Analysis of Q-learning with Adaptation and Momentum Restart for Gradient Descent

Bowen Weng[∗1], **Huaqing Xiong**[∗1], **Yingbin Liang**[1] and **Wei Zhang**[†2]
[1]Electrical and Computer Engineering, The Ohio State University, Columbus, OH, USA.
[2]Mechanical and Energy Engineering, Southern University of Science and Technology, China.
{weng.172, xiong.309, liang.889}@osu.edu, zhangw3@sustech.edu.cn

Abstract

Existing convergence analyses of Q-learning mostly focus on the vanilla stochastic gradient descent (SGD) type of updates. Despite the Adaptive Moment Estimation (Adam) has been commonly used for practical Q-learning algorithms, there has not been any convergence guarantee provided for Q-learning with such type of updates. In this paper, we first characterize the convergence rate for Q-AMSGrad, which is the Q-learning algorithm with AMSGrad update (a commonly adopted alternative of Adam for theoretical analysis). To further improve the performance, we propose to incorporate the momentum restart scheme to Q-AMSGrad, resulting in the so-called Q-AMSGradR algorithm. The convergence rate of Q-AMSGradR is also established. Our experiments on a linear quadratic regulator problem show that the two proposed Q-learning algorithms outperform the vanilla Q-learning with SGD updates. The two algorithms also exhibit significantly better performance than the DQN learning method over a batch of Atari 2600 games.

1 Introduction

Q-learning [Watkins and Dayan, 1992], as one of the most important model-free reinforcement learning (RL) algorithms, has received considerable attention in recent years [Bertsekas and Tsitsiklis, 1996; Even-Dar and Mansour, 2003; Lu *et al.*, 2018]. The vanilla Q-learning algorithm runs a step of empirical Bellman operator update of the Q-function and a step of stochastic gradient descent (SGD) in an ***alternating*** fashion. The convergence guarantee of Q-learning has been studied for the tabular case in [Bertsekas, 1995; Even-Dar and Mansour, 2003], for the case with linear function approximation in [Bertsekas and Tsitsiklis, 1996; Zou *et al.*, 2019b; Chen *et al.*, 2019b], and also for neural network parameterization in [Xu and Gu, 2019].

However, all the existing theoretical analyses focus on Q-learning algorithms that take simple SGD iterations. Such theory is not applicable to practical Q-learning algorithms that implement the Adaptive Moment Estimation (Adam) type of updates. In this paper, we study the Q-learning algorithm with Adam-type updates in terms of its theoretical convergence and the performance in benchmark experiments. It is known in optimization that Adam does not always converge, and instead, and a slightly modified variant AMSGrad proposed in [Reddi *et al.*, 2018] has been widely accepted as an alternative to justify the theoretical performance of Adam-type methods. This motivates the first question that we aim to address.

- *Q1: Can we provide the convergence guarantee for Q-learning under AMSGrad updates (i.e., Q-AMSGrad)?*

In conventional optimization problems, *restart* has been incorporated into the gradient descent algorithm with momentum as a simple yet effective scheme to facilitate the acceleration performance [O'donoghue and Candes, 2015]. Hence, it is natural to incorporate such a restart technique into Q-learning and ask the following question about its performance.

- *Q2: Does Q-AMSGrad with momentum restart (i.e., Q-AMSGradR) still converge?*

As aforementioned, both Q-AMSGrad and Q-AMSGradR update alternatingly between one step of Bellman operator update of the Q-function and one step of adaptive momentum update. This is in contrast to the well-known deep Q-Network (DQN) learning [Mnih *et al.*, 2015], which runs in a nested-loop manner with the outer loop consisting of an one-step update of the Q-function and the inner loop consisting of *many* iterations of supervised learning to fit a *target Q-function*. It is conventionally known that taking *just one* gradient step toward the target Q-function results in the instability of original Q-learning update. Thus the simple alternating update manner mainly aroused interest in theory instead of practice. We are interested in whether the adaptive momentum updates can improve the stability of Q-learning without using the supervised learning process for target network fitting. Therefore, the third question we want to address is to compare these two types of Q-learning algorithms from the experimental perspective.

