Going Beyond Linear RL: Sample Efficient Neural Function Approximation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep Reinforcement Learning (RL) powered by neural net approximation of the Q function has had enormous empirical success. While the theory of RL has traditionally focused on linear function approximation (or eluder dimension) approaches, little is known about nonlinear RL with neural net approximations of the Q functions. This is the focus of this work, where we study function approximation with two-layer neural networks (considering both ReLU and polynomial activation functions). Our first result is a computationally and statistically efficient algorithm in the generative model setting under completeness for two-layer neural networks. Our second result considers this setting but under only realizability of the neural net function class. Here, assuming deterministic dynamics, the sample complexity scales linearly in the algebraic dimension. In all cases, our results significantly improve upon what can be attained with linear (or eluder dimension) methods.

1 Introduction

In reinforcement learning (RL), an agent aims to learn the optimal decision-making rule by interacting with unknown environment [Sutton and Barto, 2018]. Deep Reinforcement Learning, empowered by deep neural networks [LeCun et al., 2015, Goodfellow et al., 2016], has achieved tremendous success in various real-world applications, such as Go [Silver et al., 2016], Atari [Mnih et al., 2013a], Dota2 [Berner et al., 2019], Texas Holdém poker [Moravčík et al., 2017], and autonomous driving [Shalev-Shwartz et al., 2016]. Those modern RL applications are characterized by large state-action spaces, and the empirical success of deep RL corroborates deep neural networks can extrapolate well across state-action spaces [Henderson et al., 2018, Mnih et al., 2013b, Lillicrap et al., 2015].

Although in practice non-linear function approximation scheme is prevalent, theoretical understandings of the sample complexity of RL mainly focus on tabular or linear function approximation settings [Strehl et al., 2006, Jaksch et al., 2010, Azar et al., 2017, Jin et al., 2018, Russo, 2019, Zanette and Brunskill, 2019, Abbasi-Yadkori et al., 2019, Jin et al., 2020a,b, Wang et al., 2021a]. These results rely on finite state space or exact linear approximations. Recently, sample efficient algorithms under non-linear function approximation settings are proposed [Wen and Van Roy, 2017, Dann et al., 2018, Du et al., 2019b, Dong et al., 2020, Liu et al., 2019, Wang et al., 2020a, Dong et al., 2021]. Those algorithms are based on Bellman rank [Jiang et al., 2017], eluder dimension [Russo and Van Roy, 2013b], neural tangent kernel [Jacot et al., 2018, Allen-Zhu et al., 2019, Du et al., 2019a], or sequential Rademacher complexity [Rakhlin et al., 2015a,b]. Besides, general neural network function approximation is shown to be hard by Dong et al. [2021]. Yet, there is a mismatch

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

between the empirical success of deep RL and the theoretical understanding of RL under general deep neural network function approximations, which yields the following important question:

What are the structural properties that allow sample-efficient algorithms for RL with neural network function approximation?

In this paper, we advance our understanding to the above question by considering several settings where structural reconstruction can be achieved. Specifically, we study two structures, namely two-layer neural networks and structured polynomials, under two RL settings, namely RL with simulator model and online RL. In the simulator setting [Kakade, 2003, Sidford et al., 2018], the agent can simulate the MDP at any state-action pair. In online RL, the agent can only start at an initial state and interact with the MDP step-by-step. Our goal is to find near-optimal policy while minimizing the number of samples used.

We obtain the following results. For the simulator setting, we propose sample-efficient algorithms for RL with two-layer neural network function approximation, under the Bellman completeness assumption. For online RL, we provide sample-efficient algorithms for RL with structured polynomial function approximation. We also present sample-efficient algorithms under realizability assumption [Du et al., 2020b, Wang et al., 2021b], but with deterministic transition. Our main techniques are neural network recovery [Zhong et al., 2017], and algebraic geometry [Shafarevich, 2013].

1.1 Summary of our results

Our main results in different settings are summarized in Table 1. We consider two-layer neural networks $f(x) = \langle v, \sigma(Wx)\rangle$ and rank k polynomials (see Example 5.3). We make the following

Table 1: Baselines and our main results for the sample complexity to find a ϵ-optimal policy.

	rank k polynomial		Neural Net of Width k	
	Sim. + Det. (R)	Online + Det. (R)	Sim. + Det. (R)	Sim. + Stoch. (C)
Baseline	$O(d^p)$	$O(d^p)$	$O(d^{\mathrm{poly}(1/\epsilon)})$ (*)	$O(d^{\mathrm{poly}(1/\epsilon)})$
Our results	$O(dk)$	$O(dk)$	$O(\mathrm{poly}(d)\exp(k))$	$O(\mathrm{poly}(d,k)/\epsilon^2)$

elaborations on Table 1.

