^*}\big[r_h^*(s_h, a_h) + V_{h+1}^\pi(s_{h+1}; r^*, \mathcal{P}^*) \,\big|\, s_h = s, a_h = a\big],
\end{aligned}
$$

where $\langle \cdot, \cdot \rangle_{\mathcal{A}}$ is the inner product over $\mathbb{R}^{\mathcal{A}}$, and $\mathbb{E}_{\mathcal{P}^*}[\cdot]$ is taken with respect to $s_{h+1} \sim \mathcal{P}_h^*(\cdot \mid s_h, a_h)$. The goal of RL is to find a policy $\pi^*$ that maximizes the expected total reward. Specifically, for the episodic MDP $\mathcal{V}^* = (\mathcal{S}, \mathcal{A}, H, \mathcal{P}^*, r^*)$, We define $\pi^* \in \operatorname{argmax}_\pi J(\pi; r^*, \mathcal{P}^*)$ as an optimal policy. Correspondingly, we define the optimal $Q$-function $Q_h^*$ and the optimal value function $V_h^*$ as $Q_h^*(s, a; r^*, \mathcal{P}^*) = Q_h^{\pi^*}(s, a; r^*, \mathcal{P}^*)$ and $V_h^*(s; r^*, \mathcal{P}^*) = V_h^{\pi^*}(s; r^*, \mathcal{P}^*)$ for any $(s, a) \in \mathcal{S} \times \mathcal{A}$.

## 3 Markov Decision Processes with Neural Dynamics

In this paper, our goal is to develop a provably efficient algorithm for RL problems adapted with large feature space, such as neural networks (NNs). To this end, we introduce the MDPs with neural dynamics, whose reward and transition dynamics are parameterized by NNs.

**Motivation.** Our definition is motivated by the kernelized nonlinear regulator (KNR). In a KNR model, the transition kernel takes the following form,

$$s_{h+1} = W_h^* \phi_h^*(s_h, a_h) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I_d), \tag{3.1}$$

where $\phi_h^*$ is a known nonlinear feature map. Former research proposes sample-efficient algorithms for such a model. Although such a KNR setting empowers sample efficient RL (Kakade et al. (2020)), it is relatively restrictive in the following two aspects. First, the feature map $\phi_h^*$ and the expected reward $r_h^*$ are known a priori. Second, the model only imposes nonlinearity on $(s_h, a_h)$ via the known feature map, while the conditional expectation of the next state given $s_h, a_h$ is a linear function of $\phi_h^*(s_h, a_h)$. In other words, when $\phi_h^*$ is known, the transition dynamics can be recovered via linear system identification methods such as ridge regression.

To generalize the KNR model, we interpret (3.1) as an energy-based model. More specifically, we can write the transition of the MDP in (3.1) as

$$\mathcal{P}_h^*(s_{h+1} \mid s_h, a_h) \propto \exp\big(-E(s_{h+1}, s_h, a_h)\big), \tag{3.2}$$

where the energy function $E(s_{h+1}, s_h, a_h)$ is defined as

$$E(s_{h+1}, s_h, a_h) = \|s_{h+1} - W_h^* \phi_h^*(s_h, a_h)\|_2^2 / 2. \tag{3.3}$$

Here (3.2) omits a normalization factor, which is a function of $(s_h, a_h)$. We generalize this model and impose nonlinearity on the next state $s_{h+1}$ by substituting a nonlinear feature map $\psi_{h+1}^*(s_{h+1})$ for $s_{h+1}$ in (3.3). Such a generalization allows us to incorporate the nonlinearity of the next state in the model. In addition, we assume that the nonlinear feature maps $\phi_h^*$ and $\psi_{h+1}^*$ are unknown and need to be estimated from pre-specified feature classes $\Phi$ and $\Psi$, which for example, can be two classes of NNs. We further assume that the expected reward $r^*$ is unknown and needs to be estimated from the reward function class $\mathcal{R}$. We formalize our generalization in the following definition.

**Definition 3.1** (MDPs with Neural Dynamics)**.** *An episodic MDP $(\mathcal{S}, \mathcal{A}, H, \mathcal{P}^*, r^*)$ is an MDP with neural dynamics if its reward function $r^* = \{r_h^*\}_{h=1}^H$ belongs to a reward function class $\mathcal{R}$, which is a known function class that consists of NNs, and the transition kernel of the MDP $\mathcal{P}^* = \{\mathcal{P}_h^*\}_{h=1}^H$ takes the following form,*

$$\mathcal{P}_h^*(s_{h+1} \mid s_h, a_h) \propto \exp\Big(-\big\|\psi_{h+1}^*(s_{h+1}) - \phi_h^*(s_h, a_h)\big\|_2^2 / 2\Big). \tag{3.4}$$

