SIAM J. MATH. DATA SCI.
Vol. 5, No. 4, pp. 1078–1101

Target Network and Truncation Overcome the Deadly Triad in Q-Learning*

Zaiwei Chen†, John-Paul Clarke‡, and Siva Theja Maguluri§

Abstract. Q-learning with function approximation is one of the most empirically successful while theoretically mysterious reinforcement learning (RL) algorithms and was identified in [R. S. Sutton, in *European Conference on Computational Learning Theory*, Springer, New York, 1999, pp. 11–17] as one of the most important theoretical open problems in the RL community. Even in the basic setting where linear function approximation is used, there are well-known divergent examples. In this work, we propose a stable online variant of Q-learning with linear function approximation that uses target network and truncation and is driven by a single trajectory of Markovian samples. We present the finite-sample guarantees of the algorithm, which imply a sample complexity of $\tilde{\mathcal{O}}(\epsilon^{-2})$ up to a function approximation error. Importantly, we establish the results under minimal assumptions and do not modify the problem parameters to achieve stability.

Key words. reinforcement learning, Q-learning, linear function approximation, finite-sample analysis

MSC codes. 60J20, 93E20, 90C40, 62L20

DOI. 10.1137/22M1499261

1. Introduction.

The Deep Q-Network [33], as a special instance of Q-learning with function approximation, is one of the most empirically successful algorithms to solve the reinforcement learning (RL) problem. On the other hand, the behavior of Q-learning with function approximation is in general not theoretically well understood and was identified in [39] as an open problem. In fact, the infamous deadly triad [40] is present in Q-learning with function approximation. Consequently, even in the basic setting where linear function approximation is used, the algorithm can experience divergence [2, 11].

Although theoretically unclear, it was empirically evidenced from [33] that the three ingredients of experience replay, target network, and truncation together overcome the divergence of Q-learning with function approximation. In this work, we focus on Q-learning with linear function approximation for infinite-horizon discounted Markov decision processes (MDPs) and show theoretically that target network and truncation are sufficient to provably stabilize Q-learning. The main contributions of this work are summarized in the following.

*Received by the editors June 3, 2022; accepted for publication (in revised form) September 5, 2023; published electronically December 7, 2023.

https://doi.org/10.1137/22M1499261

Funding: This work was partially supported by NSF grant EPCN-2144316, CPS-2240982, CMMI-2112533, and RTX.

†Computing + Mathematical Sciences, California Institute of Technology, Pasadena, CA 91106 USA (zchen458@caltech.edu).

‡Aerospace Engineering and Engineering Mechanics, University of Texas at Austin, Austin, TX 78712 USA (johnpaul@utexas.edu).

§Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (siva.theja@gatech.edu).

Stable algorithm design. We propose a stable variant of Q-learning with linear function approximation that uses target network and truncation. Notably, our algorithm can be implemented in an online fashion using a single trajectory of Markovian samples.

Finite-sample guarantees. We establish the finite-sample guarantees of our proposed algorithm to the optimal Q-function Q^* up to a function approximation error. The result implies a sample complexity of $\tilde{\mathcal{O}}(\epsilon^{-2})$, which matches with the sample complexity of Q-learning in the tabular setting and is known to be optimal up to a logarithmic factor. The function approximation error in our finite-sample bound well captures the approximation power of the chosen function class. In the special case of tabular setting, our result implies asymptotic convergence in the mean-square sense to Q^*.

Broad applicability. In the existing literature, to stabilize Q-learning with linear function approximation, one usually requires strong assumptions on the underlying MDP and/or the approximating function class. These assumptions include but are not limited to the function class being complete with respect to the Bellman operator, the MDP being linear (or approximately linear), a negative drift assumption, etc. In this work, we do not require any of these assumptions. In fact, our result holds as long as the policy used to collect samples enables the agent to sufficiently explore the state-action space, which is, to some extent, a necessary requirement to find an optimal policy in RL.

1.1. Related work. The Q-learning algorithm was first proposed in [46]. Since then, theoretically understanding the behavior of Q-learning has been a major topic in the RL community. In particular, the asymptotic convergence of Q-learning was established in [42, 21, 6, 25]. Beyond the asymptotic behavior, recently there has been an increasing interest in studying the finite-sample convergence guarantees of Q-learning. See [10, 5, 29, 35, 28] and the references therein. Other variants of Q-learning such as zap Q-learning and double Q-learning were proposed and studied in [12] and [20], respectively.

When using function approximation, the deadly triad (which refers to function approximation, off-policy sampling, and bootstrapping) [40] appears in Q-learning, and the algorithm can be unstable even under simple linear function approximation. This is evidenced by the divergent MDP example constructed in [2]. There are many attempts to stabilize Q-learning with linear function approximation, which are summarized below.

Strong negative drift assumption. The asymptotic convergence of Q-learning with linear function approximation was established in [32] under a "negative drift assumption." Under similar assumptions, finite-sample analysis of Q-learning, as well as its on-policy variant Sarsa, was performed in [11, 18, 25, 55] for using linear function approximation, and in [48, 7] for using neural network approximation. However, such a negative drift assumption is, in general, hard to satisfy unless the discount factor of the MDP is extremely small. See our online report [9] for a more detailed explanation. In this work, we do not require such a negative drift assumption or any of its variants to stabilize Q-learning with linear function approximation.

Q-learning with target network. Recently, new convergent variants of Q-learning with linear function approximation were proposed in [8, 54, 1, 53],[1] all of which make use of the target network, a heuristic developed in [33]. In [8, 54], the stability of Q-learning is achieved at the

[1] [53] is a concurrent work. A detailed comparison between our work and [53] is presented in the supplementary materials (supplement.pdf [local/web 301KB]).

cost of implicitly changing the discount factor of the MDP. Specifically, the problem being effectively solved is no longer the original MDP, but an MDP with a much smaller discount factor, which is why the algorithms in [8, 54] do not converge to the optimal Q-function Q^* even in the tabular setting. See our online report [9] for more details. In this work, we do not modify the original problem parameters to achieve stability. Moreover, in the special case of tabular RL, we have the convergence to Q^*. A closely related work [1] provides finite-sample analysis of Q-learning with linear function approximation using the target network under the Bellman completeness assumption (or an approximate variant of it). In contrast, we introduce truncation of the iterates which leads to stability by ensuring the finiteness of a variant of the inherent Bellman error. We also illustrate the importance of the truncation step by presenting a counterexample showing divergence in the absence of truncation. Moreover, compared with [1], we have an improved dependence on the function approximation error and an improved sample complexity in terms of its dependence on the effective horizon of the problem. On the other hand, [1] develops a sampling technique called reverse experience replay to improve the dependence on the mixing time of the underlying Markov chain. We will discuss more about our contribution relative to [1] after presenting our main results.

The Greedy-GQ algorithm. A two-time-scale variant of Q-learning with linear function approximation, known as Greedy-GQ, was proposed in [31]. The algorithm is designed based on minimizing the projected Bellman error using stochastic gradient descent. Although the Greedy-GQ algorithm is stable without needing the negative drift assumption, since the Bellman error is in general nonconvex, the Greedy-GQ algorithm can only guarantee convergence to stationary points. As a result, there are no performance guarantees on how well the limit point approximates the optimal Q-function Q^*. Although finite-sample analysis for Greedy-GQ was recently performed in [45, 30, 49], due to the lack of global optimality, the error bounds were on the gradient of the Bellman error rather than the distance to Q^*. In this work, we provide finite-sample guarantees to the optimal Q-function Q^* (up to a function approximation error).

Fitted Q-iteration and its variants. Fitted Q-iteration is proposed in [16] as an offline variant of Q-learning. The polynomial sample complexity of fitted Q-iteration (or more generally fitted value iteration) was established in [41, 34, 52] under bounded inherent Bellman error. The boundedness of the inherent Bellman error was imposed as an assumption in [52], while in [41, 34], the authors employed a truncation technique to ensure the boundedness of the inherent Bellman error. Such a truncation technique dates back to [19], which also inspires the truncation technique in this work. In the special case of linear function approximation, Q-learning with target network can be viewed as an approximate way of implementing fitted Q-iteration, where stochastic gradient descent was used as a way of performing such a fitting. Compared to [41, 34], the main difference of this work is that our algorithm allows for online implementation with a single trajectory of Markovian samples. Furthermore, we have an $\mathcal{O}(\epsilon^{-2})$ sample complexity instead of only polynomial sample complexity (i.e., $\tilde{\mathcal{O}}(\epsilon^{-n})$ for some positive integer n) [34]. More recently, [47] proposed a variant of batch RL algorithms called BVFT, where the authors establish an $\tilde{\mathcal{O}}(\epsilon^{-4})$ sample complexity under the realizability assumption. Another variant of fitted Q-iteration targeting finite-horizon MDPs was proposed in [14] using a distribution shift checking oracle, where polynomial sample complexity was established.

Linear MDP model. In the special case that the MDP has linear transition dynamics and linear reward, convergent variants of Q-learning with linear function approximation were designed and analyzed in [50, 23, 27] and the references therein. Such a linear model assumption can be relaxed to the case where the MDP is approximately linear. In this work, we do not make any structural assumptions on the underlying MDP, although at the cost of introducing a function approximation error in the finite-sample bound.

Other work. [13] studies Q-learning with function approximation for deterministic MDPs. The Deep Q-Network was studied in [17] under the Bellman completeness assumption.

2. Background.

We model the RL problem as an infinite-horizon discounted MDP defined by a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{P} = \{P_a \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|} \mid a \in \mathcal{A}\}$ is a set of *unknown* transition probability matrices, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto [0,1]$ is an *unknown* reward function, and $\gamma \in (0,1)$ is the discount factor.

Given a policy $\pi : \mathcal{S} \mapsto \Delta^{|\mathcal{A}|}$ (where $\Delta^{|\mathcal{A}|}$ denotes the $|\mathcal{A}|$-dimensional probability simplex), we define the state-action value function of π as $Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k \mathcal{R}(S_k, A_k) \mid S_0 = s, A_0 = a]$ for all (s,a), where we use $\mathbb{E}_\pi[\cdot]$ to indicate that the actions are chosen according to the policy π. The goal in RL is to find an optimal policy π^* so that its associated Q-function (denoted Q^*) is uniformly maximized for all (s,a). A well-known relation between the optimal Q-function and any optimal policy π^* states that $\pi^*(\cdot|s)$ is supported on the set $\arg\max_{a\in\mathcal{A}} Q^*(s,a)$ for all s [3]. Therefore, to find an optimal policy, it is enough to find the optimal Q-function.

The Q-learning algorithm is designed to find Q^* by solving the Bellman equation $Q^* = \mathcal{H}(Q^*)$ using stochastic approximation [36], where $\mathcal{H} : \mathbb{R}^{|\mathcal{S}||\mathcal{A}|} \mapsto \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ is the Bellman operator defined as $[\mathcal{H}(Q)](s,a) = \mathcal{R}(s,a) + \gamma\mathbb{E}[\max_{a'\in\mathcal{A}} Q(S_{k+1}, a') \mid S_k = s, A_k = a]$ for all Q and (s,a). While Q-learning provably converges, the algorithm becomes computationally intractable for MDPs with large state-action spaces. This motivates the use of function approximation, where the idea is to approximate the optimal Q-function from a prespecified function class.