- We show the dependence on the feature dimension d, network width or polynomial rank k, precision ϵ, and degree p.
- For the setting: Sim. denotes simulator model, Online denotes online RL, Det. denotes deterministic transitions, Stoch. denotes stochastic transitions, (R) denotes realizability assumption only, and (C) denotes completeness assumption.
- For the deterministic transition baseline, we apply Du et al. [2020c] and for the stochastic transition baseline we apply Du et al. [2020a]. We are unaware of any methods that can directly learn MDP with neural network value function approximation[1].
- In polynomial case, the baseline first vectorizes the tensor $\begin{pmatrix}1\\x\end{pmatrix}^{\otimes p}$ into a vector in $\mathbb{R}^{(d+1)^p}$ and then performs on this vector. In the neural network case, the baseline uses a polynomial of degree $1/\epsilon$ to approximate the neural network with precision ϵ and then vectorizes the polynomial into a vector in $\mathbb{R}^{d^{\mathrm{poly}(1/\epsilon)}}$. The baseline method for realizable model (denoted by (*)) needs a further gap assumption of gap $\geq d^{\mathrm{poly}(1/\epsilon)}\epsilon$ to avoid the approximation error from propagating Du et al. [2020c]; note for small ϵ this condition never holds but we include it in the table for the sake of comparison.
- In rank k polynomial case, our result $O(dk)$ in simulator model can be found in Theorem 5.7 and our result $O(dk)$ in online RL model can be found in Theorem 5.8. These results only require realizability assumption. Efficient explorations are guaranteed by algebraic geometric arguments. In neural network model, our result $O(\mathrm{poly}(d)\exp(k))$ in simulator model can

be found in Theorem 4.4. This result also only relies on the realizability assumption. For stochastic transitions, our result $O(\mathrm{poly}(d,k)/\epsilon^2)$ works for both policy complete and Bellman complete setting, as in Theorem 4.5 and Theorem 4.6 respectively.

2 Related Works

Linear Function Approximation. RL with linear function approximation has been widely studied under various settings, including linear MDP and linear mixture MDP [Jin et al., 2020b, Zanette et al., 2020, Yang and Wang, 2020]. While these papers have proved efficient regret and sample complexity bounds, their analyses relied heavily on two techniques: they used the confidence ellipsoid to quantify the uncertainty, and they used the elliptical potential lemma to bound the cumulative amount of uncertainty [Abbasi-Yadkori et al., 2011]. These two techniques were integral to their analyses, but were so restrictive that they could hardly be extended to nonlinear cases.

Eluder Dimension. Russo and Van Roy [2013a], Osband et al. [2013] came up with the notion of eluder dimension, which served as a complexity measure of the function space. In addition, they proved regret and sample complexity bounds that scaled with the eluder dimension, and showed that the eluder dimension is small (i.e. not exponentially large) in several settings, including generalized linear models and LQR. Wang et al. [2020b], Jin et al. [2021] extend Eluder dimension to reinforcement learning. However, as shown in [Dong et al., 2021], the eluder dimension could be exponentially large even with a single ReLU neuron, which suggested eluder dimension would face difficulty in dealing with neural network cases.

Bellman Rank and Completeness. Jiang et al. [2017], Sun et al. [2019] studied RL with general function approximation. They used Bellman rank to measure the error of the function class under the Bellman operator, and gave proved bounds in the term of it. Recently, Du et al. [2021] propose Bilinear rank and encompass more function approximation models. However, it is hard to bound the Bellman rank or Bilinear rank for neural nets, so their results are not known to include the neural network approximation setting. Another line of work (Du et al. [2020a], Weisz et al. [2020], Wang et al. [2021b]) shows that exponential sample complexity is unavoidable even with good representations.

3 Preliminaries

An episodic Markov Decision Process (MDP) is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action set, H is the number of time steps in each episode, $\mathbb{P}$ is the transition kernel and r is the reward function. In each episode the agent starts at a fixed initial state s_1 and at each time step $h \in [H]$ it takes action a_h, receives reward $r_h(s_h, a_h)$ and transits to $s_{h+1} \sim \mathbb{P}(\cdot|s_h, a_h)$.

A deterministic policy π is a length-H sequence of functions $\pi = \{\pi_h : \mathcal{S} \mapsto \mathcal{A}\}_{h=1}^H$. Given a policy π, we define the value function $V_h^\pi(s)$ as the expected sum of reward under policy π starting from $s_h = s$:

$$V_h^\pi(s) := \mathbb{E}\left[\sum_{t=h}^{H} r_t(s_t, a_t)|s_h = s\right]$$

and we define the Q function $Q_h^\pi(s,a)$ as the the expected sum of reward taking action a in state $s_h = s$ and then following π:

$$Q_h^\pi(s,a) := \mathbb{E}\left[\sum_{t=h}^{H} r_t(s_t, a_t)|s_h = s, a_h = 1\right].$$

[1]Prior work on neural function approximation has focused on neural tangent kernels, which would require $d^{\mathrm{poly}(1/\epsilon)}$ to approximate a two-layer network Ghorbani et al. [2021].

The Bellman operator $\mathcal{T}_h$ applied to Q-function Q_{h+1} is defined as follow

$$\mathcal{T}_h(Q_{h+1})(s,a) := r_h(s,a) + \mathbb{E}_{s'\sim\mathbb{P}(\cdot|s,a)}[\max_{a'} Q_{h+1}(s',a')].$$

There exists an optimal policy π^* that gives the optimal value function for all states, i.e. $V_h^{\pi^*}(s) = \sup_\pi V_h^\pi(s)$ for all $h \in [H]$ and $s \in \mathcal{S}$. For notational simplicity we abbreviate V^{π^*} as V^* and correspondingly Q^{π^*} as Q^*. Therefore Q^* satisfies the following Bellman optimality equations for all $s \in \mathcal{S}$, $a \in \mathcal{A}$ and $h \in [H]$:

$$Q_h^*(s,a) = \mathcal{T}_h(Q_{h+1}^*)(s,a).$$

The goal is to find a policy π that is ϵ-optimal in the sense that $V_1^*(s_1) - V_1^\pi(s_1) \le \epsilon$, within a small number of samples. We consider two query models of interacting with MDP:

- In the simulator model (Kakade [2003], Sidford et al. [2018]), the agent interacts with a black-box that simulates the MDP. At each time step $h \in [H]$, the agent can start at a state-action pair (s,a) and interact with the black box by executing some policy π chosen by the agent.
- In online RL, the agent can only start at the initial state and interact with the MDP by using a policy and observing the rewards and the next states. In each episode k, the agent proposes a policy π^k based on all history up to episode $k-1$ and executes π^k to generate a single trajectory $\{s_h^k, a_h^k\}_{h=1}^H$ with $a_h^k = \pi_h^k(s_h^k)$ and $s_{h+1}^k \sim \mathbb{P}_h(\cdot|s_h^k, a_h^k)$.