- *Q3: Do Q-AMSGrad and Q-AMSGradR perform competitively or even better than DQN in experiment?*

This paper addresses the above theoretical and experimental questions with affirmative answers.

1.1 Main Contributions

Theoretically, we show that with linear function approximation (which is almost the only structure that the current tools

[∗]Equal contribution
[†]Corresponding author

for analysis of Q-learning can handle), both Q-AMSGrad and Q-AMSGradR converge to the global optimal solution under standard assumptions for Q-learning. To the best of our knowledge, this is the first non-asymptotic convergence guarantee on Q-learning that incorporates Adam-type update. Furthermore, a slight adaptation of our proof provides the convergence rate for the AMSGrad for conventional strongly convex optimization which has not been studied before and can be of independent interest.

Experimentally, we demonstrate that the practical versions of Q-AMSGrad and Q-AMSGradR (referred to as Q-Adam and Q-AdamR) exhibit appealing experimental performance. In a batch of 23 Atari 2600 games, our experiments show that both Q-Adam and Q-AdamR outperform DQN by 50% on average. Furthermore, Q-AdamR effectively reduces the performance variance and achieves a much more stable learning process. In our experiments for the linear quadratic regulator (LQR) problems, Q-AdamR converges even faster than the model-based value iteration (VI) solution. This is a rather surprising result given that the model-based VI has been treated as the performance upper bound for the Q-learning (including DQN) algorithms with target update [Lewis and Vrabie, 2009; Yang *et al.*, 2019].

Detailed proofs and more experimental results will be available in the extended version of this paper on arXiv.org after the official publication of IJCAI Proceedings.

1.2 Related Work

We briefly review the related work as follows.

Theoretical analysis of Q-learning. Since proposed in [Watkins and Dayan, 1992], the convergence of Q-learning has been extensively studied, particularly for the case with linear function approximation such as [Bertsekas and Tsitsiklis, 1996; Zou *et al.*, 2019b; Chen *et al.*, 2019b; Du *et al.*, 2019], to name a few, and more recently for the case with neural networks in [Xu and Gu, 2019], where the analysis exploits the approximate linear structure of neural networks in the overparamterized regime. All these existing analysis of Q-learning considers the vanilla SGD update, whereas our study is the first to analyze the more involved case with Adam-type updates.

Convergence analysis of Adam-type algorithms in conventional optimization. Adam was proposed in [Kingma and Ba, 2015] for speeding up the training of deep neural networks, and the regret bounds were characterized for Adam/AMSGrad in [Kingma and Ba, 2015; Reddi *et al.*, 2018; Tran and others, 2019] for online convex optimization. Recently, convergence analysis of Adam/AMSGrad was provided for nonconvex optimization in [Zou *et al.*, 2019a; Zhou *et al.*, 2018; Chen *et al.*, 2019a] and policy gradient [Xiong *et al.*, 2020], in which such Adam-type algorithms were guaranteed to converge to a stationary point. To the best of our knowledge, our study provides the first convergence analysis of the Adam-type algorithms for Q-learning.

Empirical performance of Q-learning. DQN learning and its improved variants of dueling network structure [Wang *et al.*, 2016], double Q-learning [Van Hasselt *et al.*, 2016] and variance exploration and sampling schemes [Schaul *et al.*, 2015] have achieved significant success due to their superb performance in practice. In contrast to such nested-loop algorithms (which involves the fitting of a target Q-function periodically), the Q-learning algorithms that strictly follow the alternating updates are much less explored in practice. [Mnih *et al.*, 2016] proposed the asynchronous alternating Q-learning with competitive performance against DQN. However, the algorithm still relies on a slowly moving target network similar to DQN, and the multi-thread learning also complicates the computational setup. [Lu *et al.*, 2018] studied the problem of value overestimation and proposed the non-delusional Q-learning algorithm that employs the so-called pre-conditioned Q-networks, which is also computationally complex. [Knight and Lerner, 2018] proposed a natural gradient propagation to improve the performance, where the gradient implementation is complex. Our experiments in this paper demonstrate that simple alternating Q-learning algorithms Q-AMSGrad and Q-AMSGradR without the complex designs as in [Mnih *et al.*, 2016; Lu *et al.*, 2018; Knight and Lerner, 2018] have competitive and sometimes better performance than DQN.

