Going Beyond Linear RL: Sample Efficient Neural Function Approximation

Baihe Huang*  Kaixuan Huang†  Sham M. Kakade‡
Jason D. Lee§  Qi Lei¶  Runzhe Wang‖  Jiaqi Yang**

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 an 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 the observation that 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, b; 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 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 of 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 (generative model) 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 a 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. 2020c; Wang et al. 2021b], but with a 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 4.3). We make the following elaborations on Table 1.

Table 1: Baselines and our main results for the sample complexity to find an $\epsilon$-optimal policy.

<table>
<thead>
<tr>
<th>rank $k$ polynomial</th>
<th>Neural Net of Width $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sim. + Det. (R)</td>
<td>Online + Det. (R)</td>
</tr>
<tr>
<td>Baseline</td>
<td>$O(d^p)$</td>
</tr>
<tr>
<td>Our results</td>
<td>$O(dk)$</td>
</tr>
</tbody>
</table>

| Sim. + Det. (R)     | $O(d^p)$                 |
| Our results         | $O(dk)$                  |

| Sim. + Stoch. (C)   | $O(d^\text{poly}(1/\epsilon))$ (*) | $O(d^\text{poly}(1/\epsilon))$ |
| Our results         | $O(d^\exp(k))$             | $O(d^{1+p\text{poly}(k)/\epsilon^2})$ |

- We show the dependence on the feature dimension $d$, network width or polynomial rank $k$, precision $\epsilon$, 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., 2020d] and for the stochastic transition baseline we apply [Du et al., 2020b]. We are unaware of any methods that can directly learn MDP with neural network value function approximation
- In polynomial case, the baseline first vectorizes the tensor $\left(\frac{1}{x}\right)^\otimes p$ 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 $\epsilon$ and then vectorizes the polynomial into a vector in $\mathbb{R}^{d^{\text{poly}(1/\epsilon)}}$. The baseline method for realizable model (denoted by (*)) needs a further gap assumption of $\text{gap} \geq d^{\text{poly}(1/\epsilon)} \epsilon$ to avoid the approximation error from propagating [Du et al., 2020d]; note for small $\epsilon$ 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 4.7 and our result $O(dk)$ in online RL model can be found in Theorem 4.8. These results only require a realizability assumption. Efficient explorations are guaranteed by algebraic-geometric arguments. In neural network model, our result $O(\text{poly}(d) \exp(k))$ in simulator model can be found in Theorem 3.4. This result also only relies on the realizability assumption. For stochastic transitions, our result $O(\text{poly}(d,k)/\epsilon^2)$ works for both policy complete and Bellman complete setting, as in Theorem 3.5 and Theorem 3.6 respectively.

1.2 Related Work

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 total uncertainty [Abbasi-Yadkori et al., 2011]. These two techniques were integral to their analyses but are so restrictive that they generally do not extend to nonlinear cases.

Eluder Dimension. Russo and Van Roy [2013a], Osband et al. [2013] proposed eluder dimension, a complexity measure of the function space, and proved regret and sample complexity bounds that scaled with the eluder dimension, for bandits and reinforcement learning [Wang et al., 2020b, Jin et al., 2021]. They also showed that the eluder dimension is small in several settings, including generalized linear models and LQR. However, as shown in [Dong et al., 2021], the eluder dimension could be exponentially large even with a single ReLU neuron, which suggested the eluder dimension would face difficulty in dealing with neural network cases. The eluder dimension is only known to give non-trivial bounds for linear function classes and monotone functions of linear function classes. For structured polynomial classes, the eluder dimension simply embeds into an ambient linear space of dimension $d^p$, where $d$ is the dimension, and $p$ is the degree. This parallels the lower bounds in linearization / neural tangent kernel (NTK) works [Wei et al. 2019, Ghorbani et al., 2019, Allen-Zhu and Li, 2019], which show that linearization also incurs a similarly large penalty of $d^p$ sample complexity, and more advanced algorithm 1

1 Prior work on neural function approximation has focused on neural tangent kernels, which would require $d^{\text{poly}(1/\epsilon)}$ to approximate a two-layer network [Ghorbani et al., 2021].
design is need to circumvent linearization\cite{Bai and Lee, 2020, Chen et al., 2020, Fang et al., 2020, Woodworth et al., 2019, Gao et al., 2019, Nacson et al., 2019, Ge et al., 2018, Moroshko et al., 2020, HaoChen et al., 2020, Wang et al., 2020c, Damian et al., 2021}.

**Bellman Rank and Completeness.** \cite{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, \cite{Du et al., 2021} propose bilinear rank and encompass more function approximation models. However, it is hard to bound either the Bellman rank or the bilinear rank for neural nets. Therefore, their results are not known to include the neural network approximation setting. Another line of work shows that exponential sample complexity is unavoidable even with good representations \cite{Du et al., 2020b, Weisz et al., 2020, Wang et al., 2021b}.

**Deterministic RL** Deterministic system is often the starting case in the study of sample-efficient algorithms, where the issue of exploration and exploitation trade-off is more clearly revealed since both the transition kernel and reward function are deterministic. The seminal work \cite{Wen and Van Roy, 2013} proposes a sample-efficient algorithm for Q-learning that works for a family of function classes. Recently, \cite{Du et al., 2020d} studies the agnostic setting where the optimal Q-function can only be approximated by a function class with approximation error. The algorithm in \cite{Du et al., 2020d} learns the optimal policy with the number of trajectories linear with the eluder dimension.

## 2 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 $s_{h+1} \sim \mathbb{P}(.|s_h, a_h)$.

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

$$V^\pi_h(s) := \mathbb{E} \left[ \sum_{t=h}^H r_t(s_t, a_t) | s_h = s \right]$$
and we define the Q function $Q^\pi_h(s, a)$ as the the expected sum of reward taking action $a$ in state $s_h = s$ and then following $\pi$:

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

The Bellman operator $T_h$ applied to Q-function $Q_{h+1}$ is defined as follow:

$$T_h(Q_{h+1})(s, a) := r_h(s, a) + \mathbb{E}_{s' \sim \mathbb{P}(. \mid s, a)}[\max_{a'} Q_{h+1}(s', a')].$$

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

$$Q^*_h(s, a) = T_h(Q^*_{h+1})(s, a).$$

The goal is to find a policy $\pi$ that is $\epsilon$-optimal in the sense that $V^*_1(s_1) - V^*_1(s_1) \leq \epsilon$, 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 $\pi$ 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 $\pi^k$ based on all history up to episode $k - 1$ and executes $\pi^k$ to generate a single trajectory $\{s^k_h, a^k_h\}_{h=1}^H$ with $a^k_h = \pi^k_h(s^k_h)$ and $s^k_{h+1} \sim \mathbb{P}_h(. \mid s^k_h, a^k_h)$.

### 2.1 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 2.1 (Realizability)**. $Q^*_h \in \mathcal{F}_h$ for all $h \in [H]$. 

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

**Assumption 2.2 (Dense features).** We assume the image of \( \phi \) contains the ball \( \{ u \in \mathbb{R}^d : \| u \|_2^2 \leq d \cdot \text{polylog}(d) \} \).

**Notation** 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 \( \kappa(\cdot) := s_{\max}(\cdot)/s_{\min}(\cdot) \). 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 \( \{1, 2, \ldots, H\} \).

### 3 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 3.

**Definition 3.1 (Neural Network Function Class).** We use \( \mathcal{F}_{\text{NN}} \) to denote the function class of \( f(\phi(s, a)) : S \times A \mapsto \mathbb{R} \) where \( f(x) = \langle v, \sigma(Wx) \rangle \) is a two-layer neural network where \( \sigma \) is ReLU, \( \| W \|_F \leq B_W \), \( v \in \{-1, 1\}^k \), \( \prod_{i=1}^k s_i(W)/s_{\min}(W) \leq \lambda \), \( s_{\max}(W)/s_{\min}(W) \leq \kappa \) and \( k \leq d \).

We introduce the following completeness properties in the setting of value function approximations. Along with Assumption 2.1, they are commonly adopted in the literature.

**Definition 3.2 (Policy complete).** Given MDP \( \mathcal{M} = (S, A, P, r, H) \), function class \( \mathcal{F}_h : S \times A \mapsto \mathbb{R}, h \in [H] \) is called policy complete iff for all \( \pi \) and \( h \in [H] \), \( Q^\pi_h \in \mathcal{F}_h \).

**Definition 3.3 (Bellman complete).** Given MDP \( \mathcal{M} = (S, A, P, r, H) \), function class \( \mathcal{F}_h : S \times A \mapsto \mathbb{R}, h \in [H] \) is called Bellman complete iff for all \( h \in [H] \) and \( Q_{h+1} \in \mathcal{F}_{h+1} \), \( T_h(Q_{h+1}) \in \mathcal{F}_h \).

### 3.1 Warmup: Realizable \( Q^* \) with deterministic transition

We start by considering the setting that transition kernel is deterministic. In this case only Assumption 2.1 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 most samples fall in the feature set \( \{ u \in \mathbb{R}^d : \| u \|_2 \leq B_\phi \} \). We next construct an estimate \( \hat{Q}_h \) of \( Q^* \) by using \( \pi_{h+1}, \ldots, \pi_H \) as the roll-out. Since the transition is deterministic, \( \hat{Q}_h(s_i^h, a_i^h) = Q^*(s_i^h, a_i^h) \) and for all the samples \( \{(s_i^h, a_i^h) : i \in [n]\} \). 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 A.4. Making using of exact recovery of \( Q^*_h \), the algorithm can thus find optimal policy \( \pi^*_h \).

Algorithm 1 Learning realizable \( Q^* \) with deterministic transition

1: for \( h = H, \ldots, 1 \) do
2: Sample \( \phi(s_i^h, a_i^h), i \in [n] \) from standard Gaussian \( N(0, I_d) \)
3: \[ \triangleright \text{ If } \| \phi(s_i^h, a_i^h) \|_2 \geq B_\phi \text{ then skip this trial.} \]
4: for \( i \in [n] \) do
5: Query \( (s_i^h, a_i^h) \) 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
\[ \hat{Q}_h^{\pi_{h+1}, \ldots, \pi_H}(s_i^h, a_i^h) = r_h^{(i)} + \cdots + r_H^{(i)} \]
7: end for
8: Recover \( v_h \) and \( W_h \) from samples \( \{ \phi(s_i^h, a_i^h), \hat{Q}_h^{\pi_{h+1}, \ldots, \pi_H}(s_i^h, a_i^h) : i \in [n]\} \)
using method of moments and gradient descent.
9: Set \( \hat{Q}_h(s, a) \leftarrow v_h^\top \sigma(W_h \phi(s, a)) \)
10: Set \( \pi_h(s) \leftarrow \arg \max_{a \in S} \hat{Q}_h(s, a) \)
11: end for
12: Return \( \pi_1, \ldots, \pi_H \)

Theorem 3.4. Suppose sample complexity \( n \geq d \cdot \text{poly}(\kappa, k, \lambda, \log d, B_W, B_\phi, H) \), then with high probability Algorithm 1 learns optimal policy \( \pi^* \).

The complete proof is deferred to the 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

\(^2\)Here with high probability is with respect to \( d \).
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.

### 3.2 Policy complete neural function approximation

Now we consider general stochastic transitions. Difficulties arise in this scenario due 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 $F_{NN}$ is policy complete with regard to MDP $\mathcal{M}$. Thus for all $\pi \in \Pi$, we can denote $Q^\pi_h(s, a) = \langle v^\pi, \sigma(W^\pi \phi(s, a)) \rangle$.

