which scales linearly with the capacity of joint action space. Besides, if $|\mathcal{B}| = 1$, the minimax-FQI algorithm reduces to Algorithm 1. In this case, (5.12) also recovers the error rate of Algorithm 1. Furthermore, the statistical rate $n^{(\alpha^*-1)/2}$ achieves the optimal ℓ_2-norm error of regression for nonparametric regression with a compositional structure, which indicates that the statistical error in (5.11) can not be further improved.

Proof. See §D for a detailed proof. □

6 Proof of the Main Theorem

In this section, we present a detailed proof of Theorem 4.4.

Proof. The proof requires two key ingredients. First in Theorem 6.1 we quantify how the error of action-value function approximation propagates through each iteration of Algorithm 1. Then in Theorem 6.2 we analyze such one-step approximation error for ReLU networks.

Theorem 6.1 (Error Propagation)**.** Recall that $\{\widetilde{Q}_k\}_{0\le k\le K}$ are the iterates of Algorithm 1. Let π_K be the one-step greedy policy with respect to $\widetilde{Q}_K$, and let Q^{π_K} be the action-value function corresponding to π_K. Under Assumption 4.3, we have

$$\|Q^* - Q^{\pi_K}\|_{1,\mu} \le \frac{2\phi_{\mu,\sigma}\gamma}{(1-\gamma)^2} \cdot \varepsilon_{\max} + \frac{4\gamma^{K+1}}{(1-\gamma)^2} \cdot R_{\max}, \tag{6.1}$$

where we define the maximum one-step approximation error as $\varepsilon_{\max} = \max_{k\in[K]} \|T\widetilde{Q}_{k-1} - \widetilde{Q}_k\|_\sigma$. Here $\phi_{\mu,\sigma}$ is a constant that only depends on the probability distributions μ and σ.

Proof. See §C.1 for a detailed proof. □

We remark that similar error propagation result is established for the state-value function in Munos and Szepesvári (2008) for studying the fitted value iteration algorithm, which is further extended by Lazaric et al. (2016); Scherrer et al. (2015); Farahmand et al. (2010, 2016) for other batch reinforcement learning methods.

In the sequel, we establish an upper bound for the one-step approximation error $\|T\widetilde{Q}_{k-1} - \widetilde{Q}_k\|_\sigma$ for each $k \in [K]$.

Theorem 6.2 (One-step Approximation Error)**.** Let $\mathcal{F} \subseteq \mathcal{B}(\mathcal{S}\times\mathcal{A}, V_{\max})$ be a class of measurable functions on $\mathcal{S}\times\mathcal{A}$ that are bounded by $V_{\max} = R_{\max}/(1-\gamma)$, and let σ be a probability distribution on $\mathcal{S}\times\mathcal{A}$. Also, let $\{(S_i, A_i)\}_{i\in[n]}$ be n i.i.d. random variables in $\mathcal{S}\times\mathcal{A}$ following σ. For each $i \in [n]$, let R_i and S_i' be the reward and the next state corresponding to (S_i, A_i). In addition, for any fixed $Q \in \mathcal{F}$, we define $Y_i = R_i + \gamma \cdot \max_{a\in\mathcal{A}} Q(S_i', a)$. Based on $\{(X_i, A_i, Y_i)\}_{i\in[n]}$, we define $\widehat{Q}$ as the solution to the least-squares problem

$$\underset{f\in\mathcal{F}}{\text{minimize}} \frac{1}{n}\sum_{i=1}^{n}\big[f(S_i, A_i) - Y_i\big]^2. \tag{6.2}$$

Meanwhile, for any $\delta > 0$, let $\mathcal{N}(\delta, \mathcal{F}, \|\cdot\|_\infty)$ be the minimal δ-covering set of $\mathcal{F}$ with respect to ℓ_∞-norm, and we denote by N_δ its cardinality. Then for any $\epsilon \in (0, 1]$ and any $\delta > 0$, we have

$$\|\widehat{Q} - TQ\|_\sigma^2 \le (1+\epsilon)^2 \cdot \omega(\mathcal{F}) + C \cdot V_{\max}^2/(n \cdot \epsilon) \cdot \log N_\delta + C' \cdot V_{\max} \cdot \delta, \tag{6.3}$$

where C and C' are two positive absolute constants and $\omega(\mathcal{F})$ is defined as

$$\omega(\mathcal{F}) = \sup_{g \in \mathcal{F}} \inf_{f \in \mathcal{F}} \|f - Tg\|_\sigma^2. \tag{6.4}$$

Proof. See §C.2 for a detailed proof. □

This theorem characterizes the bias and variance that arise in estimating the action-value functions using deep ReLU networks. Specifically, $\omega(\mathcal{F})$ in (6.4) corresponds to the bias incurred by approximating the target function Tf using ReLU neural networks. It can be viewed as a measure of completeness of $\mathcal{F}$ with respect to the Bellman operator T. In addition, $V_{\max}^2/n \cdot \log N_\delta + V_{\max} \cdot \delta$ controls the variance of the estimator, where the covering number N_δ is used to obtain a uniform bound over $\mathcal{F}_0$.

To obtain an upper bound for $\|T\widetilde{Q}_{k-1} - \widetilde{Q}_k\|_\sigma$ as required in Theorem 6.1, we set $Q = \widetilde{Q}_{k-1}$ in Theorem 6.2. Then according to Algorithm 1, $\widehat{Q}$ defined in (6.2) becomes $\widetilde{Q}_k$. We set the function class $\mathcal{F}$ in Theorem 6.2 to be the family of ReLU Q-networks $\mathcal{F}_0$ defined in (4.1). By setting $\epsilon = 1$ and $\delta = 1/n$ in Theorem 6.2, we obtain

$$\|\widetilde{Q}_{k+1} - T\widetilde{Q}_k\|_\sigma^2 \le 4 \cdot \omega(\mathcal{F}_0) + C \cdot V_{\max}^2/n \cdot \log N_0, \tag{6.5}$$

where C is a positive absolute constant and

$$N_0 = \big|\mathcal{N}(1/n, \mathcal{F}_0, \|\cdot\|_\infty)\big| \tag{6.6}$$

is the $1/n$-covering number of $\mathcal{F}_0$. In the subsequent proof, we establish upper bounds for $\omega(\mathcal{F}_0)$ defined in (6.4) and $\log N_0$, respectively. Recall that the family of composite Hölder smooth functions $\mathcal{G}_0$ is defined in (4.2). By Assumption 4.2, we have $Tg \in \mathcal{G}_0$ for any $g \in \mathcal{F}_0$. Hence, we have

$$\omega(\mathcal{F}_0) = \sup_{f' \in \mathcal{G}_0} \inf_{f \in \mathcal{F}_0} \|f - f'\|_\sigma^2 \le \sup_{f' \in \mathcal{G}_0} \inf_{f \in \mathcal{F}_0} \|f - f'\|_\infty^2, \tag{6.7}$$

where the right-hand side is the ℓ_∞-error of approximating the functions in $\mathcal{G}_0$ using the family of ReLU networks $\mathcal{F}_0$.