Linear function approximation framework. In this work, we focus on using linear function approximation. Let $\phi_i \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, $i = 1, 2, \ldots, d$, be the basis vectors, and denote $\phi(s,a) = (\phi_1(s,a), \ldots, \phi_d(s,a)) \in \mathbb{R}^d$ for all (s,a). We assume without loss of generality that the basis vectors $\{\phi_i\}_{1\le i\le d}$ are linearly independent and normalized so that $\|\phi(s,a)\|_2 \le 1$ for all (s,a) [3]. Let $\Phi \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times d}$ be the feature matrix defined as $\Phi = [\phi_1, \phi_2, \ldots, \phi_d]$. Using the feature matrix Φ, the linear subspace spanned by $\{\phi_i\}_{1\le i\le d}$ can be compactly written as $\mathcal{W} = \{Q_\theta \mid Q_\theta = \Phi\theta,\ \theta \in \mathbb{R}^d\}$. With $\mathcal{W}$ defined, the goal of Q-learning with linear function approximation is to design a stable algorithm that provably finds an approximation of the optimal Q-function Q^* from the linear subspace $\mathcal{W}$.

3. Algorithm design and finite-sample analysis.

In this section, we first present the algorithm of Q-learning with linear function approximation using target network and truncation. Then we present the finite-sample guarantees.

3.1. Stable algorithm design.

To present our algorithm, we first introduce the truncation operator. Given a positive truncation radius $r > 0$, let $\mathcal{T} : \mathbb{R}^{|\mathcal{S}||\mathcal{A}|} \mapsto \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ be defined as $\mathcal{T}(Q)(s,a) = \mathrm{sgn}(Q(s,a))r$ when $|Q(s,a)| > r$, and $\mathcal{T}(Q)(s,a) = Q(s,a)$ when $|Q(s,a)| \le r$ for all $Q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ and (s,a). Since $\|Q^*\|_\infty \le 1/(1-\gamma)$, we choose $r = 1/(1-\gamma)$ to ensure that our proposed algorithm (presented in Algorithm 3.1) does not exclude the optimal Q-function.

Algorithm 3.1. Q-learning with linear function approximation: target network and truncation.

1: **Input:** Integers T, K, initializations $\hat{\theta}_0 = \mathbf{0}$, $\theta_{t,0} = \mathbf{0}$ for all t, behavior policy π_b.
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: **for** $k = 0, 1, \ldots, K-1$ **do**
4: Sample $A_k \sim \pi_b(\cdot|S_k)$ and observe $S_{k+1} \sim P_{A_k}(S_k, \cdot)$
5: $\theta_{t,k+1} = \theta_{t,k} + \alpha_k \phi(S_k, A_k)(\mathcal{R}(S_k, A_k) + \gamma \max_{a' \in \mathcal{A}} \mathcal{T}(\phi(S_{k+1}, a')^\top \hat{\theta}_t) - \phi(S_k, A_k)^\top \theta_{t,k})$
6: **end for**
7: $\hat{\theta}_{t+1} = \theta_{t,K}$ and $S_0 = S_K$
8: **end for**
9: **Output:** $\hat{\theta}_T$

In view of Algorithm 3.1, it is simple and easy to implement. In addition to $\{\theta_{t,k}\}$, we introduce $\{\hat{\theta}_t\}$ as the target network parameter, which is fixed in the inner loop where we update $\theta_{t,k}$. The target network was previously used in [33] as a heuristic for the design of the celebrated Deep Q-Network. Finally, before using the Q-function estimate associated with the target network in the inner loop, we first truncate it with radius $r = 1/(1-\gamma)$ (cf. Algorithm 3.1, line 5). Note that the location where we impose the truncation operator is different from that in the Deep Q-Network [33], where the truncation is performed for the entire temporal difference $\mathcal{R}(S_k, A_k) + \gamma \max_{a' \in \mathcal{A}} \phi(S_{k+1}, a')^\top \hat{\theta}_t - \phi(S_k, A_k)^\top \theta_{t,k}$ instead of only for $\phi(S_{k+1}, a')^\top \hat{\theta}_t$. Similar truncation techniques have been employed in [34, 22]. The reason that target network and truncation together ensure the stability of Q-learning with linear function approximation will be illustrated in detail in section 4.

On the practical side, Algorithm 3.1 uses a *single trajectory* of Markovian samples generated by the behavior policy π_b (see Algorithm 3.1, lines 4 and 7). Therefore, the agent does not have to constantly reset the system. Our result can be easily generalized to the case where one uses a time-varying behavior policy (i.e., the behavior policy is updated across the iterations of the target network) as long as it ensures sufficient exploration. For example, one can use the ϵ-greedy policy or the Boltzmann exploration policy with respect to the Q-function estimate associated with the target network $\hat{\theta}_t$ (i.e., $\Phi\hat{\theta}_t$) as the behavior policy.

3.2. Finite-sample guarantees.

To present the finite-sample bound, we first formally state our assumption and introduce the necessary notation.

Assumption 3.1. The behavior policy π_b satisfies $\pi_b(a|s) > 0$ for all (s, a) and induces an irreducible and aperiodic Markov chain $\{S_k\}$.

This assumption ensures that the behavior policy adequately explores the state-action space and is commonly imposed for value-based RL algorithms in the literature [43]. Since we work with finite MDPs, Assumption 3.1 implies that the Markov chain $\{S_k\}$ induced by π_b has a unique stationary distribution, denoted by $\mu \in \Delta^{|\mathcal{S}|}$, which satisfies $\mu(s) > 0$ for all $s \in \mathcal{S}$. In addition, the induced Markov chain $\{S_k\}$ mixes at a geometric rate [26], that is, there exist $C > 0$ and $\rho \in (0, 1)$ such that $\max_{s \in \mathcal{S}} d_{\mathrm{TV}}(P^k_{\pi_b}(s, \cdot), \mu(\cdot)) \leq C\rho^k$ for all $k \geq 0$, where P_{π_b} stands for the transition probability matrix of the Markov chain induced by π_b,

and $d_{\mathrm{TV}}(\cdot,\cdot)$ represents the total variation distance between probability distributions. For MDPs with continuous but compact state-action spaces, the geometric mixing is also needed in general; see, for example, [1, Assumption 3]. As a result of geometric mixing, letting $\tau_\delta = \min\{k \geq 0 : \max_{s\in\mathcal{S}} d_{\mathrm{TV}}(P_{\pi_b}^k(s,\cdot),\mu(\cdot)) \leq \delta\}$ be the mixing time of the induced Markov chain $\{S_k\}$ with precision $\delta > 0$, under Assumption 3.1, we have $\tau_\delta = \mathcal{O}(\log(1/\delta))$.

Notation. Let $D \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}||\mathcal{A}|}$ be a diagonal matrix with diagonal components being $\{\mu(s)\pi_b(a|s)\}_{(s,a)\in\mathcal{S}\times\mathcal{A}}$, and let $\|\cdot\|_D$ be a weighted ℓ_2-norm defined as $\|Q\|_D = (Q^\top DQ)^{1/2}$ for all $Q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$. Since D has positive diagonal components and Φ is full column rank, the matrix $\Phi^\top D\Phi$ is positive definite, the smallest eigenvalue of which is denoted by $\lambda_{\min}$. Let $\mathrm{Proj}_{\mathcal{W}}(\cdot)$ be the projection operator onto the linear subspace $\mathcal{W}$ with respect to the weighted ℓ_2-norm $\|\cdot\|_D$, which is a linear operator and is explicitly given as $\mathrm{Proj}_{\mathcal{W}}(Q) = \Phi(\Phi^\top D\Phi)^{-1}\Phi^\top DQ$ for all $Q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$. We define the function approximation error as

$$\mathcal{E}_{\mathrm{approx}} := \sup_{Q=\mathcal{T}(\Phi\theta),\,\theta\in\mathbb{R}^d} \|\mathcal{T}(\mathrm{Proj}_{\mathcal{W}}\mathcal{H}(Q)) - \mathcal{H}(Q)\|_\infty, \tag{3.1}$$

which captures the approximation power of the chosen function class. Our function approximation error $\mathcal{E}_{\mathrm{approx}}$ is closely related to the inherent Bellman error introduced in the existing literature [34, 52, 1, 44, 15]. A detailed discussion of the connection between $\mathcal{E}_{\mathrm{approx}}$ and the inherent Bellman error is presented in the supplementary materials (supplement.pdf [local/web 301KB]).

For simplicity, denote $\hat{Q}_t = \mathcal{T}(\Phi\hat{\theta}_t)$ as the truncated Q-function associated with the target network $\hat{\theta}_t$. We next present the finite-sample bounds of Algorithm 3.1, where the explicit requirement for choosing the stepsizes is presented in Appendix 5.2.

Theorem 3.2. *Under Assumption 3.1, we have the following results.*

(1) *When using properly chosen constant stepsize (i.e., $\alpha_k \equiv \alpha$), the following inequality holds for all $K \geq \tau_\alpha$ and $T \geq 0$:*

$$\mathbb{E}[\|\hat{Q}_T - Q^*\|_\infty] \leq \underbrace{\frac{\mathcal{E}_{approx}}{1-\gamma}}_{:=\mathcal{E}_1} + \underbrace{\gamma^T\|\hat{Q}_0 - Q^*\|_\infty}_{:=\mathcal{E}_2} + \underbrace{\frac{2\left((1-\alpha\lambda_{\min})^{\frac{K-\tau_\alpha}{2}} + \frac{12\sqrt{\alpha\tau_\alpha}}{\sqrt{\lambda_{\min}}}\right)}{(1-\gamma)^2\sqrt{\lambda_{\min}}}}_{:=\mathcal{E}_3}. \tag{3.2}$$

(2) *When using diminishing stepsizes $\alpha_k = \alpha/(k+h)$ (where α and h are appropriately chosen), the following inequality holds for all $K \geq k_0 := \min\{k \mid k \geq \tau_{\alpha_k}\}$ and $T \geq 0$:*

$$\mathbb{E}[\|\hat{Q}_T - Q^*\|_\infty] \leq \mathcal{E}_1 + \mathcal{E}_2 + \underbrace{\frac{2\left(\sqrt{k_0+h} + \frac{47\alpha\sqrt{\tau_{\alpha_K}}}{\sqrt{\alpha\lambda_{\min}-1}}\right)\sqrt{K+h}}{(1-\gamma)^2\sqrt{\lambda_{\min}}}}_{:=\mathcal{E}_3'}. \tag{3.3}$$

Remark 3.3. Recall that τ_{α_k} is the mixing time of the Markov chain $\{S_k\}$ (induced by π_b) with precision α_k. Since Assumption 3.1 implies geometric mixing and α_k is polynomial in k, we have $\tau_{\alpha_k} = \mathcal{O}(\log(k))$. Therefore, the threshold k_0 in Theorem 3.2(2) is well defined.

In our finite-sample bound, the term $\mathcal{E}_1$ captures the error due to using function approximation. Compared with [1], where the corresponding term that captures the function

approximation error is of $\mathcal{O}(\mathcal{E}_{\text{approx}}(1-\gamma)^{-1}\lambda_{\min}^{-1})$ (see [1, Remark 2]), we have an improvement by a factor of $\lambda_{\min}^{-1}$. As an aside, note that the $1/(1-\gamma)$ factor in $\mathcal{E}_1$ also appears in temporal difference learning with linear function approximation [43].