**Algorithm 2** Learn policy complete NN with simulator.

1. **Input:** Precision $\epsilon$.
2. **for** $h = H, \ldots, 1$ **do**
3. Sample $\phi(s^i_h, a^i_h), i \in [n]$ from standard Gaussian $N(0, I_d)$
4. **if** $\|\phi(s^i_h, a^i_h)\|_2 \geq B_\phi$ **then** skip this trial.
5. **for** $i \in [n]$ **do**
6. Query $(s^i_h, a^i_h)$ and use $\pi_{h+1}, \ldots, \pi_H$ as the roll-out to collect rewards $r^{(i)}_h, \ldots, r^{(i)}_H$
7. Construct unbiased estimation
   $Q^\pi_{h+1,\ldots,\pi_H}(s^i_h, a^i_h) = r^{(i)}_h + \cdots + r^{(i)}_H$
8. **end for**
9. Retrieve $v_h$ and $W_h$ from samples $\{\phi(s^i_h, a^i_h), Q^\pi_{h+1,\ldots,\pi_H}(s^i_h, a^i_h) : i \in [n]\}$
10. Set $\tilde{Q}_h(s, a) \leftarrow v^\top_h \sigma(W_h \phi(s, a))$
11. Set $\pi_h(s) \leftarrow \arg\max_{a \in S} \tilde{Q}_h(s, a)$
12. **end for**
13. **Return** $\pi_1, \ldots, \pi_H$

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 $N(0, I_d)$. The algorithm then queries $(s, a)$ and uses learned policy $\pi_{h+1}, \ldots, \pi_H$ as roll out to collect an unbiased estimate of Q-function $\tilde{Q}^\pi_{h+1,\ldots,\pi_H}(s, a)$. Recall that $Q^\pi_{h+1,\ldots,\pi_H}(s, a) \in F_{NN}$ is a two-layer neural network, it can then be recovered from estimates. Details of this step can be found in
Appendix A.4 The algorithm then finds optimal policy $\pi_h$ and proceed to the level $h - 1$.

**Theorem 3.5.** Suppose sample complexity $n \geq \epsilon^{-2} \cdot d \cdot \text{poly}(\kappa, k, \log d, B_W, B_\phi, H)$, then with high probability Algorithm 2 learns an $\epsilon$-optimal policy $\pi$.

The sample complexity does not depend on $\lambda$, unlike the case of Theorem 3.4.

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 $\pi_{h+1}, \ldots, \pi_H$ in level $h$, then from policy completeness $Q_{h+1}^\pi_{h+1}, \ldots, Q_{h}^\pi_{h}$ belongs to $\mathcal{F}_{NN}$ and we can interact with the simulator to obtain its estimate $\hat{Q}_{h}$. If $\|\hat{Q}_{h} - Q_{h+1}^\pi_{h+1}, \ldots, Q_{h}^\pi_{h}\|_\infty$ is small, the optimistic planning based on $\hat{Q}_{h}$ is not far from the optimal policy of $Q_{h+1}^\pi_{h+1}, \ldots, Q_{h}^\pi_{h}$. This reasoning can then be performed to level $h - 1$. The full proof of Theorem 3.5 is deferred to Section A.2.

### 3.3 Bellman complete neural function approximation

In addition to policy completeness, we show that neural network function approximation can also learn efficiently under the setting where 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 are $v_{Q_{h+1}}$ and $W_{Q_{h+1}}$ such that $T_{h}(Q_{h+1})(s, a) = \langle v_{Q_{h+1}}, \sigma(W_{Q_{h+1}}\phi(s, a)) \rangle$.

Algorithm 3 is similar to the algorithm in previous section. In level $h$, the algorithm has constructed the Q-function $\hat{Q}_{h}(s, a) = v_{h}^\top\sigma(W_{h}\phi(s, a))$ that is $(H - h)\epsilon/H$-close to the optimal $Q_{h+1}^*$. It then recovers weights $v_{h}, W_{h}$ from $T_{h}(\hat{Q}_{h})(s, a) = \langle v_{\hat{Q}_{h}}, \sigma(W_{\hat{Q}_{h}}\phi(s, a)) \rangle$, using unbiased estimates $r_{h}(s_t, a_t) + \hat{V}_{h+1}(s_{h+1})$ for all $i \in [n]$. Details of this step can be found in Appendix A.4. The Q-function $\hat{Q}_{h}(s, a) = v_{h}^\top\sigma(W_{h}\phi(s, a))$ constructed from weights $v_{h}, W_{h}$ is thus $(H - h + 1)\epsilon/H$-close to the optimal $Q_{h}^*$.

**Theorem 3.6.** Suppose sample complexity $n \geq \epsilon^{-2} \cdot d \cdot \text{poly}(\kappa, k, \log d, B_W, B_\phi, H)$, then with high probability Algorithm 3 learns an $\epsilon$-optimal policy $\pi$.

Similar to Theorem 3.5, the sample complexity does not explicitly depend on $\lambda$.

Due to Bellman completeness, the error of estimation $\hat{Q}_{h}$ can be controlled recursively. In fact, we can show $\|\hat{Q}_{h} - Q^*(s, a)\|_\infty$ is small by induction. The full proof of Theorem 3.6 is deferred to Section A.3.

In the above cases, the exploration is conducted in a way that guarantees an upper bound of $L_\infty$ error of learned candidate function. This can be seen where for any value function $Q$ realizable in the function class, the algorithm recovers a candidate function in this class deviating from $Q$ by at most $\epsilon$ uniformly for all state-action pairs in the domain of interest. This notion of learning guarantee has received study
Algorithm 3 Learn Bellman complete NN with simulator.

1: **Input:** Precision $\epsilon$.
2: **for** $h = H, \ldots, 1$ **do**
3:  Sample $\phi(s^i_h, a^i_h), i \in [n]$ from standard Gaussian $N(0, I_d)$
4:  \hspace{1em} $\triangleright$ If $\|\phi(s^i_h, a^i_h)\|_2 \geq B_\phi$ then skip this trial.
5: **for** $i \in [n]$ **do**
6:  Query $(s^i_h, a^i_h)$ and observe $r_h(s^i_h, a^i_h), s^i_{h+1}$
7: **end for**
8: Retrieve $v_h$ and $W_h$ from $\{\phi(s^i_h, a^i_h), r_h(s^i_h, a^i_h) + \hat{V}_{h+1}(s^i_{h+1}) : i \in [n]\}$
9: Set $\hat{Q}_h(s, a) \leftarrow v_h^\top \sigma(W_h \phi(s, a))$ and $\hat{V}_h \leftarrow \max_{a \in A} \hat{Q}_h(s, a)$
10: Set $\pi_h(s) \leftarrow \arg\max_{a \in A} \hat{Q}_h(s, a)$
11: **end for**
12: **Return** $\pi_1, \ldots, \pi_H$

in active learning [Hanneke, 2014, Krishnamurthy et al., 2017] and recently gain interest in contextual bandits [Foster et al., 2018]. In general, provably efficient algorithm can be designed on any exploration scheme that meets this $L_\infty$ recovery guarantee.

### 4 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 — the generative model setting and the online RL setting. Specifically, we show that if the image of the feature map $\phi_h(s_h, a_h)$ satisfies some positive measure conditions, then by random exploring, we can identify the optimal policy with samples linear in the algebraic dimension of the underlying polynomial class. We also provide a lower bound example showing the separation between the two settings.

Next, we introduce the notion of **Admissible Polynomial Families**, which are the families of structured polynomials that enable efficient learning.

**Definition 4.1 (Admissible Polynomial Families).** For $x \in \mathbb{R}^d$, denote $\tilde{x} = [1, x^\top]^\top$. Let $\mathcal{X} := \{\tilde{x}^{\otimes p} : x \in \mathbb{R}^d\}$. For any algebraic variety $\mathcal{V}$, we define $\mathcal{F}_\mathcal{V} := \{f_\Theta(x) = \langle \Theta, \tilde{x}^{\otimes p} \rangle : \Theta \in \mathcal{V}\}$ as the polynomial family parameterized by $\Theta \in \mathcal{V}$. We say $\mathcal{F}_\mathcal{V}$ is admissible w.r.t. $\mathcal{X}$, if for any $\Theta \in \mathcal{V}$, $\dim(\mathcal{X} \cap \{X \in \mathcal{X} : \langle X, \Theta \rangle = 0\}) < \dim(\mathcal{X})$.

\(^3\)Admissible means the dimension of $\mathcal{X}$ decreases by one when there is an additional linear constraint $\langle \Theta, X \rangle = 0$
The following theorem shows that to learn an admissible polynomial family, the sample complexity only scales with the algebraic dimension of the polynomial family.

**Theorem 4.2 (Huang et al. [2021]).** Consider the polynomial family $\mathcal{F}_V$ of dimension $D$. For $n \geq 2D$, there exists a Lebesgue-measure zero set $N \in \mathbb{R}^d \times \cdots \mathbb{R}^d$, 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 $f \in \mathcal{F}_V$.

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

**Example 4.3.** (Low-rank Polynomial of rank $k$) The function $f \in \mathcal{F}_V$ 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 4.4.** The function $f \in \mathcal{F}_V$ 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.

**Definition 4.5.** We say a measurable set $E \in \mathbb{R}^d$ is of positive measure if $\mu(E) > 0$, where $\mu$ 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 its definition is that for all admissible polynomial families, the set of $(x_1, \cdots, x_n)$ with "redundant information" about learning the parameter $\Theta$ 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 sufficient 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 $s \in S$, while under generative model setting, we only require the union set $\bigcup_{s \in S} \{\phi_h(s, a) | a \in A\}$ 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 4.6** (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 $F_{V_h}$ with dimension bounded by $D$.

**Theorem 4.7.** For the generative model setting, assume that the set $\{\phi_h(s, a) | s \in S, a \in A\}$ is of positive measure at any level $h$. Under the polynomial realizability, Algorithm 4 almost surely learns the optimal policy $\pi^*$ with at most $N = 2DH$ samples.

**Theorem 4.8.** For the online RL setting, assume that $\{\phi_h(s, a) | a \in A\}$ 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 $\pi^*$ almost surely.

**Algorithm 4** Dynamic programming under generative model settings

1. for $h = H, \cdots, 1$ do
2. 1. Sample $2D$ points $\{\phi_h(s_h^{(i)}, a_h^{(i)})\}_{i=1}^{2D}$ according to $\mathbb{P}_{x \in \mathcal{N}(0, I_d)}(x| x \in E_h)$ where $E_h = \{\phi_h(s, a) | s \in S, a \in A\}$.
3. 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 $s_h^{(i)}$ and reward $r_h^{(i)}$.
4. 3. Solve for $Q_h^*$ with the $2D$ equations $Q_h^*(s_h^{(i)}, a_h^{(i)}) = r_h^{(i)} + V_{h+1}^*(s_h^{(i)})$.
5. 4. Set $\pi^*_h(s) = \arg \max_a Q_h^*(s, a)$ and $V_h^*(s) = \max_a Q_h^*(s, a)$.
6. end for
7. Output $\pi^*$

We remark that our Theorem 4.8 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, this assumption is necessary in some sense, as is suggested by our lower bound example in the next subsection.

### 4.1 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.
Algorithm 5 Dynamic programming under online RL settings

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

MDP construction We briefly introduce the intuition of our construction. Consider a family of MDPs with only two states \( S = \{ S_{\text{good}}, S_{\text{bad}} \} \). We set the feature map \( \phi_h \) such that, for the good state \( S_{\text{good}} \), it allows the learner to explore randomly, i.e., \( \{ \phi_h(s_{\text{good}}, a) | a \in A \} \) is of positive measure.

However, for the bad state \( S_{\text{bad}} \), all actions are mapped to some restricted set, which forbids random exploration, i.e., \( \{ \phi_h(s_{\text{bad}}, a) | a \in A \} \) is measure zero. This is illustrated in Figure 1.

Specifically, at least \( \Omega(d^p) \) actions are needed to identify the ground-truth polynomial of \( Q_h^* \) for \( S_{\text{bad}} \), while \( O(d) \) actions suffice for \( S_{\text{good}} \).

The transition \( \mathbb{P}_h \) is constructed as \( \mathbb{P}_h(s_{\text{bad}}|s, a) = 1 \) for all \( s \in S, a \in A \), 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 \( \phi_h \) at \( S_{\text{good}} \) is of positive measure, while the image of \( \phi_h \) at \( S_{\text{bad}} \) is not. This makes it difficult to learn under the online RL setting.
Theorem 4.9. There exists a family of MDPs satisfying Assumption 4.6 such that the set \( \{ \phi_h(s, a) \mid s \in S, a \in A \} \) is of positive measure at any level \( h \), but for all \( h \) there is some \( s_{bad} \in S \) such that \( \{ \phi_h(s_{bad}, a) \mid a \in A \} \) is measure zero. 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.

5 Conclusions

In this paper, we consider neural network and polynomial 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 study of the stability of roots of polynomial systems with noise, which is a fundamental mathematical problem and even unsolved for homogeneous polynomials. In particular, noisy tensor decomposition approaches combined with zeroth-order optimization may allow for stochastic transitions [Huang et al. 2021].

In the online RL setting, we can only show efficient learning under a very strong 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.

Finally, in future work, we hope to consider deep neural networks where parameter recovery or \( \ell_\infty \) error is unattainable, and deep reinforcement learning with representation learning [Yang et al. 2020], [Du et al. 2020a].