*Here $\phi_h^* \in \Phi : \mathbb{R}^d \times \mathcal{A} \to \mathbb{R}^m$ and $\psi_{h+1}^* \in \Psi : \mathbb{R}^d \to \mathbb{R}^m$ are two unknown feature maps, and $\Phi$, $\Psi$ are two known feature map classes that consist of NNs. We denote by $\mathcal{M}$ the set of all the transition kernels that take the form of (3.4), and let $\mathcal{X} \in \mathbb{R}^m$ denote the union of the image spaces of the feature maps, namely,*

$$\mathcal{X} = \{\varphi(s, a) \mid (\varphi, s, a) \in \Phi \times \mathcal{S} \times \mathcal{A}\} \cup \{\varphi(s) \mid (\varphi, s) \in \Psi \times \mathcal{S}\} \subseteq \mathbb{R}^m.$$

**Generality of Definition 3.1.** We remark that Definition 3.1 is a significant generalization of stochastic nonlinear systems beyond KNR. For instance, when $\psi_{h+1}^*(s_{h+1}) = s_{h+1}$, the transition kernel in Definition 3.1 takes the following form,

$$\mathcal{P}_h^*(s_{h+1} \mid s_h, a_h) \propto \exp\Big(-\big\|s_{h+1} - \phi_h^*(s_h, a_h)\big\|_2^2 / 2\Big),$$

which is the transition kernel in Ren et al. (2021). Therefore, we recover the model in Ren et al. (2021) when $\psi_{h+1}^*$ is known to be the identity map and the reward function is known. Moreover, the transition kernel defined in (3.4) also includes a class of nonlinear dynamics satisfying

$$s_{h+1} = (\psi_{h+1}^*)^{-1}\big(\phi_h^*(s_h, a_h) + \epsilon_h\big),$$

where $\mathcal{S} \subseteq \mathbb{R}^m$, $\psi_{h+1}^* \colon \mathbb{R}^m \to \mathbb{R}^m$, the determinant of the Jacobian matrix of $\psi_{h+1}^*$ is a constant, and $\epsilon_h$ is a Gaussian noise. Our model significantly generalizes such a model by allowing a possibly noninvertible feature map $\psi_{h+1}^*$.

**Relationship with Kernelized Linear MDP.** Recall that $K(x, y) = \exp(-\|x - y\|_2^2/2)$ is also known as the Gaussian RBF kernel, which induces a reproducing kernel Hilbert space (RKHS) defined on $\mathbb{R}^m$ (Rahimi et al., 2007). (See Appendix §D for a brief introduction of RKHS.) Intuitively, $K(x, y)$ measures the proximity between $x$ and $y$ in the kernel space. From this perspective, the transition

kernel in (3.4) specifies the next state $s_{h+1}$ by measuring the proximity of the representations $\phi_h^*(s_h, a_h)$ and $\psi_{h+1}^*(s_{h+1})$. Besides, since $K(x, y)$ can be written as $\langle k(x), k(y)\rangle_{\mathcal{H}}$, where the feature map of the RKHS $k$ is defined as $k(x) = K(x, \cdot)$, and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is the inner product of the RKHS respectively. Thus, (3.4) can be equivalently written as

$$\mathcal{P}_h^*(s_{h+1} \mid s_h, a_h) = \Big\langle Z_h^*(s_h, a_h) \cdot k\big(\phi_h^*(s_h, a_h)\big), k\big(\psi_{h+1}^*(s_{h+1})\big)\Big\rangle_{\mathcal{H}}, \tag{3.5}$$

where $Z_h^*(s_h, a_h)$ is the normalization factor in (3.4). Thus, when $Z_h^*$ is known, our model can be regarded as an RKHS extension of the linear MDP model (Jin et al., 2020). This is the case when $\psi^*$ is the identity maps but unknown to the learner, and $Z_h^*(s_h, a_h)$ becomes a constant (Ren et al., 2021). **See Appendix §D for more details of the relationship between the model in (3.4) and RKHS.**

**Role of NNs in Our Model.** We would like to remark that the model specified in Definition 3.1 is not restricted to NNs. In fact, the definition only requires proper function classes of the reward function, representations of $(s_h, a_h)$ and $s_{h+1}$, namely, $\mathcal{R}$, $\Phi$, and $\Psi$. Thus, our model can also be defined for other function approximators such as polynomial spline (Unser et al., 1993), classification and regression tree (Syrgkanis & Zampetakis, 2020). Meanwhile, as we will see in the sequel, both our algorithm and the theoretical results do not hinge on NNs and can employ general function classes with bounded capacity. Here we call our model neural dynamics in order to **highlight that our work is the first one that covers arbitrary NN classes with bounded log-covering numbers**.

## 4 Algorithm

In this section, we introduce an algorithm for solving MDPs with neural dynamics in the online setting. We first introduce the motivation of the algorithm, and then introduce the procedure in detail.