By the definition of $\mathcal{G}_0$ in (4.2), for any $f \in \mathcal{G}_0$ and any $a \in \mathcal{A}$, $f(\cdot, a) \in \mathcal{G}(\{(p_j, t_j, \beta_j, H_j)\}_{j \in [q]})$ is a composition of Hölder smooth functions, that is, $f(\cdot, a) = g_q \circ \cdots \circ g_1$. Recall that, as defined in Definition 2.3, g_{jk} is the k-th entry of the vector-valued function g_j. Here $g_{jk} \in \mathcal{C}_{t_j}([a_j, b_j]^{t_j}, \beta_j, H_j)$ for each $k \in [p_{j+1}]$ and $j \in [q]$. In the sequel, we construct a ReLU network to approximate $f(\cdot, a)$ and establish an upper bound of the approximation error on the right-hand side of (6.7). We first show that $f(\cdot, a)$ can be reformulated as a composition of Hölder functions defined on a hypercube. We define $h_1 = g_1/(2H_1) + 1/2$,

$$h_j(u) = g_j(2H_{j-1}u - H_{j-1})/(2H_j) + 1/2, \quad \text{for all} \quad j \in \{2, \ldots, q-1\},$$

and $h_q(u) = g_q(2H_{q-1}u - H_{q-1})$. Then we immediately have

$$f(\cdot, a) = g_q \circ \cdots \circ g_1 = h_q \circ \cdots \circ h_1. \tag{6.8}$$

Furthermore, by the definition of Hölder smooth functions in Definition 2.2, for any $k \in [p_2]$, we have that h_{1k} takes value in $[0,1]$ and $h_{1k} \in \mathcal{C}_{t_1}([0,1]^{t_1}, \beta_1, 1)$. Similarly, for any $j \in \{2, \ldots, q-1\}$ and $k \in [p_{j+1}]$, h_{jk} also takes value in $[0,1]$ and

$$h_{jk} \in \mathcal{C}_{t_j}\big([0,1]^{t_j}, \beta_j, (2H_{j-1})^{\beta_j}\big). \tag{6.9}$$

Finally, recall that we use the convention that $p_{q+1} = 1$, that is, h_q is a scalar-valued function that satisfies

$$h_q \in \mathcal{C}_{t_q}\big([0,1]^{t_q}, \beta_q, H_q(2H_{q-1})^{\beta_q}\big).$$

In the following, we show that the composition function in (6.8) can be approximated by an element in $\mathcal{F}(L^*, \{d_j^*\}_{j=1}^{L+1}, s^*)$ when the network hyperparameters are properly chosen. Our proof consists of three steps. In the first step, we construct a ReLU network $\widetilde{h}_{jk}$ that approximates each h_{jk} in (6.9). Then, in the second step, we approximate $f(\cdot, a)$ by the composition of $\{\widetilde{h}_j\}_{j\in[q]}$ and quantify the architecture of this network. Finally, in the last step, we prove that this network can be embedded into class $\mathcal{F}(L^*, \{d_j^*\}_{j=1}^{L+1}, s^*)$ and characterize the final approximation error.

Step (i). Now we employ the following lemma, obtained from Schmidt-Hieber (2020+), to construct a ReLU network that approximates each h_{jk}, which combined with (6.8) yields a ReLU network that is close to $f(\cdot, a)$. Recall that, as defined in Definition 2.2, we denote by $\mathcal{C}_r(\mathcal{D}, \beta, H)$ the family of Hölder smooth functions with parameters β and H on $\mathcal{D} \subseteq \mathbb{R}^r$.

Lemma 6.3 (Theorem 5 in Schmidt-Hieber (2020+))**.** For any integers $m \geq 1$ and $N \geq \max\{(\beta+1)^r, (H+1)e^r\}$, let $L = 8 + (m+5)\cdot(1 + \lceil \log_2(r+\beta)\rceil)$, $d_0 = r$, $d_j = 6(r + \lceil\beta\rceil)N$ for each $j \in [L]$, and $d_{L+1} = 1$. For any $g \in \mathcal{C}_r([0,1]^r, \beta, H)$, there exists a ReLU network $f \in \mathcal{F}(L, \{d_j\}_{j=0}^{L+1}, s, V_{\max})$ as defined in Definition 2.1 such that

$$\|f - g\|_\infty \leq (2H+1)\cdot 6^r \cdot N \cdot (1 + r^2 + \beta^2)\cdot 2^{-m} + H \cdot 3^\beta \cdot N^{-\beta/r},$$

where the parameter s satisfies $s \leq 141 \cdot (r + \beta + 1)^{3+r} \cdot N \cdot (m+6)$.

Proof. See Appendix B in Schmidt-Hieber (2020+) for a detailed proof. The idea is to first approximate the Hölder smooth function by polynomials via local Taylor expansion. Then, neural networks are constructed explicitly to approximate each monomial terms in these local polynomials. □

We apply Lemma 6.3 to $h_{jk}\colon [0,1]^{t_j} \to [0,1]$ for any $j \in [q]$ and $k \in [p_{j+1}]$. We set $m = \eta \cdot \lceil \log_2 n \rceil$ for a sufficiently large constant $\eta > 1$, and set N to be a sufficiently large integer depending on n, which will be specified later. In addition, we set

$$L_j = 8 + (m+5)\cdot(1 + \lceil \log_2(t_j + \beta_j)\rceil) \tag{6.10}$$

and define

$$W = \max\Big\{\max_{1\le j\le q-1}(2H_{j-1})^{\beta_j}, H_q(2H_{q-1})^{\beta_q}, 1\Big\}. \tag{6.11}$$

We will later verify that $N \ge \max\{(\beta+1)^{t_j}, (W+1)e^{t_j}\}$ for all $j \in [q]$. Then by Lemma 6.3, there exists a ReLU network $\widehat{h}_{jk}$ such that

$$\|\widehat{h}_{jk} - h_{jk}\|_\infty \le (2W+1)\cdot 6^{t_j}\cdot N\cdot 2^{-m} + W\cdot 3^{\beta_j}\cdot N^{-\beta_j/t_j}. \tag{6.12}$$

Furthermore, we have $\widehat{h}_{jk} \in \mathcal{F}(L_j, \{t_j, \widetilde{d}_j, \ldots, \widetilde{d}_j, 1\}, \widetilde{s}_j)$ with

$$\widetilde{d}_j = 6(t_j + \lceil\beta_j\rceil)N, \qquad \widetilde{s}_j \le 141\cdot(t_j+\beta_j+1)^{3+t_j}\cdot N\cdot(m+6). \tag{6.13}$$

Meanwhile, since $h_{j+1} = (h_{(j+1)k})_{k\in[p_{j+2}]}$ takes input from $[0,1]^{t_{j+1}}$, we need to further transform $\widehat{h}_{jk}$ so that it takes value in $[0,1]$. In particular, we define $\sigma(u) = 1-(1-u)_+ = \min\{\max\{u,0\},1\}$ for any $u\in\mathbb{R}$. Note that σ can be represented by a two-layer ReLU network with four nonzero weights. Then we define $\widetilde{h}_{jk} = \sigma\circ\widehat{h}_{jk}$ and $\widetilde{h}_j = (\widetilde{h}_{jk})_{k\in[p_{j+1}]}$. Note that by the definition of $\widetilde{h}_{jk}$, we have $\widetilde{h}_{jk} \in \mathcal{F}(L_j+2, \{t_j, \widetilde{d}_j, \ldots, \widetilde{d}_j, 1\}, \widetilde{s}_j+4)$, which yields

$$\widetilde{h}_j \in \mathcal{F}\big(L_j+2, \{t_j, \widetilde{d}_j\cdot p_{j+1}, \ldots, \widetilde{d}_j\cdot p_{j+1}, p_{j+1}\}, (\widetilde{s}_j+4)\cdot p_{j+1}\big). \tag{6.14}$$