3.1 Function approximation

In reinforcement learning with value function approximation, the learner is given a function class $\mathcal{F} = \mathcal{F}_1 \times \cdots \times \mathcal{F}_H$, where $\mathcal{F}_h \subset \{f : \mathcal{S}\times\mathcal{A} \mapsto [0,1]\}$ is a set of candidate functions to approximate Q^*. The following assumption is commonly adopted in the literature.

Assumption 3.1 (Realizability)**.** $Q_h^* \in \mathcal{F}_h$ for all $h \in [H]$.

The function approximation is equipped with feature mapping $\phi : \mathcal{S}\times\mathcal{A} \mapsto \{u \in \mathbb{R}^d : \|u\|_2 \le B_\phi\}$ that is known to the agent. We make the following assumption about ϕ that allows efficient explorations.

Assumption 3.2 (Dense features)**.** We assume the image of ϕ contains the ball $\{u \in \mathbb{R}^d : \|u\|_2 \le d\text{polylog}(d)\}$.

Notations For any vector $x \in \mathbb{R}^d$, let $x_{\max} := \max_{i\in[d]} x_i$ and $x_{\min} := \min_{i\in[d]} x_i$. Let $s_i(\cdot)$ denote the i-th singular value, $s_{\min}(\cdot)$ denotes the minimum eigenvalue and $s_{\max}(\cdot)$ denotes the maximum eigenvalue. The conditional number is defined by $\kappa(\cdot) := s_{\max}(\cdot)/s_{\max}(\cdot)$. We use $\otimes$ to denote Kronecker product and $\circ$ to denote Hadamard product. For a given integer H, we use $[H]$ to denote the set $\{1, 2, \ldots, H\}$.

4 Neural Network Function Approximation

In this section we show sample efficient algorithms with neural network function approximations. The function class of interest is given in the following definition. More general neural network class is discussed in Section 4.

Definition 4.1 (Neural Network Function Class)**.** We use $\mathcal{F}_{NN}$ to denote the function class of $f(\phi(s,a)) : \mathcal{S}\times\mathcal{A} \mapsto \mathbb{R}$ where $f(x) = \langle v, \sigma(Wx)\rangle$ is a two-layer neural network where σ is ReLU, $\|W\|_F \le B_W$, $\prod_{i=1}^k s_i(W)/s_{\min}(W) \le \lambda$, $s_{\max}(W)/s_{\min}(W) \le \kappa$ and $k \le d$.

We introduce the following completeness properties in the setting of value function approximations. Along with Assumption 3.1, they are commonly adopted in the literature.

Definition 4.2 (Policy complete)**.** Given MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, r, H)$, function class $\mathcal{F}_h : \mathcal{S}\times\mathcal{A} \mapsto \mathbb{R}, h \in [H]$ is called policy complete iff for all π and $h \in [H]$, $Q_h^\pi \in \mathcal{F}_h$.

Definition 4.3 (Bellman complete)**.** Given MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, r, H)$, function class $\mathcal{F}_h : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}, h \in [H]$ is called Bellman complete iff for all $h \in [H]$ and $Q_{h+1} \in \mathcal{F}_{h+1}$, $\mathcal{T}_h(Q_{h+1}) \in \mathcal{F}_h$.

4.1 Warmup: Realizable Q^* with deterministic transition

We start by considering the setting that transition kernel is deterministic. In this case only Assumption 3.1 is required for neural network function approximations. The learning procedure, presented in Algorithm 1, learns optimal policy from time step H to 1. Suppose we have learned optimal policy $\pi_{h+1}, \ldots, \pi_H$ at level h, we can then query features from a standard Gaussian distribution and if $\|\phi(s,a)\|_2 \geq B_\phi$ then it simply skips this trial. Notice that $B_\phi \geq d^2$, so with high probability [2] most samples fall in the feature set $\{u \in \mathbb{R}^d : \|u\|_2 \leq B_\phi\}$. We next construct an estimate $\widehat{Q}_h$ of Q^* by using $\pi_{h+1}, \ldots, \pi_H$ as the roll-out. Since the transition is deterministic, $\widehat{Q}_h(s_h^i, a_h^i) = Q^*(s_h^i, a_h^i)$ and for all the samples $\{(s_h^i, a_h^i) : i \in [n]\}$. Recall Q_h^* is a two-layer neural network, we can now recover its parameters in Step 8 exactly by invoking techniques in neural network optimization (see, e.g. Janzamin et al. [2015], Ge et al. [2017], Zhong et al. [2017]). Details of this step can be found in Appendix D.3. Making using of exact recovery of Q_h^*, the algorithm can thus find optimal policy π_h^*.