**Motivation.** To strike a balance between exploration and exploitation, our algorithm follows the principle of *Optimism in the Face of Uncertainty* (Lattimore & Szepesvári, 2020). When we know the true feature maps $\{\phi_h^*\}_{h=1}^H$, we can apply kernel LSVI (Yang et al., 2020) to construct the exploration bonus since the energy-base transition admits a kernel structure. (See §3 for the details.) However, in MDPs with neural dynamics, we do not know $\{\phi_h^*\}_{h=1}^H$. A straightforward solution for handling the unknown feature maps is to learn the feature maps from the data we collect and construct the bonus based on the learned features. However, the bonus constructed by the learned features might be invalid since the learned features have errors. We handle the error in the learned feature by purposefully taking uniform actions when exploring the environment. Such a sampling scheme gives us more diverse data for model estimation. Based on this motivation, we design an iterative algorithm that outputs a policy after $N$ iterations. In particular, in each iteration $n \in [N]$, our algorithm performs the following four steps: (i) sampling new data from the environment, (ii) estimating the model via maximum likelihood estimation, (iii) constructing exploration incentives using the features of the learned model, and (iv) updating the online policy for exploration via planning on the learned model.

**Sampling Scheme.** As we mentioned in §3, the transition of MDPs with neural dynamics can be written as an energy-based model and admits the Gaussian RBF kernel. To exploit the kernel structure in the transition, we explore the environment using the exploration bonus induced by the Gaussian RBF kernel and the feature maps learned from the data, which is motivated by Yang et al. (2020). However, since the bonus is not induced by the true underlying feature, it might fail to indicate the most uncertain state-action pairs for exploration. To mitigate such an issue, we combine the uniform policy, which samples action from the uniform distribution over the action space, with the optimistic policy during the sampling procedure. Intuitively, such a sampling scheme provides a wider coverage over the state-action space and better explores the environment.

To simplify the presentation in the main text, we present the sketch of the sampling scheme as follows. The rigorous presentation of the sampling scheme for the boundary case is deferred to Appendix §B. In the $n$-th iteration of our algorithm, given the previously collected dataset $D_{h,i}^{n-1}$ for $i \in \{0, 1, 2\}$ and $h \in [H]$, we interact with the MDP following the policy $\pi^n = \{\pi_h^n\}_{h=1}^H$ and obtain the new dataset $D_{h,i}^n$ for $i \in \{0, 1, 2\}$ and $h \in [H]$. Specifically, for any $h \in \{-1, \ldots, H\}$, we start from the initial state $s_1$ and choose the action $a_{\bar{h}} \sim \pi_{\bar{h}}^n(\cdot \mid s_{\bar{h}})$ in the $\bar{h}$-th step when $\bar{h} \in \{1, \cdots, h\}$, and choose the action $a_{\bar{h}} \sim \mathcal{U}(\mathcal{A})$ when $\bar{h} \in \{h+1, h+2\}$. Here $\mathcal{U}(\mathcal{A})$ is the uniform distribution over the action space $\mathcal{A}$. By following such a procedure, we obtain the following trajectory,

$$s_1, a_1, r_1, \ldots, s_{h+2}, a_{h+2}, r_{h+2}, s_{h+3}, \tag{4.1}$$

where $s_1 = s_{\text{init}}$. Then, we label the obtained trajectory as follows,

$$s^n_{h+i,i} = s_{h+i}, \qquad a^n_{h+i,i} = a_{h+i}, \qquad r^n_{h+i,i} = r_{h+i}, \qquad \bar{s}^n_{h+i,i} = s_{h+i+1}, \tag{4.2}$$

for any $i \in \{0, 1, 2\}$. We then update the dataset $\mathcal{D}^n_{h,i}$ by $\mathcal{D}^n_{h,i} = \mathcal{D}^{n-1}_{h,i} \cup \{(s^n_{h,i}, a^n_{h,i}, r^n_{h,i}, \bar{s}^n_{h+i,i})\}$ for any $i \in \{0, 1, 2\}$ and $h \in [H]$. We use $i$ in (4.2) to indicate how many steps of the uniform policy we need to execute to obtain such a dataset. Intuitively, the dataset with a bigger $i$ has a better coverage over the state-action space $\mathcal{S} \times \mathcal{A}$. We summarize the sampling scheme in Algorithm 1. See Figure 1 in Appendix §B for an illustration of the sampling scheme, and see Algorithm 3 in Appendix §B for the formal presentation of Algorithm 1.

**Algorithm 1** Sampling Scheme (Informal)

1: **Input:** Policy $\pi^n = \{\pi^n_h\}_{h=1}^H$, datasets $\mathcal{D}^{n-1}_{h,i}$ for $i \in \{0, 1, 2\}$ and $h \in [H]$.
2: **for** $h = -1, \ldots, H$ **do**
3: Interact with the environment to obtain the trajectory in (4.1) by first executing $\pi^n$ from $s_1$ to $s_h$, and then execute $\mathcal{U}(\mathcal{A})$ for two more steps. Label the obtained trajectory as (4.2).
4: **end for**
5: Set $\mathcal{D}^n_{h,i} \leftarrow \{(s^\tau_{h,i}, a^\tau_{h,i}, r^\tau_{h,i}, \bar{s}^\tau_{h+i,i})\}_{\tau=1}^n$ for $h \in [H]$ and $i \in \{0, 1, 2\}$, where the tuple $(s^\tau_{h,i}, a^\tau_{h,i}, r^\tau_{h,i}, \bar{s}^\tau_{h+i,i})$ is the data we label in (4.2). ▷ Updating the datasets
6: **Return:** Datasets $\{\mathcal{D}^n_{h,i}\}_{h=1,i=0}^{h=H,i=2}$.