Acknowledgment

JDL acknowledges support of the ARO under MURI Award W911NF-11-1-0303, the Sloan Research Fellowship, NSF CCF 2002272, and an ONR Young Investigator Award.
References


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


A Omitted Proofs in Section 3

A.1 Proof of Section 3.1

**Theorem A.1** (Formal statement of Theorem 3.4). Consider MDP $\mathcal{M}$ where the transition is deterministic. Assume the function class in Definition 3.1 satisfies Assumption 2.1. Then for all $t \in (0, 1)$ such that $d \geq \Omega(\log(B_W B_\phi))$, suppose sample complexity $n \geq d \cdot \text{poly}(\kappa, k, \lambda, \log d, B_W, B_\phi, H, \log(1/t))$, then with probability at least $1 - t$ Algorithm 1 learns the optimal policy $\pi^*$.

**Proof.** Use $\pi_1^*, \ldots, \pi_H^*$ to denote the global optimal policy. We prove that Algorithm 1 learns $\pi_h^*$ from $h = H$ to $h = 1$.

At level $H$, the query obtains exact $Q_H^*(s, a)$. Therefore by Theorem A.14, $\hat{Q}_H = Q_H^*$ and thus the optimal planning finds $\pi_H^* = \pi_H^*$. Suppose we have learned $\pi_{h+1}^*, \ldots, \pi_H^*$ at level $h$. Due to deterministic transition, the query obtains exact $Q_h^*(s, a)$. Therefore by Theorem A.14, $\hat{Q}_h = Q_h^*$ and thus the optimal planning finds $\pi_h^* = \pi_h^*$. Recursively applying this process to $h = 1$, we complete the proof. \qed

A.2 Proof of Section 3.2

**Theorem A.2** (Formal statement of Theorem 3.5). Assume the function class in Definition 3.1 satisfies Assumption 2.1 and is policy complete. Then for all $\epsilon > 0$ and $t \in (0, 1)$ such that $d \geq \Omega(\log(B_W B_\phi/\epsilon))$, suppose sample complexity $n \geq \epsilon^{-2} \cdot d \cdot \text{poly}(\kappa, k, \log d, B_W, B_\phi, H, \log(1/t))$, then with probability at least $1 - t$ Algorithm 2 learns an $\epsilon$-optimal policy $\pi$.

**Proof.** Use $\pi_1^*, \ldots, \pi_H^*$ to denote the global optimal policy. We prove for all $s \in S$

$$V_{\pi_h^*, \pi_{h+1}^*, \ldots, \pi_H^*}^H(s) - V_{\pi_h^*, \pi_{h+1}^*, \ldots, \pi_H^*}^H(s) \leq \frac{(H - h + 1)\epsilon}{H}.$$

At level $H$, let $e_H(s^i_H, a^i_H) = r(s^i_H, a^i_H) - Q_H^*(s^i_H, a^i_H)$, then $e_H(s^i_H, a^i_H) = 0$. From Theorem A.12 we have $\hat{Q}_H(s, a) := v_H^\top \sigma(W_H \phi(s, a))$ satisfies $|\hat{Q}_H(s, a) - Q_H^*(s, a)| \leq \frac{\epsilon}{2H}$ for all $s \in S, a \in A$. Therefore for all $s \in S$

$$V_H^*(s) - V_H^*(s) = \mathbb{E}_{a \sim \pi_H^*}[Q_H^*(s, a)] - \mathbb{E}_{a \sim \pi_H^*}[\hat{Q}_H(s, a)]$$

$$+ \mathbb{E}_{a \sim \pi_H^*}[\hat{Q}_H(s, a)] - \mathbb{E}_{a \sim \pi_H^*}[\hat{Q}_H(s, a)]$$

$$+ \mathbb{E}_{a \sim \pi_H^*}[\hat{Q}_H(s, a)] - \mathbb{E}_{a \sim \pi_H^*}[Q_H^*(s, a)]$$

$$\leq \frac{\epsilon}{H}$$

24
where in the second step we used \( \mathbb{E}_{a \sim \pi_H} [\hat{Q}_H(s, a)] \leq \mathbb{E}_{a \sim \pi_H} [\tilde{Q}_H(s, a)] \) by optimality of \( \pi_H \) and \(|\tilde{Q}_H(s, a) - Q_H^i(s, a)| \leq \frac{\epsilon}{2H^2} \).

Suppose we have learned policies \( \pi_h, \ldots, \pi_H \), we use \( \tilde{\pi}_h \) to denote the optimal policy of \( Q_{\pi_{h+1}, \ldots, \pi_H}^i(s, a) \). Let 
\[
\epsilon_h(s_h, a_h^i) = \hat{Q}_{\pi_{h+1}, \ldots, \pi_H}^i(s_h, a_h^i) - Q_{\pi_{h+1}, \ldots, \pi_H}^i(s_h, a_h^i)
\]
then \( \epsilon_h(s_h, a_h^i) \) is zero mean \( H^2 \) sub-Gaussian (notice that \( \hat{Q}_{\pi_{h+1}, \ldots, \pi_H}^i(s_h, a_h^i) \) is unbiased estimate of \( Q_{\pi_{h+1}, \ldots, \pi_H}^i(s_h, a_h^i) \), and \( \hat{Q}_{\pi_{h+1}, \ldots, \pi_H}^i(s_h, a_h^i) \leq H \)). From Theorem \( \text{A.12} \) we have \( \tilde{Q}_h(s, a) := \nu_h^\top (W_h \phi(s, a)) \) satisfies \(|\tilde{Q}_h(s, a) - Q_{\pi_{h+1}, \ldots, \pi_H}^i(s, a)| \leq \frac{\epsilon}{2H} \) for all \( s \in S, a \in A \). Therefore for all \( s \in S \),
\[
V_{\pi_{h+1}, \ldots, \pi_H}(s) - V_{\pi_{h+1}, \ldots, \pi_H}^i(s) = \mathbb{E}_{a \sim \tilde{\pi}_h} [\tilde{Q}_h(s, a)] - \mathbb{E}_{a \sim \tilde{\pi}_h} [\hat{Q}_h(s, a)]
\]
\[
= \mathbb{E}_{a \sim \tilde{\pi}_h} [Q_{\pi_{h+1}, \ldots, \pi_H}^i(s, a)] - \mathbb{E}_{a \sim \tilde{\pi}_h} [\tilde{Q}_h(s, a)]
\]
\[
+ \mathbb{E}_{a \sim \tilde{\pi}_h} [\hat{Q}_h(s, a)] - \mathbb{E}_{a \sim \tilde{\pi}_h} [Q_{\pi_{h+1}, \ldots, \pi_H}^i(s, a)]
\]
\[
\leq \frac{\epsilon}{H}
\]
where in the second step we used \( \mathbb{E}_{a \sim \tilde{\pi}_h} [\hat{Q}_h(s, a)] \leq \mathbb{E}_{a \sim \tilde{\pi}_h} [\tilde{Q}_h(s, a)] \) by optimality of \( \tilde{\pi}_h \) and \(|\tilde{Q}_h(s, a) - Q_{\pi_{h+1}, \ldots, \pi_H}^i(s, a)| \leq \frac{\epsilon}{2H}\).

Therefore for all \( s \in S \),
\[
V_{\pi_{h+1}, \ldots, \pi_H}(s) - V_{\pi_{h+1}, \ldots, \pi_H}^i(s) = V_{\pi_{h+1}, \ldots, \pi_H}^i(s) - V_{\pi_{h+1}, \ldots, \pi_H}(s)
\]
\[
+ V_{\pi_{h+1}, \ldots, \pi_H}^i(s) - V_{\pi_{h+1}, \ldots, \pi_H}(s)
\]
\[
+ V_{\pi_{h+1}, \ldots, \pi_H}^i(s) - V_{\pi_{h+1}, \ldots, \pi_H}(s)
\]
\[
\leq V_{\pi_{h+1}, \ldots, \pi_H}^i(s) - V_{\pi_{h+1}, \ldots, \pi_H}(s) + \frac{\epsilon}{H}
\]
\[
\leq \ldots
\]
\[
\leq \frac{(H - h + 1)\epsilon}{H}
\]
where in the second step we use \( V_{\pi_{h+1}, \ldots, \pi_H}^i(s) \leq V_{\pi_{h+1}, \ldots, \pi_H}(s) \) from optimality of \( \tilde{\pi}_h \). \( \square \)

### A.3 Proof of Section 3.3

**Theorem A.3** (Formal statement of Theorem 3.6). Assume the function class in Definition 3.1 satisfies Assumption 2.1 and is Bellman complete. Then for all
\( \epsilon > 0 \) and \( t \in (0, 1) \) such that \( d \geq \Omega(\log(B_W B_\phi / \epsilon)) \), suppose sample complexity \( n \geq \epsilon^{-2} \cdot d \cdot \text{poly}(\kappa, k, \log d, B_W, B_\phi, H, \log(1/t)) \), then with probability at least \( 1 - t \) Algorithm 3 learns an \( \epsilon \)-optimal policy \( \pi \).

**Proof.** Use \( \pi_1^*, \ldots, \pi_H^* \) to denote the global optimal policy. We prove

\[
|\hat{Q}_h(s, a) - Q_h^*(s, a)| \leq \frac{(H - h + 1)\epsilon}{H}
\]

for all \( s \in \mathcal{S}, a \in \mathcal{A} \).

At level \( H \), let

\[
e_H(s_H^i, a_H^i) = r_H(s_H^i, a_H^i) - Q_h^*(s_H^i, a_H^i)
\]

then \( e_H(s_H^i, a_H^i) = 0 \). From Theorem A.12 we have \( \hat{Q}_H(s, a) := v_H^T \sigma(W_H \phi(s, a)) \) satisfies \( |\hat{Q}_H(s, a) - Q_h^*(s, a)| \leq \frac{\epsilon}{H} \) for all \( s \in \mathcal{S}, a \in \mathcal{A} \).

Suppose we have learned \( \hat{Q}_{h+1}(s, a) \) with \( |\hat{Q}_{h+1}(s, a) - Q_{h+1}^*(s, a)| \leq \frac{(H - h)\epsilon}{H} \).

At level \( h \), let

\[
e_h(s_h^i, a_h^i) = r_h(s_h^i, a_h^i) + \hat{V}_{h+1}(s_h^i) - \mathcal{T}_h(\hat{Q}_{h+1})(s_h^i, a_h^i)
\]

then \( e_h(s_h^i, a_h^i) \) is zero mean \( H^2 \) sub-Gaussian (notice that \( r_h(s_h^i, a_h^i) + \hat{V}_{h+1}(s_h^i) \) is unbiased estimate of \( \mathcal{T}_h(\hat{Q}_{h+1})(s_h^i, a_h^i) \), and \( r_h(s_h^i, a_h^i) + \hat{V}_{h+1}(s_h^i) \leq H \)). From Theorem A.12 we have \( \hat{Q}_H(s, a) := v_H^T \sigma(W_H \phi(s, a)) \) satisfies \( |\hat{Q}_H(s, a) - \mathcal{T}_h(\hat{Q}_{h+1})(s_h^i, a_h^i)| \leq \frac{\epsilon}{H} \) for all \( s \in \mathcal{S}, a \in \mathcal{A} \). Therefore

\[
|\hat{Q}_h(s, a) - Q_h^*(s, a)| \leq |\hat{Q}_h(s, a) - \mathcal{T}_h(\hat{Q}_{h+1})(s, a)| + |\mathcal{T}_h(\hat{Q}_{h+1})(s, a) - Q_h^*(s, a)|
\]

\[
\leq \frac{\epsilon}{H} + \max_{s \in \mathcal{S}, a \in \mathcal{A}} |\hat{Q}_{h+1}(s, a) - Q_{h+1}^*(s, a)|
\]

\[
\leq \frac{(H - h + 1)\epsilon}{H}
\]

holds for all \( s \in \mathcal{S}, a \in \mathcal{A} \).

