Finite-Time Theory for Momentum Q-learning

Bowen Weng[1] **Huaqing Xiong**[1] **Lin Zhao**[2] **Yingbin Liang**[1] **Wei Zhang**[3]

[1]Department of Electrical and Computer Engineering, The Ohio State University, Columbus, Ohio, USA
[2]Department of Electrical and Computer Engineering , National University of Singapore , Singapore, Republic of Singapore
[3]Department of Mechanical and Energy Engineering, Southern University of Science and Technology, Shenzhen, China

Abstract

Existing studies indicate that momentum ideas in conventional optimization can be used to improve the performance of Q-learning algorithms. However, the finite-time analysis for momentum-based Q-learning algorithms is only available for the tabular case without function approximation. This paper analyzes a class of momentum-based Q-learning algorithms with finite-time convergence guarantee. Specifically, we propose the MomentumQ algorithm, which integrates the Nesterov's and Polyak's momentum schemes, and generalizes the existing momentum-based Q-learning algorithms. For the infinite state-action space case, we establish the convergence guarantee for MomentumQ with linear function approximation under Markovian sampling. In particular, we characterize a finite-time convergence rate which is provably faster than the vanilla Q-learning. This is the first finite-time analysis for momentum-based Q-learning algorithms with function approximation. For the tabular case under synchronous sampling, we also obtain a finite-time convergence rate that is slightly better than the SpeedyQ [Azar et al., 2011]. Finally, we demonstrate through various experiments that the proposed MomentumQ outperforms other momentum-based Q-learning algorithms.

1 INTRODUCTION

Reinforcement learning (RL) aims to design strategies for an agent to find a desirable policy through interacting with an environment in order to maximize an accumulative reward for a task. RL has received drastically growing attention in recent years and accomplished tremendous success in various application domains such as playing video games [Mnih et al., 2013], bipedal walking robot [Castillo et al., 2019], board game [Silver et al., 2017], to name a few. This paper focuses on Q-learning, which is a widely used model-free RL algorithm for finding the action-value function (known as the Q-function) of the optimal policy.

Q-learning was first proposed in Watkins and Dayan [1992] and has been studied extensively since then. For scenarios with a finite state-action space, the Q-function can be conveniently represented as a tabular function. The convergence of Q-learning in the tabular case was proved in Jaakkola et al. [1994]. In the case with a continuous state-action space, one typically approximates the Q-function with a parameterized function class of a relatively small parameter dimension. Among the rich approximation classes, linear function approximation [Bertsekas and Tsitsiklis, 1996, Sutton and Barto, 2018] and neural network function approximation [Mnih et al., 2013] are often adopted in the literature. We will review these studies in more details in Section 1.2.

The central idea of Q-learning algorithms is to solve an optimal Bellman equation [Bertsekas and Tsitsiklis, 1996] iteratively as a fixed point problem. Since the Bellman operator is expressed as the expected value over the underlying Markov decision processes (MDP) which is unknown, Q-learning (as a model-free algorithm) approximates it via its sampled version, and such an update can be viewed analogously to the first-order (stochastic) gradient descent algorithm [Baird, 1995]. This connection thus motivated several studies on accelerating Q-learning by incorporating various momentum schemes, such as Heavy-ball (HB) [Polyak, 1964] and Nesterov's accelerated gradient (NAG) [Nesterov, 2013] which were shown to accelerate gradient descent in conventional optimization algorithms. For example, speedy Q-learning (SpeedyQ) proposed in Azar et al. [2011] can be viewed as incorporating NAG to Q-learning and has been proved to accelerate the convergence with particularly designed learning rate in the tabular case. Devraj et al. [2019] applied HB and NAG to Q-learning separately with a matrix learning rate in the tabular case and analyzed their asymptotic properties. Vieillard et al. [2019] proposed momentum-based value iteration by viewing the greedy policy as an analog of gradient

Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021), PMLR 161:665–674.

ascent. However, the convergence is not guaranteed. To the best of our knowledge, the *finite-time* convergence rate has not been established for momentum-based Q-learning algorithms with function approximation. The focus of this paper is to address the above important question.

1.1 MAIN CONTRIBUTIONS

This paper investigates a general momentum-based Q-learning scheme (referred to as MomentumQ hereafter), which involves both NAG-type and HB-type of history information for accelerating Q-learning. The main contribution of this paper is three-fold.

First, we establish the finite-time convergence rate for MomentumQ with linear function approximation, and we show that this algorithm provably accelerates vanilla Q-learning. To the best of our knowledge, this is the first finite-time convergence guarantee for momentum-based Q-learning with linear function approximation.

Second, the only existing finite-time baseline bounds for momentum-based Q-learning is given by SpeedyQ [Azar et al., 2011] and Generalized SpeedQ [John et al., 2020] for the tabular case. Hence, to be able to compare with such a baseline, we also provide a finite-time analysis of MomentumQ in the tabular case. We show that it achieves a better (but order-wisely the same) convergence rate than SpeedyQ. Technically, due to the additional momentum terms in MomemtumQ, its analysis is more challenging than SpeedyQ and requires substantial new technical developments.

Finally, our numerical results show that the proposed MomentumQ outperforms the vanilla Q-learning as well as the other existing momentum-based Q-learning algorithms for both tabular and function approximation cases.

1.2 RELATED WORK

We review the most relevant studies on Q-learning here with a focus on the theoretical convergence analysis.

Q-learning with function approximation: When the state-action space is considerably large or even continuous, it is practical to properly discretize the space [Shah and Xie, 2018], or parameterize the Q-function with a certain function class. For linear MDP, Melo and Ribeiro [2007], Yang and Wang [2019] proposed provably sample-efficient Q-learning algorithms with linear function approximation. For more general MDPs with linear function approximation of the Q-function, finite-time convergence analysis was established in Zou et al. [2019], Chen et al. [2019] under Markovian sampling, in Du et al. [2019] on exploration samples and in Weng et al. [2020a] by incorporating Adam-type updates. Recently, Cai et al. [2019], Fan et al. [2019] and Xu and Gu [2019] established the convergence rate of Q-learning with neural network approximation in the over-parameterized regime under i.i.d. and non-i.i.d sampling, respectively.