**Model Estimation.** To estimate the model, we solve the following optimization problems,

$$\widehat{r}^n_h = \operatorname*{argmin}_{r \in \mathcal{R}} \sum_{i=1}^{2} \sum_{(s_h, a_h, r_h, s_{h+1}) \in \mathcal{D}^n_{h,i}} \big[r_h - r(s_h, a_h)\big]^2, \tag{4.3}$$

$$\widehat{\mathcal{P}}^n_h = \operatorname*{argmin}_{\mathcal{P} \in \mathcal{M}} - \sum_{i=1}^{2} \sum_{(s_h, a_h, r_h, s_{h+1}) \in \mathcal{D}^n_{h,i}} \log \mathcal{P}(s_{h+1} \mid s_h, a_h). \tag{4.4}$$

Here $\mathcal{R}$ and $\mathcal{M}$ are the reward function class and the transition kernel class defined in Definition 3.1. We denote by $\phi^n_h$ and $\psi^n_{h+1}$ the feature maps that correspond to the transition kernel $\widehat{\mathcal{P}}^n_h$ estimated in (4.4). To simplify our analysis, we assume that there exists an oracle that returns the global minimum of the optimization problems (4.3) and (4.4). Similar assumption also arises in the previous study of RL (Fan et al., 2020; Kakade et al., 2020; Uehara et al., 2021; Jin et al., 2021a). When the normalization factor in (3.5) is a constant, (4.4) can be easily implemented since it is equivalent with

$$\widehat{\mathcal{P}}^n_h = \operatorname*{argmin}_{\phi \in \Phi, \psi \in \Psi} \sum_{i=1}^{2} \sum_{(s_h, a_h, r_h, s_{h+1}) \in \mathcal{D}^n_{h,i}} \|\psi(s_{h+1}) - \phi(s_h, a_h)\|_2^2 .$$

**Remark 4.1** (Transition Estimation)**.** *We would like to remark that the method for estimating the transition is not restricted to maximum likelihood estimation (MLE). Methods including variational autoencoder (Kingma & Welling, 2013), score matching (Hyvärinen & Dayan, 2005) can also be used for transition estimation. Our sample complexity bound holds for any transition estimator whose total variance error has an upper bound.*

**Exploration Bonus.** The transition kernel in Definition 3.1 is closely related to the radial basis function (RBF) kernel. In the sequel, we define the bonuses for exploration and update the policy based on such bonuses. Specifically, for a fixed feature map $\phi : \mathcal{S} \to \mathbb{R}$, we define the Gram matrix $K^n_h[\phi]$ and the function $k^n_h[\phi] : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^n$ as follows,

$$K^n_h[\phi] = \Big[K\big(\phi(s^\tau_{h,1}, a^\tau_{h,1}), \phi(s^{\tau'}_{h,1}, a^{\tau'}_{h,1})\big)\Big]_{\tau,\tau'=1}^n \in \mathbb{R}^{n \times n},$$

$$k^n_h[\phi](s, a) = \Big[K\big(\phi(s, a), \phi(s^1_{h,1}, a^1_{h,1})\big), \ldots, K\big(\phi(s, a), \phi(s^n_{h,1}, a^n_{h,1})\big)\Big]^\top \in \mathbb{R}^n, \ \forall (s, a) \in \mathcal{S} \times \mathcal{A},$$

where $\{(s^\tau_{h,1}, a^\tau_{h,1}, r^\tau_{h,1}, \bar{s}^\tau_{h,1})\}_{\tau=1}^n \in \mathcal{D}^n_{h,1}$. We then define the bonus $u^n_h$ as follows,

$$\begin{aligned} &u^n_h(s, a) = \min\big\{2H + 2, \beta \widetilde{u}^n_h(s, a)/\lambda\big\}, \\ &\text{where} \quad \widetilde{u}^n_h(s, a) = 1 - k^n_h[\phi^n_h](s, a)^\top \big(\lambda I + K^n_h[\phi^n_h]\big)^{-1} k^n_h[\phi^n_h](s, a). \end{aligned} \tag{4.5}$$

Here $\beta > 0$ and $\lambda > 0$ are the tuning parameters. We remark that the form of the bonus in (4.5) aligns with the bonus in other previous works that use kernel functions for function approximation (Srinivas et al., 2009; Yang et al., 2020).