It thus follows that for all \( s_1 \in \mathcal{S} \),

\[
V_{\pi_1^* \ldots \pi_H^*}(s_1) - V_{\pi_1 \ldots \pi_H}(s_1) = E_{a_1 \sim \pi_1^*}[Q_1^*(s_1, a)] - E_{a_1 \sim \pi_1}[Q_1^{\pi_2 \ldots \pi_H}(s_1, a)]
\]

\[
\leq E_{a_1 \sim \pi_1^*}[\hat{Q}_1(s_1, a)] - E_{a_1 \sim \pi_1}[Q_1^{\pi_2 \ldots \pi_H}(s_1, a)] + \epsilon
\]

\[
\leq E_{a_1 \sim \pi_1}[\hat{Q}_1(s_1, a) - Q_1^{\pi_2 \ldots \pi_H}(s_1, a)] + \epsilon
\]

\[
\leq E_{a_1 \sim \pi_1}[Q_1^*(s_1, a) - Q_1^{\pi_2 \ldots \pi_H}(s_1, a)] + 2\epsilon
\]

\[
\leq E_{a_1 \sim \pi_1}E_{s_2 \sim P(\cdot|s, a)}[V_2^{\pi_2 \ldots \pi_H}(s_2) - V_2^{\pi_2 \ldots \pi_H}(s_2)] + 2\epsilon
\]

\[
\leq \ldots
\]

\[
\leq 2H\epsilon
\]
where the first step comes from definition of value function; the second step comes from Eq (1); the third step comes from optimality of $\pi_1$; the fourth step comes from Eq (1); the fifth step comes from Bellman equation.

A.4 Neural network recovery

This section considers recovering neural network $\langle v, \sigma(Wx) \rangle$ from the following two models:

- Noisy samples from 
  \[ x \sim \mathcal{N}^{\leq B}(0, I_d), \quad y = \langle v, \sigma(Wx) \rangle + \xi \] 
  where $\xi$ is $\vartheta$ sub-Gaussian noise.
- Noiseless samples from 
  \[ x \sim \mathcal{N}^{\leq B}(0, I_d), \quad y = \langle v, \sigma(Wx) \rangle \]

We use $\mathcal{N}^{\leq B}(0, I_d)$ to denote the following distribution, to sample from $\mathcal{N}^{\leq B}(0, I_d)$, first sampling from standard Gaussian $\mathcal{N}(0, I_d)$ and then discard the samples that $\|x\|_2^2 > B$. Recall that $B = d \cdot \text{poly log}(d)$. We consider more general homogeneous activation functions, specified by the assumption that follow. This section mainly follows Zhong et al. [2017].

Assumption A.4 (Property 3.1 of Zhong et al. [2017]). Assume $\sigma'(x)$ is nonnegative and homogeneously bounded, i.e. $0 \leq \sigma'(x) \leq L_1 |x|^p$ for some constants $L_1 > 0$ and $p \geq 0$.

Definition A.5 (Part of property 3.2 of Zhong et al. [2017]). Define $\rho(z) := \min\{\beta_0(z) - \alpha_0^2(z) - \alpha_1^2(z), \beta_2(z) - \alpha_1^2(z), \alpha_0(z) \alpha_2(z) - \alpha_2^2(z)\}$, where $\alpha_q(z) := \mathbb{E}_{x \sim \mathcal{N}(0,1)}[\sigma'(zx)x^q], q \in \{0, 1, 2\}$, and $\beta_q(z) := \mathbb{E}_{x \sim \mathcal{N}(0,1)}[(\sigma')^2(zx)x^q]$ for $q \in \{0, 2\}$.

Assumption A.6 (Part of property 3.2 of Zhong et al. [2017]). The first derivative $\sigma'(z)$ satisfies that, for all $z > 0$, we have $\rho(z) > 0$.

Assumption A.7 (Property 3.3 of Zhong et al. [2017]). The second derivative $\sigma''(x)$ is either (a) globally bounded or (b) $\sigma''(x) = 0$ except for finite points.

Notice that ReLU, squared ReLU, leaky ReLU, and polynomial activation function functions all satisfies the above assumption. We make the following assumption on the dimension of feature vectors, which corresponds to how features can extract information about neural networks from noisy samples.
**Assumption A.8** (Rich feature). Assume $d \geq \Omega(\log(\|W\|/\epsilon))$.

First we introduce a notation from [Zhong et al. 2017](#).

**Definition A.9.** Define outer product $\tilde{\otimes}$ as follows. For a vector $v \in \mathbb{R}^d$ and an identity matrix $I \in \mathbb{R}^{d \times d}$,

$$v \tilde{\otimes} I = \sum_{j=1}^{d} [v \otimes e_j \otimes e_j + e_j \otimes v \otimes e_j + e_j \otimes e_j \otimes v].$$

For a symmetric rank-$r$ matrix $M = \sum_{i=1}^{r} s_i v_i v_i^\top$ and an identity matrix $I \in \mathbb{R}^{d \times d}$,

$$M \tilde{\otimes} I = \sum_{i=1}^{r} s_i \sum_{j=1}^{d} \sum_{l=1}^{d} A_{i, i, j},$$

where $A_{1, i, j} = v_i \otimes v_i \otimes e_j \otimes e_j$, $A_{2, i, j} = v_i \otimes e_j \otimes v_i \otimes e_j$, $A_{3, i, j} = e_j \otimes v_i \otimes v_i \otimes e_j$, $A_{4, i, j} = v_i \otimes e_j \otimes e_j \otimes v_i$, $A_{5, i, j} = e_j \otimes v_i \otimes e_j \otimes v_i$, $A_{6, i, j} = e_j \otimes e_j \otimes v_i \otimes v_i$.

Now we define some moments.

**Definition A.10.** Define $M_1, M_2, M_3, M_4, m_{1,i}, m_{2,i}, m_{3,i}, m_{4,i}$ as follows:

- $M_1 := \mathbb{E}[y \cdot x]$
- $M_2 := \mathbb{E}[y \cdot (x \otimes x - I)]$
- $M_3 := \mathbb{E}[y \cdot (x \otimes^3 - x \otimes I)]$
- $M_4 := \mathbb{E}[y \cdot (x \otimes^4 - (x \otimes x) \otimes I + I \otimes I)]$
- $\gamma_j(x) := \mathbb{E}_{z \sim \mathcal{N}(0, 1)}[\sigma(x \cdot z) z^j], \forall j \in \{0, 1, 2, 3, 4\}$
- $m_{1,i} := \gamma_1(\|w_i\|)$
- $m_{2,i} := \gamma_2(\|w_i\|) - \gamma_0(\|w_i\|)$
- $m_{3,i} := \gamma_3(\|w_i\|) - 3\gamma_1(\|w_i\|)$
- $m_{4,i} := \gamma_4(\|w_i\|) + 3\gamma_0(\|w_i\|) - 6\gamma_2(\|w_i\|)$

The expectations are all with respect to $x \sim \mathcal{N}(0, I_d)$ and $y = \langle v, \sigma(Wx) \rangle$.

**Assumption A.11** (Assumption 5.3 of [Zhong et al. 2017](#)). Assume the activation function satisfies the followings:

- If $M_i \neq 0$, then $m_{j,i} \neq 0$ for all $i \in [k]$.
- At least one of $M_3$ and $M_4$ is not zero.
• If $M_1 = M_3 = 0$, then $\sigma(z)$ is an even function.
• If $M_2 = M_4 = 0$, then $\sigma(z)$ is an odd function.

Now we state the theoretical result that recovers neural networks from noisy data.

**Theorem A.12** (Neural network recovery from noisy data). Let the activation function $\sigma$ satisfies Assumption A.4 and Assumption A.11. Let $\kappa$ be the condition number of $W$. Given $n$ samples from Eq (2). For any $t \geq 1$ and $\epsilon \in (0, 1)$ such that Assumption A.8 holds, if sample complexity

$$n \geq \epsilon^{-2} \cdot d \cdot \text{poly}(t, \kappa, k, \vartheta, \log d)$$

then there exists an algorithm that takes $\tilde{O}(nkd)$ time and outputs a matrix $\hat{W} \in \mathbb{R}^{k \times d}$ and a vector $\hat{v} \in \{\pm 1\}^k$ such that with probability at least $1 - d^{-\Omega(t)}$,

$$\|\hat{W} - W\|_F \leq \epsilon \text{poly}(k, \kappa) \cdot \|W\|_F, \text{ and } \hat{v} = v.$$

The algorithm and proof are shown in Appendix A.4.1. By Assumption A.4, the following corollary is therefore straightforward.

**Corollary A.13.** In the same setting as Theorem A.12. Suppose $\|W\|_F \leq B_W$ and Assumption A.8 holds. Given $n$ samples from Eq (2). If sample complexity

$$n \geq \epsilon^{-2} \cdot d \cdot \text{poly}(t, \kappa, k, \log d, B_W, B_\phi, \vartheta)$$

then there exists an algorithm that takes $\tilde{O}(nkd)$ time and outputs a matrix $\hat{W} \in \mathbb{R}^{k \times d}$ and a vector $\hat{v} \in \{\pm 1\}^k$ such that with probability at least $1 - d^{-\Omega(t)}$, for all $\|x\|_2 \leq B_\phi$

$$|\langle \hat{v}, \sigma(\hat{W}x) \rangle - \langle v, \sigma(Wx) \rangle| \leq \epsilon.$$

In particular, when $B_\phi^2 = O(d \cdot \text{poly log } d)$ the following sample complexity suffices

$$n \geq \epsilon^{-2} \cdot d^{1+p} \cdot \text{poly}(t, \kappa, k, \log d, B_W, \vartheta).$$

Now we state the theoretical result that precisely recovers neural networks from noiseless data. The proof and method are shown in Appendix A.4.2.

**Theorem A.14** (Exact neural network recovery from noiseless data). Let the activation function satisfies Assumption A.4 and Assumption A.11. Let $\kappa$ be the condition number of $W$. Given $n$ samples from Eq (3). Suppose sample complexity

$$n \geq d \cdot \text{poly}(\kappa, t, k, \lambda, \log d)$$

then there exists an algorithm that output exact $W$ and $v$ with probability at least $1 - d^{-\Omega(t)}$.  

A.4.1 Recover neural networks from noisy data

In this section we prove Theorem A.12. Denote $W = [w_1, \ldots, w_k]^{\top}$ where $w_i \in \mathbb{R}^d$ and $\overline{w}_i = w_i / \|w_i\|_2$.

Definition A.15. Given a vector $\alpha \in \mathbb{R}^d$, Define $P_2 := M_{j_2}(I, I, \alpha, \ldots, \alpha)$ where $j_2 = \min\{j \geq 2 : M_j \neq 0\}$ and $P_3 := M_{j_3}(I, I, \alpha, \ldots, \alpha)$ where $j_3 = \min\{j \geq 3 : M_j \neq 0\}$.

The method of moments is presented in Algorithm 6. There are three main steps. First it computes the span of the rows of $W$. By power method, Line 7 finds the top-$k$ eigenvalues of $CI + \hat{P}_2$ and $CI - \hat{P}_2$. It then picks the largest $k$ eigenvalues from $CI + \hat{P}_2$ and $CI - \hat{P}_2$, by invoking TopK in Line 15. Finally it orthogonalizes the corresponding eigenvectors in Line 19 and finds an orthogonal matrix $V$ in the subspace spanned by $\{\overline{w}_1, \ldots, \overline{w}_k\}$.