Tabular Q-learning: Q-learning was first proposed in Watkins and Dayan [1992] under finite state-action space. Regarding the theoretical studies, research of tabular Q-learning has focused on the asymptotic convergence which was usually studied via its connection to the corresponding stochastic approximation algorithm (see, for example, Tsitsiklis [1994], Jaakkola et al. [1994], Borkar and Meyn [2000], Melo [2001]). More recently, Lee and He [2019] provided asymptotic results for asynchronous Q-learning by formulating it as a switching affine system. Another research line has focused on the finite-time (i.e., non-asymptotic) analysis. Finite-time performance for Q-learning was first established in Szepesvári [1998]. Considering both synchronous and asynchronous Q-learning, Even-Dar and Mansour [2003] investigated the convergence rates under different choices of the learning rates. Sharper bounds on the finite-time convergence rate have been established in more recent work [Wainwright, 2019a, Qu and Wierman, 2020, Li et al., 2020].

Momentum-based Q-learning: For tabular Q-learning, several studies incorporated the momentum idea in conventional optimization to accelerate the convergence. Azar et al. [2011] proposed the SpeedyQ algorithm and characterized the finite-time performance. John et al. [2020] generalized SpeedyQ by introducing a relaxation parameter which was used to modify sample distribution and to redistribute the momentum. Devraj et al. [2019] extended HB and NAG with a matrix learning rate on the momentum. The asymptotic performance was analyzed under simplified assumptions. Vieillard et al. [2019] proposed a momentum-based value iteration and generalized the scheme to DQN. While some theoretical properties of the algorithms were explored in the tabular case, the convergence of the algorithm was not established. Among these studies, only Azar et al. [2011], John et al. [2020] characterized the finite-time rate for SpeedyQ in the tabular case, and such finite-time analysis for momentum-based Q-learning algorithms has not been provided for the function approximation case, which is the focus of this paper.

Other variants of Q-learning: Other than the above momentum-based Q-learning algorithms, which mainly exploit the acceleration ideas in conventional optimization, Q-learning also inspires a number of other variants, including residual Q-learning [Baird, 1995], phased Q-learning [Kearns and Singh, 1999], Zap Q-learning [Devraj and Meyn, 2017], periodic Q-learning [Lee and He, 2020], variance reduced Q-learning [Wainwright, 2019b, Li et al., 2020], and double Q-learning [Hasselt, 2010, Weng et al., 2020b, Xiong et al., 2020b] to name a few. These algorithms are proposed to speed up convergence rates or improve the performance by mitigating various issues in the implementation of Q-learning. In this paper, we mainly focus on the

momentum-based Q-learning algorithm motivated by the optimization idea.

2 PRELIMINARIES

In this section, we provide the necessary background.

2.1 MARKOV DECISION PROCESS

We consider the standard reinforcement learning setting, where a learning agent interacts with a (possibly stochastic) environment modeled as a discrete-time discounted Markov decision process (MDP). Such an MDP is characterized by a quintuple $(\mathcal{X}, \mathcal{U}, P, R, \gamma)$, where $\mathcal{X}$ is the state space, $\mathcal{U}$ is the action space, $P : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \mapsto [0, 1]$ is the probability transition kernel, namely, $P(\cdot|x, u)$ denotes the probability that the system takes the next state given the current state x and action u. In addition, $R : \mathcal{X} \times \mathcal{U} \mapsto [0, R_{\max}]$ denotes the reward function (or negative of the cost function) mapping the state-action pairs to a bounded subset of $\mathbb{R}$, and $\gamma \in (0, 1)$ is the discount factor. A policy $\pi : \mathcal{X} \mapsto \mathcal{U}$ represents a strategy to take actions, i.e., it captures the probability of taking each action at any given state. By following a policy π, we perform an action u_k with probability $\pi(u_k|x_k)$ at time k, observe a reward $r_k = R(x_k, u_k)$, and evolve to the next state x_{k+1} with the probability $P(x_{k+1}|x_k, u_k)$.

We define the value function as the expected return of following policy π and starting from state x, given by $V^\pi(x) = \mathbb{E}_P \sum_{k=0}^{\infty} \gamma^k r_k$, where $\mathbb{E}_P$ denotes the expectation with respect to the transition probability P. The Q-function is defined as the state-action value function $Q^\pi(x, u) = R(x, u) + \gamma \sum_{y \in \mathcal{X}} P(y|x, u)V^\pi(y)$, which is the return of performing action u at state s at the first step and following policy π thereafter.

2.2 TABULAR Q-LEARNING

Q-learning seeks to maximize the expected discounted return over policy π as formulated below.

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & V^\pi(x_0) = \mathbb{E}_P \left[\sum_{k=0}^{\infty} \gamma^k R(x_k, \pi(x_k)) \right], \\ \text{subject to} \quad & x_{k+1} \sim P(\cdot|x_k, \pi(x_k)), \end{aligned} \tag{1}$$

We let $\pi^\star$ denote the optimal stationary policy $\pi^\star : \mathcal{X} \mapsto \mathcal{U}$ which is the solution of the above optimization problem.

Define the Bellman operator $\mathcal{T}$ pointwisely as

$$\mathcal{T}Q(x, u) = R(x, u) + \gamma \mathbb{E}_P \max_{u' \in U(x')} Q(x', u'), \tag{2}$$

where $x' \sim P(\cdot|x, u)$ and $U(x)$ denotes the admissible set of actions at state x. It can be shown that the Bellman operator $\mathcal{T}$ is γ-contractive in the supremum norm $\|Q\| := \sup_{x,u} |Q(x, u)|$, i.e., it satisfies

$$\|\mathcal{T}Q(x, u) - \mathcal{T}Q'(x, u)\| \leq \gamma \|Q(x, u) - Q'(x, u)\|. \tag{3}$$

Thus, $\mathcal{T}$ has a unique fixed point $Q^\star$ given by

$$Q^\star(x, u) = R(x, u) + \gamma \mathbb{E}_P \max_{u' \in U(x')} Q^\star(x', u'). \tag{4}$$

The above property suggests that starting with an arbitrary initial Q-function, we can apply the Bellman operator $\mathcal{T}$ iteratively to learn $Q^\star$. Hence, the optimal policy can be obtained from the optimal Q-function as:

$$\pi^\star(x) = \underset{u \in U(x)}{\text{argmax}}\, Q^\star(x, u), \forall x \in \mathcal{X}. \tag{5}$$

Note that the knowledge of the transition probability P is not needed in (5), which is one advantage of Q-learning.