Moreover, since both $\widetilde{h}_{jk}$ and h_{jk} take value in $[0,1]$, by (6.12) we have

$$\begin{aligned}\|\widetilde{h}_{jk} - h_{jk}\|_\infty &= \|\sigma\circ\widehat{h}_{jk} - \sigma\circ h_{jk}\|_\infty \le \|\widehat{h}_{jk} - h_{jk}\|_\infty \\ &\le (2W+1)\cdot 6^{t_j}\cdot N\cdot n^{-\eta} + W\cdot 3^{\beta_j}\cdot N^{-\beta_j/t_j},\end{aligned} \tag{6.15}$$

where the constant W is defined in (6.11). Since we can set the constant η in (6.15) to be sufficiently large, the second term on the right-hand side of (6.15) is the leading term asymptotically, that is,

$$\|\widetilde{h}_{jk} - h_{jk}\|_\infty \lesssim N^{-\beta_j/t_j}. \tag{6.16}$$

Thus, in the first step, we have shown that there exists $\widetilde{h}_{jk} \in \mathcal{F}(L_j+2, \{t_j, \widetilde{d}_j, \ldots, \widetilde{d}_j, 1\}, \widetilde{s}_j+4)$ satisfying (6.16).

Step (ii). In the second step, we stack $\widetilde{h}_j$ defined in (6.14) to approximate $f(\cdot, a)$ in (6.8). Specifically, we define $\widetilde{f}\colon \mathcal{S}\to\mathbb{R}$ as $\widetilde{f} = \widetilde{h}_q\circ\cdots\circ\widetilde{h}_1$, which falls in the function class

$$\mathcal{F}(\widetilde{L}, \{r, \widetilde{d}, \ldots, \widetilde{d}, 1\}, \widetilde{s}), \tag{6.17}$$

where we define $\widetilde{L} = \sum_{j=1}^q (L_j+2)$, $\widetilde{d} = \max_{j\in[q]}\widetilde{d}_j\cdot p_{j+1}$, and $\widetilde{s} = \sum_{j=1}^q(\widetilde{s}_j+4)\cdot p_{j+1}$. Recall that L_j is defined in (6.10). Then when n is sufficiently large, we have

$$\begin{aligned}\widetilde{L} &\le \sum_{j=1}^q \big\{8 + \eta\cdot(\log_2 n + 6)\cdot\big[1 + \lceil\log_2(t_j+\beta_j)\rceil\big]\big\} \\ &\lesssim \sum_{j=1}^q \eta\cdot\log_2(t_j+\beta_j)\cdot\log_2 n \lesssim (\log n)^{1+\xi},\end{aligned} \tag{6.18}$$

where $\xi > 0$ is an absolute constant. Here the last inequality follows from (4.6). Moreover, for $\widetilde{d}$ defined in (6.17), by (4.6) we have

$$N \cdot \max_{j\in[q]}\{p_{j+1}\cdot(t_j+\beta_j)\} \lesssim \widetilde{d} \le 6\cdot N\cdot \big(\max_{j\in[q]} p_j\big)\cdot\big[\max_{j\in[q]}(t_j+\beta_j)\big] \lesssim N\cdot(\log n)^{2\xi}. \tag{6.19}$$

In addition, combining (6.13), (4.6), and the fact that $t_j \le p_j$, we obtain

$$\begin{aligned}\widetilde{s} &\lesssim N\cdot\log n\cdot\bigg[\sum_{j=1}^{q} p_{j+1}\cdot(t_j+\beta_j+1)^{3+t_j}\bigg]\\ &\lesssim N\cdot\log n\cdot\big(\max_{j\in[q]} p_j\big)\cdot\bigg[\sum_{j=1}^{q}(t_j+\beta_j+1)^{3+t_j}\bigg] \lesssim N\cdot(\log n)^{1+2\xi}.\end{aligned} \tag{6.20}$$

Step (iii). In the last step, we show that the function class in (6.17) can be embedded in $\mathcal{F}(L^*,\{d_j^*\}_{j=1}^{L^*+1},s^*)$ and characterize the final approximation bias, where L^*, $\{d_j^*\}_{j=1}^{L^*+1}$, and s^* are specified in (4.7). To this end, we set

$$N = \big\lceil \max_{1\le j\le q} C\cdot n^{t_j/(2\beta_j^*+t_j)}\big\rceil, \tag{6.21}$$

where the absolute constant $C>0$ is sufficiently large. Note that we define $\alpha^* = \max_{j\in[q]} t_j/(2\beta_j^*+t_j)$. Then (6.21) implies that $N \asymp n^{\alpha^*}$. When n is sufficiently large, it holds that $N \ge \max\{(\beta+1)^{t_j},(W+1)e^{t_j}\}$ for all $j\in[q]$. When ξ^* in (4.7) satisfies $\xi^* \ge 1+2\xi$, by (6.18) we have

$$\widetilde{L} \le L^* \lesssim (\log n)^{\xi^*}.$$

In addition, (6.19) and (4.7) implies that we can set $d_j^* \ge \widetilde{d}$ for all $j\in[L^*]$. Finally, by (6.20) and (6.21), we have $\widetilde{s}\lesssim n^{\alpha^*}\cdot(\log n)^{\xi^*}$, which implies $\widetilde{s}+(L^*-\widetilde{L})\cdot r \le s^*$. For an $\widetilde{L}$-layer ReLU network in (6.17), we can make it an L^*-layer ReLU network by inserting $L^*-\widetilde{L}$ identity layers, since the inputs of each layer are nonnegative. Thus, ReLU networks in (6.17) can be embedded in

$$\mathcal{F}\big[L^*,\{r,r,\ldots,r,\widetilde{d},\ldots,\widetilde{d},1\},\widetilde{s}+(L^*-\widetilde{L})d\big],$$

which is a subset of $\mathcal{F}(L^*,\{d_j^*\}_{j=1}^{L+1},s^*)$ by (4.7).