Algorithm 1 Learning realizable Q^* with deterministic transition

1: **for** $h = H, \ldots 1$ **do**
2: Sample $\phi(s_h^i, a_h^i), i \in [n]$ from standard Gaussian $N(0, I_d)$
3: ▷ If $\|\phi(s_h^i, a_h^i)\|_2 \geq B_\phi$ then skip this trial.
4: **for** $i \in [n]$ **do**
5: Query (s_h^i, a_h^i) and use $\pi_{h+1}, \ldots, \pi_H$ as the roll-out to collect rewards $r_h^{(i)}, \ldots, r_H^{(i)}$
6: Construct estimation

$$\widehat{Q}_h^{\pi_{h+1},\ldots,\pi_H}(s_h^i, a_h^i) = r_h^{(i)} + \cdots + r_H^{(i)}$$

7: **end for**
8: Recover v_h and W_h from samples $\{\phi(s_h^i, a_h^i), \widehat{Q}_h^{\pi_{h+1},\ldots,\pi_H}(s_h^i, a_h^i) : i \in [n]\}$ using method of moments and gradient descent.
9: Set $\widehat{Q}_h(s,a) \leftarrow v_h^\top \sigma(W_h \phi(s,a))$
10: Set $\pi_h(s) \leftarrow \arg\max_{a \in \mathcal{S}} \widehat{Q}_h(s,a)$
11: **end for**
12: **Return** $\pi_1, \ldots, \pi_H$

Theorem 4.4. *Suppose sample complexity $n \geq d \cdot \mathrm{poly}(\kappa, k, \lambda, \log d, B_W, B_\phi, H)$, then with high probability Algorithm 2 learns optimal policy π^*.*

The complete proof is deferred to Appendix. The main idea of exact neural network recovery can be summarized in the following. We first use method of moments to find a 'rough' parameter recovery. If this 'rough' recovery is sufficiently close to the true parameter, the loss function is locally strongly convex and there is unique global minimum. Then we can apply gradient descent to find this global minimum which is exactly the true parameter.

4.2 Policy complete neural function approximation

Now we consider general stochastic transitions. Difficulty arises in this scenario duo to noises in the estimation of Q-functions. In the presence of model misspecification, these noises cause estimation errors to amplify through levels and require samples to be exponential in H. In this section we show that neural network function approximation is still learnable assuming the function class $\mathcal{F}_{NN}$ is policy complete with regard to MDP $\mathcal{M}$. Thus for all $\pi \in \Pi$, we can denote $Q_h^\pi(s,a) = \langle v^\pi, \sigma(W^\pi \phi(s,a)) \rangle$.

Algorithm 2 learns policy from level $H, H-1, \ldots, 1$. In level h, the algorithm has learned policy $\pi_{h+1}, \ldots, \pi_H$ that is only sub-optimal by $(H-h)\epsilon/H$. Then it selects features $\phi(s,a)$ from $\mathcal{N}(0, I_d)$.

[2] Here with high probability is with respect to d.

Algorithm 2 Learn policy complete NN with simulator.

Input: Precision ϵ.
for $h = H, \dots 1$ **do**
  Sample $\phi(s_h^i, a_h^i), i \in [n]$ from standard Gaussian $N(0, I_d)$
  ▷ If $\|\phi(s_h^i, a_h^i)\|_2 \geq B_\phi$ then skip this trial.
  for $i \in [n]$ **do**
    Query (s_h^i, a_h^i) and use $\pi_{h+1}, \dots, \pi_H$ as the roll-out to collect rewards $r_h^{(i)}, \dots, r_H^{(i)}$
    Construct unbiased estimation

$$\widehat{Q}_h^{\pi_{h+1},\dots,\pi_H}(s_h^i, a_h^i) = r_h^{(i)} + \cdots + r_H^{(i)}$$

  end for
  Retrieve v_h and W_h from samples $\{\phi(s_h^i, a_h^i), \widehat{Q}_h^{\pi_{h+1},\dots,\pi_H}(s_h^i, a_h^i) : i \in [n]\}$
  Set $\widehat{Q}_h(s, a) \leftarrow v_h^\top \sigma(W_h \phi(s, a))$
  Set $\pi_h(s) \leftarrow \arg\max_{a \in \mathcal{S}} \widehat{Q}_h(s, a)$
end for
Return $\pi_1, \dots, \pi_H$

The algorithm then queries (s, a) and uses learned policy $\pi_{h+1}, \dots, \pi_H$ as roll out to collect an unbiased estimate of Q-function $\widehat{Q}_h^{\pi_{h+1},\dots,\pi_H}(s, a)$. Recall that $Q_h^{\pi_{h+1},\dots,\pi_H}(s, a) \in \mathcal{F}_{NN}$ is a two-layer neural network, it can then be recovered from estimates. Details of this step can be found in Appendix D.3. The algorithm then finds optimal policy π_h and proceed to the level $h - 1$.

Theorem 4.5. *Suppose sample complexity $n \geq \epsilon^{-2} \cdot d \cdot \mathrm{poly}(\kappa, k, \log d, B_W, B_\phi, H)$, then with high probability Algorithm 2 learns an ϵ-optimal policy π.*

The sample complexity does not depend on λ, unlike the case of Theorem 4.4.

The main idea of the proof is that at each time step a neural network surrogate of Q^* can be constructed by the policy already learned. Suppose we have learned $\pi_{h+1}, \dots, \pi_H$ in level h, then from policy completeness $Q_h^{\pi_{h+1},\dots,\pi_H}$ belongs to $\mathcal{F}_{NN}$ and we can interact with the simulator to obtain its estimate $\widehat{Q}_h$. If $\|\widehat{Q}_h - Q_h^{\pi_{h+1},\dots,\pi_H}\|_\infty$ is small, the optimistic planning based on $\widehat{Q}_h$ is not far from the optimal policy of $Q_h^{\pi_{h+1},\dots,\pi_H}$. This reasoning can then be performed to level $h - 1$. The full proof of Theorem 4.5 is deferred to Section D.1.