**Remark 4.2** (Dependency of Rewards on Features.)**.** *Here we do not require that the reward depends on the feature in the transition kernel, which is different from the literature in linear MDPs (Cai et al., 2019; Jin et al., 2020). Common sense seems to dictate that we can not characterize the uncertainty using the feature without such a dependency. However, the estimation error of the reward estimators $\widehat{r}_h^n$ in the empirical measure can be bounded from the above by the property of the least square estimator. Therefore, the uncertainty of $r_h^*$ in the distribution induced by any new policy, such as $\pi^*$, can be bounded from the above by the distribution shift, which is characterized by $u_{h-1}^n$. Such an observation allows us to characterize the uncertainty of the reward estimator even when the reward does not depend on the feature in the transition kernel.*

**Policy Update.** We update the policy $\pi^{n+1}$ by setting it as the optimal policy of the learned model, which can be efficiently computed by dynamic programming. Due to the space limit, we defer the details of the planning algorithm to Appendix §B. We remark that we can also apply other model-based algorithms, including *Dyna* (Sutton, 1991) and *Gradient-Aware Model-based Policy Search* (D'Oro et al., 2020), to compute the optimal policy of the learned model, and the suboptimality of the output of Algorithm 2 is bounded when the error of the planning oracle is bounded.

**Remark 4.3** (Computational Efficiency)**.** *Our algorithm is oracle efficient in the sense that our algorithm is computationally efficient given an optimization oracle for model estimation, which also appears in the previous study (Fan et al., 2020; Kakade et al., 2020; Uehara et al., 2021). More specifically, the bonus and the policy in each iteration can be efficiently computed by* (4.5) *and Algorithm 4 in the appendix. The existing literature on general function approximation requires either global optimism over the confidence set (Kakade et al., 2020; Jin et al., 2021a) or posterior sampling over the hypothesis set (Ren et al., 2021), which can not be computed efficiently.*

**Algorithm 2** Exploration with Learnable Neural Features (ELNF)

1: **Input:** Failure probability $\delta > 0$, tuning parameters $\beta, \lambda > 0$.
2: Initialize: Set $\pi^1 = \{\pi_h^1\}_{h=-1}^H$, where $\pi_h^1(\cdot \mid s) = \mathcal{U}(\mathcal{A})$ for any $(s,h) \in \mathcal{S} \times [H]$, and set $\mathcal{D}_{h,i}^0 = \emptyset$ for all $(h,i) \in [H] \times \{0,1,2\}$.
3: **for** $n = 1, \ldots, N$ **do**
4: Set $\{\mathcal{D}_{h,i}^n\}_{h=1,i=0}^{h=H,i=2}$ by applying Algorithm 1 (Sampling Scheme) with the policy $\pi^n$ and the datasets $\{\mathcal{D}_{h,i}^{n-1}\}_{h=1,i=0}^{h=H,i=2}$ as the input. ▷ Sampling
5: Set $\{\widehat{r}_h^n\}_{h=1}^H$ and $\{\widehat{\mathcal{P}}_h^n\}_{h=1}^H$ as in (4.3) and (4.4), respectively. ▷ Model estimation
6: Set $\{\phi_h^n\}_{h=1}^H$ and $\{\psi_{h+1}^n\}_{h=1}^H$ as the feature maps corresponding to $\{\widehat{\mathcal{P}}_h^n\}_{h=1}^H$.
7: Set $\{u_h^n\}_{h=1}^H$ as in (4.5). ▷ Feature estimation and bonus construction
8: Set $\pi^{n+1}$ by applying Algorithm 4 (Planning Algorithm) in the appendix with the learned model $\{\widehat{r}_h^n\}_{h=1}^H$, $\{\mathcal{P}_h^n\}_{h=1}^H$, and the bonuses $\{\widehat{u}_h^n\}_{h=1}^H$ as the input. ▷ Planning
9: **end for**
10: **Return:** $\widehat{\pi} = \mathcal{U}\big(\{\pi^n\}_{n=2}^{N+1}\big)$.

## 5 Theory

In this section, we present the analysis of ELNF. We first present the boundedness assumption on the model.

**Assumption 5.1** (Boundedness of Model)**.** *We assume that the state space $\mathcal{S}$ is a bounded set of $\mathbb{R}^d$, and the Lebesgue measure of $\mathcal{S}$ is an absolute constant. We also assume that $\max\{\|\phi(s,a)\|_2, \|\psi(s)\|_2\} \leq R$ for all $(s,a,\phi,\psi) \in \mathcal{S} \times \mathcal{A} \times \Phi \times \Psi$. We further assume that $0 \leq r(s,a) \leq 1$ for any $(s,a,r) \in \mathcal{S} \times \mathcal{A} \times \mathcal{R}$.*

Since $\mathcal{S}$ is bounded, Assumption 5.1 is a reasonable regularity condition on the model. Such a regularity assumption is standard and is also assumed in the previous works (Cai et al., 2019; Jin et al., 2020; 2021b). Next, we introduce the following assumption, which characterizes the complexity of the NN classes.