In the second step, the algorithm forms third order tensor $R_3 = P_5(V, V, V) \in \mathbb{R}^{k \times k \times k}$ and use tensor decomposition in \cite{Kuleshov15} to find $\hat{u}$ which estimates $s_iV^\top \overline{w}_i$ with unknown signs. In the third step the algorithm recovers $s, v$ and $w_i, i \in [k]$. Due to homogeneous activation function, we assume $v_i \in \{\pm 1\}$ and $m_{j,i} = c_j\|w_i\|^{p+1}$ for universal constants $c_j$. We define $Q_1$ and $Q_2$ as follows.

$$Q_1 = M_{l_1}(I, \alpha, \ldots, \alpha) = \sum_{(l_1 - 1) \alpha} v_i c_i \|w_i\|^{p+1} (\alpha^\top \overline{w}_i)^{l_1-1} \overline{w}_i,$$  \tag{4}$$

$$Q_2 = M_{l_2}(V, V, \alpha, \ldots, \alpha) = \sum_{(l_2 - 2) \alpha} v_i c_i \|w_i\|^{p+1} (\alpha^\top \overline{w}_i)^{l_2-2} (V^\top \overline{w}_i) (V^\top \overline{w}_i)^\top,$$  \tag{5}$$

where $l_1 \geq 1$ such that $M_{l_1} \neq 0$ and $l_2 \geq 2$ such that $M_{l_2} \neq 0$ are specified later. Then consider the following linear system.

$$z^* = \arg \min_{z \in \mathbb{R}^k} \left\| \sum_{i=1}^k z_i s_i \overline{w}_i - Q_1 \right\|,$$

$$r^* = \arg \min_{r \in \mathbb{R}^k} \left\| \sum_{i=1}^k r_i V^\top \overline{w}_i (V^\top \overline{w}_i)^\top - Q_2 \right\|_F.$$  \tag{6}$$

The solutions of the above linear systems are

$$z_i^* = v_i s_i^{l_1} c_i \|w_i\|^{p+1} (\alpha^\top s_i \overline{w}_i)^{l_1-1},$$

$$r_i = v_i s_i^{l_2} c_i \|w_i\|^{p+1} (\alpha^\top s_i \overline{w}_i)^{l_2-2}.$$  

Since $\hat{u}$ estimates $s_i V^\top \overline{w}_i$, we approximate $s_i \overline{w}_i$ by $V \hat{u}_i$ and approximate $Q_1$ and $Q_2$ by their empirical versions $\hat{Q}_1$ and $\hat{Q}_2$ in Line 28. Then we can solve the following
Suppose \( m \), then with probability at least \( 1 \),

\[
\hat{z} = \arg \min_{z \in \mathbb{R}^k} \left\| \sum_{i=1}^{k} z_i V \hat{u}_i - \hat{Q}_1 \right\|, \quad \text{and} \quad \hat{r} = \arg \min_{r \in \mathbb{R}^k} \left\| \sum_{i=1}^{k} r_i \hat{u}_i \hat{u}_i^\top - \hat{Q}_2 \right\|.
\]

In this position we can estimate the magnitude \( \| w_i \| \) by \((\| \hat{z}_i/(c_j \alpha \top V \hat{u}_i)\|^{1-\epsilon})^{1/(p+1)}\) and recover \( v_i \) and \( s_i \) by the standard procedures in [Zhong et al. 2017]:

1. If \( M_1 = M_3 = 0 \), we choose \( l_1 = l_2 = \min\{j \in \{2, 4\} | M_j \neq 0\} \). Return \( v_i^{(0)} = \text{sign}(\hat{r}_i c_{l_2}) \) and \( s_i^{(0)} \) being \(-1\) or \(1\).

2. If \( M_2 = M_4 = 0 \), we choose \( l_1 = \min\{j \in \{1, 3\} | M_j \neq 0\} \), \( l_2 = 3 \). Return \( v_i^{(0)} \) being \(-1\) or \(1\) and \( s_i^{(0)} = \text{sign}(v_i^{(0)} \hat{z}_i c_{l_1}) \).

3. Otherwise, we choose \( l_1 = \min\{j \in \{1, 3\} | M_j \neq 0\} \), \( l_2 = \min\{j \in \{2, 4\} | M_j \neq 0\} \). Return \( v_i^{(0)} = \text{sign}(\hat{r}_i c_{l_2}) \) and \( s_i^{(0)} = \text{sign}(v_i^{(0)} \hat{z}_i c_{l_1}) \).

Since Algorithm 6 carries out the same computation as [Zhong et al. 2017], the time complexity is the same. The difference of sample complexity comes from the noise \( \xi \) in the model and the truncation of standard Gaussian. The proof entails bounding the error in estimating \( P_2 \) in Line 4, \( R_3 \) in Line 20 and \( Q_1, Q_2 \) in Line 28. In the following, unless further specified, the expectations are all with respect to \( x \sim \mathcal{N}(0, I_d) \) and \( y = \langle v, \sigma(Wx) \rangle \).

**Lemma A.16.** Let \( \hat{P}_2 \) be computed in Line 4 of Algorithm 6 and \( P_2 \) defined in Definition A.15. Suppose \( m_0 = \min_{i \in [k]} \{ |m_{j_2,i}|^2 (\bar{w}_i \alpha)^{2(j_2 - 2)} \} \) and

\[
|S| \gtrsim \bar{d} \cdot \text{poly}(\kappa, t, \vartheta, \log d) / (\epsilon^2 m_0)
\]

then with probability at least \( 1 - d^{-\Omega(t)} \),

\[
\| P_2 - \hat{P}_2 \| \leq \epsilon \sum_{i=1}^{k} |v_i m_{j_2,i} (\bar{w}_i \alpha)^{j_2 - 2} |.
\]

**Proof.** It suffices to bound \( \| M_2 - \hat{M}_2 \|, \| M_3(I, I, \alpha) - \hat{M}_3(I, I, \alpha) \| \) and \( \| M_4(I, I, \alpha) - \hat{M}_4(I, I, \alpha, \alpha) \| \).

Specifically, we show that with probability at least \( 1 - d^{-t} \),

\[
\| M_2 - \hat{M}_2 \| \leq \epsilon \sum_{i=1}^{k} |v_i m_{2,i}|.
\]
where the first step comes from Assumption A.4 and the second step comes from Claim D.2 and Claim D.3.

Define \( B = \frac{1}{\alpha} \) independent of \( B \) using Claim D.1. For any constant \( \alpha \),

\[
\| M_3(I, I, \alpha) - \hat{M}_3(I, I, \alpha) \| \leq \epsilon \sum_{i=1}^{k} |v_i m_{3,i}(\bar{w}_i^T \alpha)|. \tag{9}
\]

\[
\| M_4(I, I, \alpha, \alpha) - \hat{M}_4(I, I, \alpha, \alpha) \| \leq \epsilon \sum_{i=1}^{k} |v_i m_{4,i}(\bar{w}_i^T \alpha)^2|. \tag{10}
\]

Recall that for sample \( (x_j, y_j) \in S, y_j = \sum_{i=1}^{k} v_i \sigma(w_i^T x_j) + \xi_j \) where \( \xi_j \) is independent of \( x_j \). Consider each component \( i \in [k] \). Define \( C_i(x_j), B_i(x_j) \in \mathbb{R}^{d \times d} \) as follows:

\[
C_i(x_j) = \mathbb{I}_{\|x_j\| \leq B}(\sigma(w_i^T x_j) + \xi_j) \cdot (x_j^{\otimes 4} - (x_j \otimes x_j) \otimes I + I \otimes I)(I, I, \alpha, \alpha)
\]

\[
= \mathbb{I}_{\|x_j\| \leq B}(\sigma(w_i^T x_j) + \xi_j) \cdot ((x^T \alpha)^2 x^{\otimes 2} - (\alpha^T x)^2 I - 2(\alpha^T x)(x \alpha^T + \alpha x^T) - xx^T + 2 \alpha^T + I).
\]

Define \( B_i(x_j) = (\sigma(w_i^T x_j) + \xi_j) \cdot ((x^T \alpha)^2 x^{\otimes 2} - (\alpha^T x)^2 I - 2(\alpha^T x)(x \alpha^T + \alpha x^T) - xx^T + 2 \alpha^T + I) \in \mathbb{R}^{d \times d} \). Then from Claim D.5 we have \( \mathbb{E}[B_i(x_j)] = m_{i,j}(\bar{w}_i^T \alpha)^2 \bar{w}_i \).

By Assumption A.4

\[
\sigma(w_i^T x_j) \cdot (x_j^{\otimes 4} - (x_j \otimes x_j) \otimes I + I \otimes I)(I, I, \alpha, \alpha)
\]

\[
\preceq |w_i^T x_j|^{p+1} + |\phi(0)| \cdot ((x_j^T \alpha)^2 \|x_j\|^2 + 1 + \|x_j\|^2 + (\alpha^T x_j)^2)
\]

\[
\leq |w_i|^{p+1} \cdot |x_j|^{p+5},
\]

using Claim D.1 and \( B \geq d \cdot \text{poly log}(d) \) we have

\[
\| \mathbb{E}[C_i(x_j)] - m_{i,j}(\bar{w}_i^T \alpha)^2 \bar{w}_i \bar{w}_i^T \| \leq \mathbb{E}[\mathbb{I}_{\|x_j\| \geq B}|w_i|^{p+1} \cdot |x_j|^{p+5}]
\]

\[
\leq (\|w_i\| d)^{p+5} \cdot e^{-\Omega(d \log d)}
\]

\[
\leq \epsilon.
\]

Also, \( \frac{1}{2} m_{i,j}(\bar{w}_i^T \alpha)^2 \leq \| \mathbb{E}[C_i(x_j)] \| \leq 2 m_{i,j}(\bar{w}_i^T \alpha)^2 \).

For any constant \( t \geq 1 \), we have with probability \( 1 - 1/(d^t) \),

\[
\| C_i(x) \| \preceq (|w_i^T x_j|^{p+1} + |\phi(0)| + |\xi_j|) \cdot ((x_j^T \alpha)^2 \|x_j\|^2 + 1 + \|x_j\|^2 + (\alpha^T x_j)^2)
\]

\[
\preceq (\|w_i\|^{p+1} + |\phi(0)| + \|d\| \text{poly(log } d, t) \]

where the first step comes from Assumption A.4 and the second step comes from Claim D.2 and Claim D.3.

32
Using Claim D.4, we have
\[
\|\mathbb{E}[C_i(x)]\| \lesssim \left(\mathbb{E}[\phi^4(x_j)]\right)^{1/2} \left(\mathbb{E}[\|x_j\|^4]\right)^{1/2} \lesssim (\|w_i\|^{p+1} + |\phi(0)| + \vartheta)^2d.
\]

Furthermore we have,
\[
\max_{\|a\|=1} \left(\mathbb{E} \left[ (a^T C_i(x_j)a)^2 \right] \right)^{1/2} \lesssim \left(\mathbb{E} \left[ (\phi^4(w_i^T x_j + \xi_j)]\right)\right)^{1/4} \lesssim \|w_i\|^{p+1} + |\phi(0)| + \vartheta.
\]

Then by Claim D.6 with probability at least \(1 - d^{-t}\),
\[
\left\| m_{4,i}(\overline{w}_i^T \alpha)^2\overline{w}_i\overline{w}_i^T - \frac{1}{|S|} \sum_{x_j \in S} C_i(x_j) \right\| \\
\leq \left\| m_{4,i}(\overline{w}_i^T \alpha)^2\overline{w}_i\overline{w}_i^T - \mathbb{E}[C_i(x_j)] \right\| + \left\| \mathbb{E}[C_i(x_j)] - \frac{1}{|S|} \sum_{x_j \in S} C_i(x_j) \right\| \\
\leq \epsilon |m_{4,i}|(\overline{w}_i^T \alpha)^2.
\]

Summing up all components \(i \in [k]\), we proved Eq (10). Eq (8) and Eq (9) can be shown similarly.

Lemma A.17. Let \(V \in \mathbb{R}^{d \times k}\) be an orthogonal matrix. Let \(\hat{R}_3\) be computed in Line 20 of Algorithm 6 and \(R_3 = P_3(V, V, V)\). Suppose
\[
m_0 = \min_{i \in [k]} \{|m_{j3,i}|^2(\overline{w}_i^T \alpha)^2(j_3-3)\}
\]
and
\[
|S| \gtrsim d \cdot \text{poly}(\kappa, t, \vartheta, \log d)/(\epsilon^2 m_0)
\]
then with probability at least \(1 - d^{-\Omega(t)}\),
\[
\|R_3 - \hat{R}_3\| \leq \epsilon \sum_{i=1}^k |v_i m_{j3,i}(\overline{w}_i^T \alpha)^{j_3-3}|.
\]

Proof. From the definition of \(R_3\), it suffices to bound \(\|M_3(V, V, V) - \hat{M}_3(V, V, V)\|\) and \(\|M_4(V, V, V, \alpha) - \hat{M}_4(V, V, V, \alpha)\|\).
Specifically, we show that with probability at least $1 - d^{-t}$,

$$\|M_3(V, V, V) - \tilde{M}_3(V, V, V)\| \leq \epsilon \sum_{i=1}^{k} |v_i m_{3,i}|. \quad (11)$$

$$\|M_4(V, V, V, \alpha) - \tilde{M}_4(V, V, V, \alpha)\| \leq \epsilon \sum_{i=1}^{k} |v_i m_{4,i}(w_i^T \alpha)|. \quad (12)$$

Recall that for sample $(x_j, y_j) \in S$, $y_j = \sum_{i=1}^{k} v_i \sigma(w_i^T x_j) + \xi_j$ where $\xi_j$ is independent of $x_j$. Consider each component $i \in [k]$. Define $T_i(x_j), S_i(x_j) \in \mathbb{R}^{k \times k \times k}$:

$$T_i(x_j) = (\sigma(w_i^T x_j) + \xi_j) \cdot (x_i^T \alpha \cdot v(x)^{\otimes 3} - (V^T \alpha) \otimes (v(x) \otimes v(x)) - \alpha^T x \cdot v(x) \otimes I + (V^T \alpha) \otimes I),$$

$$S_i(x_j) = \mathbb{1}_{\|x_j\| \leq B} T_i(x_j)$$

where $v(x) = V^T x$. Flatten $T_i(x_j)$ along the first dimension to obtain $B_i(x_j) \in \mathbb{R}^{k \times k^2}$, flatten $S_i(x_j)$ along the first dimension to obtain $C_i(x_j) \in \mathbb{R}^{k \times k^2}$.