4.3 Bellman complete neural function approximation

In addition to policy completeness, we show that neural network function approximation is also learning in the setting that the function class $\mathcal{F}_{NN}$ is Bellman complete with regard to MDP $\mathcal{M}$. Specifically, for $Q_{h+1} \in \mathcal{F}_{h+1}$, there is $\mathcal{T}_h(Q_{h+1})(s, a) = \langle v^{Q_{h+1}}, \sigma(W^{Q_{h+1}} \phi(s, a)) \rangle$.

Algorithm 3 is similar to the algorithm in previous section. In level h, the algorithm has constructed the Q-function $\widehat{Q}_{h+1}(s, a) = v_{h+1}^\top \sigma(W_{h+1} \phi(s, a))$ that is $(H - h)\epsilon/H$-close to the optimal Q_{h+1}^*. It then recovers weights v_h, W_h from $\mathcal{T}_h(\widehat{Q}_{h+1})(s, a) = \langle v^{\widehat{Q}_{h+1}}, \sigma(W^{\widehat{Q}_{h+1}} \phi(s, a)) \rangle$, using unbiased estimates $r_h(s_h^i, a_h^i) + \widehat{V}_{h+1}(s_{h+1}^i)$ for all $i \in [n]$. Details of this step can be found in Appendix D.3. The Q-function $\widehat{Q}_h(s, a) = v_h^\top \sigma(W_h \phi(s, a))$ constructed from weights v_h, W_h is thus $(H - h + 1)\epsilon/H$-close to the optimal Q_h^*.

Theorem 4.6. *Suppose sample complexity $n \geq \epsilon^{-2} \cdot d \cdot \mathrm{poly}(\kappa, k, \log d, B_W, B_\phi, H)$, then with high probability Algorithm 3 learns an ϵ-optimal policy π.*

Similar to Theorem 4.5, the sample complexity does not explicitly depend on λ.

Due to Bellman completeness, the error of estimation $\widehat{Q}_h$ can be controlled recursively. In fact, we can show $\|\widehat{Q}_h - Q^*(s, a)\|_\infty$ is small by induction. The full proof of Theorem 4.6 is deferred to Section D.2.

Algorithm 3 Learn Bellman complete NN with simulator.

Input: Precision ϵ.
for $h = H, \dots 1$ **do**
  Sample $\phi(s_h^i, a_h^i), i \in [n]$ from standard Gaussian $N(0, I_d)$
  $\triangleright$ If $\|\phi(s_h^i, a_h^i)\|_2 \geq B_\phi$ then skip this trial.
  for $i \in [n]$ **do**
    Query (s_h^i, a_h^i) and observe $r_h(s_h^i, a_h^i), s_{h+1}^i$
  end for
  Retrieve v_h and W_h from $\{\phi(s_h^i, a_h^i), r_h(s_h^i, a_h^i) + \widehat{V}_{h+1}(s_{h+1}^i) : i \in [n]\}$
  Set $\widehat{Q}_h(s, a) \leftarrow v_h^\top \sigma(W_h \phi(s, a))$ and $\widehat{V}_h \leftarrow \max_{a \in \mathcal{A}} \widehat{Q}_h(s, a)$
  Set $\pi_h(s) \leftarrow \arg\max_{a \in \mathcal{A}} \widehat{Q}_h(s, a)$
end for
Return $\pi_1, \dots, \pi_H$

5 Polynomial Realizability

In this section, we study the sample complexity to learn deterministic MDPs under polynomial realizability. We identify sufficient and necessary conditions for efficiently learning the MDPs for two different settings — generative model setting and online RL setting. Specifically, we show that if the image of the feature map $\phi_h(s_h, a_h)$ satisfies some postive measure conditions, then by random exploring we can identify the optimal policy with samples that is linear in the algebraic dimension of the underlying polynomial class. We also provide a lower bound example showing the seperation between the two settings.

Next, we introduce the notion of **Admissible Polynomial Families**, which are the families of strucutured polynomial that enables efficient learning.

Definition 5.1 (Admissible Polynomial Families)**.** For $x \in \mathbb{R}^d$, denote $\widetilde{x} = [1, x^\top]^\top$. Let $\mathcal{X} := \left\{\widetilde{x}^{\otimes p} : x \in \mathbb{R}^d\right\}$. For any algebraic variety $\mathcal{V}$, we define $\mathcal{F}_\mathcal{V} := \{f_\Theta(x) = \langle \Theta, \widetilde{x}^{\otimes p} \rangle : \Theta \in \mathcal{V}\}$ as the polynomial family parameterized by $\Theta \in \mathcal{V}$. We say $\mathcal{F}_\mathcal{V}$ is admissible[3] w.r.t. $\mathcal{X}$, if for any $\Theta \in \mathcal{V}$, $\dim(\mathcal{X} \cap \{X \in \mathcal{X} : \langle X, \Theta \rangle = 0\rangle\}) < \dim(\mathcal{X}) = d$. We define the dimension D of the family to be the dimension of $\mathcal{V}$.

Next theorem shows that to learn an admissible polynomial family, the sample complexity only scales with the algebraic dimension of the polynomial family.

Theorem 5.2 (Anonymous)**.** *Consider the polynomial family $\mathcal{F}_\mathcal{V}$ of dimension D. For $n \geq 2D$, there exists a Lebesgue-measure zero set $N \in \mathbb{R}^d \times \dots \mathbb{R}^d$, such that if $(x_1, \cdots, x_n) \notin N$, for any y_i, there is a unique f (or no such f) to the system of equations $y_i = f(x_i)$ for $f \in \mathcal{F}_\mathcal{V}$.*

We give two important examples of admissible polynomial families with low dimension.