In practice, exact evaluation of the Bellman operator (2) is usually not feasible due to the lack of the knowledge of the transition probability kernel. Instead, the sample-based *empirical Bellman operator* is used as an estimator. Specifically, for the kth round of iteration at the state-action pair (x, u), we sample the next state $y_k \sim P(\cdot|x, u)$, and then evaluate the empirical Bellman operator $\hat{\mathcal{T}}_k$ as

$$\hat{\mathcal{T}}_k Q_k(x, u) = R(x, u) + \gamma \max_{u' \in U(y_k)} Q_k(y_k, u'). \tag{6}$$

Then the update of tabular Q-learning is implemented as

$$Q_{k+1} = Q_k - \alpha_k(Q_k - \hat{\mathcal{T}}_k Q_k), \tag{7}$$

where α_k is the stepsize and we omit the dependence on (x, u) for simplicity when there is no confusion.

2.3 Q-LEARNING WITH LINEAR FUNCTION APPROXIMATION

For relatively large or even infinite state-action space $\mathcal{X} \times \mathcal{U}$, it is impractical to express the Q-function in an explicit tabular form with respect to each state-action pair. In such a case, the update rule of (7) is no longer directly applicable.

To handle such cases, a parametric function $\hat{Q}(x, u; \theta)$ is adopted as an approximation of the Q-function, where the parameter vector θ is of small dimension. Our focus here is the linear function class, which is often considered in the literature for establishing the finite-time analysis [Zou et al., 2019, Chen et al., 2019, Du et al., 2019]. Then the Q-function $\hat{Q}(x, u; \theta)$ can be written as

$$\hat{Q}(x, u; \theta) = \Phi(x, u)^T \theta, \tag{8}$$

where $\theta \in \mathbb{R}^d$, and $\Phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ is a vector function of size d, and the elements of Φ represent the nonlinear kernel

(feature) functions. Correspondingly, the updating rule of Q-learning with linear function approximation is given by

$$\theta_{k+1} = \theta_k - \alpha_k \Phi(x_k, u_k) \left[\Phi(x_k, u_k)^T \theta_k - R(x_k, u_k) - \gamma \max_{u' \in U(x_{k+1})} \Phi(x_{k+1}, u')^T \theta_k \right], \quad (9)$$

where α_k is the stepsize.

3 MOMENTUMQ ALGORITHM

In this section, we introduce the MomentumQ algorithm.

3.1 TABULAR MOMENTUMQ

Overall, MomentumQ integrates the Nesterov's momentum [Nesterov, 2013] and Polyak's Momentum [Polyak, 1964] together, with the learning rates flexibly interpolating between the two to optimize the momentum performance. Specifically, MomentumQ takes the form given by

$$\begin{aligned} S_k &= (1 - a_k) Q_{k-1} + a_k \hat{\mathcal{T}}_k Q_{k-1}, \\ P_k &= (1 - a_k) Q_k + a_k \hat{\mathcal{T}}_k Q_k, \\ Q_{k+1} &= P_k + \underbrace{b_k (P_k - S_k)}_{\text{Nesterov's momentum}} + \underbrace{c_k (Q_k - Q_{k-1})}_{\text{Polyak's momentum}}, \end{aligned} \quad (10)$$

where a_k, b_k, c_k determine the learning rates. Algorithm 1 implements MomentumQ with a particular family of learning rates under synchronous sampling [Even-Dar and Mansour, 2003]. One special feature of the algorithm is the additional freedom introduced by the hyperparameter m. This hyperparameter actually stems from our finite analysis of MomentumQ in the tabular case. It helps to better bound the propagation of the learning error (i.e., $Q * - Q_k$) via the particularly designed learning rate. Unlike SpeedyQ, which only admits a fixed set of learning rate (namely, $1/k$), MomentumQ's learning rate is more flexible due to the introduction of this hyperparameter. In practice, we found that in general setting $m = 1/\gamma + c$ for some arbitrary small number $c > 0$ (e.g., $c = 1$), will generally yields a very good performance in suppressing the overshoot of the learning error which was otherwise very significant in SpeedyQ learning. We will also see later in the simulation that the proposed algorithm accelerates the convergence for a number of arbitrarily chosen m that satisfies $m \geq 1/\gamma$.

Note that the proposed MomentumQ algorithm in (10) contains not only the momentum term $\hat{\mathcal{T}}_k Q_{k-1}$ in the update, but also the historical information Q_{k-1} explicitly. We compare MomentumQ with SpeedyQ [Azar et al., 2011] given by

$$Q_{k+1} = Q_k + a_k (\hat{\mathcal{T}}_k Q_k - Q_k) + (1 - a_k)(\hat{\mathcal{T}}_k Q_k - \hat{\mathcal{T}}_k Q_{k-1}). \quad (11)$$

Algorithm 1 Synchronous Tabular MomentumQ

Input: Initial action-value function Q_0 and $Q_{-1} = Q_0$, discount factor γ, parameter $m \geq \frac{1}{\gamma}$, and iteration number T
for $k = 0, 1, 2, \cdots, T-1$ **do**
  $a_k = \frac{1}{k+1}$, $b_k = k - m - 1$, $c_k = \frac{-k^2 + (m+1)k + 1}{k+1}$;
  for each $(x, u) \in \mathcal{X} \times U(x)$ **do**
    Generate the next state sample $y_k \sim P(\cdot|x, u)$;
    $\hat{\mathcal{T}}_k Q_{k-1}(x, u) = R(x, u) + \gamma \max_{u \in U(y_k)} Q_{k-1}(y_k, u)$;
    $\hat{\mathcal{T}}_k Q_k(x, u) = R(x, u) + \gamma \max_{u \in U(y_k)} Q_k(y_k, u)$;
    $S_k(x, u) = (1 - a_k) Q_{k-1}(x, u) + a_k \hat{\mathcal{T}}_k Q_{k-1}(x, u)$;
    $P_k(x, u) = (1 - a_k) Q_k(x, u) + a_k \hat{\mathcal{T}}_k Q_k(x, u)$;
    $Q_{k+1}(x, u) = P_k(x, u) + b_k (P_k(x, u) - S_k(x, u)) + c_k (Q_k(x, u) - Q_{k-1}(x, u))$;
  end for
end for
Output: Q_T

We see from (11) that SpeedyQ contains only the momentum term $\hat{\mathcal{T}}_k Q_{k-1}$ in the update. In contrast, MomentumQ additionally incorporates the historical information Q_{k-1} explicitly. Indeed, the simulation in Section 5 indicates that MomentumQ effectively smooths out the large overshoots that are present in SpeedyQ and converges faster by incorporating additional historical terms. The finite-time analysis of MomentumQ is more challenging than SpeedyQ due to this difference, since the additional Q_{k-1} term increases the order of the recursion. We will discuss in more details later. Furthermore, (11) simply involves $\hat{\mathcal{T}}_k Q_k - \hat{\mathcal{T}}_k Q_{k-1}$ as the only momentum term, while our algorithm designs this part more systematically. We directly use two consecutive outputs of the empirical Bellman operators to update the Q-function and obtain S_k and P_k. Compared to (11), the resulted Nesterov's momentum is potentially a better estimation of the "gradient" for updating the Q function. This intuition is also verified in our numerical results.