The term $\mathcal{E}_2$ goes to zero geometrically fast as T goes to infinity. In fact, the term $\mathcal{E}_2$ captures the error due to fixed-point iteration. That is, if we had a complete basis (hence no function approximation error) and were able to perform value iteration to solve the Bellman equation $Q^* = \mathcal{H}(Q^*)$ (hence no stochastic error), $\mathcal{E}_2$ is the only error term.

The term $\mathcal{E}_3$ (or $\mathcal{E}_3'$) represents the error bound for the inner loop of Algorithm 3.1 when using constant (or diminishing) stepsizes. The update equation in Algorithm 3.1, line 5, is a stochastic approximation algorithm [36] under Markovian noise. When using a constant stepsize, the error bound consists of a geometrically decaying term (which is usually called bias, or optimization error) and a constant term (which is usually called variance, or statistical error) that is proportional to $\sqrt{\alpha\tau_\alpha}$. Since geometric mixing implies $\tau_\alpha = \mathcal{O}(\log(1/\alpha))$, the variance term $\sqrt{\alpha\tau_\alpha}$ can be arbitrarily small using a sufficiently small constant stepsize. When using diminishing stepsizes with a suitable decay rate, since $\tau_{\alpha_K} = \mathcal{O}(\log(K))$, both the bias term and the variance term go to zero at a rate of $\mathcal{O}(\log(K)/K)$. This agrees with the existing literature studying stochastic approximation [38, 11]. Since the additional factor of τ_α (or τ_{α_K}) arises due to using a single trajectory of Markovian samples, it can be removed by using other advanced sampling techniques, one of which is the reverse experience replay developed in [1].

Based on Theorem 3.2, we next derive the sample complexity of Algorithm 3.1 in the following. Note that the sample complexity is orderwise the same for using either a constant stepsize or diminishing stepsizes.

Corollary 3.4. *Given $\epsilon > 0$, to achieve $\mathbb{E}[\|\hat{Q}_T - Q^*\|_\infty] \le \epsilon + \mathcal{E}_{approx}/(1-\gamma)$, the sample complexity is $\tilde{\mathcal{O}}\left(\epsilon^{-2}(1-\gamma)^{-4}\right)$.*

Remark 3.5. While commonly used in the existing literature studying RL with function approximation, it was argued in [24] that the sample complexity is rigorously speaking not well defined when the asymptotic error is nonzero. Here, we present the "sample complexity" in the same sense as in the existing literature to enable a fair comparison.

Corollary 3.4 states that Algorithm 3.1 achieves an $\tilde{\mathcal{O}}(\epsilon^{-2})$ sample complexity for finding the optimal Q-function up to a function approximation error. Notably, the effective horizon appears as $(1-\gamma)^{-4}$ in the sample complexity, which has an improvement over [1, Theorem 1] by a factor of $(1-\gamma)^{-1}$. Up to a function approximation error, our result also implies an $\tilde{\mathcal{O}}(\epsilon^{-2})$ sample complexity measured by the suboptimality gap of the output policy, i.e., $\mathbb{E}_{(S,A)\sim\rho_o}[Q^*(S,A) - Q^{\pi_T}(S,A)]$, where ρ_o is an arbitrary distribution on the state-action space and π_T is the policy greedily induced from $\hat{Q}_T$. This follows from

$$\mathbb{E}_{(S,A)\sim\rho_o}[Q^*(S,A) - Q^{\pi_T}(S,A)] \le \mathbb{E}[\|Q^{\pi_T} - Q^*\|_\infty] \le \frac{2}{1-\gamma}\mathbb{E}[\|\hat{Q}_T - Q^*\|_\infty],$$

where the second inequality is from [37].

In the existing literature studying lower bounds of RL [44, 51], it was shown that, even if Q^* belongs to the approximation subspace, unless certain assumptions (such as Bellman completeness) are imposed, it is in general not possible to develop a polynomial algorithm that is capable of approximating Q^* with an *arbitrary precision.* That is, given an arbitrary

$\epsilon > 0$, an exponential sample complexity is needed in general to achieve an ϵ-approximation error to Q^*. In this work, we take a different goal since we allow for some *fixed error* (i.e., our function approximation error) in approximating Q^*. In this case, Corollary 3.4 states that, given an arbitrary $\epsilon > 0$, an $\tilde{\mathcal{O}}(\epsilon^{-2})$ sample complexity is enough to achieve an $(\epsilon + \mathcal{E}_{\text{approx}}/(1-\gamma))$-approximation error to Q^*. However, suppose we want to find an approximation to Q^* with an error less than $\mathcal{E}_{\text{approx}}/(1-\gamma)$ (provided that Q^* is in the approximating linear subspace); then our algorithm in general is not guaranteed to deliver an output that satisfies this requirement. Therefore, our approach can be viewed as a way of obtaining an improved sample complexity at the cost of introducing an approximation error.

4. The reason that target network and truncation stabilize Q-learning. In the previous section, we presented the stable algorithm and the finite-sample bound. In this section, we elaborate in detail on why target network and truncation are enough to stabilize Q-learning.

Summary. We start with the classical semigradient Q-learning with linear function approximation in section 4.1, which unfortunately is not necessarily stable, as evidenced by the divergent counterexample constructed in [2]. In section 4.2, we show that by adding the target network to Q-learning, the resulting algorithm successfully overcomes the divergence issue in the MDP example in [2]. However, beyond the example in [2], the target network alone is not sufficient to stabilize Q-learning. In fact, we show in section 4.3 that Q-learning with target network diverges for another MDP example constructed in [11]. In section 4.4, we show that by further adding truncation, the resulting algorithm (i.e., Algorithm 3.1) is provably stable. The reason that truncation successfully stabilizes Q-learning is due to an insightful observation regarding the relation between truncation and a particular projection.

4.1. Classical semigradient Q-learning. We begin by presenting the classical semigradient Q-learning with linear function approximation [3, 40]. With a trajectory of samples $\{(S_k, A_k)\}$ collected under the behavior policy π_b and an initialization $\theta_0 \in \mathbb{R}^d$, the semigradient Q-learning algorithm updates the parameter θ_k according to the following formula:

$$\theta_{k+1} = \theta_k + \alpha_k \phi(S_k, A_k)(\mathcal{R}(S_k, A_k) + \gamma \max_{a' \in \mathcal{A}} \phi(S_{k+1}, a')^\top \theta_k - \phi(S_k, A_k)^\top \theta_k). \tag{4.1}$$

The reason that (4.1) is called semigradient Q-learning is that it can be interpreted as a one-step stochastic semigradient descent for minimizing the Bellman error. See [3] for more details. Unfortunately, algorithm (4.1) does not necessarily converge, as evidenced by the divergent example provided in [2]. The MDP example constructed in [2] has seven states and two actions. To perform linear function approximation, 14 linearly independent basis vectors are chosen. See [2] for a complete description of this MDP. The important thing to note about this example is that the number of basis vectors is equal to the size of the state-action space, i.e., $d = |\mathcal{S}||\mathcal{A}|$. Therefore, rather than doing function approximation, we are essentially doing a change of basis. Even in this setting, algorithm (4.1) surprisingly diverges. To achieve stability for algorithm (4.1), the negative drift assumption was imposed in [32, 11, 25].

By viewing algorithm (4.1) as a stochastic approximation algorithm (which is a framework for solving root-finding problems with incomplete information [36]), the target equation algorithm (4.1) is trying to solve is

$$\mathbb{E}_{S_k \sim \mu}[\phi(S_k, A_k)(\mathcal{R}(S_k, A_k) + \gamma \max_{a' \in \mathcal{A}} \phi(S_{k+1}, a')^\top \theta - \phi(S_k, A_k)^\top \theta)] = 0.$$

The previous equation can be written compactly using the Bellman operator $\mathcal{H}(\cdot)$ and the diagonal matrix D as

$$\Phi^\top D(\mathcal{H}(\Phi\theta) - \Phi\theta) = 0 \tag{4.2}$$

and is further equivalent to the fixed-point equation

$$\theta = \mathcal{H}_\Phi(\theta), \tag{4.3}$$

where the operator $\mathcal{H}_\Phi : \mathbb{R}^d \mapsto \mathbb{R}^d$ is defined by $\mathcal{H}_\Phi(\theta) = (\Phi^\top D\Phi)^{-1}\Phi^\top D\mathcal{H}(\Phi\theta)$. Equation (4.3) is closely related to the so-called projected Bellman equation. To see this, since Φ has linearly independent columns, (4.3) is equivalent to

$$\Phi\theta = \Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D\mathcal{H}(\Phi\theta) = \mathrm{Proj}_{\mathcal{W}}\mathcal{H}(\Phi\theta). \tag{4.4}$$

We next show that in the complete basis setting, i.e., $d = |\mathcal{S}||\mathcal{A}|$, which covers the counterexample in [2] as a special case, the operator $\mathcal{H}_\Phi(\cdot)$ is, in fact, a contraction mapping with $\theta^* = \Phi^{-1}Q^*$ being its unique fixed-point. This implies that the design of the classical semigradient Q-learning algorithm (4.1) is flawed because *if it were designed as a stochastic approximation algorithm effectively performing fixed-point iteration to solve* (4.3), *it would converge.* Instead, it was designed as a stochastic approximation algorithm based on (4.2). While (4.2) is equivalent to (4.3), their corresponding stochastic approximation algorithms have different behavior in terms of their convergence or divergence.

To show the contraction property of $\mathcal{H}_\Phi(\cdot)$, first observe that in the complete basis setting we have $\mathcal{H}_\Phi(\theta) = (\Phi^\top D\Phi)^{-1}\Phi^\top D\mathcal{H}(\Phi\theta) = \Phi^{-1}\mathcal{H}(\Phi\theta)$. Let $\|\cdot\|_{\Phi,\infty}$ be a norm on $\mathbb{R}^d$ defined as $\|\theta\|_{\Phi,\infty} = \|\Phi\theta\|_\infty$ for all θ. Since Φ has linearly independent columns, $\|\cdot\|_{\Phi,\infty}$ is indeed a norm. Then we have

$$\|\mathcal{H}_\Phi(\theta_1) - \mathcal{H}_\Phi(\theta_2)\|_{\Phi,\infty} = \|\mathcal{H}(\Phi\theta_1) - \mathcal{H}(\Phi\theta_2)\|_\infty \le \gamma\|\Phi(\theta_1 - \theta_2)\|_\infty = \gamma\|\theta_1 - \theta_2\|_{\Phi,\infty}$$

for all $\theta_1, \theta_2 \in \mathbb{R}^d$, where the inequality follows from the Bellman operator $\mathcal{H}(\cdot)$ being a contraction mapping with respect to the ℓ_∞-norm. It follows that the operator $\mathcal{H}_\Phi(\cdot)$ is a contraction mapping with respect to $\|\cdot\|_{\Phi,\infty}$. Moreover, since

$$\mathcal{H}_\Phi(\theta^*) = \Phi^{-1}\mathcal{H}(\Phi\theta^*) = \Phi^{-1}\mathcal{H}(Q^*) = \Phi^{-1}Q^* = \theta^*,$$

the point θ^* is the unique fixed-point of the operator $\mathcal{H}_\Phi(\cdot)$. The previous analysis suggests that we should aim at designing Q-learning with linear function approximation as a fixed-point iteration (implemented in a stochastic manner due to sampling in RL) to solve (4.3). The resulting algorithm would at least converge for the counterexample in [2].