To obtain the approximation error $\|\widetilde{f}-f(\cdot,a)\|_\infty$, we define $G_j = h_j\circ\cdots\circ h_1$ and $\widetilde{G}_j = \widetilde{h}_j\circ\cdots\circ\widetilde{h}_1$ for any $j\in[q]$. By triangle inequality, for any $j>1$ we have

$$\begin{aligned}\|G_j-\widetilde{G}_j\|_\infty &\le \|h_j\circ\widetilde{G}_{j-1}-h_j\circ G_{j-1}\|_\infty + \|\widetilde{h}_j\circ\widetilde{G}_{j-1}-h_j\circ\widetilde{G}_{j-1}\|_\infty\\ &\le W\cdot\|G_{j-1}-\widetilde{G}_{j-1}\|_\infty^{\beta_j\wedge 1} + \|h_j-\widetilde{h}_j\|_\infty,\end{aligned} \tag{6.22}$$

where the second inequality holds since h_j is Hölder smooth. To simplify the notation, we define $\lambda_j = \prod_{\ell=j+1}^{q}(\beta_\ell\wedge 1)$ for any $j\in[q-1]$, and set $\lambda_q = 1$. By applying recursion to (6.22), we obtain

$$\|f(\cdot,a)-\widetilde{f}\|_\infty = \|G_q-\widetilde{G}_q\|_\infty \le W\sum_{j=1}^{q}\|\widetilde{h}_j-h_j\|_\infty^{\lambda_j}, \tag{6.23}$$

where the constant W is defined in (6.11). Here in (6.23) we use the fact that $(a+b)^\alpha \le a^\alpha + b^\alpha$ for all $\alpha \in [0,1]$ and $a, b > 0$.

In the sequel, we combine (6.7), (6.15), (6.23), and (6.21) to obtain the final bound on $\omega(\mathcal{F}_0)$. Also note that $\beta_j^* = \beta_j \cdot \prod_{\ell=j+1}^{q}(\beta_\ell \wedge 1) = \beta_j \cdot \lambda_j$ for all $j \in [q-1]$. Thus we have $\beta_j^* = \beta_j \cdot \lambda_j$ for all $j \in [q]$. Combining (6.23) and (6.16), we have

$$\|f(\cdot, a) - \widetilde{f}\|_\infty \lesssim \sum_{j=1}^{q} \big(N^{-\beta_j/t_j}\big)^{\lambda_j} = \sum_{j=1}^{q} N^{-\beta_j^*/t_j} \lesssim \max_{j\in[q]} N^{-\beta_j^*/t_j}. \tag{6.24}$$

Thus, we combine (6.7), (6.21), and (6.24) to obtain

$$\omega(\mathcal{F}_0) \le \big(\max_{j\in[q]} N^{-\beta_j^*/t_j}\big)^2 \asymp \max_{j\in[q]} n^{-2\beta_j^*/(2\beta_j^*+t_j)} = n^{\alpha^*-1}. \tag{6.25}$$

As the final step of the proof, it remains to control the covering number of $\mathcal{F}_0$ defined in (4.1). By definition, for any $f \in \mathcal{F}_0$, we have $f(\cdot, a) \in \mathcal{F}(L^*, \{d_j^*\}_{j=1}^{L^*+1}, s^*)$ for any $a \in \mathcal{A}$. For notational simplicity, we denote by $\mathcal{N}_\delta$ the δ-covering of $\mathcal{F}(L^*, \{d_j^*\}_{j=1}^{L^*+1}, s^*)$, that is, we define

$$\mathcal{N}_\delta = \mathcal{N}\big[\delta, \mathcal{F}(L^*, \{d_j^*\}_{j=1}^{L^*+1}, s^*), \|\cdot\|_\infty\big].$$

By the definition of covering, for any $f \in \mathcal{F}_0$ and any $a \in \mathcal{A}$, there exists $g_a \in \mathcal{N}_\delta$ such that $\|f(\cdot, a) - g_a\|_\infty \le \delta$. Then we define a function $g\colon \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ by $g(s, a) = g_a(s)$ for any $(s, a) \in \mathcal{S}\times\mathcal{A}$. By the definition of g, it holds that $\|f - g\|_\infty \le \delta$. Therefore, the cardinality of $\mathcal{N}(\delta, \mathcal{F}_0, \|\cdot\|_\infty)$ satisfies

$$\big|\mathcal{N}(\delta, \mathcal{F}_0, \|\cdot\|_\infty)\big| \le |\mathcal{N}_\delta|^{|\mathcal{A}|}. \tag{6.26}$$

Now we utilize the following lemma in Anthony and Bartlett (2009) to obtain an upper bound of the cardinality of $\mathcal{N}_\delta$.

Lemma 6.4 (Covering Number of ReLU Network)**.** Recall that the family of ReLU networks $\mathcal{F}(L, \{d_j\}_{j=0}^{L+1}, s, V_{\max})$ is given in Definition 2.1. Let $D = \prod_{\ell=1}^{L+1}(d_\ell + 1)$. For any $\delta > 0$, we have

$$\log\Big|\mathcal{N}\big[\delta, \mathcal{F}(L, \{d_j\}_{j=0}^{L+1}, s, V_{\max}), \|\cdot\|_\infty\big]\Big| \le (s+1)\cdot\log\big[2\delta^{-1}\cdot(L+1)\cdot D^2\big].$$

Proof. See Theorem 14.5 in Anthony and Bartlett (2009) for a detailed proof. □

Recall that we denote $\mathcal{N}(1/n, \mathcal{F}_0, \|\cdot\|_\infty)$ by N_0 in (6.6). By combining (6.26) with Lemma 6.4 and setting $\delta = 1/n$, we obtain that

$$\log N_0 \le |\mathcal{A}|\cdot\log|\mathcal{N}_\delta| \le |\mathcal{A}|\cdot(s^*+1)\cdot\log\big[2n\cdot(L^*+1)\cdot D^2\big],$$

where $D = \prod_{\ell=1}^{L^*+1}(d_\ell^* + 1)$. By the choice of L^*, s^*, and $\{d_j^*\}_{j=0}^{L^*+1}$ in (4.7), we conclude that

$$\log N_0 \lesssim |\mathcal{A}|\cdot s^*\cdot L^*\max_{j\in[L^*]}\log(d_j^*) \lesssim n^{\alpha^*}\cdot(\log n)^{1+2\xi^*}. \tag{6.27}$$

Finally, combining (6.1), (6.5), (6.25), and (6.27), we conclude the proof of Theorem 4.4. □

7 Conclusion

We study deep Q-network from the statistical perspective. Specifically, by neglecting the computational issues, we consider the fitted Q-iteration algorithm with ReLU networks, which can be viewed as a modification of DQN that fully captures its key features. Under mild assumptions, we show that DQN creates a sequence of policies whose corresponding value functions converge to the optimal value function, when both the sample size and the number of iteration go to infinity. Moreover, we establish a precise characterization of both the statistical and the algorithmic rates of convergence. As a byproduct, our results provide theoretical justification for the trick of using a target network in DQN. Furthermore, we extend DQN to two-player zero-sum Markov games by proposing the Minimax-DQN algorithm. Utilizing the analysis of DQN, we establish theoretical guarantees for Minimax-DQN. To further extend this work, one future direction is to analyze reinforcement learning methods targeting at MDP with continuous action spaces, e.g., example, soft Q-learning (Haarnoja et al., 2017) and deep deterministic policy gradient (DDPG) (Lillicrap et al., 2016). Another promising direction is to combine results on optimization for deep learning with our statistical analysis to gain a unified understanding of the statistical and computational aspects of DQN.

A Deep Q-Network

We first present the DQN algorithm for MDP in details, which is proposed by Mnih et al. (2015) and adapted here to discounted MDP. As shown in Algorithm 3 below, DQN features two key tricks that lead to its empirical success, namely, experience replay and target network.