**Assumption 5.2** (Decay Rate of Covering Number). *There exists constants $C_{\text{net}} > 0$ and $\gamma \geq 0$ that only depend on $(\mathcal{R}, \Phi, \Psi)$, such that*

$$H_c(\epsilon) \triangleq \max\big\{H_\infty(\mathcal{R}, \epsilon), H_2(\Phi, \epsilon), H_2(\Psi, \epsilon)\big\} \leq C_{\text{net}} \cdot \big(1 + \log(1/\epsilon)\big)/\epsilon^\gamma.$$

In Assumption 5.2, $\gamma$ characterizes the complexity of the NN class by quantifying the growth rate of the covering number when the covering radius $\epsilon$ decays. We remark that previous research bounds the covering number of NN classes from the above at the same scale as Assumption 5.2. For example, Schmidt-Hieber (2020) and Chen et al. (2019) show that NN classes with specific structures satisfy Assumption 5.2 with $\gamma = 0$. See Lemmas C.3 and C.6 in Appendix §C for the details.

**Theorem 5.3** (Sample Complexity of ELNF). *We assume that Assumption 5.2 holds with $\gamma < 1/2$, and we can obtain the exact solution to the optimization problems (4.3) and (4.4). We set the tuning parameters $\lambda$ and $\beta$ as*

$$\lambda = C' N^{\gamma/(1+\gamma)} m \log(48HRN/\delta),$$
$$\beta = C'' H |\mathcal{A}|^{1/2} m^{1/2} N^{3\gamma/(4+4\gamma)} \sqrt{\log(48HRN/\delta)}$$

*in ELNF, where $m$ is the dimension of the image of the feature maps, $C'$, $C''$ are constants that only depend on the regularity parameters in Assumption 5.1. Under Assumption 5.1 and 5.2, for the policy $\widehat{\pi}$ returned by ELNF, it holds with probability at least $1 - \delta$ that*

$$J(\pi^*; r^*, \mathcal{P}^*) - J(\widehat{\pi}; r^*, \mathcal{P}^*) \leq C H^5 \cdot |\mathcal{A}|^2 \cdot \xi \cdot N^{(2\gamma-1)/(2+2\gamma)} (\log N)^{m+1}.$$

*Here $C$ is a constant that only depends on the dimension $m$, the bound of the feature maps $R$, and $C_{\text{net}}$ in Assumption 5.2, and $\xi = (\log(48HRN/\delta))^{5/2}$ is a logarithmic factor.*

*Proof.* See Appendix §E for a detailed proof. □

In Theorem 5.3, $\lambda$ is the regularization parameter that trades off between bias and variance, and $\beta$ is the uncertainty coefficient, which scales with $\gamma$ and $N$. We remark that our analysis is not restricted to NN classes, and can be extended to other bounded function classes with bounded covering numbers. We further remark that $m$ in Theorem 5.3 is the dimension of the image of the feature maps, which can be much smaller than the dimension of the state.

Moreover, in Appendix §E, we show that the suboptimality bound in Theorem 5.3 reduces to $\widetilde{\mathcal{O}}(d_{\text{eff}} N^{(2\gamma-1)/(2+2\gamma)})$ in terms of $N$, where $d_{\text{eff}}$ is the effective dimension in Definition E.4 in the appendix. Such a bound connects the sample efficiency of ELNF to the effective dimension of the RKHS and the covering number of NN classes. We further remark that when $\gamma = 0$ in Theorem 5.3, the suboptimality bound is sublinear in $N$, which aligns with the previous theoretical research. A detailed comparison with the related work can be found in Appendix §A.

## 6 Proof Sketch of Theorem 5.3

In this section, we sketch the proof of Theorem 5.3 and highlight the technique that allows us to remove the dependency on the Eluder dimension of NN classes. Since the policy $\widehat{\pi}$ returned by Algorithm 2 (ELNF) is the mixture of $\pi^1, \ldots, \pi^N$, we can decompose the suboptimality as

$$\begin{aligned} J(\pi^*; r^*, \mathcal{P}^*) - J(\widehat{\pi}; r^*, \mathcal{P}^*) = & \frac{1}{N} \underbrace{\sum_{n=1}^N \big[J(\pi^*; r^*, \mathcal{P}^*) - J(\pi^*; r^n + u^n, \mathcal{P}^n)\big]}_{\text{Term (a)}} \\ & + \frac{1}{N} \underbrace{\sum_{n=1}^N \big[J(\pi^*; r^n + u^n, \mathcal{P}^n) - J(\pi^{n+1}; r^n + u^n, \mathcal{P}^n)\big]}_{\text{Term (b)}} \\ & + \frac{1}{N} \underbrace{\sum_{n=1}^N \big[J(\pi^{n+1}; r^n + u^n, \mathcal{P}^n) - J(\pi^{n+1}; r^*, \mathcal{P}^*)\big]}_{\text{Term (c)}}. \end{aligned}$$

Here Term (a) is the out-of-sample estimation error of the estimated value $J$, Term (b) is the error of the planning algorithm, and Term (c) is the in-sample estimation error of the estimated value $J$. Since the bonus $u^n$ captures the uncertainty of the estimated model with high probability, and the planning algorithm finds the optimal policy of the learned model, Term (a) and Term (b) are small with high probability, which is shown in the following lemma.