4.2. Introducing target network. We begin with the following fixed-point iteration for solving the fixed-point equation (4.3):

$$\hat{\theta}_{t+1} = (\Phi^\top D\Phi)^{-1}\Phi^\top D\mathcal{H}(\Phi\hat{\theta}_t), \tag{4.5}$$

where we write $\mathcal{H}_\Phi(\cdot)$ explicitly in terms of Φ, D, and $\mathcal{H}(\cdot)$. The update in (4.5) is what we would like to perform if we had complete information on the dynamics of the underlying MDP.

Algorithm 4.1. Q-learning with linear function approximation: target network and no truncation.

1: **Input:** Integers T, K, initializations $\hat{\theta}_0 = \mathbf{0}$, $\theta_{t,0} = \mathbf{0}$ for all t, behavior policy π_b.
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: **for** $k = 0, 1, \ldots, K-1$ **do**
4: Sample $A_k \sim \pi_b(\cdot|S_k)$ and observe $S_{k+1} \sim P_{A_k}(S_k, \cdot)$
5: $\theta_{t,k+1} = \theta_{t,k} + \alpha_k \phi(S_k, A_k)(\mathcal{R}(S_k, A_k) + \gamma \max_{a' \in \mathcal{A}} \phi(S_{k+1}, a')^\top \hat{\theta}_t - \phi(S_k, A_k)^\top \theta_{t,k})$
6: **end for**
7: $\hat{\theta}_{t+1} = \theta_{t,K}$ and $S_0 = S_K$
8: **end for**
9: **Output:** $\hat{\theta}_T$

The question is whether there is a stochastic variant of such fixed-point iteration that can be implemented in the RL setting (where the environment is unknown). The answer is Q-learning with target network, which is presented in Algorithm 4.1.

We next elaborate intuitively on why Algorithm 4.1 can be viewed as a stochastic variant of the fixed-point iteration (4.5). For a fixed outer-loop iteration index t, let us focus on the update equation (cf. Algorithm 4.1, line 5) of the inner loop. Conditioning on the past, the target network parameter $\hat{\theta}_t$ is a constant. Therefore, the update equation in terms of $\theta_{t,k}$ is in fact a linear stochastic approximation algorithm for solving the linear system of equations:

$$-\Phi^\top D \Phi \theta + \Phi^\top D \mathcal{H}(\Phi \hat{\theta}_t) = 0. \tag{4.6}$$

Since the matrix $-\Phi^\top D \Phi$ is negative definite, the asymptotic convergence of the inner-loop update follows from standard results in the literature [3]. Therefore, when the stepsize sequence $\{\alpha_k\}$ is appropriately chosen and K is large enough, we expect the last iterate of the inner loop, i.e., $\theta_{t,K}$, to approximate the solution of (4.6), that is, $\theta_{t,K} \approx (\Phi^\top D \Phi)^{-1} \Phi^\top D \mathcal{H}(\Phi \hat{\theta}_t)$. Now in view of Algorithm 4.1, line 7, since the target network $\hat{\theta}_{t+1}$ is synchronized to $\theta_{t,K}$, the overall update in terms of the target network parameter is

$$\hat{\theta}_{t+1} \approx (\Phi^\top D \Phi)^{-1} \Phi^\top D \mathcal{H}(\Phi \hat{\theta}_t).$$

Therefore, Q-learning with target network is in effect performing an approximate version (due to the stochastic error in sampling) of the fixed-point iteration (4.5).

Revisiting the counterexample in [2] (where $d = |\mathcal{S}||\mathcal{A}|$), recall that the fixed-point iteration (4.5) reduces to $\hat{\theta}_{t+1} = \Phi^{-1} \mathcal{H}(\Phi \hat{\theta}_t) = \mathcal{H}_\Phi(\hat{\theta}_t)$. Since the operator $\mathcal{H}_\Phi(\cdot)$ is a contraction mapping as shown in section 4.1, the fixed-point iteration (4.5) provably converges. As a result, Q-learning with target network as a stochastic variant of the fixed-point iteration (4.5) also converges, which is stated in the following proposition.

Proposition 4.1. *Consider Algorithm 4.1. Suppose that Assumption 3.1 is satisfied and the feature matrix Φ is a square matrix (i.e., $d = |\mathcal{S}||\mathcal{A}|$) with full rank. Then the sample complexity to achieve $\mathbb{E}[\|\Phi \hat{\theta}_T - Q^*\|_\infty] < \epsilon$ is $\tilde{\mathcal{O}}(\epsilon^{-2})$.*

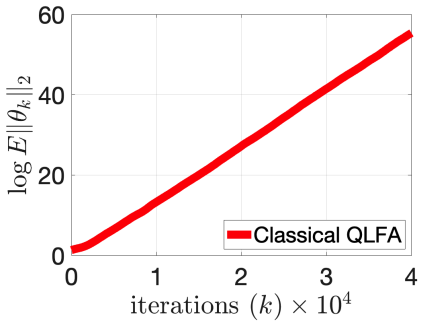

Figure 1. *Semigradient Q-learning.*

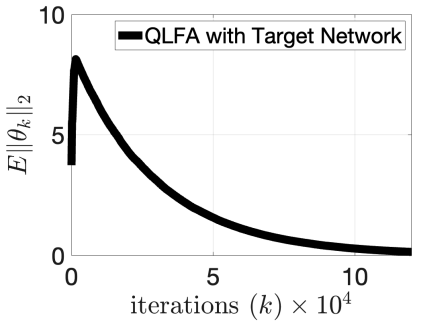

Figure 2. *Target network Q-learning.*

The proof of Proposition 4.1 is identical to that of Theorem 3.2 and hence is omitted. To further verify the stability, we conduct numerical simulations for the MDP example constructed in [2]. As we see, while classical semigradient Q-learning diverges in Figure 1 (which agrees with [2]), Q-learning with target network converges as shown in Figure 2.

4.3. Insufficiency of target network. The reason that Q-learning with target network overcomes the divergence for the MDP example in [2] is essentially that *the projected Bellman operator reduces to the regular Bellman operator (which is a contraction mapping) when we have a complete basis.* However, this is in general not the case. In the projected Bellman equation (4.4), the Bellman operator $\mathcal{H}(\cdot)$ is a contraction mapping with respect to the ℓ_∞-norm $\|\cdot\|_\infty$, while the projection operator $\text{Proj}_{\mathcal{W}}$ is a nonexpansive mapping with respect to the projection norm, in this case the weighted ℓ_2-norm $\|\cdot\|_D$. Due to the norm mismatch, the composition $\text{Proj}_{\mathcal{W}}\mathcal{H}(\cdot)$ is not necessarily a contraction mapping with respect to any norm. This is the *fundamental reason* for the divergence of Q-learning with linear function approximation, and introducing the target network alone does not overcome this issue, as evidenced by the following MDP example constructed in [11].

Example 4.2. Consider an MDP with state space $\mathcal{S} = \{s_1, s_2\}$ and action space $\mathcal{A} = \{a_1, a_2\}$. Regardless of the present state, taking action a_1 results in state s_1 with probability 1, and taking action a_2 results in state s_2 with probability 1. The reward function is defined

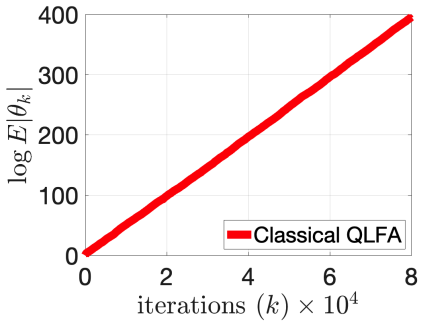

Figure 3. *Semigradient Q-learning.*

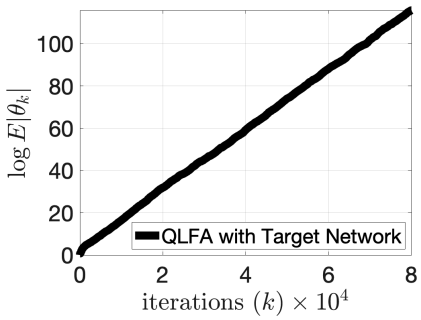

Figure 4. *Target network Q-learning.*

as $\mathcal{R}(s_1, a_1) = 1$, $\mathcal{R}(s_1, a_2) = \mathcal{R}(s_2, a_1) = 2$, and $\mathcal{R}(s_2, a_2) = 4$. The approximation linear subspace is chosen to be a span of a single basis vector $\Phi = [1, 2, 2, 4]^\top$. The behavior policy is to take each action with equal probability.

In Example 4.2, after straightforward calculation, we have the following result.

Lemma 4.3 (Appendix C of [11]). *Equation (4.3) is explicitly given as*

$$\theta = 1 + \frac{9\gamma}{10}\theta + \frac{3\gamma\theta}{10}(\mathbb{1}_{\{\theta \geq 0\}} - \mathbb{1}_{\{\theta < 0\}}).$$

When the discount factor γ is in the interval $(5/6, 1)$, for any positive initialization $\theta_0 > 0$, it is clear that performing fixed-point iteration to solve (4.3) in this example leads to divergence. Since Q-learning with target network is a stochastic variant of such fixed-point iteration, it also diverges. Numerical simulations demonstrate that performing either classical semigradient Q-learning or Q-learning with the target network leads to divergence for the MDP in Example 4.2 (cf. Figures 3 and 4). This example shows that it is in general not possible to achieve the stability of Q-learning with linear function approximation with only the target network, which motivates our truncation technique in the next subsection.

4.4. Truncation to the rescue. Recall from the previous section that Q-learning with target network is trying to perform a stochastic variant of the fixed-point iteration (4.5), which can be equivalently written as

$$\tilde{Q}_{t+1} = \mathrm{Proj}_{\mathcal{W}} \mathcal{H}(\tilde{Q}_t), \tag{4.7}$$

where we use $\tilde{Q}_t$ to denote the Q-function estimate associated with the target network $\hat{\theta}_t$, i.e., $\tilde{Q}_t = \Phi\hat{\theta}_t$. To motivate the truncation technique, we next analyze the update (4.7), the behavior of which in terms of stability aligns with the behavior of Q-learning with target network, as explained in the previous section. First note that (4.7) is equivalent to

$$\tilde{Q}_{t+1} - Q^* = \mathcal{H}(\tilde{Q}_t) - \mathcal{H}(Q^*) + \mathrm{Proj}_{\mathcal{W}}\mathcal{H}(\tilde{Q}_t) - \mathcal{H}(\tilde{Q}_t).$$

A simple calculation using the triangle inequality, the contraction property of $\mathcal{H}(\cdot)$, and telescoping yields the following error bound of the iterative algorithm (4.7):

$$\|\tilde{Q}_T - Q^*\|_\infty \le \gamma^T \|\tilde{Q}_0 - Q^*\|_\infty + \sum_{t=0}^{T} \gamma^{T-t} \underbrace{\|\mathrm{Proj}_{\mathcal{W}}\mathcal{H}(\tilde{Q}_t) - \mathcal{H}(\tilde{Q}_t)\|_\infty}_{E_t}.$$

The problem with the previous analysis is that the term E_t (which captures the error due to using linear function approximation) is not necessarily bounded unless using a complete basis or knowing beforehand that $\{\tilde{Q}_t\}$ is always contained in a bounded set. The possibility of such a function approximation error being unbounded is an alternative explanation to the divergence of Q-learning with linear function approximation. This is true for arbitrary function approximation (including neural networks) as well, since it is in general not possible to uniformly approximate unbounded functions.