From Claim D.7, $E[B_i(x_j)] = m_{4,i}(\alpha^T \overline{w}_i)(V^T \overline{w}_i)\operatorname{vec}((V^T \overline{w}_i)(V^T \overline{w}_i)^T)^T$. Therefore we have,

$$\|E[B_i(x)]\| = |m_{4,i}(\alpha^T \overline{w}_i)| \cdot \|V^T \overline{w}_i\|^3.$$ 

By Assumption A.4

$$\|B_i(x)\| \leq ((w_i^T x_j)^{p+1} + |\phi(0)| + |\xi_j|) \cdot ((x_j^T \alpha)^2 \|V^T x_j\|^3$$

$$+ 3\|V^T x_j\|^2 + 3|x_j^T \alpha||V^T x_j\| \sqrt{k} + 3|V^T \alpha| \sqrt{k}) \leq \sqrt{k} \cdot \|w_i\|^{p+1} \cdot \|x_j\|^{p+6},$$

using Claim D.1 and $B \geq d \cdot \text{poly} \log(d)$,

$$\|E[C_i(x_j)] - m_{4,i}(\alpha^T \overline{w}_i)(V^T \overline{w}_i)\operatorname{vec}((V^T \overline{w}_i)(V^T \overline{w}_i)^T)^T\|$$

$$\leq \|E[\mathbb{1}_{\|x_j\| \leq B} \sqrt{k} \cdot \|w_i\|^{p+1} \cdot \|x_j\|^{p+6}]$$

$$\leq \epsilon.$$ 

For any constant $t \geq 1$, we have with probability $1 - d^{-\Omega(t)}$,

$$\|C_i(x)\| \leq ((w_i^T x_j)^{p+1} + |\phi(0)| + |\xi_j|) \cdot ((x_j^T \alpha)^2 \|V^T x_j\|^3$$

$$+ 3\|V^T x_j\|^2 + 3|x_j^T \alpha||V^T x_j\| \sqrt{k} + 3|V^T \alpha| \sqrt{k}) \leq (\|w_i\|^{p+1} + |\phi(0)| + \vartheta)k^{3/2}\text{poly}(\log d, t)$$

34
where the first step comes from Assumption A.4 and the second step comes from Claim D.2 and Claim D.3.

Using Claim D.4, we have

\[ \|E[C_i(x)C_i(x)^\top]\| \lesssim \left( E\left[ (\phi(w_i^\top x_j + \xi_j)^4 \right] \right)^{1/2} \left( E\left[ (\alpha^\top x_j)^4 \right] \right)^{1/2} \left( E\left[ \|V^\top x_j\|^6 \right] \right)^{1/2} \]

\[ \lesssim (\|w_i\|^{p+1} + |\phi(0)| + \vartheta)^2 k^{3/2}. \]

ans

\[ \|E[C_i(x)C_i(x)^\top]\| \]

\[ \lesssim \left( E\left[ (\phi(w_i^\top x_j + \xi_j)^4 \right] \right)^{1/2} \left( E\left[ (\alpha^\top x_j)^4 \right] \right)^{1/2} \left( E\left[ ||V^\top x_j\||^4 \right] \right)^{1/2} \]

\[ \cdot \max_{\|A\|_F=1} E\left[ \langle A, (V^\top x_j)(V^\top x_j)^\top \rangle^4 \right] \]

\[ \lesssim (\|w_i\|^{p+1} + |\phi(0)| + \vartheta)^2 k^2. \]

Furthermore we have,

\[ \max_{\|a\| = \|b\| = 1} \left( E\left[ (a^\top C_i(x_j)b)^2 \right] \right)^{1/2} \]

\[ \lesssim \left( E\left[ (\phi(w_i^\top x_j + \xi_j)^4 \right] \right)^{1/4} \left( E\left[ (\alpha^\top x_j)^4 \right] \right)^{1/4} \max_{\|a\|=1} \left( E\left[ (a^\top V^\top x_j)^4 \right] \right)^{1/2} \]

\[ \cdot \max_{\|A\|_F=1} \left( E\left[ \langle A, (V^\top x_j)(V^\top x_j)^\top \rangle^4 \right] \right)^{1/2} \]

\[ \lesssim (\|w_i\|^{p+1} + |\phi(0)| + \vartheta)k. \]

Then by Claim D.6 with probability at least $1 - d^{-t}$,

\[ \left\| m_{4,i}(\alpha^\top \varpi_i)(V^\top \varpi_i)\text{vec}((V^\top \varpi_i)(V^\top \varpi_i)^\top)^\top - \frac{1}{|S|} \sum_{x_j \in S} C_i(x_j) \right\| \]

\[ \leq \left\| m_{4,i}(\alpha^\top \varpi_i)(V^\top \varpi_i)\text{vec}((V^\top \varpi_i)(V^\top \varpi_i)^\top)^\top - \mathbb{E}[C_i(x_j)] \right\| \]

\[ + \left\| \mathbb{E}[C_i(x_j)] - \frac{1}{|S|} \sum_{x_j \in S} C_i(x_j) \right\| \]

\[ \leq \epsilon |v_i m_{4,i}(\varpi_i^\top \alpha)|. \]

Summing up all neurons $i \in [k]$, we proved Eq (12). Eq (11) can be shown similarly.
Lemma A.18. Let \( \hat{Q}_1 \) and \( \hat{Q}_2 \) be computed in Line 28 of Algorithm 6. Let \( Q_1 \) be defined by Eq 4 and \( Q_2 \) be defined by Eq 5. Suppose

\[
m_0 = \min_{i \in [k]} \{|m_{j_1,i}|^2 (\overline{w}_i^\top \alpha)^{2(j_1-1)}, |m_{j_2,i}|^2 (\overline{w}_i^\top \alpha)^{2(j_2-2)}\}
\]

and

\[
|S| \gtrsim d \cdot \text{poly}(\kappa, t, \vartheta, \log d)/(\epsilon^2 m_0)
\]

then with probability at least \( 1 - d^{-\Omega(t)} \),

\[
\|Q_1 - \hat{Q}_1\| \leq \epsilon \sum_{i=1}^k |v_i m_{j_1,i} (\overline{w}_i^\top \alpha)^{j_1-1}|,
\]

\[
\|Q_2 - \hat{Q}_2\| \leq \epsilon \sum_{i=1}^k |v_i m_{j_2,i} (\overline{w}_i^\top \alpha)^{j_2-2}|.
\]

Proof. Recall the expression of \( Q_1 \) and \( Q_2 \),

\[
Q_1 = M_{j_1}(I, \alpha, \ldots, \alpha) = \sum_{i=1}^k v_i c_{j_1} \|w_i\|^{p+1} (\alpha^\top \overline{w}_i)^{j_1-1} \overline{w}_i,
\]

\[
Q_2 = M_{j_2}(V, V, \alpha, \ldots, \alpha) = \sum_{i=1}^k v_i c_{j_2} \|w_i\|^{p+1} (\alpha^\top \overline{w}_i)^{j_2-2}(V^\top \overline{w}_i)(V^\top \overline{w}_i)^\top.
\]

The proof is essentially similar to Lemma A.16 and Lemma A.17.

We also use the following Lemmata from Zhong et al. [2017].

Lemma A.19 (Lemma E.6 of Zhong et al. [2017]). Let \( P_2 \) be defined as in Definition A.15 and \( \hat{P}_2 \) be its empirical version calculated in Line 4 of Algorithm 6. Let \( U \in \mathbb{R}^{d \times k} \) be the orthogonal column span of \( W \in \mathbb{R}^{d \times k} \). Assume \( \|\hat{P}_2 - P_2\| \leq s_k(P_2)/10 \). Let \( C \) be a large enough positive number such that \( C > 2\|P_2\| \). Then after \( T = O(\log(1/\epsilon)) \) iterations, the \( V \in \mathbb{R}^{d \times k} \) computed in Algorithm 6 will satisfy

\[
\|UU^\top - VV^\top\| \lesssim \|\hat{P}_2 - P_2\|/s_k(P_2) + \epsilon,
\]

which implies

\[
\|(I - VV^\top)w_i\| \lesssim (\|\hat{P}_2 - P_2\|/s_k(P_2) + \epsilon)\|w_i\|.
\]
Lemma A.20 (Lemma E.13 in Zhong et al. [2017]). Let $U \in \mathbb{R}^{d \times k}$ be the orthogonal column span of $W^*$. Let $V \in \mathbb{R}^{d \times k}$ denote an orthogonal matrix satisfying that $\|VV^T - UU^T\| \leq \delta_2 \leq 1/(\kappa^2 \sqrt{k})$. For each $i \in [k]$, let $\hat{u}_i$ denote the vector satisfying $\|\hat{u}_i - V^T w_i\| \leq \delta_3 \leq 1/(\kappa \sqrt{k})$. Let $Q_1$ be defined as in Eq. (4) and $\hat{Q}_1$ be the empirical version of $Q_1$ such that $\|Q_1 - \hat{Q}_1\| \leq \delta_4 \|Q_1\| \leq \frac{1}{2} \|Q_1\|$.

Let $z^* \in \mathbb{R}^k$ and $\tilde{z} \in \mathbb{R}^k$ be defined as in Eq. (6) and Eq. (7). Then

$$\|\tilde{z}_i - z_i^*\| \leq (\kappa^4 k^{3/2} \delta_2 + \delta_3) \|z^*\|_1.$$ 

Lemma A.21 (Lemma E.14 in Zhong et al. [2017]). Let $U \in \mathbb{R}^{d \times k}$ be the orthogonal column span of $W^*$. Let $V \in \mathbb{R}^{d \times k}$ denote an orthogonal matrix satisfying that $\|VV^T - UU^T\| \leq \delta_2 \leq 1/(\kappa \sqrt{k})$. For each $i \in [k]$, let $\hat{u}_i$ denote the vector satisfying $\|\hat{u}_i - V^T w_i\| \leq \delta_3 \leq 1/(\sqrt{k} \kappa^3)$.

Let $Q_2$ be defined as in Eq. (5) and $\hat{Q}_2$ be the estimation of $Q_2$ such that $\|Q_2 - \hat{Q}_2\|_F \leq \delta_4 \|Q_2\|_F \leq \frac{1}{2} \|Q_2\|_F$. Let $r^* \in \mathbb{R}^k$ and $\hat{r} \in \mathbb{R}^k$ be defined as in Eq. (6) and Eq. (7). Then

$$\|\hat{r}_i - r_i^*\| \leq (k^3 \kappa^8 \delta_3 + \kappa^2 k^2 \delta_4) \|r^*\|.$$ 

Now we are in the position of proving Theorem A.12.

Proof. Consider Algorithm 6. Using triangle inequality and $V^TV = I$ we have the following,

$$\|\tilde{w}_i - s_i V \hat{u}_i\| \leq \|VV^T \tilde{w}_i - \tilde{w}_i\| + \|VV^T \tilde{w}_i - V s_i \hat{u}_i\|$$

$$= \|VV^T \tilde{w}_i - \tilde{w}_i\| + \|V^T \tilde{w}_i - s_i \hat{u}_i\|. \quad (13)$$

For $\|VV^T \tilde{w}_i - \tilde{w}_i\|$, we have

$$\|VV^T \tilde{w}_i - \tilde{w}_i\| \leq (\|\hat{P}_2 - P_2\|/s_k(P_2) + \epsilon)$$

$$\leq (\text{poly}(k, \kappa) \|\hat{P}_2 - P_2\| + \epsilon)$$

$$\leq \text{poly}(k, \kappa) \epsilon, \quad (14)$$

where the first step comes from Lemma A.19, the second step comes from $s_k(P_2) \geq 1/\text{poly}(k, \kappa)$, and the last step comes from $\|\hat{P}_2 - P_2\| \leq \epsilon \text{poly}(k, \kappa)$ if the number of samples is proportional to $\tilde{O}(d/\epsilon^2)$ as shown in Lemma A.16.