Algorithm 3 Deep Q-Network (DQN)

Input: MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, replay memory $\mathcal{M}$, number of iterations T, minibatch size n, exploration probability $\epsilon \in (0,1)$, a family of deep Q-networks $Q_\theta \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, an integer T_{target} for updating the target network, and a sequence of stepsizes $\{\alpha_t\}_{t \geq 0}$.

Initialize the replay memory $\mathcal{M}$ to be empty.

Initialize the Q-network with random weights θ.

Initialize the weights of the target network with $\theta^\star = \theta$.

Initialize the initial state S_0.

for $t = 0, 1, \ldots, T$ **do**

With probability ϵ, choose A_t uniformly at random from $\mathcal{A}$, and with probability $1-\epsilon$, choose A_t such that $Q_\theta(S_t, A_t) = \max_{a \in \mathcal{A}} Q_\theta(S_t, a)$.

Execute A_t and observe reward R_t and the next state S_{t+1}.

Store transition (S_t, A_t, R_t, S_{t+1}) in $\mathcal{M}$.

Experience replay: Sample random minibatch of transitions $\{(s_i, a_i, r_i, s_i')\}_{i \in [n]}$ from $\mathcal{M}$.

For each $i \in [n]$, compute the target $Y_i = r_i + \gamma \cdot \max_{a \in \mathcal{A}} Q_{\theta^\star}(s_i', a)$.

Update the Q-network: Perform a gradient descent step

$$\theta \leftarrow \theta - \alpha_t \cdot \frac{1}{n} \sum_{i \in [n]} \big[Y_i - Q_\theta(s_i, a_i) \big] \cdot \nabla_\theta Q_\theta(s_i, a_i).$$

Update the target network: Update $\theta^\star \leftarrow \theta$ every T_{target} steps.

end for

Define policy $\overline{\pi}$ as the greedy policy with respect to Q_θ.

Output: Action-value function Q_θ and policy $\overline{\pi}$.

Furthermore, in the following, we present the details of the Minimax-DQN algorithm that extends DQN to two-player zero-sum Markov games introduced in §5. Similar to DQN, this algorithm also utilizes the experience replay and target networks. The main difference is that here the target Y_i in (5.7) is obtained by solving a zero-sum matrix game. In Algorithm 4 we present the algorithm for the second player, which can be easily modified for the first player. We note that for the second player, similar to (5.6),the equilibrium joint policy is defined as

$$\big[\widetilde{\pi}_Q(\cdot \,|\, s), \widetilde{\nu}_Q(\cdot \,|\, s)\big] = \operatorname*{argmax}_{\nu' \in \mathcal{P}(\mathcal{B})} \operatorname*{argmin}_{\pi' \in \mathcal{P}(\mathcal{A})} \mathbb{E}_{a \sim \pi', b \sim \nu'} \big[Q(s,a,b)\big], \qquad \forall s \in \mathcal{S}. \tag{A.1}$$

Algorithm 4 Minimax Deep Q-Network (Minimax-DQN) for the second player

Input: Zero-Sum Markov game $(\mathcal{S}, \mathcal{A}, \mathcal{B}, P, R, \gamma)$, replay memory $\mathcal{M}$, number of iterations T, minibatch size n, exploration probability $\epsilon \in (0,1)$, a family of deep Q-networks $Q_\theta \colon \mathcal{S} \times \mathcal{A} \times \mathcal{B} \to \mathbb{R}$, an integer T_{target} for updating the target network, and a sequence of stepsizes $\{\alpha_t\}_{t \geq 0}$.

Initialize the replay memory $\mathcal{M}$ to be empty.

Initialize the Q-network with random weights θ.

Initialize the weights of the target network by letting $\theta^\star = \theta$.

Initialize the initial state S_0.

for $t = 0, 1, \ldots, T$ **do**

With probability ϵ, choose B_t uniformly at random from $\mathcal{B}$, and with probability $1-\epsilon$, sample B_t according to the equilibrium policy $\widetilde{\nu}_{Q_\theta}(\cdot \,|\, S_t)$ defined in (A.1).

Execute B_t and observe the first player's action A_t, reward R_t satisfying $-R_t \sim R(S_t, A_t, B_t)$, and the next state $S_{t+1} \sim P(\cdot \,|\, S_t, A_t, B_t)$.

Store transition $(S_t, A_t, B_t, R_t, S_{t+1})$ in $\mathcal{M}$.

Experience replay: Sample random minibatch of transitions $\{(s_i, a_i, b_i, r_i, s_i')\}_{i \in [n]}$ from $\mathcal{M}$.

For each $i \in [n]$, compute the target

$$Y_i = r_i + \gamma \cdot \max_{\nu' \in \mathcal{P}(\mathcal{B})} \min_{\pi' \in \mathcal{P}(\mathcal{A})} \mathbb{E}_{a \sim \pi', b \sim \nu'} \big[Q_{\theta^*}(s_i', a, b) \big].$$

Update the Q-network: Perform a gradient descent step

$$\theta \leftarrow \theta - \alpha_t \cdot \frac{1}{n} \sum_{i \in [n]} \big[Y_i - Q_\theta(s_i, a_i, b_i) \big] \cdot \nabla_\theta Q_\theta(s_i, a_i, b_i).$$

Update the target network: Update $\theta^\star \leftarrow \theta$ every T_{target} steps.

end for

Output: Q-network Q_θ and equilibrium joint policy with respect to Q_θ.

B Computational Aspect of DQN

Recall that in Algorithm 1 we assume the global optima of the nonlinear least-squares problem in (3.1) is obtained in each iteration. We make such an assumption as our focus is on the statistical analysis. In terms of optimization, it has been shown recently that, when the neural network is overparametrized, (stochastic) gradient descent converges to the global minima of the empirical function. Moreover, the generalization error of the obtained neural network can also be established. The intuition behind these results is that, when the neural network is overparametrized, it behaves similar to the random feature model (Rahimi and Recht, 2008, 2009). See, e.g., Du et al. (2019b,a); Zou et al. (2018); Chizat et al. (2019); Allen-Zhu et al. (2019a,b); Jacot et al. (2018); Cao and Gu (2019); Arora et al. (2019); Weinan et al. (2019); Mei et al. (2019); Yehudai and Shamir (2019); Bietti and Mairal (2019); Yang and Salman (2019); Yang (2019); Gao et al. (2019); Bai and Lee (2019); Huang et al. (2020) and the references therein. Also see Fan et al. (2019) for a detailed sur-

vey. In this section, we make an initial attempt in providing a unified statistical and computational analysis of DQN.

In the sequel, we consider the reinforcement learning problem with the state space $\mathcal{S} = [0,1]^r$ and a finite action space $\mathcal{A}$. To simplify the notation, we represent action a using one-hot embedding and thus identify it as an element in $\{0,1\}^{|\mathcal{A}|} \subseteq \mathbb{R}^{|\mathcal{A}|}$. In practice, categorical actions are often embedded into the Euclidean space (Dulac-Arnold et al., 2015). Thus, we can pack the state s and the action a together and obtain a vector (s,a) in $\mathbb{R}^d$, where we denote $r + |\mathcal{A}|$ by d. Moreover, without loss of generality, we assume that $\|(s,a)\|_2 \le 1$.