Suppose that we are able to somehow control the size of the estimate $\tilde{Q}_t$ so that it is always contained in a bounded set. Then the term E_t is guaranteed to be finite and effectively captures the approximation power of the chosen function class. To achieve the boundedness of the associated Q-function estimate $\tilde{Q}_t$ of the target network, tracing back to Algorithm 4.1, a natural approach is to first project $\tilde{Q}_t = \Phi\hat{\theta}_t$ onto the ℓ_∞-norm ball $B_r := \{Q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|} \mid \|Q\|_\infty \le r\}$ before using it as the target Q-function in the inner loop, resulting in Algorithm 4.2.

Algorithm 4.2. Impractical Q-learning with linear function approximation: target network and projection.

Input: Integers T, K, initializations $\hat{\theta}_0 = \mathbf{0}$, $\theta_{t,0} = \mathbf{0}$ for all t, behavior policy π_b.
for $t = 0, 1, \ldots, T-1$ **do**
  for $k = 0, 1, \ldots, K-1$ **do**
    Sample $A_k \sim \pi_b(\cdot|S_k)$ and observe $S_{k+1} \sim P_{A_k}(S_k, \cdot)$
    $\theta_{t,k+1} = \theta_{t,k} + \alpha_k \phi(S_k, A_k)(\mathcal{R}(S_k, A_k) + \gamma \max_{a' \in \mathcal{A}} \tilde{Q}_t(S_{k+1}, a') - \phi(S_k, A_k)^\top \theta_{t,k})$
  end for
  $\hat{\theta}_{t+1} = \theta_{t,K}$ and $S_0 = S_K$
  $\tilde{Q}_{t+1} = \Pi_{B_r} \Phi\hat{\theta}_{t+1}$
end for
Output: $\hat{\theta}_T$

In Algorithm 4.2, line 8, the operator Π_{B_r} represents the projection onto the ℓ_∞-norm ball B_r with respect to some suitable norm $\|\cdot\|$. The specific norm $\|\cdot\|$ chosen to perform the projection turns out to be irrelevant as a result of a key observation between truncation and projection, which will be revealed soon.

Although Algorithm 4.2 stabilizes the Q-function estimate $\tilde{Q}_t$, it is not implementable in practice. To see this, recall that the point of using linear function approximation is to avoid directly working with $|\mathcal{S}||\mathcal{A}|$-dimensional variables. However, to implement Algorithm 4.2, line 8, one has to first compute $\Phi\hat{\theta}_{t+1} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, and then project it onto B_r. Therefore, the last difficulty we need to overcome is to find a way to implement Algorithm 4.2 without working with $|\mathcal{S}||\mathcal{A}|$-dimensional variables. The solution relies on the following lemma.

Lemma 4.4. *For any $Q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ and any weighted ℓ_p-norm $\|\cdot\|$ (the weights can be arbitrary and $p \in [1,\infty]$), we have $\mathcal{T}(Q) \in \arg\min_{Q' \in B_r} \|Q - Q'\|$.*

Remark 4.5. Note that $\arg\min_{Q' \in B_r} \|Q - Q'\|$ is, in general, a set because the projection onto B_r may not be unique. As an example, observe that any point in the set $\{(x,1) \mid x \in [-1,1]\}$ is a projection of the point $(0,2)$ onto the ℓ_∞-norm unit ball $\{(x,y) \mid x,y \in [-1,1]\}$ with respect to the ℓ_∞-norm.

Proof of Lemma 4.4. Let $\{\omega(s,a)\}_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ be arbitrary positive weights. We denote the weighted ℓ_p-norm with weights $\{\omega(s,a)\}_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ by $\|\cdot\|_{\omega,p}$. For any $Q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, we have

$$\begin{aligned}
\min_{Q' \in B_r} \|Q - Q'\|_{\omega,p} &= \min_{Q' \in B_r} \left(\sum_{s,a} \omega(s,a) |Q(s,a) - Q'(s,a)|^p \right)^{1/p} \\
&= \left(\sum_{s,a} \omega(s,a) \min_{-r \le Q'(s,a) \le r} |Q(s,a) - Q'(s,a)|^p \right)^{1/p} \\
&= \left(\sum_{s,a} \omega(s,a) |Q(s,a) - \mathcal{T}(Q(s,a))|^p \right)^{1/p} \\
&= \|Q - \mathcal{T}(Q)\|_{\omega,p}.
\end{aligned}$$

Therefore, we have $\mathcal{T}(Q) \in \arg\min_{Q' \in B_r} \|Q - Q'\|_{\omega,p}$. ∎

Lemma 4.4 states that, for any $Q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, if we simply truncate the vector Q, the resulting vector must belong to the projection set of Q onto the ℓ_∞-norm ball with radius r, for a wide class of projection norms. This seemingly simple but important result enables us to replace the projection $\tilde{Q}_{t+1} = \Pi_{B_r}\Phi\hat{\theta}_{t+1}$ with the truncation $\tilde{Q}_{t+1} = \mathcal{T}(\Phi\hat{\theta}_{t+1})$ in Algorithm 4.2, line 8. Unlike projection, truncation is a componentwise operation. Therefore, $\tilde{Q}_{t+1} = \mathcal{T}(\Phi\hat{\theta}_{t+1})$ is equivalent to $\tilde{Q}_{t+1}(s,a) = \mathcal{T}(\phi(s,a)^\top\hat{\theta}_{t+1})$ for all (s,a).

The last issue is that we need to perform truncation for all state-action pairs (s,a), which, as illustrated earlier, violates the purpose of doing function approximation. However, observe that the target network is used only in line 5 of Algorithm 4.2, where only the component of $\tilde{Q}_t$ visited by the sample trajectory is needed to perform the update. In light of this observation, instead of truncating $\phi(s,a)^\top\hat{\theta}_t$ for all (s,a), we only need to truncate $\phi(S_{k+1},a')^\top\hat{\theta}_t$ in Algorithm 4.2, line 5, which leads to our stable version of Q-learning with linear function approximation presented in Algorithm 3.1. The following proposition shows that target network

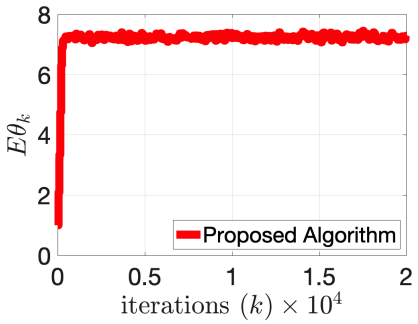

Figure 5. *Algorithm 3.1 for* [2, Example 1].

and truncation together stabilize Q-learning with linear function approximation and serve as a middle step to proving Theorem 3.2.

For simplicity of notation, we denote $\bar{\theta}_t := (\Phi^\top D\Phi)^{-1}\Phi^\top D\mathcal{H}(\hat{Q}_t)$ for all t, where we recall that $\hat{Q}_t = \mathcal{T}(\Phi\hat{\bar{\theta}}_t) = \mathcal{T}(\Phi\theta_{t-1,K})$. Note that we have from the explicit expression of the projection operator $\text{Proj}_{\mathcal{W}}$ that $\Phi\bar{\theta}_t = \text{Proj}_{\mathcal{W}}(\mathcal{H}(\hat{Q}_t))$.

Proposition 4.6. *The following inequality holds:*

$$\mathbb{E}[\|\hat{Q}_T - Q^*\|_\infty] \le \gamma^T\|\hat{Q}_0 - Q^*\|_\infty + \frac{\mathcal{E}_{approx}}{1-\gamma} + \sum_{t=0}^{T-1}\gamma^{T-t-1}\mathbb{E}[\|\theta_{t,K} - \bar{\theta}_t\|_2]. \tag{4.8}$$

Due to truncation, the error arising from using function approximation is bounded and is captured by $\mathcal{E}_{\text{approx}}$. This is crucial to prevent the divergence of Q-learning with linear function approximation. The last term on the right-hand side (RHS) of (4.8) captures the error in the inner loop of Algorithm 3.1 and eventually leads to the terms $\mathcal{E}_3$ and $\mathcal{E}_3'$ in Theorem 3.2.

Revisiting Example 4.2, where either semigradient Q-learning or Q-learning with target network diverges, Algorithm 3.1 converges as demonstrated in Figure 5. Moreover, observe that Algorithm 3.1 seems to converge to a positive scalar, which we denote by θ^*. As a result, the policy π induced greedily by $\Phi\theta^*$ is to always take action a_2. It can be easily verified that π is indeed the optimal policy. This is an interesting observation since the optimal Q-function Q^* in this case does not belong to the linear subspace $\mathcal{W}$ (which is spanned by a single basis vector $(1,2,2,4)^\top$). However, performing Algorithm 3.1 converges and the induced policy is optimal. Figure 6 shows that Algorithm 3.1 also converges for the MDP example in [2].

Discussion about the truncation technique. The truncation technique we use here can be viewed as a means to regularize the Q-function associated with the weight vector, i.e., $\Phi\theta$. The reason that it works is due to Lemma 4.4, which enables us to regularize $|\mathcal{S}||\mathcal{A}|$-dimensional variables by working with d-dimensional variables (which is the point of performing linear function approximation). In the existing literature, another popular regularization technique (perhaps a seemingly more natural one) is to directly project the weight vector θ onto a prespecified bounded set after each iteration of Q-learning [4, 48, 55]. Compared to directly projecting θ, truncating $\Phi\theta$ has three main advantages.

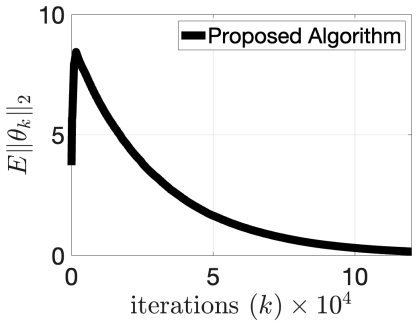

Figure 6. *Algorithm 3.1 for Example 4.2.*

(1) Since the goal of Q-learning is to estimate Q^*, it is more desirable to bound $Q^* - \Phi\theta$ instead of $\theta - \theta^*$, where θ^* is usually the solution (the existence and uniqueness of which need to be assumed) of a properly defined projected Bellman equation [32].
(2) Since the set onto which θ is projected must contain the solution θ^* of the projected Bellman equation, and the estimate of the size of θ^* involves unknown parameters such as the stationary distribution of the Markov chain $\{(S_k, A_k)\}$ under π_b [32, 55, 43], it is not clear how to perform such projection in practice. On the contrary, we use truncation and aim to control the Q-function associated with θ (and not θ itself). Since $\|Q^*\|_\infty \le 1/(1-\gamma)$, we simply choose a truncation radius of $r = 1/(1-\gamma)$ for $\Phi\theta$ as a form of regularization, and this does not involve any unknown parameters.
(3) For existing results using the approach of projecting θ, all of them require the "negative drift assumption," such as [48, Assumption 5.3] and [7, Assumption 6.1]. One of our main contributions is to remove the negative drift assumption.

5. Proof of Theorem 3.2.

In this section, we present the full proof of Theorem 3.2. The high-level idea of the proof is essentially presented in the previous section. Specifically, we view the inner loop of Algorithm 3.1 as a stochastic approximation algorithm for estimating $\mathrm{Proj}_{\mathcal{W}}\mathcal{H}(\mathcal{T}(\hat{Q}_t))$, which is then used in the outer loop for the fixed-point iteration $\hat{Q}_{t+1} = \mathrm{Proj}_{\mathcal{W}}\mathcal{H}(\mathcal{T}(\hat{Q}_t))$.