For $\|V^T \tilde{w}_i - s_i \hat{u}_i\|$, we have

$$\|V^T \tilde{w}_i - s_i \hat{u}_i\| \leq \text{poly}(k, \kappa) \|\hat{R}_3 - R_3\| \leq \epsilon \text{poly}(k, \kappa), \quad (15)$$

where the first step comes from Theorem 3 in Kuleshov et al. [2015] and the last step comes from Lemma A.17.
Plugging (14) and (15) into Eq. (13), we have
\[ \|\bar{w}_i - s_i V \hat{u}_i\| \leq \epsilon \text{poly}(k, \kappa). \] (16)

Next we bound the error in $\hat{r}$ and $\hat{z}$. We have,
\[ |\hat{r}_i - r^*_i| \leq \text{poly}(k, \kappa) \max\{\|Q_1 - \hat{Q}_1\|, \|Q_2 - \hat{Q}_2\|\} \cdot \|r^*\| \leq \epsilon \text{poly}(k, \kappa) \cdot \|r^*\|. \] (17)

where the first step comes from Lemma A.20 and the second step comes from Lemma A.18. Furthermore,
\[ |\hat{z}_i - z^*_i| \leq \text{poly}(k, \kappa) \max\{\|Q_1 - \hat{Q}_1\|, \|Q_2 - \hat{Q}_2\|\} \cdot \|z^*\|_1 \leq \epsilon \text{poly}(k, \kappa) \|z^*\|_1, \] (18)

where the first step comes from Lemma A.21 and the second step comes from Lemma A.18. Combining Eq (17), Eq (18) and Eq (16), the output in Line 32 satisfies
\[ \|w_i^{(0)} - w_i\|_F \leq \epsilon \text{poly}(k, \kappa) \cdot \|w_i\|_F. \] Since $v_i$ are discrete values, they are exactly recovered.

A.4.2 Exact recovery of neural networks from noiseless data

In this section we prove Theorem A.14. Similar to Appendix A.4.1, denote $W = [w_1, \cdots, w_k]^\top$ where $w_i \in \mathbb{R}^d$ and $\hat{W} = [\hat{w}_1, \cdots, \hat{w}_k]^\top$. We define the empirical loss and the population loss as follows,
\[ L_n(\hat{W}) = \frac{1}{2n} \sum_{i=1}^n \left( \sum_{i=1}^k v_i \sigma(\hat{w}_i^\top x_i) - y_i \right)^2, \] (19)
\[ L(\hat{W}) = \frac{1}{2} \mathbb{E} \left[ \left( \sum_{i=1}^k v_i \sigma(\hat{w}_i^\top x) - y \right)^2 \right]. \] (20)

**Definition A.22.** Let $s_i$ be the $i$-th singular value of $W$, $\lambda := \prod_{i=1}^k (s_i/s_k)$. Let $\tau = (3s_1/2)^{4p}/\min_{z \in [s_k/2, 3s_1/2]} \{\rho^2(z)\}$.

We use the follow results adapted from Zhong et al. [2017]. The only difference is that we use truncated Gaussian $N^{\leq B}(0, I_d)$, and due to $B = d \text{poly} \log(d)$ we can bound its difference between standard Gaussian in the same way as Appendix A.4.1.

**Lemma A.23 (Concentration, adapted from Lemma D.11 in Zhong et al. [2017]).** Let samples size $n \geq \epsilon^{-2} d \text{poly}(\log d, t)$, then with probability at least $1 - d^{-\Omega(t)}$,
\[ \|\nabla^2 L_n(W) - \nabla^2 L(W)\| \lesssim k s_1^{2p} \epsilon \] (21)
Lemma A.24 (Adapted from Lemma D.16 in [Zhong et al., 2017]). Let \( \{x_1, \ldots, x_n\} \) denote a set of samples from Distribution \( D \). Assume activation \( \sigma(\cdot) \) satisfies Assumption A.7 and Assumption A.6. Then for any \( t \geq 1 \), if \( n \geq d^\text{poly}(\log d, t) \), with probability at least \( 1 - d^{-t} \), for any \( \hat{W} \) (which is not necessarily to be independent of samples) satisfying \( \|W - \hat{W}\| \leq s_k/4 \), we have

\[
\|\nabla^2 L_n(\hat{W}) - \nabla^2 L_n(W)\| \leq ks_k^p \|W - \hat{W}\| d^{(p+1)/2}.
\]

Now we prove Theorem A.14.

**Proof.** The exact recovery consists of first finding (exact) \( v \) and (approximate) \( \hat{W} \) close enough to \( W \) by tensor method (Appendix A.4.1), and then minimizing the empirical loss \( L_n(\cdot) \). We will prove that \( L_n(\cdot) \) is locally strongly convex, thus we find the precise \( W \).

From Lemma D.3 from [Zhong et al., 2017] we know:

\[
\Omega(\rho(s_k)/\lambda) I \preceq \nabla^2 L(W) \preceq O(ks_k^2p) I.
\] (22)

From Lemma A.23, we need \( n \geq \frac{k^3\lambda^2s_k^4p}{\rho^2(s_k)} d^\text{poly}(\log d, t) \) for \( \nabla^2 L_n(W) \) to be positive definite.

Next we uniformly bound Lipschitzness of \( \nabla^2 L_n \). From Lemma A.24 there exists a universal constant \( c \), for all \( \hat{W} \) that satisfies \( \|W - \hat{W}\| \leq cks_k^2p/(ks_k^2d^{(p+1)/2}) = cs_k^p d^{-(p+1)/2} \), \( \nabla L_n^2(\hat{W}) \gtrsim ks_k^2p \) uniformly. So there is a unique minimizer of \( L_n \) in this region.

Notice \( L_n(W) = 0 \), therefore we can find \( W \) by directly minimizing the empirical loss as long as we find any \( \hat{W} \) in this region. This can be achieved by tensor method in Appendix A.4.1. We thus complete the proof.

**B Omitted Proofs in Section 4**

For the proofs of Theorem 4.2, Example 4.3, and Example 4.4, we refer the readers to [Huang et al., 2021].

**Lemma B.1.** Consider the polynomial family \( \mathcal{F}_V \) of dimension \( D \). Assume that \( n > 2D \). For any \( E \in \mathbb{R}^d \) that is of positive measure, by sampling \( n \) samples \( \{x_i\} \) i.i.d. from \( \mathbb{P}_{x \in \mathcal{N}(0, I_d)}(\cdot | x \in E) \) and observing the noiseless feedbacks \( y_i = f^*(x_i) \), one can almost surely uniquely determine the \( f^* \) by solving the system of equations \( y_i = f(x_i), i = 1, \ldots, n, \) for \( f \in \mathcal{F}_V \).
Proof. By Theorem 4.2, there exists a set \( N \in \mathbb{R}^d \times \cdots \times \mathbb{R}^d \) of Lebesgue measure zero, such that if \((x_1, \ldots, x_n) \notin N\), one can uniquely determine the \( f^* \) by the observations on the \( n \) samples. Therefore, we only need to show that with probability 1, the sampling procedure returns \((x_1, \ldots, x_n) \notin N\). This is because
\[
\mathbb{P}(x_1, \ldots, x_n \in N) = \frac{\mathbb{P}_{x_i \in N(0, I_d)}((x_1, \ldots, x_n) \in N \cap (E \times \cdots \times E))}{\mathbb{P}_{x_i \in N(0, I_d)}((x_1, \ldots, x_n) \in (E \times \cdots \times E))} = 0.
\]

By Lemma B.1 above, it is not hard to see that Algorithms 4 and 5 work.

C Omitted Constructions and Proofs in Subsection 4.1

Construction of the Reward Functions The following construction of the polynomial hard case is adopted from [Huang et al., 2021].

Let \( d \) be the dimension of the feature space. Let \( e_i \) denotes the \( i \)-th standard orthonormal basis of \( \mathbb{R}^d \), i.e., \( e_i \) has only one 1 at the \( i \)-th entry and 0’s for other entries. Let \( p \) denote the highest order of the polynomial. We assume \( d \gg p \). We use \( \Lambda \) to denote a subset of the \( p \)-th multi-indices
\[
\Lambda = \{(\alpha_1, \ldots, \alpha_p) | 1 \leq \alpha_1 \leq \cdots \leq \alpha_p \leq d \}.
\]

For an \( \alpha = (\alpha_1, \ldots, \alpha_p) \in \Lambda \), denote \( M_{\alpha} = e_{\alpha_1} \otimes \cdots \otimes e_{\alpha_p}, x_\alpha = e_{\alpha_1} + \cdots + e_{\alpha_p} \).

The model space \( \mathcal{M} \) is a subset of rank-1 \( p \)-th order tensors, which is defined as
\[
\mathcal{M} = \{M_\alpha | \alpha \in \Lambda \}. \quad \mathcal{F} = \text{conv}(\mathcal{F}_0).
\]

For \( M_\alpha \in \mathcal{M} \), \( x \in \mathcal{F} \), define \( r(M_\alpha, x) = \langle M_\alpha, x^{\otimes p} \rangle = \prod_{i=1}^p \langle e_{\alpha_i}, x \rangle \). We assume that for each level \( h \), there is a \( M^{(h)} = M^{(h)}_\alpha \in \mathcal{M} \), and the noiseless reward is \( r_h(s, a) = r(M^{(h)}, \phi_h(s, a)) \).

We have the following properties.

**Proposition C.1** [Huang et al. (2021)]. For \( M_\alpha \in \mathcal{M} \) and \( x_{\alpha'} \in \mathcal{F}_0 \), we have
\[
r(M_\alpha, x_{\alpha'}) = \mathbb{I}_{\{\alpha = \alpha'\}}.
\]

**Proposition C.2.** For \( M_\alpha \in \mathcal{M} \), we have
\[
\max_{x \in \mathcal{F}} r(M_\alpha, x) = 1.
\]
proof of Proposition C.2  For all $x \in \mathcal{F}$, since $\mathcal{F} = \text{conv}(\mathcal{F}_0)$, we can write
\[ x = \sum_{\alpha \in \Lambda} p_\alpha (e_{\alpha_1} + \cdots + e_{\alpha_p}), \]
where $\sum_{\alpha \in \Lambda} p_\alpha = 1$ and $p_\alpha \geq 0$. Therefore,
\[ r(M_{\alpha'}, x) = \prod_{i=1}^{p} \langle e_{\alpha'_i}, x \rangle. \]
Plug in the expression of $x$, we have
\[ \langle e_{\alpha'_i}, x \rangle = \sum_{\alpha} p_\alpha \langle e_{\alpha'_i}, e_{\alpha_1} + \cdots + e_{\alpha_p} \rangle \]
\[ = \sum_{\alpha} p_\alpha \mathbb{1}_{\{e_{\alpha'_i} \in \alpha\}} \]
\[ \leq \sum_{\alpha} p_\alpha = 1. \]
Therefore,
\[ r(M_{\alpha'}, x) = \prod_{i=1}^{p} \langle e_{\alpha'_i}, x \rangle \]
\[ = \left( \sum_{\alpha} p_\alpha \mathbb{1}_{\{e_{\alpha'_1} \in \alpha\}} \right) \cdots \left( \sum_{\alpha} p_\alpha \mathbb{1}_{\{e_{\alpha'_p} \in \alpha\}} \right) \]
\[ \leq 1. \]
Finally, since $r(M_{\alpha'}, x_{\alpha'}) = 1$, we have $\max_{x \in \mathcal{F}} r(M_\alpha, x) = 1$. \qed

MDP constructions  Consider a family of MDPs with only two states $\mathcal{S} = \{S_{\text{good}}, S_{\text{bad}}\}$. The action set $\mathcal{A}$ is set to be $\mathcal{F}$. Let $f$ be a mapping from $\mathcal{F}$ to $\mathcal{F}_0$ such that $f$ is identity when restricted to $\mathcal{F}_0$. For all level $h \in [H]$, we define the feature map $\phi_h : \mathcal{S} \times \mathcal{A} \to \mathcal{F}$ to be
\[ \phi_h(s, a) = \begin{cases} a & \text{if } s = S_{\text{good}}, \\ f(a) & \text{if } s = S_{\text{bad}}. \end{cases} \]
Given an unknown sequence of indices $\alpha^{(1)}, \ldots, \alpha^{(H)}$, the reward function at level $h$ is $r_h(s, a) = r(M_{\alpha^{(h)}}, \phi_h(s, a))$. Specifically, we have
\[ r_h(S_{\text{good}}, a) = r(M_{\alpha^{(h)}}, a), \quad r_h(S_{\text{bad}}, a) = r(M_{\alpha^{(h)}}, f(a)). \]
The transition $P_h$ is constructed as

$$ P_h(S_{\text{bad}}|s, a) = 1 \text{ for all } s \in S, a \in A. $$

This construction means it is impossible for the online scenarios to reach the good state for $h > 1$.