**Lemma 6.1** (Informal)**.** *We define* $\zeta = \sqrt{C_2 \log(20HRN/\delta) N^{\gamma/(1+\gamma)}}$*, where* $C_2$ *is a constant that depends on the regularity parameters and the parameters of the NN class. Following the same condition of Theorem 5.3, we have* $\mathrm{Term(a)} \leq 8H|\mathcal{A}|\zeta\sqrt{N}$ *holds with probability at least* $1-\delta$*. We also have* $\mathrm{Term(b)} \leq 0$ *when we use Algorithm 4 for planning.*

*Proof.* This can be directly proved by Lemma E.5 and Lemma E.6. See Appendix §E for the details. □

The remaining analysis is to connect Term (c) with the complexity measure of the model class. Former research (Wang et al., 2020; Jin et al., 2021a) that allows neural network parameterization quantifies the uncertainty by the level set, and the Eluder dimension naturally appears when we telescope the in-sample error. To remove the dependency on the Eluder dimension, we want to quantify the uncertainty by the bonus defined by the true feature. However, since the true feature is unknown, we can only quantify the uncertainty using the learned feature. Inspired by Uehara et al. (2021), we obtain the following lemma, which connects the bonus defined by the learned feature with the bonus defined by the true unseen feature.

**Lemma 6.2** (Bonus Equivalence for the True Model, Informal)**.** *Following the same condition with Theorem 5.3, we have*

$$J(\pi; u^n, \mathcal{P}^*) \leq 2\,|\mathcal{A}|\,\beta d_{\text{eff}}^{1/2}/\sqrt{n} + J(\pi; u^{*,n}, \mathcal{P}^*)$$

*holds for any policy* $\pi$ *and* $n \geq 2$ *with high probability, where the bonus defined by the true feature* $u^{*,n} = \{u_h^{*,n}\}_{h=1}^H$ *is defined in Lemma I.1.*

*Proof.* This is Lemma I.2. See Appendix §I.5 for the details. □

We remark that we cannot obtain the above lemma by directly applying the technique in Uehara et al. (2021) since they only consider the finite-dimensional inner product. Using the lemma above, we have $\mathrm{Term(c)} \leq 32H^2\,|\mathcal{A}|\,\zeta\sqrt{d_{\text{eff}}N} + 2\sum_{n=1}^N J(\pi^{n+1}; u^{*,n}, \mathcal{P}^*)$ holds with high probability. Finally, we show that the sum of the bonuses defined by the true feature can be bounded by the effective dimension of the RKHS induced by the noise instead of the Eluder dimension of NN classes. We conclude the proof of Theorem 5.3 by combining the upper bounds of Term (a), Term (b), and Term (c).

**Removing Dependency on Eluder Dimension.** The existing literature on RL using general function approximators relies on the Eluder dimension when bounding the regret or the suboptimality (Wang et al., 2020; Jin et al., 2021a; Ren et al., 2021), which is exponentially large for a simple neural network class (Dong et al., 2021). However, we can remove such dependency in MDPs with neural dynamics. Our key insight is that we can regard MDPs with neural dynamics as kernel MDPs (Yang et al., 2020) whose feature is the composite map of the neural network and the feature map of the RKHS since the energy-base transition admits a kernel structure, which is shown in §3. Therefore, we can characterize the effect of the distribution shifts by the bonus defined by the true feature, whose sum is bounded by the effective dimension of the RKHS instead of the Eluder dimension of the NN class, without knowing the true feature.

**Role of Uniform Policy in Proof.** The uniform policy in the sampling scheme enables us to bound the influence of the distribution shift and show that the bonuses defined by the learned feature are valid uncertainty quantification. In order to show that the estimated value defined by the learned bonus is almost optimistic (Lemma E.5) for any policy, we need to bound the influence of the distribution shift. The data that we obtain from the uniform sampling has better coverage over $\mathcal{S} \times \mathcal{A}$, and the bonuses we construct with these data quantify the uncertainty with the presence of the distribution shift. Therefore, we take uniform actions when exploring the environment to obtain a valid uncertainty quantification without knowing the true feature, and $|\mathcal{A}|^2$ in the suboptimality bound is the price paid for the uniform sampling.

## REFERENCES


Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pp. 2312–2320, 2011.

Yasin Abbasi-Yadkori, Peter Bartlett, Kush Bhatia, Nevena Lazic, Csaba Szepesvari, and Gellért Weisz. Politex: Regret bounds for policy iteration using expert prediction. In *International Conference on Machine Learning*, pp. 3692–3702. PMLR, 2019.

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang. Model-based reinforcement learning with value-targeted regression. In *International Conference on Machine Learning*, pp. 463–474. PMLR, 2020.

Mikhail Belkin. Approximation beats concentration? an approximation view on inference with smooth radial kernels. In *Conference On Learning Theory*, pp. 1348–1361. PMLR, 2018.

Qi Cai, Zhuoran Yang, Chi Jin, and Zhaoran Wang. Provably efficient exploration in policy optimization. *arXiv preprint arXiv:1912.05830*, 2019.

Minshuo Chen, Xingguo Li, and Tuo Zhao. On generalization bounds of a family of recurrent neural networks. *arXiv preprint arXiv:1910.12947*, 2019.