5.1. Analysis of the outer loop (proof of Proposition 4.6).

Using our notation $\hat{Q}_t = \mathcal{T}(\Phi\hat{\theta}_t) = \mathcal{T}(\Phi\theta_{t-1,K})$ and the fact that $Q^* = \mathcal{H}(Q^*)$, we have for any $t = 1, 2, \ldots, T$ that

$$
\begin{aligned}
&\|\hat{Q}_t - Q^*\|_\infty \\
&= \|\mathcal{H}(\hat{Q}_{t-1}) - \mathcal{H}(Q^*) + \hat{Q}_t - \mathcal{T}(\mathrm{Proj}_{\mathcal{W}}\mathcal{H}(\hat{Q}_{t-1})) + \mathcal{T}(\mathrm{Proj}_{\mathcal{W}}\mathcal{H}(\hat{Q}_{t-1})) - \mathcal{H}(\hat{Q}_{t-1})\|_\infty \\
&\le \|\mathcal{H}(\hat{Q}_{t-1}) - \mathcal{H}(Q^*)\|_\infty + \|\hat{Q}_t - \mathcal{T}(\mathrm{Proj}_{\mathcal{W}}\mathcal{H}(\hat{Q}_{t-1}))\|_\infty \\
&\quad + \|\mathcal{T}(\mathrm{Proj}_{\mathcal{W}}\mathcal{H}(\hat{Q}_{t-1})) - \mathcal{H}(\hat{Q}_{t-1})\|_\infty \\
&\le \|\mathcal{H}(\hat{Q}_{t-1}) - \mathcal{H}(Q^*)\|_\infty + \|\mathcal{T}(\Phi\theta_{t-1,K}) - \mathcal{T}(\mathrm{Proj}_{\mathcal{W}}\mathcal{H}(\hat{Q}_{t-1}))\|_\infty + \mathcal{E}_{\mathrm{approx}}, \qquad (5.1)
\end{aligned}
$$

where the last line follows from the definition of the function approximation error in (3.1).

We next bound the first two terms on the RHS of the previous inequality. Since the Bellman operator $\mathcal{H}(\cdot)$ is a contraction mapping with respect to $\|\cdot\|_\infty$, with contraction factor γ [3], we have for all $t = 1, 2, \ldots, T$ that

$$\|\mathcal{H}(\hat{Q}_{t-1}) - \mathcal{H}(Q^*)\|_\infty \le \gamma \|\hat{Q}_{t-1} - Q^*\|_\infty. \tag{5.2}$$

As for the second term on the RHS of (5.1), first observe that

$$\begin{aligned}\|\mathcal{T}(Q_1) - \mathcal{T}(Q_2)\|_\infty &= \max_{s,a} |\mathcal{T}(Q_1)(s,a) - \mathcal{T}(Q_2)(s,a)| \\ &\le \max_{s,a} |Q_1(s,a) - Q_2(s,a)| \\ &= \|Q_1 - Q_2\|_\infty \quad \forall\, Q_1, Q_2 \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|},\end{aligned}$$

which can also be viewed as a nonexpansive property of the truncation operator with respect to $\|\cdot\|_\infty$ (cf. Lemma 4.4). Then we have from the previous inequality and $\bar{\theta}_{t-1} = (\Phi^\top D \Phi)^{-1} \Phi^\top D \mathcal{H}(\hat{Q}_{t-1})$ that

$$\begin{aligned}\|\mathcal{T}(\Phi\theta_{t-1,K}) - \mathcal{T}(\mathrm{Proj}_{\mathcal{W}}\mathcal{H}(\hat{Q}_{t-1}))\|_\infty &\le \|\Phi\theta_{t-1,K} - \mathrm{Proj}_{\mathcal{W}}\mathcal{H}(\hat{Q}_{t-1})\|_\infty \\ &= \|\Phi(\theta_{t-1,K} - \bar{\theta}_{t-1})\|_\infty \\ \text{(Definition of } \|\cdot\|_\infty) \qquad &= \max_{s,a} |\phi(s,a)^\top (\theta_{t-1,K} - \bar{\theta}_{t-1})| \\ \text{(Cauchy–Schwarz inequality)} \qquad &\le \max_{s,a} \|\phi(s,a)\|_2 \|\theta_{t-1,K} - \bar{\theta}_{t-1}\|_2 \\ &\le \|\theta_{t-1,K} - \bar{\theta}_{t-1}\|_2,\end{aligned}$$

where the last line follows from $\|\phi(s,a)\|_2 \le 1$ for all (s,a). Substituting the previous bound and the bound in (5.2) into (5.1), we have

$$\|\hat{Q}_t - Q^*\|_\infty \le \gamma \|\hat{Q}_{t-1} - Q^*\|_\infty + \|\theta_{t-1,K} - \bar{\theta}_{t-1}\|_2 + \mathcal{E}_{\mathrm{approx}}$$

for all $t = 1, 2, \ldots, T$. Recursively using the previous inequality and then taking expectation, we obtain

$$\mathbb{E}[\|\hat{Q}_T - Q^*\|_\infty] \le \gamma^T \|\hat{Q}_0 - Q^*\|_\infty + \sum_{t=0}^{T-1} \gamma^{T-t-1} \mathbb{E}[\|\theta_{t,K} - \bar{\theta}_t\|_2] + \frac{\mathcal{E}_{\mathrm{approx}}}{1-\gamma}. \tag{5.3}$$

This proves Proposition 4.6.

5.2. Analysis of the inner loop. We first present the inner loop of Algorithm 3.1 in the following, where we omit the outer-loop iteration index t. The result from this section can be combined with (5.3) in a straightforward fashion by using the conditional expectation and the Markov property.

We next reformulate Algorithm 5.1 as a Markovian stochastic approximation algorithm for solving some suitable equation. Let $\{X_k\}$ be a Markov chain defined as $X_k = (S_k, A_k, S_{k+1})$ for all $k \ge 0$. Note that the state space of $\{X_k\}$ is given by $\mathcal{X} = \{(s,a,s') \mid s \in \mathcal{S}, \pi_b(a|s) > 0, P_a(s,s') > 0\}$. Let P_X be the transition probability matrix of the Markov chain $\{X_k\}$. Under Assumption 3.1, the Markov chain $\{X_k\}$ also has a unique stationary distribution, denoted by ν, which satisfies $\nu(s,a,s') = \mu(s)\pi_b(a|s)P_a(s,s')$ for all $(s,a,s') \in \mathcal{X}$, where we recall that $\mu(\cdot)$ is the stationary distribution of the Markov chain $\{S_k\}$ induced by π_b. See the paragraph after Assumption 3.1.

Algorithm 5.1. Inner loop of Algorithm 3.1.

1: **Input:** Integer K, initialization $\theta_0 = \mathbf{0}$, target network $\hat{\theta}$, behavior policy π_b.
2: **for** $k = 0, 1, \ldots, K-1$ **do**
3: Sample $A_k \sim \pi_b(\cdot|S_k)$ and observe $S_{k+1} \sim P_{A_k}(S_k, \cdot)$
4: $\theta_{k+1} = \theta_k + \alpha_k \phi(S_k, A_k)(\mathcal{R}(S_k, A_k) + \gamma \max_{a' \in \mathcal{A}} \mathcal{T}(\phi(S_{k+1}, a')^\top \hat{\theta}) - \phi(S_k, A_k)^\top \theta_k)$
5: **end for**
6: **Output:** θ_K

Let $F : \mathbb{R}^d \times \mathcal{X} \mapsto \mathbb{R}^d$ be an operator defined as

$$F(\theta, s, a, s') = \phi(s, a)(\mathcal{R}(s, a) + \gamma \max_{a' \in \mathcal{A}} \mathcal{T}(\phi(s', a')^\top \hat{\theta}) - \phi(s, a)^\top \theta)$$

for all $\theta \in \mathbb{R}^d$ and $(s, a, s') \in \mathcal{X}$. Let $\bar{F} : \mathbb{R}^d \mapsto \mathbb{R}^d$ be defined as $\bar{F}(\theta) = \mathbb{E}_{X \sim \nu(\cdot)}[F(\theta, X)]$. Then the update equation in Algorithm 5.1, line 4, can be compactly written as

$$\theta_{k+1} = \theta_k + \alpha_k F(\theta_k, X_k), \tag{5.4}$$

which is a Markovian stochastic approximation algorithm for solving the equation $\bar{F}(\theta) = 0$. Due to the wide applicability of stochastic approximation, there are many existing works that perform finite-sample analysis [38, 4, 11]. Specifically, we will apply [11, Theorem 1] (which is presented in Theorem A.2 for completeness) to establish the finite-sample bound of Algorithm 5.1 (or equivalently, Algorithm 5.4). To achieve that, we next formally state the requirement for choosing the stepsizes, and we verify [11, Assumptions 1, 2, and 3] (stated in Assumption A.1) in the following sequence of lemmas. For simplicity of notation, we denote $\tau_k := \tau_{\alpha_k}$ as the mixing time of the Markov chain $\{S_k\}$ induced by π_b with precision α_k.

Condition 5.1. *The stepsize sequence $\{\alpha_k\}$ is nonincreasing and satisfies $\sum_{i=k-\tau_k}^{k-1} \alpha_i \le \frac{\lambda_{\min}}{130}$ for all $k \ge \tau_k$. When using $\alpha_k = \alpha/(k+h)$, we additionally require $\alpha > 1/\lambda_{\min}$.*

Lemma 5.2. *It holds that $\max_{x \in \mathcal{X}} d_{TV}(P_X^{k+1}(x, \cdot), \nu(\cdot)) \le C\rho^k$ for all $k \ge 0$.*

Proof of Lemma 5.2. For any $k \ge 0$ and $x = (s_0, a_0, s_1) \in \mathcal{X}$, we have

$$\begin{aligned}
d_{\mathrm{TV}}(P_X^{k+1}(x, \cdot), \nu(\cdot)) &= \frac{1}{2} \sum_{x' \in \mathcal{X}} |P_X^{k+1}(x, x'), \nu(x')| \\
&\le \frac{1}{2} \sum_{(s_0', a_0', s_1') \in \mathcal{X}} |P_{\pi_b}^k(s, s_0') - \mu(s_0')| \pi_b(a_0'|s_0') P_{a_0'}(s_0', s_1') \\
&= \frac{1}{2} \sum_{s_0' \in \mathcal{S}} |P_{\pi_b}^k(s, s_0') - \mu(s_0')| \\
&\le \max_{s \in \mathcal{S}} d_{\mathrm{TV}}(P_{\pi_b}^k(s, \cdot), \mu(\cdot)) \\
&\le C\rho^k,
\end{aligned}$$

where the last line is a consequence of Assumption 3.1. Since the RHS of the previous inequality does not depend on x, we in fact have $\max_{x \in \mathcal{X}} d_{\mathrm{TV}}(P_X^{k+1}(x, \cdot), \nu(\cdot)) \le C\rho^k$. ∎

Lemma 5.3. *It holds for any* $\theta_1, \theta_2 \in \mathbb{R}^d$ *and* $x \in \mathcal{X}$ *that* (1) $\|F(\theta_1, x) - F(\theta_2, x)\|_2 \leq \|\theta_1 - \theta_2\|_2$, *and* (2) $\|F(\mathbf{0}, x)\|_2 \leq 1/(1-\gamma)$.

Proof of Lemma 5.3.