We represent the Q-network by the family of two-layer neural networks

$$Q\big(s,a;b,W\big) = \frac{1}{\sqrt{2m}} \sum_{j=1}^{2m} b_j \cdot \sigma[W_j^\top (s,a)], \qquad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{B.1}$$

Here $2m$ is the number of neurons, $b_j \in \mathbb{R}$ and $W_j \in \mathbb{R}^d$ for all $j \in [2m]$, and $\sigma(u) = \max\{u, 0\}$ is the ReLU activation function. Here $b = (b_1, \ldots, b_{2m})^\top \in \mathbb{R}^{2m}$ and $W = (W_1, \ldots, W_{2m}) \in \mathbb{R}^{d \times 2m}$ are the weights of the neural network.

For such class of neural networks, for any $k \ge 1$, in k-th iteration of the neural FQI algorithm, the optimization problem in (3.1) becomes

$$\underset{b,W}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^{n} \big[Y_i - Q(S_i, A_i; b, W)\big]^2, \tag{B.2}$$

where $Y_i = R_i + \gamma \cdot \max_{a \in \mathcal{A}} \widetilde{Q}_{k-1}(S_i', a)$ is the target and $\widetilde{Q}_{k-1}$ is the Q-network computed in the previous iteration. Notice that this problem is a least-squares regression with overparameterized neural networks. For computational efficiency, we propose to solve (B.2) via stochastic gradient descent (SGD). Specifically, in each iteration of SGD, we sample a fresh observation (S, A, R, S') with (S, A) drawn from the sampling distribution σ, $R \sim R(\cdot \,|\, S, A)$, and $S' \sim P(\cdot \,|\, S, A)$. Then an estimator of the gradient is computed based on (S, A, R, S'), which is used to update the network parameters. We run the SGD updates for a total of n iterations and denote the output by $\widetilde{Q}_k$.

Besides, in each FQI-step, we initialize the parameters via the symmetric initialization scheme (Gao et al., 2019; Bai and Lee, 2019) as follows. For any $j \in [m]$, we set $b_j \overset{\text{i.i.d.}}{\sim} \text{Unif}(\{-1, 1\})$ and $W_j \overset{\text{i.i.d.}}{\sim} N(0, I_d/d)$, where I_d is the identity matrix in $\mathbb{R}^d$. For any $j \in \{m+1, \ldots, 2m\}$, we set $b_j = -b_{j-m}$ and $W_j = W_{j-m}$. We remark that such initialization implies that the initial Q-network is a zero function, which is used only to simply the theoretical analysis. Besides, for ease of presentation, during training we fix the value of b at its initial value and only optimize over W. We initialize b and W at the very beginning of our algorithm and in each FQI subproblem, we update the Q-network starting from the same initialization. Hereafter, we denote the initial value of W and b by $W^{(0)} \in \mathbb{R}^{d \times 2m}$ and $b^{(0)} \in \mathbb{R}^{2m}$, respectively, and let $Q(\cdot, \cdot; W)$ denote $Q(\cdot, \cdot; b^{(0)}, W)$. In order to have bounded functions, we further restrict the weight W to a Frobenius norm ball centered at $W^{(0)}$ with radius $B > 0$, i.e., we define

$$\mathcal{B}_B = \big\{W \in \mathbb{R}^{d \times 2m} \colon \|W - W^{(0)}\|_{\text{fro}} \le B\big\}, \tag{B.3}$$

where B is a sufficiently large constant. Thus, the population version of the k-th iteration of the FQI algorithm becomes

$$\underset{W\in\mathcal{B}_B}{\text{minimize}}\, L(W) = \mathbb{E}\big\{\big[Y - Q(S,A;W)\big]^2\big\}, \tag{B.4}$$

where $(S,A)\sim\sigma$ and Y is computed using $\widetilde{Q}_{k-1}$. We solve this optimization problem via projected SGD, which generates a sequence of weight matrices $\{W^{(t)}\}_{t\geq 0}\subseteq\mathcal{B}_B$ satisfying

$$W^{(t)} = \Pi_{\mathcal{B}_B}\Big[W^{(t-1)} - \eta\cdot\big[Y_t - Q\big(S_t,A_t;W^{(t-1)}\big)\big]\cdot\nabla_W Q\big(S_t,A_t;W^{(t-1)}\big)\Big],\quad \forall t\geq 1, \tag{B.5}$$

where $\Pi_{\mathcal{B}_B}$ is the projection operator onto $\mathcal{B}_B$ with respect to the Frobenius norm, $\eta>0$ is the step size, and (S_t,A_t,Y_t) is a random observation. We present the details of fitted Q-iteration method with projected SGD in Algorithm 5.

Algorithm 5 Fitted Q-Iteration Algorithm with Projected SGD Updates

Input: MDP $(\mathcal{S},\mathcal{A},P,R,\gamma)$, function class $\mathcal{F}$, sampling distribution σ, number of FQI iterations K, number of SGD iterations T, the initial estimator $\widetilde{Q}_0$.

Initialize the weights $b^{(0)}$ and $W^{(0)}$ of Q-network via the symmetric initialization scheme.

for $k=0,1,2,\ldots,K-1$ **do**

for $t=1,\ldots,T$ **do**

Draw an independent sample (S_t,A_t,R_t,S_t') with (S_t,A_t) drawn from distribution σ.

Compute $Y_t = R_t + \gamma\cdot\max_{a\in\mathcal{A}}\widetilde{Q}_k(S_t',a)$.

Perform projected SGD update

$$\widetilde{W}^{(t)} \leftarrow W^{(t-1)} - \eta\cdot\big[Y_t - Q\big(S_t,A_t;W^{(t-1)}\big)\big]\cdot\nabla_W Q\big(S_t,A_t;W^{(t-1)}\big)$$
$$W^{(t)} \leftarrow \Pi_{\mathcal{B}_B}\big(\widetilde{W}^{(t)}\big) = \underset{W\in\mathcal{B}_B}{\operatorname{argmin}}\,\|W-\widetilde{W}^{(t)}\|_{\text{fro}}.$$

end for

Update the action-value function $\widetilde{Q}_{k+1}(\cdot,\cdot)\leftarrow Q(\cdot,\cdot;W_{k+1})$ where $W_{k+1} = T^{-1}\sum_{t=1}^{T}W^{(t)}$.

end for

Define policy π_K as the greedy policy with respect to $\widetilde{Q}_K$.

Output: An estimator $\widetilde{Q}_K$ of Q^* and policy π_K.