The next proposition shows that $Q_h^*$ is polynomial realizable and falls into the case of Example 4.4.

**Proposition C.3.** We have for all $h \in [H]$ and $s \in S$, $a \in A$, $V_h^*(s) = H - h + 1$ and $Q_h^*(s, a) = r_h(s, a) + H - h + 1$. Furthermore, $Q_h^*(s, a)$, viewed as the function of $\phi_h(s, a)$, is a polynomial of the form $q_h(U_h\phi_h(s, a))$ for some degree-$p$ polynomial $q_h$ and $U_h \in \mathbb{R}^{p \times d}$.

**Proof of Proposition C.3.** First notice that by Proposition C.2, for all $h \in [H]$ and $s \in S$, we have

$$ \max_{a \in A} r_h(s, a) = 1. $$

Therefore, by induction, suppose we have proved for all $s'$, $V_{h+1}^*(s') = H - h$, then we have

$$ V_h^*(s) = \max_{a \in A} Q_h^*(s, a) $$

$$ = \max_{a \in A} \{r_h(s, a) + \mathbb{E}_{s' \sim P_h(|s, a)}[V_{h+1}^*(s')]\} $$

$$ = 1 + H - h. $$

Then we have $Q_h^*(s, a) = r_h(s, a) + H - h + 1$.

Furthermore, we have

$$ Q_h^*(s, a) = r_h(s, a) + H - h + 1 $$

$$ = r(M_{\alpha(h)}, \phi_h(s, a)) + H - h + 1 $$

$$ = \prod_{i=1}^p \langle e_{\alpha_i(h)}, \phi_h(s, a) \rangle + H - h + 1 $$

$$ = q_h(U_h\phi_h(s, a)), $$

where $q_h(x_1, \ldots, x_p) = x_1 x_2 \cdots x_p + (H - h + 1)$ and $U_h \in \mathbb{R}^{p \times d}$ is a matrix with $e_{\alpha_i(h)}$ as the $i$-th row.

**Theorem C.4.** Under the online RL setting, any algorithm needs to play at least $(\binom{d}{p} - 1) = \Omega(d^p)$ episodes to identify $\alpha^{(2)}, \ldots, \alpha^{(H)}$ and thus to identify the optimal policy.
proof of Theorem C.4. Under the online RL setting, any algorithm enters and remains in $S_{\text{bad}}$ for $h > 1$. When $s_h = S_{\text{bad}}$, no matter what $a_h$ the algorithm chooses, we have $\phi_h(s_h, a_h) = f(a_h) \in F_0$. Notice that for any $M_{\alpha(h)} \in M$ and any $x_\alpha \in F_0$, we have $r(M_{\alpha(h)}, x_\alpha) = I_{\{\alpha = \alpha(h)\}}$ as Proposition C.1 suggests. Hence, we need to play $(\binom{d}{p} - 1)$ times at level $h$ in the worst case to find out $\alpha(h)$. The argument holds for all $h = 2, 3, \ldots, H$.

Theorem C.5. Under the generative model setting, by querying $2d(p + 1)^p H = O(dH)$ samples, we can almost surely identify $\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(H)}$ and thus identify the optimal policy.

proof of Theorem C.5. By Proposition C.3, we know that $Q^*_h(s, a)$, viewed as the function of $\phi_h(s, a)$, falls into the case of Example 4.4 with $k = p$.

Next, notice that for all $h \in [H]$, $\{\phi_h(s, a) \mid s \in S, a \in A\} = F$. Although $F$ is not of positive measure, we can actually know the value of $Q^*_h$ when $\phi_h(s, a)$ is in $\text{conv}(F, 0)$ since the reward is $p$-homogenous. Specifically, for every feature of the form $c \cdot \phi_h(s, a)$, where $0 \leq c \leq 1$ and $\phi_h(s, a) \in F$, the reward is $c^p$ times the reward of $(s, a)$. Therefore, to get the reward at $c \cdot \phi_h(s, a)$, we only need to query the generative model at $(s, a)$ of level $h$, and then multiply the reward by $c^p$.

Notice that $\text{conv}(F, 0)$ is of positive Lebesgue measure. By Theorem 4.7, we know that only $2d(p + 1)^p H = O(dH)$ samples are needed to determine the optimal policy almost surely.

D Technical claims

Claim D.1. Let $\chi^2(d)$ denote $\chi^2$-distribution with freedom $d$. For any $t > 0$ we have,

$$\Pr_{z \sim \chi^2(d)}(z \geq d + 2t + 2\sqrt{dt}) \leq e^{-t}$$

We use the following facts from [Zhong et al., 2017].

Claim D.2. Given a fixed vector $z \in \mathbb{R}^d$, for any $C \geq 1$ and $n \geq 1$, we have

$$\Pr_{x \sim \mathcal{N}(0, I_d)}(\|x\|^2 \leq 5C\|z\|^2 \log n) \geq 1 - 1/(nd^C).$$

Claim D.3. For any $C \geq 1$ and $n \geq 1$, we have

$$\Pr_{x \sim \mathcal{N}(0, I_d)}(\|x\|^2 \leq 5Cd \log n) \geq 1 - 1/(nd^C).$$
Claim D.4. Let \( a, b, c \geq 0 \) be three constants, let \( u, v, w \in \mathbb{R}^d \) be three vectors, we have
\[
\mathbb{E}_{x \sim \mathcal{N}(0, I_d)} \left[ |u^T x|^a |v^T x|^b |w^T x|^c \right] \approx \|u\|^a \|v\|^b \|w\|^c.
\]

Claim D.5. Let \( M_{j, j} \in [4] \) be defined in Definition A.10. For each \( j \in [4] \), 
\[
M_j = \sum_{i=1}^k v_i^* m_{j,i} \otimes j^i.
\]

Claim D.6. Let \( B \) denote a distribution over \( \mathbb{R}^{d_1 \times d_2} \). Let \( d = d_1 + d_2 \). Let \( B_1, B_2, \ldots, B_n \) be i.i.d. random matrices sampled from \( B \). Let \( \bar{B} = \mathbb{E}_{B \sim B}[B] \) and \( \hat{B} = \frac{1}{n} \sum_{i=1}^n B_i \). For parameters \( m \geq 0, \gamma \in (0, 1), \nu > 0, L > 0 \), if the distribution \( B \) satisfies the following four properties,
\begin{enumerate}
  \item \( \Pr_{B \sim B} [\|B\| \leq m] \geq 1 - \gamma; \)
  \item \( \left\| \mathbb{E}_{B \sim B} [B] \right\| > 0; \)
  \item \( \max \left( \left\| \mathbb{E}_{B \sim B} [BB^T] \right\|, \left\| \mathbb{E}_{B \sim B} [B^T B] \right\| \right) \leq \nu; \)
  \item \( \max_{\|a\|=\|b\|=1} \left( \left\| \mathbb{E}_{B \sim B} \left[ (a^\top B b)^2 \right] \right\| \right)^{1/2} \leq L. \)
\end{enumerate}

Then we have for any \( 0 < \epsilon < 1 \) and \( t \geq 1 \), if
\[
n \geq (18 t \log d) \cdot (\nu + \|\bar{B}\|^2 + m \|\bar{B}\| \epsilon)/(\epsilon^2 \|\bar{B}\|^2) \quad \text{and} \quad \gamma \leq (\epsilon \|\bar{B}\|/(2L))^2
\]
with probability at least \( 1 - 1/d^{2t} - n\gamma \),
\[
\|\hat{B} - \bar{B}\| \leq \epsilon \|\bar{B}\|.
\]

Claim D.7. Let \( P_2 \) and \( P_3 \) be defined in Definition A.15. Then
\[
P_2 = \sum_{i=1}^k v_i m_{j_2,i} (\alpha^\top \bar{w}_i)^{j_2 - 2} \bar{w}_i^\otimes 2
\]
and
\[
P_3 = \sum_{i=1}^k v_i m_{j_3,i} (\alpha^\top \bar{w}_i)^{j_3 - 3} \bar{w}_i^\otimes 3.
\]
Algorithm 6 Method of moments

1: Inputs: $S = \{(x_i, y_i) : i \in [n]\}$
2: Choose $\alpha$ to be a random unit vector
3: Partition $S$ into $S_1, S_2, S_3, S_4$ of equal size
4: $\hat{P}_2 \leftarrow \mathbb{E}_{S_1}[P_2], C \leftarrow 3\|P_2\|$, $T \leftarrow$ sufficient large constant
5: Choose $\hat{V}_1^{(0)}, \hat{V}_2^{(0)} \in \mathbb{R}^{d \times k}$ to be random matrices \hspace{1cm} \triangleright\text{Estimate subspace } V
6: for $t = 1, \ldots, T$ do
7: \hspace{1cm} $\hat{V}_1^{(t)} \leftarrow \text{QR}(C\hat{V}_1^{(t-1)} + \hat{P}_2\hat{V}_2^{(t-1)}), \hat{V}_2^{(t)} \leftarrow \text{QR}(C\hat{V}_2^{(t-1)} - \hat{P}_2\hat{V}_1^{(t-1)})$
8: end for
9: for $j = 1, 2$ do
10: \hspace{1cm} $\hat{V}_1^{(T)} \leftarrow [\hat{V}_{j,1}, \ldots, \hat{V}_{j,k}]$
11: \hspace{1cm} for $i \in [k]$ do
12: \hspace{1.5cm} $\lambda_{j,i} \leftarrow |\hat{V}_{j,i}\hat{P}_2\hat{V}_{j,i}|$
13: \hspace{1cm} end for
14: end for
15: $\pi_1, \pi_2, k_1, k_2 \leftarrow \text{TOPK}(\lambda, k)$
16: for $j = 1, 2$ do
17: \hspace{1cm} $V_j \leftarrow [\hat{V}_{j,\pi_j(1)}, \ldots, \hat{V}_{j,\pi_j(k_j)}]$\hspace{1cm} \triangleright\text{Learn } s_jV_j^\top \mathbb{w}_i$
18: end for
19: $\hat{V}_2 \leftarrow \text{QR}((I - V_1V_1^\top)V_2), V \leftarrow [V_1, \hat{V}_2]$
20: $\hat{R}_3 \leftarrow \mathbb{E}_{S_2}[P_3(V, V, V)], \{\hat{w}_i\}_{i \in [k]} \leftarrow \text{TENSORDECOMPOSITION}(\hat{R}_3)$ \hspace{1cm} \triangleright\text{Learn } s_1 V_i^\top \mathbb{w}_i$
21: if $M_1 = M_3 = 0$ then
22: \hspace{1cm} $l_1, l_2 = \min\{j \in \{2, 4\} : M_j \neq 0\}$
23: else if $M_2 = M_4 = 0$ then
24: \hspace{1cm} $l_1 = \min\{j \in \{1, 3\} : M_j \neq 0\}, l_2 \leftarrow 3$
25: else
26: \hspace{1cm} $l_1 \leftarrow \min\{j \in \{1, 3\} : M_j \neq 0\}, l_2 = \min\{j \in \{2, 4\} : M_j \neq 0\}$
27: end if
28: $\hat{Q}_1 \leftarrow \mathbb{E}_{S_1}[Q_1], \hat{Q}_2 \leftarrow \mathbb{E}_{S_4}[Q_2]$
29: $\hat{z} \leftarrow \arg \min_i \|\sum_{i=1}^k z_i V\hat{u}_i - \hat{Q}_1\|, \hat{r} \leftarrow \arg \min_{r_i} \|\sum_{i=1}^k r_i\hat{u}_i\hat{u}_i - \hat{Q}_2\|_F$
30: for $i = 1, \ldots, k$ do
31: \hspace{1cm} $v^{(0)}_i \leftarrow \text{sign}(\hat{r}_i c_{i2}), s^{(0)}_i \leftarrow \text{sign}(v^{(0)}_i \hat{z}_i c_{i1})$
32: \hspace{1cm} $w^{(0)}_i \leftarrow s^{(0)}_i (|\frac{z_i}{c_{i1}(\alpha + V_i\mathbb{w}_i)^{-1})|^{1/(p+1)}) V\hat{u}_i$
33: end for
34: $W^{(0)} \leftarrow [w^{(0)}_1, \ldots, w^{(0)}_k]$
35: Return $v^{(0)}, W^{(0)}$