3.2 MOMENTUMQ WITH LINEAR FUNCTION APPROXIMATION

For the case where the state-action space is considerably large, we use the linear function approximation to estimate the Q-function to overcome the curse of dimensionality.

Consider the case where the Q-function is approximated by a linear parameterized function. We propose MomentumQ for this case as

$$\theta_{k+1} = \theta_k + (b_k + c_k)(\theta_k - \theta_{k-1}) - a_k (1 + b_k) g_k + a_k b_k g_{k-1}, \quad (12)$$

where

$$g_k := g(\theta_k; x_k, u_k, x_{k+1})$$

Algorithm 2 MomentumQ with linear approximation

Input: Initial parameters $\theta_{-1} = \theta_0$; discount factor γ; iteration number T.
for $k = 0, 1, 2, \cdots, T-1$ **do**
  Assign a_k, b_k, c_k;
  Sample $u_k \sim \pi, x_{k+1} \sim P(\cdot|x_k, u_k)$;
  Compute g_k as (13);
  Update $\theta_{k+1} = \theta_k + (b_k + c_k)(\theta_k - \theta_{k-1})$
  $-a_k(1 + b_k)g_k + a_k b_k g_{k-1}$;
end for
Output: θ_{out}.

$$= \Big(\Phi(x_k, u_k)^T \theta_k - \gamma \max_{u' \in U(x_{k+1})} \Phi(x_{k+1}, u')^T \theta_k - R(x_k, u_k) \Big) \Phi(x_k, u_k). \quad (13)$$

We focus on the more practical Markovian sampling model, in which the data tuples are sequentially drawn from a single trajectory under an unknown stationary distribution. More implementation details are referred to Algorithm 2.

4 FINITE-TIME CONVERGENCE RESULTS

In this section, we present our main results on the finite-time convergence rate guarantee for MomentumQ. We focus on linear function class and provide the first finite-time analysis for momentum-based Q-learning with function approximation. We also present our study of tabular MomentumQ in order to make a comparison with the only existing theory baseline for momentum-based Q-learning, which was established for tabular SpeedyQ.

4.1 MOMENTUMQ WITH LINEAR FUNCTION APPROXIMATION UNDER MARKOVIAN SAMPLING

In this section, we characterize the finite-time convergence guarantee for the proposed MomentumQ algorithm with linear function approximation under Markovian sampling. To proceed the convergence analysis, we first define

$$\bar{g}(\theta) := \mathop{\mathbb{E}}_{\mu}[g(\theta; x, u, x')] = \mathop{\mathbb{E}}_{\mu}\left[\Phi(x, u)^T \theta - R(x, u) - \gamma \max_{u' \in U(x')} \Phi(x', u')^T \theta \right] \Phi(x, u), \quad (14)$$

where the expectation is taken over the stationary distribution of the sampling tuple (x, u, x').

We take the following standard assumptions in our analysis.

Assumption 1. *The columns of Φ are linearly independent and $\|\Phi\|_2 \leq 1$.*

Assumption 2. *The term $\bar{g}(\cdot)$ has a unique root denoted as $\theta^\star$, i.e., $\bar{g}(\theta^\star) = 0$. There exists a constant $\delta > 0$, such that for any $\theta \in \mathbb{R}^d$ we have*

$$(\theta - \theta^\star)^T \bar{g}(\theta) \geq \delta \|\theta - \theta^\star\|_2^2. \quad (15)$$

Assumption 3. *The domain of the approximation parameters θ is contained in a ball $\mathcal{B}$ centered at $\theta = 0$ with a bounded diameter $D_{\max}$ and the optimal parameter $\theta^\star \in \mathcal{B}$. That is, there exists $D_{\max}$, such that $\|\theta - \theta'\|_2 \leq D_{\max}, \forall \theta, \theta' \in \mathcal{B}$, and $\theta^\star \in \mathcal{B}$.*

Assumption 4. *There exist constants $\sigma > 0$ and $\rho \in (0, 1)$ such that*

$$\sup_{x \in \mathcal{X}} d_{TV}(\mathbb{P}(x_k \in \cdot | x_0 = x), \mu) \leq \sigma \rho^k \quad \forall k,$$

where $d_{TV}(\mu, \nu)$ denotes the total-variation distance between the probability measures μ and ν.

Assumptions 1 and 2 are standard in the literature on theoretical analysis of Q-learning algorithms with linear function approximation [Bhandari et al., 2018, Chen et al., 2019, Zou et al., 2019]. The boundedness condition in Assumption 1 can be justified by normalization and hence does not lose generalization. Assumption 4 can easily hold for irreducible and aperiodic Markov chains, and is widely adopted in the literature on theoretical analysis of RL algorithms under Markovian sampling [Bhandari et al., 2018, Chen et al., 2019, Zou et al., 2019, Xu and Gu, 2019, Xiong et al., 2020a]. For Assumption 4, we further define the quantity of the mixing time $\tau^{mix}(\cdot)$ as follows, which denotes the duration of the time for the Markov chain to approach sufficiently close to its steady-state

$$\tau^* := \tau^{mix}(\kappa) := \min \left\{ k = 1, 2, \ldots | \sigma \rho^k \leq \kappa \right\}. \quad (16)$$

To understand the challenges of analyzing Markovian sampling in MomemtumQ, we first illustrate how a non-zero bias is introduced if the Markovian sampling is considered. For simplicity, we denote $O_k := (x_k, u_k, x_{k+1})$ as the data at time step k sampled from a Markov chain. Recall $g_k(\theta; O_k)$ in (13), and $\bar{g}(\theta) = \mathbb{E}[g(\theta; O_k)]$ in (14) where the expectation is taken over the marginal distribution of O_k since θ is fixed. However, if θ is random and dependent on O_k, the equality no longer holds. In particular, since θ_k is dependent on the historical tuples $\{O_1, O_2, \ldots, O_k\}$, we have

$$\bar{g}(\theta_k) \neq \mathbb{E}[g(\theta_k; O_k) | \theta_k].$$

Thus, we have a non-zero bias due to Markovian sampling to approximate the expectation of $g_k^T(\theta_k - \theta^\star)$. Namely,

$$\mathbb{E}[g_k^T(\theta_k - \theta^\star)] = \mathbb{E}[\bar{g}(\theta_k)^T(\theta_k - \theta^\star)] + \mathbb{E}[(g_k - \bar{g}(\theta_k))^T(\theta_k - \theta^\star)],$$

where the second term on the right hand side captures the bias, which is the key challenge of the analysis under this setting. The following lemma develops an important upper bound on the bias term, which is a key step in the convergence analysis.