Kefan Dong, Jiaqi Yang, and Tengyu Ma. Provable model-based nonlinear bandit and reinforcement learning: Shelve optimism, embrace virtual curvature. *arXiv preprint arXiv:2102.04168*, 2021.

Pierluca D'Oro, Alberto Maria Metelli, Andrea Tirinzoni, Matteo Papini, and Marcello Restelli. Gradient-aware model-based policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3801–3808, 2020.

Simon S Du, Sham M Kakade, Jason D Lee, Shachar Lovett, Gaurav Mahajan, Wen Sun, and Ruosong Wang. Bilinear classes: A structural framework for provable generalization in rl. *arXiv preprint arXiv:2103.10897*, 2021.

Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. A theoretical analysis of deep q-learning. In *Learning for Dynamics and Control*, pp. 486–489. PMLR, 2020.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6, 2005.

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low bellman rank are pac-learnable. In *International Conference on Machine Learning*, pp. 1704–1713. PMLR, 2017.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pp. 2137–2143. PMLR, 2020.

Chi Jin, Qinghua Liu, and Sobhan Miryoosefi. Bellman eluder dimension: New rich classes of rl problems, and sample-efficient algorithms. *arXiv preprint arXiv:2102.00815*, 2021a.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pp. 5084–5096. PMLR, 2021b.

Sham Kakade, Akshay Krishnamurthy, Kendall Lowrey, Motoya Ohnishi, and Wen Sun. Information theoretic regret bounds for online nonlinear control. *arXiv preprint arXiv:2006.12466*, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec):1107–1149, 2003.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Ofir Nachum and Mengjiao Yang. Provable representation learning for imitation with contrastive fourier features. *Advances in Neural Information Processing Systems*, 34:30100–30112, 2021.

Ian Osband and Benjamin Van Roy. Model-based reinforcement learning and the eluder dimension. *Advances in Neural Information Processing Systems*, 27, 2014.

Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, volume 3, pp. 5. Citeseer, 2007.

Tongzheng Ren, Tianjun Zhang, Csaba Szepesvári, and Bo Dai. A free lunch from the noise: Provable and practical exploration for representation learning. *arXiv preprint arXiv:2111.11485*, 2021.

Tongzheng Ren, Tianjun Zhang, Lisa Lee, Joseph E Gonzalez, Dale Schuurmans, and Bo Dai. Spectral decomposition representation for reinforcement learning. *arXiv preprint arXiv:2208.09515*, 2022.

Daniel Russo and Benjamin Van Roy. Eluder dimension and the sample complexity of optimistic exploration. In *NIPS*, pp. 2256–2264. Citeseer, 2013.

Johannes Schmidt-Hieber. Nonparametric regression using deep neural networks with relu activation function. *The Annals of Statistics*, 48(4):1875–1897, 2020.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Berfin Simsek, François Ged, Arthur Jacot, Francesco Spadaro, Clément Hongler, Wulfram Gerstner, and Johanni Brea. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In *International Conference on Machine Learning*, pp. 9722–9732. PMLR, 2021.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

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

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Vasilis Syrgkanis and Manolis Zampetakis. Estimation and inference with trees and forests in high dimensions. In *Conference on learning theory*, pp. 3453–3454. PMLR, 2020.

Masatoshi Uehara, Xuezhou Zhang, and Wen Sun. Representation learning for online and offline rl in low-rank mdps. *arXiv preprint arXiv:2110.04652*, 2021.

Michael Unser, Akram Aldroubi, and Murray Eden. A family of polynomial spline wavelet transforms. *Signal processing*, 30(2):141–162, 1993.

Sara A Van de Geer. *Applications of empirical process theory*, volume 91. Cambridge University Press Cambridge, 2000.

Lingxiao Wang, Qi Cai, Zhuoran Yang, and Zhaoran Wang. Neural policy gradient methods: Global optimality and rates of convergence. *arXiv preprint arXiv:1909.01150*, 2019.

Ruosong Wang, Ruslan Salakhutdinov, and Lin F Yang. Provably efficient reinforcement learning with general value function approximation. *arXiv preprint arXiv:2005.10804*, 2020.

Holger Wendland. *Scattered data approximation*, volume 17. Cambridge university press, 2004.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Zhuoran Yang, Chi Jin, Zhaoran Wang, Mengdi Wang, and Michael I Jordan. On function approximation in reinforcement learning: Optimism in the face of large state spaces. *arXiv preprint arXiv:2011.04622*, 2020.

Tianjun Zhang, Tongzheng Ren, Mengjiao Yang, Joseph Gonzalez, Dale Schuurmans, and Bo Dai. Making linear mdps practical via contrastive representation learning. In *International Conference on Machine Learning*, pp. 26447–26466. PMLR, 2022a.

Xuezhou Zhang, Yuda Song, Masatoshi Uehara, Mengdi Wang, Alekh Agarwal, and Wen Sun. Efficient reinforcement learning in block mdps: A model-free representation learning approach. In *International Conference on Machine Learning*, pp. 26517–26547. PMLR, 2022b.