Example 5.3. (Low-rank Polynomial of rank k) The function $f \in \mathcal{F}_\mathcal{V}$ is a polynomial with k terms, that is

$$F(x) = \sum_{i=1}^{k} \lambda_i \langle v_i, x \rangle^{p_i},$$

where $p = \max\{p_i\}$. The dimension of this family is upper bounded by $D \leq dk$. Neural network with monomial/polynomial activation functions are low-rank polynomials.

Example 5.4. The function $f \in \mathcal{F}_\mathcal{V}$ is of the form $f(x) = q(Ux)$, where $U \in \mathbb{R}^{k \times d}$ and q is a degree p polynomial. The polynomial q and matrix U are unknown. The dimension of this family is upper bounded by $D \leq d(k+1)^p$.

Next we introduce the notion of positive measure.

[3] Admissible means the dimension of $\mathcal{X}$ decreases by one when there is an additional linear constraint $\langle \Theta, X \rangle = 0$

Definition 5.5. We say a measurable set $E \in \mathbb{R}^d$ is of positive measure, if $\mu(E) > 0$, where μ is the standard Lebesgue measure on $\mathbb{R}^d$.

If a measurable set E satisfies $\mu(E) > 0$, then there exists a procedure to draw samples from E, such that for any $N \subset \mathbb{R}^d$ of Lebesgue-measure zero, the probability that the sample falls in N is zero. In fact, the sampling probability can be given by $\mathbb{P}_{x\in\mathcal{N}(0,I_d)}(\cdot|x \in E)$. The intuition behind is definition is that for all admissible polynomial families, the set of $(x_1, \cdots, x_n)$ that has "redundant information" about learning the parameter Θ is of Lebesgue-measure zero. Therefore, a positive measure set allows you to query randomly and avoids getting coherent measurements.

Next two theorems identify the sufficent conditions for efficiently learning deterministic MDPs under polynomial realizability. Specifically, under online RL setting, we require the strong assumption that the set $\{\phi_h(s, a)|a \in \mathcal{A}\}$ is of positive measure for all $h \in [H]$ and all $s \in \mathcal{S}$, while under generative model setting, we only require the union set $\bigcup_{s\in\mathcal{S}}\{\phi_h(s, a)|a \in \mathcal{A}\}$ to be of positive measure for all $h \in [H]$. The algorithms for solving the both cases are summarized in Algorithms 4 and 5.

Assumption 5.6 (Polynomial Realizability)**.** For all $h \in [H]$, $Q^*_h(s_h, a_h)$, viewed as the function of $\phi_h(s_h, a_h)$, lies in some admissible polynomial family $\mathcal{F}_{\mathcal{V}_h}$ with dimension bounded by D.

Theorem 5.7. *For the generative model setting, assume that the set $\{\phi_h(s, a) \mid s \in \mathcal{S}, a \in \mathcal{A}\}$ is of positive measure at any level h. Under the polynomial realizability, Algorithm 4 almost surely learns the optimal policy $\pi^\star$ with at most $N = 2DH$ samples.*

Theorem 5.8. *For the online RL setting, assume that $\{\phi_h(s, a) \mid a \in \mathcal{A}\}$ is of positive measure for every state s at every level h. Under polynomial realizability, within $T = 2DH$ episodes, Algorithm 5 learns the optimal policy $\pi^\star$ almost surely.*

Algorithm 4 Dynamic programming under generative model settings

for $h = H, \cdots, 1$ **do**
  1. Sample $2D$ points $\{\phi_h(s_h^{(i)}, a_h^{(i)})\}_{i=1}^{2D}$ according to $\mathbb{P}_{x\in\mathcal{N}(0,I_d)}(\cdot|x \in E_h)$ where $E_h = \{\phi_h(s, a) \mid s \in \mathcal{S}, a \in \mathcal{A}\}$.
  2. Query the generative model with state-action pair $(s_h^{(i)}, a_h^{(i)})$ at level h for $i = 1, \ldots, 2D$, and observe the next state $\widetilde{s}_h^{(i)}$ and reward $r_h^{(i)}$.
  3. Solve for Q^*_h with the $2D$ equations $Q^*_h(s_h^{(i)}, a_h^{(i)}) = r_h^{(i)} + V^*_{h+1}(\widetilde{s}_h^{(i)})$.
  4. Set $\pi^*_h(s) = \arg\max_a Q^*_h(s, a)$ and $V^*_h(s) = \max_a Q^*_h(s, a)$.
end for
Output π^*

Algorithm 5 Dynamic programming under online RL settings

for $h = H, \cdots, 1$ **do**
  1. Fix any action sequence $a_1, \cdots, a_{h-1}$.
  2. Play $a_1, \cdots, a_{h-1}$ for the first $h - 1$ levels and reaches a state s_h. Sample $2D$ points $\{\phi_h(s_h, a_h^{(i)})\}_{i=1}^{2D}$ according to $\mathbb{P}_{x\in\mathcal{N}(0,I_d)}(\cdot|x \in E_h)$ where $E_h = \{\phi_h(s_h, a) \mid a \in \mathcal{A}\}$.
  3. play $a_h^{(i)}$ at s_h for $i = 1, \ldots, 2D$, and observe the next state $\widetilde{s}_h^{(i)}$ and reward $r_h^{(i)}$.
  4. Solve for Q^*_h with the $2D$ equations $Q^*_h(s_h^{(i)}, a_h^{(i)}) = r_h^{(i)} + V^*_{h+1}(\widetilde{s}_h^{(i)})$.
  5. Set $\pi^*_h(s) = \arg\max_a Q^*_h(s, a)$ and $V^*_h(s) = \max_a Q^*_h(s, a)$.
end for
Output π^*

We remark that our Theorem 5.8 for learning MDPs under the online RL setting relies on a very strong assumption that allows the learner to explore randomly for any state. However, in some sense this assumption is necessary, as suggested by our lower bound example in the next subsection.