Lemma 1. *Suppose that Assumptions 1-4 hold and fix $\kappa > 0$ in* (16)*. Let MomentumQ update as* (12) *by choosing non-increasing a_k, b_k, c_k and denote $\beta_k = b_k + c_k$ with $\beta_k \in (0,1)$. Then we have*

$$\mathbb{E}[(g_k - \bar{g}(\theta_k))^T(\theta_k - \theta^\star)] \leq \begin{cases} \eta_1 \sum_{i=1}^{k-1} \beta_i + \eta_2 \sum_{i=1}^{k-1} a_i, & k \leq \tau^*; \\ 4D_{\max}G_{\max}\kappa + \eta_1\tau^*\beta_{k-\tau^*} + \eta_2\tau^* a_{k-\tau^*}, & k > \tau^*, \end{cases}$$

where $\eta_1 = 2D_{\max}((1+\gamma)D_{\max} + G_{\max}), \eta_2 = 6G_{\max}((1+\gamma)D_{\max} + G_{\max})$ with $G_{\max} = 2D_{\max} + R_{\max}$.

With the bias term bounded, we are ready to provide the convergence result for MomentumQ with linear function approximation under Markovian sampling.

Theorem 1. *(MomentumQ with constant learning rate) Suppose that Assumptions 1-4 hold and fix $\kappa > 0$ in* (16)*. Let $a_k = \alpha, b_k + c_k = \beta\lambda^k$ where $\beta, \lambda \in (0,1)$ and $\alpha \in (0, \frac{1-\lambda}{2\delta})$. After running T steps of Algorithm 2 under Markovian sampling, we take the output $\theta_{out} = \theta_T$ and have*

$$\mathbb{E}\left\|\theta_{out} - \theta^\star\right\|_2^2 \tag{17}$$

$$\leq \prod_{i=0}^{T-1}(1 - 2\alpha\delta(1+b_i))\left\|\theta_0 - \theta^\star\right\|_2^2 + \beta\left(\frac{2\eta_1\tau^*}{\delta} + \frac{C}{1-2\alpha\delta-\lambda}\right)(1-2\alpha\delta)^{T-1-\tau^*} + \frac{15G_{\max}^2\alpha}{2\delta} + \frac{2\eta_2\tau^*\alpha}{\delta} + \frac{8D_{\max}G_{\max}\kappa}{\delta}, \tag{18}$$

where $C = 5D_{\max}^2 + 2\alpha D_{\max}G_{\max} + 4\alpha\eta_1\tau^*\lambda$ with $G_{\max} = 2D_{\max} + R_{\max}$, and η_1, η_2 are defined in Lemma 1.

Theorem 1 indicates that the convergence behavior is determined by five terms. The first two terms capture the convergence rate as T changes, indicating that with a constant learning rate, MomentumQ enjoys an exponential convergence rate to a neighborhood of the global optimum. Since $\prod_{i=0}^{T-1}(1-2\alpha\delta(1+b_i)) < (1-2\alpha\delta)^T$, the dominant term of the convergence rate is the second term. The last three terms capture the convergence error. Since one usually chooses $\kappa = \alpha_k = \alpha$, the convergence error can be made as small as possible by choosing a sufficiently small learning rate.

As a comparison, the convergence of the vanilla Q-learning under similar assumptions and Markovian sampling is obtained in Chen et al. [2019] as $\mathbb{E}\left\|\theta_{\text{out}} - \theta^\star\right\|_2^2 \leq (1-2\delta\alpha)^T\left\|\theta_0 - \theta^\star\right\|_2^2 + \alpha C_1 + \kappa C_2$ for some constants C_1, C_2. Clearly, the dominant order in (18) can have a smaller coefficient than that of the vanilla Q-learning by setting a small β, so that MomentumQ can enjoy a better convergence rate.

In addition, one can also observe that α, β control a set of tradeoffs. First, while smaller α yields a smaller convergence error, it also slows down the convergence rate. As for β, although smaller β yields a smaller coefficient in the dominant term, it can also slow down the convergence rate because b_i in the first term needs to be small.

Next, we seek to remove the convergence error and balance the tradeoff caused by the choice of a_k. To this end, we can choose a diminishing learning rate and obtain the following theorem.

Theorem 2. *(MomentumQ with diminishing learning rate) Suppose that Assumptions 1-4 hold and fix $\kappa > 0$. Let $a_k = \frac{\alpha}{\sqrt{k}}, b_k + c_k = \beta\lambda^k$ with $\alpha > 0, \beta, \lambda \in (0,1)$. After running T steps of Algorithm 2 under Markovian sampling, we take the output $\theta_{out} = \frac{1}{T}\sum_{k=1}^{T}\theta_k$ and have*

$$\mathbb{E}\left\|\theta_{out} - \theta^\star\right\|_2^2 \leq \frac{D_{\max}^2/\alpha + 30\alpha G_{\max}^2 + 16\tau^*\alpha\eta_2}{2\delta\sqrt{T}} + \frac{8D_{\max}G_{\max}\kappa}{\delta} + \frac{1}{T}\left[\frac{5\beta D_{\max}^2}{2\alpha\delta(1-\lambda)^2} + \frac{D_{\max}G_{\max}\beta\lambda + 4\tau^*\eta_1\beta\lambda}{\delta(1-\lambda)}\right],$$

where η_1, η_2 are defined in Lemma 1.