To understand the convergence of the projected SGD updates in (B.5), we utilize the fact that the dynamics of training overparametrized neural networks is captured by the neural tangent kernel (Jacot et al., 2018) when the width is sufficiently large. Specifically, since $\sigma(u) = u\cdot\mathbb{1}\{u>0\}$, the gradient of the Q-network in (B.1) is given by

$$\nabla_{W_j}Q(s,a;b,W) = 1/\sqrt{2m}\cdot b_j\cdot\mathbb{1}\{W_j^\top(s,a)>0\}\cdot(s,a),\qquad \forall j\in[2m]. \tag{B.6}$$

Recall that we initialize parameters b and W as $b^{(0)}$ and $W^{(0)}$ and that we only update W during training. We define a function class $\mathcal{F}_{B,m}^{(t)}$ as

$$\mathcal{F}_{B,m}^{(t)} = \bigg\{\widehat{Q}(s,a;W) = \frac{1}{\sqrt{2m}}\sum_{j=1}^{2m}b_j^{(0)}\cdot\mathbb{1}\big\{(W_j^{(t)})^\top(s,a)>0\big\}\cdot W_j^\top(s,a)\colon W\in\mathcal{B}_B\bigg\}. \tag{B.7}$$

By (B.6), for each function $\widehat{Q}(\cdot,\cdot;W) \in \mathcal{F}_{B,m}^{(t)}$, we can write it as

$$\widehat{Q}(\cdot,\cdot;W) = Q(\cdot,\cdot;W^{(t)}) + \big\langle \nabla_W Q(\cdot,\cdot;W^{(t)}), W - W^{(t)} \big\rangle, \qquad \forall W \in \mathcal{B}_B,$$

which is the first-order linearization of $Q(\cdot,\cdot;W^{(t)})$ at $W^{(t)}$. Furthermore, since B in (B.3) is a constant, for each weight matrix W in $\mathcal{B}_B$, when m goes to infinity, $\|W_j - W_j^{(0)}\|_2$ would be small for almost all $j \in [2m]$, which implies that $\mathbb{1}\{W_j^\top(s,a) > 0\} = \mathbb{1}\{(W_j^{(0)})^\top(s,a) > 0\}$ holds with high probability for all $j \in [2m]$ and $(s,a) \in \mathcal{S} \times \mathcal{A}$. As a result, when m is sufficiently large, $\mathcal{F}_{B,m}^{(t)}$ defined in (B.7) is close to

$$\mathcal{F}_{B,m}^{(0)} = \bigg\{ \widehat{Q}(s,a;W) = \frac{1}{\sqrt{2m}} \sum_{j=1}^{2m} b_j^{(0)} \cdot \mathbb{1}\big\{(W_j^{(0)})^\top(s,a) > 0\big\} \cdot W_j^\top(s,a) \colon W \in \mathcal{B}_B \bigg\}, \tag{B.8}$$

where $b^{(0)}$ and $W^{(0)}$ are the initial parameters. Specifically, as proved in Lemma A.2 in Wang et al. (2019), when the sampling distribution σ is regular in the sense that Assumption (B.2) specified below is satisfied, for any $W_1, W_2 \in \mathcal{B}_B$, we have

$$\mathbb{E}_{\text{init}} \Big[\big\| \langle \nabla_W Q(\cdot,\cdot;W_1) - \nabla_W Q(\cdot,\cdot;W^{(0)}), W_2 \rangle \big\|_\sigma^2 \Big] = \mathcal{O}(B^3 \cdot m^{-1/2}),$$

where $\mathbb{E}_{\text{init}}$ denotes that the expectation is taken with respect to the initialization of the network parameters. Thus, when the network width $2m$ is sufficiently large such that $B^3 \cdot m^{-1/2} = o(1)$, the linearized function classes $\{\mathcal{F}_{B,m}^{(t)}\}_{t\in[T]}$ are all close to $\mathcal{F}_{B,m}^{(0)}$.

To simplify the notation, for $b \in \{-1, 1\}$ and $W \in \mathbb{R}^d$, we define feature mapping $\phi(\cdot,\cdot;b,W) \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ as

$$\phi(s,a;b,W) = b \cdot \mathbb{1}\{W^\top(s,a) > 0\} \cdot (s,a), \qquad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{B.9}$$

Besides, for all $j \in [2m]$, we let ϕ_j denote $\phi(\cdot,\cdot;b_j^{(0)}, W_j^{(0)})$. Due to the symmetric initialization scheme, $\{\phi_j\}_{j\in[m]}$ are i.i.d. random feature functions and $\phi_j = -\phi_{j+m}$ for all $j \in [m]$. Thus, each $\widehat{Q}(\cdot,\cdot;W)$ in (B.8) can be equivalently written as

$$\widehat{Q}(s,a;W) = \frac{1}{\sqrt{2m}} \sum_{j=1}^{2m} \phi_j(s,a)^\top W_j = \frac{1}{\sqrt{2m}} \sum_{j=1}^{m} \phi_j(s,a)^\top (W_j - W_{j+m}). \tag{B.10}$$

Let $W_j' = (W_j - W_{j+m})/\sqrt{2}$. Since $W \in \mathcal{B}_B$, we have

$$\sum_{j=1}^{m} \|W_j'\|_2^2 = \frac{1}{2} \sum_{j=1}^{m} \big\| (W_j - W_j^{(0)}) - (W_{j+m} - W_{j+m}^{(0)}) \big\|_2^2 \le \sum_{j=1}^{2m} \|W_j - W_j^{(0)}\|^2 \le B^2, \tag{B.11}$$

where we use the fact that $W_j^{(0)} = W_{j+m}^{(0)}$ for all $j \in [m]$. Thus, combining (B.10) and (B.11), we conclude that $\mathcal{F}_{B,m}^{(0)}$ in (B.8) is a subset of $\mathcal{F}_{B,m}$ defined as

$$\mathcal{F}_{B,m} = \bigg\{ \widehat{Q}(s,a;W) = \frac{1}{\sqrt{m}} \sum_{j=1}^{m} \phi_j(s,a)^\top W_j \colon W \in \{W' \in \mathbb{R}^{d\times m} \colon \|W'\|_{\text{fro}} \le B\} \bigg\}. \tag{B.12}$$

Notice that each function in $\mathcal{F}_{B,m}$ is a linear combination of m i.i.d. random features. In particular, let $\beta \in \mathrm{Unif}(\{-1,1\})$ and $\omega \sim N(I_d/d)$ be two independent random variables and let μ denote their joint distribution. Then the random feature $\phi(\cdot,\cdot;\beta,\omega)$ induces a reproducing kernel Hilbert space $\mathcal{H}$ (Rahimi and Recht, 2008, 2009; Bach, 2017) with kernel $K\colon (\mathcal{S}\times\mathcal{A})\times(\mathcal{S}\times\mathcal{A})\to\mathbb{R}$ given by

$$\begin{aligned} K\big[(s,a),(s',a')\big] &= \mathbb{E}_\mu\big[\phi(s,a;\beta,\omega)^\top\phi(s,a;\beta,\omega)\big] \\ &= \mathbb{E}_\omega\big[\mathbb{1}\{\omega^\top(s,a)>0\}\cdot\mathbb{1}\{\omega^\top(s',a')>0\}\cdot(s,a)^\top(s',a')\big]. \end{aligned} \tag{B.13}$$

Each function in $\mathcal{H}$ can be represented by a mapping $\alpha\colon\{-1,1\}\times\mathbb{R}^d\to\mathbb{R}^d$ as

$$f_\alpha(\cdot,\cdot) = \int_{\{-1,1\}\times\mathbb{R}^d} \phi(\cdot,\cdot;b,W)^\top\alpha(b,W)\,\mathrm{d}\mu(b,W) = \mathbb{E}_\mu\big[\phi(\cdot,\cdot;\beta,\omega)^\top\alpha(\beta,\omega)\big].$$

For two functions f_{α_1} and f_{α_2} in $\mathcal{H}$ represented by α_1 and α_2, respectively, their inner product is given by

$$\langle f_{\alpha_1}, f_{\alpha_2}\rangle_{\mathcal{H}} = \int_{\{-1,1\}\times\mathbb{R}^d} \alpha_1(b,W)^\top\alpha_2(b,W)\,\mathrm{d}\mu(b,W) = \mathbb{E}_\mu\big[\alpha_1(\beta,\omega)^\top\alpha_2(\beta,\omega)\big].$$

We let $\|\cdot\|_{\mathcal{H}}$ denote the RKHS norm of $\mathcal{H}$. Then, when m goes to infinity, $\mathcal{F}_{B,m}$ in (B.12) converges to the RKHS norm ball $\mathcal{H}_B = \{f\in\mathcal{H}\colon \|f\|_{\mathcal{H}}\le B\}$.