(1) For any $\theta_1, \theta_2 \in \mathbb{R}^d$ and $x = (s, a, s') \in \mathcal{X}$, we have by definition of $F(\cdot)$ that

$$\begin{aligned}\|F(\theta_1, x) - F(\theta_2, x)\|_2 &= \|\phi(s,a)\phi(s,a)^\top(\theta_1 - \theta_2)\|_2 \\ &= \|\phi(s,a)\|_2 |\phi(s,a)^\top(\theta_1 - \theta_2)| \\ &\leq \|\phi(s,a)\|_2^2 \|\theta_1 - \theta_2\|_2 \\ &\leq \|\theta_1 - \theta_2\|_2,\end{aligned}$$

where the last line follows from $\|\phi(s,a)\|_2 \leq 1$ for all (s,a).

(2) For any $x = (s, a, s') \in \mathcal{X}$, we have

$$\begin{aligned}\|F(\mathbf{0}, x)\|_2 &= \|\phi(s,a)(\mathcal{R}(s,a) + \gamma \max_{a' \in \mathcal{A}} \mathcal{T}(\phi(s', a')^\top \hat{\theta})\|_2 \\ &\leq \|\phi(s,a)\|_2 (|\mathcal{R}(s,a)| + \gamma |\mathcal{T}(\phi(s', a')^\top \hat{\theta})|) \\ &\leq \frac{1}{1-\gamma}.\end{aligned}$$

Lemma 5.4. *The operator* $\bar{F}(\cdot)$ *has the following properties:*

(1) $\bar{F}(\theta) = 0$ *has a unique solution* $\bar{\theta} = (\Phi^\top D \Phi)^{-1} \Phi^\top D \mathcal{H}(\mathcal{T}(\Phi \hat{\theta}))$,

(2) $\langle \bar{F}(\theta_1) - \bar{F}(\theta_2), \theta_1 - \theta_2 \rangle \leq -\lambda_{\min} \|\theta_1 - \theta_2\|_2^2$ *for any* $\theta_1, \theta_2 \in \mathbb{R}^d$.

Proof of Lemma 5.4.

(1) Using the definition of $\bar{F}(\cdot)$, we have

$$\bar{F}(\theta) = \Phi^\top D(\mathcal{H}(\mathcal{T}(\Phi \hat{\theta})) - \Phi \theta) \quad \forall\, \theta \in \mathbb{R}^d.$$

Since $\Phi^\top D \Phi$ is positive definite, the equation $\bar{F}(\theta) = 0$ has a unique solution

$$\bar{\theta} = (\Phi^\top D \Phi)^{-1} \Phi^\top D \mathcal{H}(\mathcal{T}(\Phi \hat{\theta})).$$

(2) For any $\theta_1, \theta_2 \in \mathbb{R}^d$, we have

$$\langle \bar{F}(\theta_1) - \bar{F}(\theta_2), \theta_1 - \theta_2 \rangle = \langle -\Phi^\top D \Phi(\theta_1 - \theta_2), \theta_1 - \theta_2 \rangle \leq -\lambda_{\min} \|\theta_1 - \theta_2\|_2^2,$$

where we recall that $\lambda_{\min}$ is the minimum eigenvalue of $\Phi^\top D \Phi$.

Lemma 5.5. *It holds that* $\|\bar{\theta}\|_2 \leq \frac{1}{\sqrt{\lambda_{\min}}(1-\gamma)}$.

Proof of Lemma 5.5. Using the explicit expression of $\bar{\theta}$, we have

$$\begin{aligned}\|\Phi \bar{\theta}\|_D &= \|\Phi(\Phi^\top D \Phi)^{-1} \Phi^\top D \mathcal{H}(\mathcal{T}(\Phi \hat{\theta}))\|_D \\ &= \|\mathrm{Proj}_{\mathcal{W}} \mathcal{H}(\mathcal{T}(\Phi \hat{\theta}))\|_D \\ &\leq \|\mathcal{H}(\mathcal{T}(\Phi \hat{\theta}))\|_D,\end{aligned}$$

where the last line follows from $\mathrm{Proj}_{\mathcal{W}}$ being a nonexpansive operator with respect to $\|\cdot\|_D$. To proceed, observe that for any Q satisfying $\|Q\|_\infty \le 1/(1-\gamma)$, we have

$$\begin{aligned}\|\mathcal{H}(\mathcal{T}(\Phi\hat{\theta}))\|_D &\le \|\mathcal{H}(\mathcal{T}(\Phi\hat{\theta}))\|_\infty \\ &= \max_{s,a}\left|\mathcal{R}(s,a)+\gamma\sum_{s'}P_a(s,s')\max_{a'}Q(s',a')\right| \\ &\le \max_{s,a}|\mathcal{R}(s,a)|+\gamma\|Q\|_\infty \\ &\le \frac{1}{1-\gamma}.\end{aligned}$$

Therefore, we have $\|\Phi\bar{\theta}\|_D \le 1/(1-\gamma)$. To connect $\|\Phi\bar{\theta}\|_D$ with $\|\theta\|_2$, note that

$$\|\Phi\bar{\theta}\|_D^2 = \theta^\top\Phi^\top D\Phi\theta \ge \lambda_{\min}\|\theta\|_2^2,$$

which implies $\|\theta\|_2 \le \|\Phi\bar{\theta}\|_D/\sqrt{\lambda_{\min}} \le 1/[\sqrt{\lambda_{\min}}(1-\gamma)]$. ∎

Now we are ready to apply Theorem A.2. We start with constant stepsize. When $\alpha_k \equiv \alpha$, and α is chosen such that $\alpha\tau_\alpha \le \frac{\lambda_{\min}}{130}$, we have for all $k \ge \tau_\alpha$ that

$$\begin{aligned}\mathbb{E}[\|\theta_k-\bar{\theta}\|_2^2] &\le \left(\frac{1}{\sqrt{\lambda_{\min}}(1-\gamma)}+1\right)^2\left((1-\lambda_{\min}\alpha)^{k-\tau_\alpha}+\frac{130\alpha\tau_\alpha}{\lambda_{\min}}\right) \\ &\le \frac{4\left((1-\lambda_{\min}\alpha)^{k-\tau_\alpha}+\frac{130\alpha\tau_\alpha}{\lambda_{\min}}\right)}{\lambda_{\min}(1-\gamma)^2},\end{aligned} \tag{5.5}$$

where the last inequality follows from

$$\lambda_{\min} = \min_{\theta:\|\theta\|_2=1}\theta^\top\Phi^\top D\Phi\theta \le \min_{\theta:\|\theta\|_2=1}\sum_{s,a}\mu(s)\pi_b(a|s)\|\phi(s,a)\|_2^2\|\theta\|_2^2 \le 1.$$

When using diminishing stepsizes of the form $\alpha_k = \alpha/(k+h)$, where $\alpha > 1/\lambda_{\min}$ and h is chosen such that Condition 5.1 is satisfied, similarly we have for all $k \ge k_0$ that

$$\mathbb{E}[\|\theta_k-\bar{\theta}\|_2^2] \le \frac{4\left((k_0+h)+\frac{780e\alpha^2\tau_k}{(\alpha\lambda_{\min}-1)}\right)}{\lambda_{\min}(1-\gamma)^2(k+h)}. \tag{5.6}$$

5.3. Putting together. Observe that the RHS of either (5.5) or (5.6) does not depend on $\bar{\theta}$ (due to Lemma 5.5). Therefore, the same bound holds for $\mathbb{E}[\|\theta_{t,K}-\bar{\theta}_t\|_2^2]$ for all $t = 0,1,\dots,T-1$. As a result, when using constant stepsize, we have by Jensen's inequality and (5.5) that

$$\mathbb{E}[\|\theta_{t,K}-\bar{\theta}_t\|_2] \le \mathbb{E}^{1/2}[\|\theta_{t,K}-\bar{\theta}_t\|_2^2] \le \frac{2\left((1-\alpha\lambda_{\min})^{\frac{K-\tau_\alpha}{2}}+\frac{12\sqrt{\alpha\tau_\alpha}}{\sqrt{\lambda_{\min}}}\right)}{(1-\gamma)\sqrt{\lambda_{\min}}},$$

where the last line follows from $\sqrt{a+b} \le \sqrt{a}+\sqrt{b}$ for all $a,b \ge 0$. Using the previous inequality in (5.3), we have Theorem 3.2(1). Theorem 3.2(2) follows from a similar approach of combining (5.6) and (5.3).

6. Conclusion. This work makes contributions toward the understanding of Q-learning with function approximation. In particular, we show that by adding target network and truncation, the resulting Q-learning algorithm under linear function approximation is provably stable and achieves a sample complexity of $\tilde{\mathcal{O}}(\epsilon^{-2})$ up to a function approximation error. Furthermore, the establishment of our results does not require strong assumptions (e.g., linear MDP, strong negative drift assumption, sufficiently small discount factor γ) as in related literature. Potential future directions of this work include (1) extension to Q-learning with nonlinear function approximation, and (2) extension to MDPs with continuous state-action space and unbounded reward, such as linear quadratic regulator. See the supplementary materials (supplement.pdf [local/web 301KB]) for a more detailed discussion about future directions.

Appendix A. A stochastic approximation result. Consider a stochastic approximation algorithm of the form

$$w_0 = \mathbf{0}, \quad w_{k+1} = w_k + \alpha_k G(w_k, Y_k), \tag{A.1}$$

where $\{Y_k\}$ is a finite-state Markov chain with state-space $\mathcal{Y}$, $G : \mathbb{R}^n \times \mathcal{Y} \mapsto \mathbb{R}^n$ is a (possibly) nonlinear operator, and $\{\alpha_k\}$ is a positive sequence of stepsizes.

Assumption A.1.

(1) The Markov chain $\{Y_k\}$ has a unique stationary distribution ν_Y, and it holds for any $k \geq 0$ that $\max_{y\in\mathcal{Y}} d_{\mathrm{TV}}(P_Y^k(y,\cdot), \nu_Y(\cdot)) \leq C_Y \rho_Y^k$ for some constant $C_Y > 0$ and $\rho_Y \in (0,1)$.

(2) There exists $L_1 \geq 1$ such that the operator $G(\cdot,\cdot)$ satisfies $\|G(w_1,y) - G(w_2,y)\|_2 \leq L_1\|w_1 - w_2\|_2$ and $\|G(\mathbf{0},y)\|_2 \leq L_1$ for any w_1, w_2, and y.

(3) The equation $\bar{G}(w) = \mathbb{E}_{Y\sim\nu_Y}[G(w,Y)] = 0$ has a unique solution w^*, and the following inequality holds for all $w \in \mathbb{R}^n$: $\langle \bar{G}(w), w - w^*\rangle \leq -\kappa\|w - w^*\|_2^2$, where $\kappa > 0$ is a positive constant.

(4) The sequence $\{\alpha_k\}$ is nonincreasing and satisfies $\sum_{i=k-\tilde{\tau}_k}^{k-1} \alpha_i \leq \min(\frac{1}{4L_1}, \frac{\kappa}{130L_1^2})$, where $\tilde{\tau}_k$ denotes the mixing time of the Markov chain $\{Y_k\}$ with precision α_k.