In Theorem 2, if we choose $\kappa = \alpha_k = \alpha/\sqrt{T}$, then the mixing time $\tau^* = \mathcal{O}(\log T)$. Thus, MomentumQ converges to the global optimum at a rate of $\mathcal{O}(\log T/\sqrt{T})$ under a diminishing learning rate.

4.2 TABULAR MOMENTUMQ

In this subsection, we provide the finite-time analysis for tabular MomentumQ as listed in Algorithm 1. As we mention before, MomentumQ combines different types of momentum terms dynamically. This requires substantial new technical developments here in the convergence analysis.

We assume that the state space $\mathcal{X}$ and the action space $\mathcal{U}$ are finite with cardinalities $|\mathcal{X}|$ and $|\mathcal{U}|$, respectively. We denote $n = |\mathcal{X}| \cdot |\mathcal{U}|$. We also need the following assumption in our analysis.

Assumption 5. *The Q-function is uniformly bounded throughout the learning process. That is, $\exists V_{\max}$, such that $\|Q_k\| \leq V_{\max}, \forall k \geq 0$.*

Note that it is nontrivial to show the boundedness of the proposed iteration scheme. In fact, it is usually assumed for proving convergence of many such complicated stochastic

approximation algorithms [Kushner and Yin, 2003]. Alternatively, one can extend the ODE method [Borkar and Meyn, 2000] considerably to show the boundedness, which we leave for our future work.

To facilitate the analysis, we rewrite (10) in a more compact form as

$$\begin{aligned} Q_{k+1} =&(1-a_k)Q_k+[b_k(1-a_k)+c_k]\left(Q_k-Q_{k-1}\right) \\ &+a_k\left[(1+b_k)\hat{\mathcal{T}}_kQ_k-b_k\hat{\mathcal{T}}_kQ_{k-1}\right]. \end{aligned} \tag{19}$$

Our analysis first bounds the errors of approximating the exact Bellman operator $\mathcal{T}$ with empirical Bellman operators $\hat{\mathcal{T}}_k$. For convenience, we denote the last term in (19) by

$$\mathcal{D}_k\left[Q_k,Q_{k-1}\right]:=(1+b_k)\hat{\mathcal{T}}_kQ_k-b_k\hat{\mathcal{T}}_kQ_{k-1}, \tag{20}$$

for all $k\geq 0$. Note that $\mathcal{D}_k$ is a function of all samples $\{y_1,y_2,\cdots,y_k\}$ for all state-action pairs (x,u) up to round k. Let $\mathcal{F}_k$ denote the filtration generated by the sequence of these random variables $\{y_1,y_2,\cdots,y_k\}$. We see that $\mathcal{D}_k\in\mathcal{F}_k$ and $Q_{k+1}\in\mathcal{F}_k$. Then if we define $\mathcal{D}\left[Q_k,Q_{k-1}\right]$ as the conditional expectation of $\mathcal{D}_k\left[Q_k,Q_{k-1}\right]$ given $\mathcal{F}_{k-1}$, we obtain by the definition of $\mathcal{T}$ that

$$\begin{aligned} \mathcal{D}\left[Q_k,Q_{k-1}\right]:&=\mathbb{E}_P\left(\mathcal{D}_k\left[Q_k,Q_{k-1}\right]|\mathcal{F}_{k-1}\right) \\ &=(1+b_k)\mathcal{T}Q_k-b_k\mathcal{T}Q_{k-1}. \end{aligned} \tag{21}$$

Now define the error between $\mathcal{D}_k$ and $\mathcal{D}$ as follows:

$$\epsilon_k:=\mathcal{D}\left[Q_k,Q_{k-1}\right]-\mathcal{D}_k\left[Q_k,Q_{k-1}\right]. \tag{22}$$

Clearly $\mathbb{E}_P\left(\epsilon_k|\mathcal{F}_{k-1}\right)=0$. This shows that $\forall(x,u)\in\mathcal{X}\times U(x)$, the sequence of the estimation errors $\{\epsilon_k(x,u)\}_{k=0}^T$ is a martingale difference sequence with respect to the filtration $\mathcal{F}_k$. In other words, if we denote

$$E_k(x,u):=\sum_{j=0}^{k}\epsilon_j(x,u), \tag{23}$$

then E_k is a martingale sequence with respect to $\mathcal{F}_k$, $\forall(x,u)\in\mathcal{X}\times U(x)$ and $\forall k\geq 0$.

The following proposition provides the uniform bounds of $\mathcal{D}_k$ and ϵ_k.

Proposition 1. *Suppose Assumption 5 holds. Consider MomentumQ as in Algorithm 1. Then the terms $\mathcal{D}_k\left[Q_k,Q_{k-1}\right]$ defined in* (20) *and ϵ_k in* (22) *are uniformly bounded for all $k\geq 0$. Specifically, $\exists\bar{D}>0$, s.t. $\|\mathcal{D}_k[Q_k,Q_{k-1}]\|\leq\bar{D},\|\epsilon_k\|\leq 2\bar{D},\forall k\geq 0$.*

The uniform bounds proved in Proposition 1 are critical in the derivation of the main theorem below.

Theorem 3. *Suppose Assumption 5 holds. Consider Algorithm 1 where $m\geq 1/\gamma$. Then, with probability at least $1-\delta$, the output of MomentumQ satisfies for $T>m$:*

$$\|Q^\star-Q_T\|\leq\frac{\tilde{h}V_{\max}+\bar{D}\sqrt{8(T-\lfloor m\rfloor-1)\log\frac{2n}{\delta}}}{T(1-\gamma)}, \tag{24}$$

where $\tilde{h}=2\gamma(m+\lfloor m\rfloor+2)+2$, $\bar{D}$ is specified in Proposition 1, and $\lfloor m\rfloor$ denotes the largest integer that does not exceed m.

Using the Borel–Cantelli lemma, we immediately have the following corollary.

Corollary 1. *Q_k converges to $Q^\star$ almost surely at a rate of at least $\tilde{\mathcal{O}}(\frac{\sqrt{(T-\lfloor m\rfloor-1)\log n}}{(1-\gamma)^2T})$.*

This rate is slightly better than $\tilde{\mathcal{O}}(\frac{\sqrt{\log n}}{(1-\gamma)^2\sqrt{T}})$ of SpeedyQ due to the presence of $m>1$. We note that the Generalized SpeedyQ [John et al., 2020] only slightly improves the constant multipliers over SpeedyQ. In addition, implementation of Generalized SpeedyQ requires a complicated sampling procedure, which added extra computational complexity.