5.1 Necessity of Generic Feature Maps in Online RL

In this section, we consider lower bounds for learning deterministic MDPs with polynomial realizable Q^* under online RL setting. Our goal is to show that in the online setting the generic assumption on the feature maps $\phi_h(s, \cdot)$ is necessary. On the contrary, under the generative model setting one can efficiently learn the MDPs without such a strong assumption, since at every level h the we can set the state arbitrarily.

MDP construction We briefly introduce the intuition of our construction. Consider a family of MDPs with only two states $\mathcal{S} = \{S_{\text{good}}, S_{\text{bad}}\}$. we set the feature map ϕ_h such that, for the good state S_{good}, it allows the learner to explore randomly, i.e., $\{\phi_h(S_{\text{good}}, a) \mid a \in \mathcal{A}\}$ is generic. However, for the bad state S_{bad}, all actions are mapped to some restricted set, which forbids random exploration, i.e., $\{\phi_h(S_{\text{bad}}, a) \mid a \in \mathcal{A}\}$ is not generic. This is illustrated in Figure 1. Specifically, at least $\Omega(d^p)$ actions are needed to identify the groud-truth polynomial of Q_h^* for S_{bad}, while $O(d)$ actions suffice for S_{good}. The transition $\mathbb{P}_h$ is constructed as $\mathbb{P}_h(s_{\text{bad}}|s, a) = 1$ for all $a \in A$, which means it is impossible for the online scenarios to reach the good state for $h > 1$.

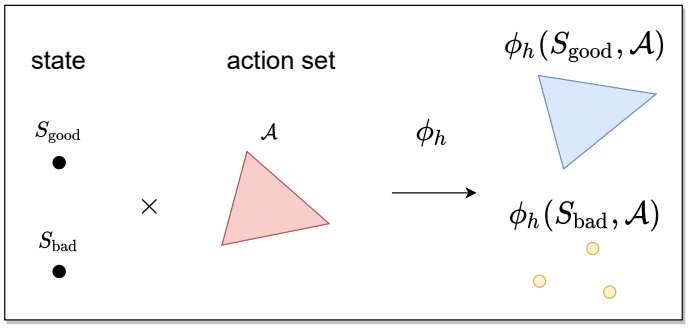

Figure 1: An illustration of the hard case for deterministic MDPs with polynomial realizable Q^*. The image of the feature map ϕ_h at S_{good} is generic, while the image of ϕ_h at S_{bad} is not. This makes it difficult to learn in the online RL setting.

Theorem 5.9. *There exists a family of MDPs satisfying Assumption 5.6, such that the set $\{\phi_h(s, a) \mid s \in \mathcal{S}, a \in \mathcal{A}\}$ is generic at any level h, but for all h there is some $s_{bad} \in \mathcal{S}$ such that $\{\phi_h(s_{bad}, a) \mid a \in \mathcal{A}\}$ is not generic. Under the online RL setting, any algorithm needs to play at least $\Omega(d^p)$ episodes to identify the optimal policy. On the contrary, under the generative model setting, only $O(d)H$ samples are needed.*

6 Conclusions

In this paper, we consider neural network and polynomial value function approximation in the simulator setting. To our knowledge, this is the first paper that shows sample-efficient reinforcement learning is possible with neural net function approximation. Our results substantially improve upon what can be achieved with existing results that primarily rely on embedding neural networks into linear function classes.

Our results for polynomial activation require deterministic transitions, since we cannot handle how noise propagates in solving polynomial equations. We leave to future work an in-depth studying the stability of roots of polynomial systems under noise, which is a fundamental mathematical problem. We believe that noisy tensor decomposition approaches combined with zeroth order optimization may allow for stochastic transitions.

In the online RL setting, we are only able to show efficient learning under a very strong assumption, yet necessary assumption on the feature mapping. We leave to future work identifying more realistic and natural conditions which permit efficient learning in the online RL setting.

References

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

Yasin Abbasi-Yadkori, Nevena Lazic, Csaba Szepesvari, and Gellert Weisz. Exploration-enhanced POLITEX. *arXiv preprint arXiv:1908.10479*, 2019.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning (ICML)*, pages 242–252, 2019.

Anonymous. Anonymous submission by same authors.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 263–272, 2017.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, and Chris Hesse. Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680, 2019.

Christoph Dann, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E. Schapire. On oracle-efficient PAC-RL with rich observations. In *Advances in Neural Information Processing Systems*, 2018.

Kefan Dong, Jian Peng, Yining Wang, and Yuan Zhou. $\sqrt{n}$-regret for learning in Markov decision processes with function approximation and low Bellman rank. In *Conference on Learning Theory*, pages 1554–1557. PMLR, 2020.

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.

Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning (ICML)*, pages 1675–1685. PMLR, 2019a.

Simon S Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudik, and John Langford. Provably efficient RL with rich observations via latent state decoding. In *International Conference on Machine Learning*, pages 1665–1674, 2019b.