Therefore, from the perspective of neural tangent kernel, when the Q-network is represented by the class of overparametrized neural networks given in (B.1) with a sufficiently large number of neurons, each population problem associated with each FQI iteration in (B.4) becomes

$$\underset{Q\in\mathcal{H}}{\mathrm{minimize}}\,\mathbb{E}\big\{[Q(S,A)-Y]^2\big\},$$

where the minimization is over a subset of $\mathcal{H}_B$ as $\mathcal{F}_{B,m}^{(0)}$ is a subset of $\mathcal{F}_{B,m}$.

Utilizing the connection between neural network training and RKHS, in the sequel, we provide a jointly statistical and computational analysis of Algorithm 5. To this end, we define a function class $\mathcal{G}_B$ as

$$\mathcal{G}_B = \bigg\{ f_\alpha(s,a) = \int_{\{-1,1\}\times\mathbb{R}^d} \phi(s,a;b,W)^\top\alpha(b,W)\,\mathrm{d}\mu(b,W)\colon \|\alpha(\cdot,\cdot)\|_\infty \le B/\sqrt{d}\bigg\}. \tag{B.14}$$

That is, each function in $\mathcal{G}_B$ is represented by a feature mapping $\alpha\colon\{-1,1\}\times\mathbb{R}^d\to\mathbb{R}^d$ which is almost surely bounded in the ℓ_∞-norm. Thus, $\mathcal{G}_B$ is a strict subset of the RKHS-norm ball $\mathcal{H}_B$. When B is sufficiently large, $\mathcal{G}_B$ is known to be a rich function class (Hofmann et al., 2008). Similar to Assumption 4.2 in §4, we impose the following assumption on the Bellman optimality operator.

Assumption B.1 (Completeness of $\mathcal{G}_B$)**.** We assume that that $TQ(\cdot,\cdot;W)\in\mathcal{G}_B$ for all for all $W\in\mathcal{B}_B$, where T is the Bellman optimality operator and $Q(\cdot,\cdot;W)$ is given in (B.1).

This assumption specifies that T maps any neural network with weight matrix W in $\mathcal{B}_B$ to a subset $\mathcal{G}_B$ of the RKHS $\mathcal{H}$. When m is sufficiently large, this assumption is similar to stating that $\mathcal{G}_B$ is approximately closed under T.

We also impose the following regularity condition on the sampling distribution σ.

Assumption B.2 (Regularity Condition on σ)**.** We assume that there exists an absolute constant $C > 0$ such that

$$\mathbb{E}_\sigma\Big[\mathbb{1}\big\{|y^\top(S,A)| \le u\big\}\Big] \le C \cdot u/\|y\|_2, \qquad \forall y \in \mathbb{R}^d,\ \forall u > 0.$$

Assumption B.2 states that the density of σ is sufficiently regular, which holds when the density is upper bounded.

Now we are ready to present the main result of this section, which characterizes the performance of π_K returned by Algorithm 5.

Theorem B.3. In Algorithm 5, we assume that each step of the fitted-Q iteration is solved by T steps of projected SGD updates with a constant stepsize $\eta > 0$. We set $T = C_1 m$ and $\eta = C_2/\sqrt{T}$, where C_1, C_2 are absolute constants that are properly chosen. Then, under Assumptions 4.3, B.1, and B.2, we have

$$\mathbb{E}_{\text{init}}\big[\|Q^* - Q^{\pi_K}\|_{1,\mu}\big] = \mathcal{O}\bigg(\frac{\phi_{\mu,\sigma}\gamma}{(1-\gamma)^2} \cdot (B^{3/2} \cdot m^{-1/4} + B^{5/4} \cdot m^{-1/8}) + \frac{\gamma^{K+1}}{(1-\gamma)^2} \cdot R_{\max}\bigg), \quad \text{(B.15)}$$

where $\mathbb{E}_{\text{init}}$ denotes that the expectation is taken with respect to the randomness of the initialization.

As shown in (B.15), the error $\mathbb{E}_{\text{init}}[\|Q^* - Q^{\pi_K}\|_{1,\mu}]$ can be similarly written as the sum of a statistical error and an algorithmic error, where the algorithmic error converges to zero at a linear rate as K goes to infinity. The statistical error corresponds to the error incurred in solving each FQI step via T projected SGD steps. As shown in (B.15), when B is regarded as a constant, with $T \asymp m$ projected SGD steps, we obtain an estimator with error $\mathcal{O}(m^{-1/8})$. Hence, Algorithm 5 finds the globally optimal policy when both m and K goes to infinity. Therefore, when using overparametrized neural networks, our fitted Q-iteration algorithm provably attains both statistical accuracy and computational efficiency.

Finally, we remark that focus on the class of two-layer overparametrized ReLU neural networks only for the simplicity of presentation. The theory of neural tangent kernel can be extended to feedforward neural networks with multiple layers and neural networks with more complicated architectures (Gao et al., 2019; Frei et al., 2019; Yang and Salman, 2019; Yang, 2019; Huang et al., 2020).

B.1 Proof of Theorem B.3

Proof. Our proof is similar to that of Theorem 4.4. For any $k \in [K]$, we define the maximum one-step approximation error as $\varepsilon_{\max} = \max_{k\in[K]} \mathbb{E}_{\text{init}}[\|T\widetilde{Q}_{k-1} - \widetilde{Q}_k\|_\sigma]$, where $\mathbb{E}_{\text{init}}$ denotes that the expectation is taken with respect to the randomness in the initialization of network weights, namely $b^{(0)}$ and $W^{(0)}$. By Theorem 6.1, we have

$$\mathbb{E}_{\text{init}}\big[\|Q^* - Q^{\pi_K}\|_{1,\mu}\big] \le \frac{2\phi_{\mu,\sigma}\gamma}{(1-\gamma)^2} \cdot \varepsilon_{\max} + \frac{4\gamma^{K+1}}{(1-\gamma)^2} \cdot R_{\max}, \quad \text{(B.16)}$$

where $\phi_{\mu,\sigma}$, specified in Assumption 4.3, is a constant that only depends on the concentration coefficients. Thus, it remains to characterize $\|T\widetilde{Q}_{k-1} - \widetilde{Q}_k\|_\sigma$ for each k, which corresponds to the prediction risk of the estimator constructed by T projected SGD steps.