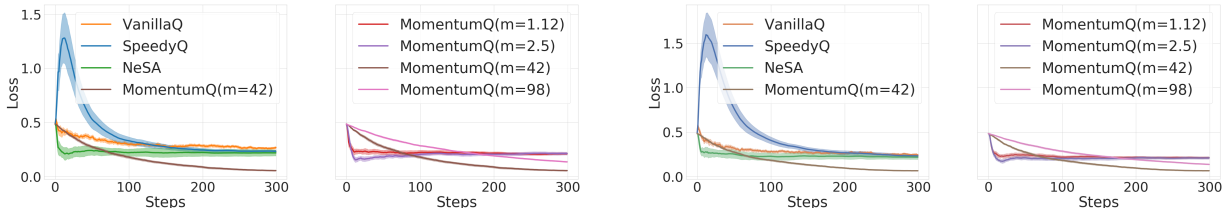
5 EXPERIMENTS

We evaluate the performance of the proposed MomentumQ scheme and compare it with other related Q-learning algorithms over a series of FrozenLake games (see the appendix for further specifications of the FrozonLake problem). We present the empirical results for both the tabular MomentumQ and MomentumQ with linear function approximation which are discussed above.

5.1 EXPERIMENTS ON TABULAR MOMENTUMQ

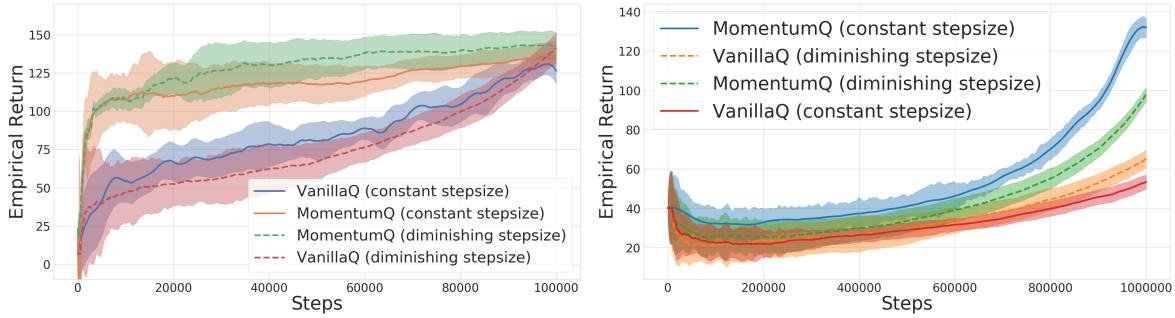
We compare MomentumQ with two other existing momentum-based Q-learning algorithms: SpeedyQ and the Nesterov stochastic approximation (NeSA) algorithm [Devraj et al., 2019]. In addition, we include comparisons with vanilla Q-learning to demonstrate the advantages of the proposed momentum techniques.

The experimental settings in this section are consistent with those of MomentumQ in Algorithm 1 and SpeedyQ in Azar et al. [2011, Algorithm 1]. Thus the numerical results should be able to give a convincing comparison between two algorithms. It is worth mentioning that the tabular MomentumQ has an additional hyperparameter m that can take a wide range of values (recall $m\geq 1/\gamma$). We experiment with several different m's. For relatively large m values (e.g., when $m>10$), the learning rates are shifted to step from $1/(m+1)$, that is, $\alpha_k=1/(m+k+1)$, for $k=0,1,2,....$ This is to avoid the large errors accumulated from initial iterations when $b_k<0$, which are reflected in the constants in (24). Note that this shift does not change the obtained



(a) FrozenLake-4 $\times$ 4 (b) FrozenLake-8 $\times$ 8

Figure 1: Comparing MomentumQ with NeSA, SpeedQ, and VanillaQ with tabular Q-function.



(a) i.i.d sampling (b) Markovian sampling

Figure 2: Comparison of MomentumQ with VanillaQ in the FrozenLake-128 $\times$ 128 task with various learning rate schemes and sampling strategies.

theoretical order of the convergence rate. We observe stable and consistent improvement in convergence and optimality across various tests when choosing large m values, which also aligns with our theoretical analysis.

Considering the randomness embedded in MDP of both FrozenLake games, we evaluate the performance of each algorithm with 20 different random seeds and then illustrate the average loss and standard deviation in Fig. 1a and Fig. 1b. For evaluation purpose, we have access to the true transition probability, and can find the ground truth optimal Q-function $Q^\star$ using dynamic programming. In both games, the loss at step k is then defined as $\|Q_k - Q^\star\|$. It can be seen from the plots that MomentumQ with a rather wide choices of m can all converge faster than vanilla Q-learning and SpeedyQ. It shows competitive performance against NeSA with smaller variance presented. Note that the high variance observed in the NeSA training aligns with the previous reported results from Devraj et al. [2019] under different tasks.

5.2 EXPERIMENTS WITH FUNCTION APPROXIMATION

We adopt the FrozenLake-128 $\times$ 128 as the benchmark task to evaluate the performance of MomentumQ with linear function approximation and compare it with the vanilla Q-learning (referred to as VanillaQ). Both algorithms are evaluated with different learning rate schemes (constant & diminishing stepsize), as well as different sampling strategies (i.i.d. and Markovian). We note that SpeedyQ and NeSA have been proposed in the literature only for the tabular setting and are thus not included here for comparison.

Note that the i.i.d. sampling is an ideal assumption and cannot be satisfied perfectly in practice. For our implementation, we perform i.i.d. sampling strategy in a similar fashion to the experience replay [Mnih et al., 2013] typically used for DQN training. A data buffer, referred to as the experience, is accumulated with data points collected across multiple training steps in the past. At each training step, the training data is then randomly uniformly sampled from the data buffer. In contrast, the Markovian sampling takes the training samples in an "on-policy" manner where the collected data points are fed in to the Q-learning process right after. At step k, the performance of the algorithm is evaluated through the total return of 150 rounds of trials. Similarly to the tabular setup, we execute each algorithm 20 times with different random seeds and illustrate the average return and standard deviation in Fig. 2a with i.i.d. sampling and Fig. 2b with Markovian sampling.

Overall, the MomentumQ algorithm has exhibited superior performance than the vanilla Q-learning. In particular, training with i.i.d. sampling is significantly faster than the Markovian sampling, which can be also expected from our theoretical results. Within the same sampling strategy, MomentumQ is also faster in convergence than the vanilla

Q-learning with the same learning rate scheme.