Simon S Du, Sham M Kakade, Ruosong Wang, and Lin F Yang. Is a good representation sufficient for sample efficient reinforcement learning? In *International Conference on Learning Representations*, 2020a.

Simon S. Du, Sham M. Kakade, Ruosong Wang, and Lin F. Yang. Is a good representation sufficient for sample efficient reinforcement learning? In *International Conference on Learning Representations*, 2020b.

Simon S Du, Jason D Lee, Gaurav Mahajan, and Ruosong Wang. Agnostic Q-learning with function approximation in deterministic systems: Tight bounds on approximation error and sample complexity. *Advances in Neural Information Processing Systems*, 2020c.

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.

Rong Ge, Jason D. Lee, and Tengyu Ma. Learning one-hidden-layer neural networks with landscape design. arXiv preprint arXiv:1711.00501, 2017.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Linearized two-layers neural networks in high dimension. *The Annals of Statistics*, 49(2):1029–1054, 2021.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems (NeurIPS)*, pages 8571–8580, 2018.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

Majid Janzamin, Hanie Sedghi, and Anankumar Anima. Beating the perils of nonconvexity: Guaranteed training of neural networks using tensor methods. arXiv preprint arXiv:1506.08473, 2015.

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low Bellman rank are PAC-learnable. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1704–1713, 2017.

Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is Q-learning provably efficient? In *Advances in Neural Information Processing Systems*, pages 4863–4873, 2018.

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

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

Chi Jin, Qinghua Liu, and Sobhan Miryoosefi. Bellman eluder dimension: New rich classes of rl problems, and sample-efficient algorithms. arXiv preprint arXiv:2102.00875, 2021.

Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, University of College London, 2003.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Boyi Liu, Qi Cai, Zhuoran Yang, and Zhaoran Wang. Neural proximal/trust region policy optimization attains globally optimal policy. arXiv preprint arXiv:1906.10306, 2019.

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

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*, 2013b.

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pages 3003–3011, 2013.

Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning via sequential complexities. *Journal of Machine Learning Research*, 16(6):155–186, 2015a.

Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Sequential complexities and uniform martingale laws of large numbers. *Probability Theory and Related Fields*, 161(1-2):111–153, 2015b.

Dan Russo and Benjamin Van Roy. Eluder dimension and the sample complexity of optimistic exploration. In *Advances in Neural Information Processing Systems*, pages 2256–2264, 2013a.

Daniel Russo. Worst-case regret bounds for exploration via randomized value functions. In *Advances in Neural Information Processing Systems*, pages 14433–14443, 2019.

Daniel Russo and Benjamin Van Roy. Eluder dimension and the sample complexity of optimistic exploration. In *Advances in Neural Information Processing Systems*, 2013b.

Igor R Shafarevich. *Basic Algebraic Geometry 1: Varieties in Projective Space*. Springer Science & Business Media, 2013.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

Aaron Sidford, Mengdi Wang, Xian Wu, Lin F Yang, and Yinyu Ye. Near-optimal time and sample complexities for solving discounted markov decision process with a generative model. arXiv preprint arXiv:1806.01492, 2018.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, and Marc Lanctot. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

Alexander L Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L Littman. PAC model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 881–888. ACM, 2006.

Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Model-based RL in contextual decision processes: PAC bounds and exponential improvements over model-free approaches. In *Conference on Learning Theory*, pages 2898–2933, 2019.

Richard S Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Lingxiao Wang, Qi Cai, Zhuoran Yang, and Zhaoran Wang. Neural policy gradient methods: Global optimality and rates of convergence. arXiv preprint arXiv:1909.01150, 2020a.

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

Yining Wang, Ruosong Wang, Simon S. Du, and Akshay Krishnamurthy. Optimism in reinforcement learning with generalized linear function approximation. In *International Conference on Learning Representations*, 2021a.

Yuanhao Wang, Ruosong Wang, and Sham M. Kakade. An exponential lower bound for linearly-realizable mdps with constant suboptimality gap. arXiv preprint arXiv:2103.12690, 2021b.

Gellert Weisz, Philip Amortila, and Csaba Szepesvári. Exponential lower bounds for planning in mdps with linearly-realizable optimal action-value functions. *arXiv preprint arXiv:2010.01374*, 2020.

Zheng Wen and Benjamin Van Roy. Efficient reinforcement learning in deterministic systems with value function generalization. *Math. Oper. Res.*, 42(3):762–782, 2017. ISSN 0364-765X.

Lin F Yang and Mengdi Wang. Reinforcement leaning in feature space: Matrix bandit, kernels, and regret bound. *International Conference on Machine Learning*, 2020.

Andrea Zanette and Emma Brunskill. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. In *International Conference on Machine Learning*, pages 7304–7312, 2019.

Andrea Zanette, Alessandro Lazaric, Mykel Kochenderfer, and Emma Brunskill. In *International Conference on Machine Learning*, 2020.

Kai Zhong, Zhao Song, Prateek Jain, Peter L. Bartlett, and Inderjit S. Dhillon. Recovery guarantees for one-hidden-layer neural networks. *Proceedings of the Thirty-fourth International Conference on Machine Learning (ICML)*, 70, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Our abstract and introduction accurately reflect our main contribution.
   (b) Did you describe the limitations of your work? [Yes] In the Preliminary section we clearly state the settings we considered and when our results apply.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work is theoretical and generally will not have negative social impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] The assumptions are clearly included in every theorem or lemma.
   (b) Did you include complete proofs of all theoretical results? [Yes] All the proofs can be found in the appendix
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]