Theorem A.2 (Theorem 1 of [11]). *Consider $\{w_k\}$ generated by algorithm (A.1). Under Assumption A.1, we have the following results.*

(1) *When using constant stepsize, i.e., $\alpha_k \equiv \alpha$, we have for all $k \geq \tilde{\tau}_\alpha$ that*

$$\mathbb{E}[\|w_k - w^*\|_2^2] \leq c_1(1-\alpha\kappa)^{K-\tilde{\tau}_\alpha} + 130c_2\frac{\alpha\tilde{\tau}_\alpha}{\kappa}, \tag{A.2}$$

where $c_1 = (\|w^*\|_2 + 1)^2$ and $c_2 = L_1^2 c_1$.

(2) *When using diminishing stepsizes of the form $\alpha_k = \alpha/(k+h)$, where $\alpha > 1/\kappa$, we have for all $k \geq \tilde{k}_0 := \min\{\tilde{k} \geq 0 \mid \tilde{k} \geq \tilde{\tau}_{\tilde{k}}\}$ that*

$$\mathbb{E}[\|w_k - w^*\|_2^2] \leq \frac{c_1(\tilde{k}_0 + h)}{k+h} + \frac{780ec_2\alpha^2\tilde{\tau}_k}{(\alpha\kappa - 1)(k+h)}. \tag{A.3}$$

REFERENCES

[1] N. Agarwal, S. Chaudhuri, P. Jain, D. M. Nagaraj, and P. Netrapalli, *Online target Q-learning with reverse experience replay: Efficiently finding the optimal policy for linear MDPs*, in Proceedings of the International Conference on Learning Representations, 2021.

[2] L. Baird, *Residual algorithms: Reinforcement learning with function approximation*, in Machine Learning Proceedings 1995, Elsevier, Amsterdam, 1995, pp. 30–37, https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=DA12DC12763657BCF6093FA405C24F33?doi=10.1.1.50.7784&rep=rep1&type=pdf.

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, Nashua, NH, 1996.

[4] J. Bhandari, D. Russo, and R. Singal, *A finite-time analysis of temporal difference learning with linear function approximation*, in Proceedings of the Conference on Learning Theory, 2018, pp. 1691–1692.

[5] V. S. Borkar, *A concentration bound for contractive stochastic approximation*, Systems Control Lett., 153 (2021), 104947.

[6] V. S. Borkar and S. P. Meyn, *The ODE method for convergence of stochastic approximation and reinforcement learning*, SIAM J. Control Optim., 38 (2000), pp. 447–469.

[7] Q. Cai, Z. Yang, J. D. Lee, and Z. Wang, *Neural temporal difference and Q-learning provably converge to global optima*, Math. Oper. Res., (2023), https://doi.org/10.1287/moor.2023.1370.

[8] D. Carvalho, F. S. Melo, and P. Santos, *A new convergent variant of Q-learning with linear function approximation*, Adv. Neural Inf. Process. Syst., 33 (2020).

[9] Z. Chen, J. P. Clarke, and S. T. Maguluri, *Target Network and Truncation Overcome the Deadly Triad in Q-Learning*, preprint, arXiv:2203.02628, 2022.

[10] Z. Chen, S. T. Maguluri, S. Shakkottai, and K. Shanmugam, *Finite-sample analysis of off-policy TD-learning via generalized Bellman operators*, Adv. Neural Inf. Process. Syst., 34 (2021), pp. 21440–21452.

[11] Z. Chen, S. Zhang, T. T. Doan, J.-P. Clarke, and S. T. Maguluri, *Finite-sample analysis of nonlinear stochastic approximation with applications in reinforcement learning*, Automatica, 146 (2022), 110623.

[12] A. M. Devraj and S. Meyn, *Zap Q-learning*, in Advances in Neural Information Processing Systems, 2017, pp. 2235–2244.

[13] S. S. Du, J. D. Lee, G. Mahajan, and R. Wang, *Agnostic Q-learning with function approximation in deterministic systems: Near-optimal bounds on approximation error and sample complexity*, in Advances in Neural Information Processing Systems, 2020.

[14] S. S. Du, Y. Luo, R. Wang, and H. Zhang, *Provably efficient Q-learning with function approximation via distribution shift error checking oracle*, in Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019, pp. 8060–8070.

[15] Y. Duan, Z. Jia, and M. Wang, *Minimax-optimal off-policy evaluation with linear function approximation*, in International Conference on Machine Learning, PMLR, 2020, pp. 2701–2709.

[16] D. Ernst, P. Geurts, and L. Wehenkel, *Tree-based batch mode reinforcement learning*, J. Mach. Learn. Res., 6 (2005), pp. 503–556.

[17] J. Fan, Z. Wang, Y. Xie, and Z. Yang, *A theoretical analysis of deep Q-learning*, in Learning for Dynamics and Control, PMLR, 2020, pp. 486–489.

[18] Z. Gao, Q. Ma, T. Başar, and J. R. Birge, *Finite-Sample Analysis of Decentralized Q-Learning for Stochastic Games*, preprint, arXiv:2112.07859, 2021.

[19] L. Györfi, M. Kohler, A. Krzyzak, H. Walk, et al., *A Distribution-free Theory of Nonparametric Regression*, Vol. 1, Springer, Berlin, 2002.

[20] H. Hasselt, *Double Q-learning*, Adv. Neural Inf. Process. Syst., 23 (2010), pp. 2613–2621.

[21] T. Jaakkola, M. I. Jordan, and S. P. Singh, *Convergence of stochastic iterative dynamic programming algorithms*, in Advances in Neural Information Processing Systems, 1994, pp. 703–710.

[22] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, *Is Q-learning provably efficient?*, in Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 4868–4878.

[23] C. Jin, Z. Yang, Z. Wang, and M. I. Jordan, *Provably efficient reinforcement learning with linear function approximation*, in Proceedings of the Conference on Learning Theory, PMLR, 2020, pp. 2137–2143.
[24] S. Khodadadian, Z. Chen, and S. T. Maguluri, *Finite-sample analysis of off-policy natural actor-critic algorithm*, in Proceedings of the International Conference on Machine Learning, PMLR, 2021, pp. 5420–5431.
[25] D. Lee and N. He, *A unified switching system perspective and convergence analysis of Q-learning algorithms*, Adv. Neural Inf. Process. Syst., 33 (2020), pp. 15556–15567.
[26] D. A. Levin and Y. Peres, *Markov Chains and Mixing Times*, AMS, Providence, RI, 2017.
[27] G. Li, Y. Chen, Y. Chi, Y. Gu, and Y. Wei, *Sample-efficient reinforcement learning is feasible for linearly realizable MDPs with limited revisiting*, Adv. Neural Inf. Process. Syst., 34 (2021).
[28] G. Li, L. Shi, Y. Chen, Y. Gu, and Y. Chi, *Breaking the sample complexity barrier to regret-optimal model-free reinforcement learning*, Adv. Neural Inf. Process. Syst., 34 (2021).
[29] G. Li, Y. Wei, Y. Chi, Y. Gu, and Y. Chen, *Sample complexity of asynchronous Q-learning: Sharper analysis and variance reduction*, Adv. Neural Inf. Process. Syst., 33 (2020), pp. 7031–7043.
[30] S. Ma, Z. Chen, Y. Zhou, and S. Zou, *Greedy-GQ with variance reduction: Finite-time analysis and improved complexity*, in Proceedings of the International Conference on Learning Representations, 2021.
[31] H. R. Maei, C. Szepesvári, S. Bhatnagar, and R. S. Sutton, *Toward off-policy learning control with function approximation*, in Proceedings of the 27th International Conference on Machine Learning, 2010, pp. 719–726.
[32] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, *An analysis of reinforcement learning with function approximation*, in Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 664–671.
[33] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., *Human-level control through deep reinforcement learning*, Nature, 518 (2015), pp. 529–533.
[34] R. Munos and C. Szepesvári, *Finite-time bounds for fitted value iteration*, J. Mach. Learn. Res., 9 (2008).
[35] G. Qu and A. Wierman, *Finite-time analysis of asynchronous stochastic approximation and Q-learning*, in Proceedings of the Conference on Learning Theory, PMLR, 2020, pp. 3185–3205.
[36] H. Robbins and S. Monro, *A stochastic approximation method*, Ann. Math. Statist., 22 (1951), pp. 400–407.
[37] S. P. Singh and R. C. Yee, *An upper bound on the loss from approximate optimal-value functions*, Mach. Learn., 16 (1994), pp. 227–233.
[38] R. Srikant and L. Ying, *Finite-time error bounds for linear stochastic approximation and TD-learning*, in Proceedings of the Conference on Learning Theory, 2019, pp. 2803–2830.
[39] R. S. Sutton, *Open theoretical questions in reinforcement learning*, in European Conference on Computational Learning Theory, Springer, Berlin, 1999, pp. 11–17.
[40] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 2018.
[41] C. Szepesvári and R. Munos, *Finite time bounds for sampling-based fitted value iteration*, in Proceedings of the 22nd International Conference on Machine Learning, 2005, pp. 880–887.
[42] J. N. Tsitsiklis, *Asynchronous stochastic approximation and Q-learning*, Mach. Learn., 16 (1994), pp. 185–202.
[43] J. N. Tsitsiklis and B. Van Roy, *An analysis of temporal-difference learning with function approximation*, IEEE Trans. Automat. Control, 42 (1997), pp. 674–690.
[44] R. Wang, D. Foster, and S. M. Kakade, *What are the statistical limits of offline RL with linear function approximation?*, in Proceedings of the International Conference on Learning Representations, 2020.
[45] Y. Wang and S. Zou, *Finite-sample analysis of Greedy-GQ with linear function approximation under Markovian noise*, in Proceedings of the Conference on Uncertainty in Artificial Intelligence, PMLR, 2020, pp. 11–20.
[46] C. J. Watkins and P. Dayan, *Q-learning*, Mach. Learn., 8 (1992), pp. 279–292.

[47] T. Xie and N. Jiang, *Batch value-function approximation with only realizability*, in Proceedings of the International Conference on Machine Learning, PMLR, 2021, pp. 11404–11413.

[48] P. Xu and Q. Gu, *A finite-time analysis of Q-learning with neural network function approximation*, in Proceedings of the International Conference on Machine Learning, PMLR, 2020, pp. 10555–10565.

[49] T. Xu and Y. Liang, *Sample complexity bounds for two timescale value-based reinforcement learning algorithms*, in Proceedings of the International Conference on Artificial Intelligence and Statistics, PMLR, 2021, pp. 811–819.

[50] L. Yang and M. Wang, *Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound*, in Proceedings of the International Conference on Machine Learning, PMLR, 2020, pp. 10746–10756.

[51] A. Zanette, *Exponential lower bounds for batch reinforcement learning: Batch RL can be exponentially harder than online RL*, in Proceedings of the International Conference on Machine Learning, PMLR, 2021, pp. 12287–12297.

[52] A. Zanette, A. Lazaric, M. Kochenderfer, and E. Brunskill, *Learning near optimal policies with low inherent Bellman error*, in Proceedings of the International Conference on Machine Learning, PMLR, 2020, pp. 10978–10989.

[53] A. Zanette and M. Wainwright, *Stabilizing Q-learning with linear architectures for provable efficient learning*, in Proceedings of the International Conference on Machine Learning, PMLR, 2022, pp. 25920–25954.

[54] S. Zhang, H. Yao, and S. Whiteson, *Breaking the deadly triad with a target network*, in Proceedings of the 38th International Conference on Machine Learning, PMLR, 2021, pp. 12621–12631.

[55] S. Zou, T. Xu, and Y. Liang, *Finite-sample analysis for SARSA with linear function approximation*, in Advances in Neural Information Processing Systems, 2019, pp. 8668–8678.