6 CONCLUSION

We proposed new momentum-based Q-learning algorithms for both the tabular and linear function approximation cases, which are respectively applicable to finite and continuous state-action spaces. We further characterized the convergence rate with empirical evaluation. The proposed algorithms accelerate the convergence in comparison to vanilla Q-learning on various challenging tasks under both tabular and parametric Q-learning settings.

Author Contributions

Bowen Weng, Huaqing Xiong, and Lin Zhao provided an equal contribution to this paper.

Acknowledgements

The work of H. Xiong and Y. Liang was supported in part by the U.S. National Science Foundation under the grants CCF-1909291 and CCF-1761506; The work of L. Zhao was supported in part by the National University of Singapore startup grant R-263-000-E60-133; This work was also supported in part by National Natural Science Foundation of China (Grant No. 62073159), and the Shenzhen Science and Technology Program (Grant No. JCYJ20200109141601708).

References

Mohammad Gheshlaghi Azar, Remi Munos, M Ghavamzadaeh, and Hilbert J Kappen. Speedy Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2411–2419, 2011.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

Dimitri P. Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*, volume 5. Athena Scientific, 1996.

Jalaj Bhandari, Daniel Russo, and Raghav Singal. A finite time analysis of temporal difference learning with linear function approximation. In *Conference on Learning Theory (COLT)*, 2018.

Vivek S Borkar and Sean P Meyn. The ode method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control and Optimization*, 38 (2):447–469, 2000.

Qi Cai, Zhuoran Yang, Jason D Lee, and Zhaoran Wang. Neural temporal-difference learning converges to global optima. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 11312–11322, 2019.

Guillermo A Castillo, Bowen Weng, Ayonga Hereid, Zheng Wang, and Wei Zhang. Reinforcement learning meets hybrid zero dynamics: A case study for rabbit. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 284–290, 2019.

Zaiwei Chen, Sheng Zhang, Thinh T. Doan, Siva Theja Maguluri, and John-Paul Clarke. Finite-time analysis of Q-learning with linear function approximation. *arXiv preprint arXiv:1905.11425*, 2019.

Adithya M Devraj and Sean Meyn. Zap Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2235–2244, 2017.

Adithya M. Devraj, Ana Bušić, and Sean Meyn. On matrix momentum stochastic approximation and applications to Q-learning. In *57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 749–756, 2019.

Simon S Du, Yuping Luo, Ruosong Wang, and Hanrui Zhang. Provably efficient Q-learning with function approximation via distribution shift error checking oracle. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8058–8068, 2019.

Eyal Even-Dar and Yishay Mansour. Learning rates for Q-learning. *Journal of Machine Learning Research*, 5 (Dec):1–25, 2003.

Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. A theoretical analysis of deep Q-learning. *arXiv preprint arXiv:1901.00137*, 2019.

Hado V Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2613–2621, 2010.

Tommi Jaakkola, Michael I Jordan, and Satinder P Singh. Convergence of stochastic iterative dynamic programming algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 703–710, 1994.

I. John, C. Kamanchi, and S. Bhatnagar. Generalized speedy q-learning. *IEEE Control Systems Letters*, 4(3):524–529, 2020.

Michael J Kearns and Satinder P Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 996–1002, 1999.

Harold Kushner and G George Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

Donghwan Lee and Niao He. A unified switching system perspective and ODE analysis of Q-learning algorithms. *arXiv preprint arXiv:1912.02270*, 2019.

Donghwan Lee and Niao He. Periodic Q-learning. *arXiv preprint arXiv:2002.09795*, 2020.

Gen Li, Yuting Wei, Yuejie Chi, Yuantao Gu, and Yuxin Chen. Sample complexity of asynchronous q-learning: Sharper analysis and variance reduction. *arXiv preprint arXiv:2006.03041*, 2020.

Francisco S Melo. Convergence of Q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep*, pages 1–4, 2001.

Francisco S Melo and M Isabel Ribeiro. Q-learning with linear function approximation. In *International Conference on Computational Learning Theory*, pages 308–322, 2007.

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

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Guannan Qu and Adam Wierman. Finite-time analysis of asynchronous stochastic approximation and Q-learning. *arXiv preprint arXiv:2002.00260*, 2020.

Devavrat Shah and Qiaomin Xie. Q-learning with nearest neighbors. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3111–3121, 2018.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017. ISSN 1476-4687.

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

Csaba Szepesvári. The asymptotic convergence-rate of Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1064–1070, 1998.

John N Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine learning*, 16(3):185–202, 1994.

Nino Vieillard, Bruno Scherrer, Olivier Pietquin, and Matthieu Geist. Momentum in reinforcement learning. *arXiv preprint arXiv:1910.09322*, 2019.

Martin J Wainwright. Stochastic approximation with cone-contractive operators: Sharp ℓ_∞-bounds for Q-learning. *arXiv preprint arXiv:1905.06265*, 2019a.

Martin J Wainwright. Variance-reduced Q-learning is minimax optimal. *arXiv preprint arXiv:1906.04697*, 2019b.

Christopher J.C.H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

Bowen Weng, Huaqing Xiong, Yingbin Liang, and Wei Zhang. Analysis of Q-learning with adaptation and momentum restart for gradient descent. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, pages 3051–3057, 2020a.

Wentao Weng, Harsh Gupta, Niao He, Lei Ying, and Srikant R. Provably-efficient double Q-learning. *arXiv preprint arXiv:arXiv:2007.05034*, 2020b.

Huaqing Xiong, Tengyu Xu, Yingbin Liang, and Wei Zhang. Non-asymptotic convergence of Adam-type reinforcement learning algorithms under markovian sampling. *arXiv preprint arXiv:2002.06286*, 2020a.

Huaqing Xiong, Lin Zhao, Yingbin Liang, and Wei Zhang. Finite-time analysis for double Q-learning. *arXiv preprint arXiv:2009.14257*, 2020b.

Pan Xu and Quanquan Gu. A finite-time analysis of Q-learning with neural network function approximation. *arXiv preprint arXiv:1912.04511*, 2019.

Lin Yang and Mengdi Wang. Sample-optimal parametric Q-learning using linearly additive features. In *International Conference on Machine Learning (ICML)*, pages 6995–7004, 2019.

Shaofeng Zou, Tengyu Xu, and Yingbin Liang. Finite-sample analysis for SARSA with linear function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8665–8675, 2019.