Notations We use $\|x\| := \|x\|_2$ to denote the ℓ_2 norm of a vector x, and use $\|x\|_\infty$ to denote the infinity norm. When x, y are both vectors, $x/y, xy, x^2, \sqrt{x}$ are all calculated in the element-wise manner, which will be used in the update of Adam and AMSGrad. We denote $[n] = 1, 2, \dots, n$, and $\lfloor x \rfloor \in \mathbb{Z}$ as the integer such that $\lfloor x \rfloor \le x < \lfloor x \rfloor + 1$.

2 Preliminaries

We consider a Markov decision process with a considerably large or continuous state space $\mathcal{S} \subset \mathbb{R}^M$ and action space $\mathcal{A} \subset \mathbb{R}^N$, a non-negative bounded reward function $R : \mathcal{S} \times \mathcal{A} \to [0, R_{\max}]$, and a transition kernel $P(s'|s,a)$ that indicates the probability from a state-action pair (s,a) to a state s'. We define $U(s) \subset \mathcal{A}$ as the admissible set of actions at state s, and $\pi : \mathcal{S} \to \mathcal{A}$ as a feasible stationary policy. We seek to solve a discrete-time sequential decision problem as follows:

$$\begin{aligned} &\underset{\pi}{\text{maximize}} \;\; J_\pi(s_0) = \mathbb{E}_P\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\right], \\ &\text{subject to} \;\; s_{t+1} \sim P(\cdot|s_t, a_t), \end{aligned} \tag{1}$$

where $\gamma \in (0,1)$ is the discount factor. Let $J^\star(s) := J_{\pi^\star}(s)$ be the optimal value function when applying the optimal policy $\pi^\star$. The corresponding optimal Q-function can be defined as

$$Q^\star(s,a) := R(s,a) + \gamma \mathbb{E}_P J^\star(s'), \tag{2}$$

where $s' \sim P(\cdot|s,a)$ and we use the same notation hereafter when no confusion arises. In other words, $Q^\star(s,a)$ represents the reward of an agent who starts from state s and takes action a at the first step and then follows the optimal policy $\pi^\star$ thereafter.

2.1 Q-learning Algorithm

This paper focuses on the Q-learning algorithm that uses a parametric function $\hat{Q}(s,a;\theta)$ to approximate the Q-function with a parameter θ having finite and relatively small dimensions. The update rule of Q-learning is given by

$$T\hat{Q}(s,a;\theta_t) = R(s,a) + \gamma \max_{a' \in U(s')} \hat{Q}(s',a';\theta_t); \tag{3}$$

$$\theta_{t+1} = \theta_t - \alpha_t \left(\hat{Q}_t(s,a;\theta_t) - T\hat{Q}(s,a;\theta_t)\right)\hat{g}_t, \tag{4}$$

$$\hat{g}_t := \hat{g}(\theta_t;s,a) = \frac{\partial}{\partial \theta_t}\hat{Q}_t(s,a;\theta_t), \tag{5}$$

where α_t is the step size at time t. It is clear that Q-learning performs the update by taking one step of temporal target update and one step of parameter learning in an alternating fashion.

2.2 Linear Function Approximation

Like most of the related work, we focus on the convergence analysis under the linear function approximation. A linear approximation of the Q-function $\hat{Q}(s,a;\theta)$ can be written as

$$\hat{Q}(s,a;\theta) = \phi(s,a)^T\theta, \tag{6}$$

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.

Remark 1. *We note that recent work [Xu and Gu, 2019] established the convergence rate of Q-learning with neural network approximation, which exploits the approximate linear structure of the neural network in the overparameterized regime. Thus, our analysis under the linear function approximation can be generalized to the function class of overparameterized neural networks by applying the techniques developed in recent work [Xu and Gu, 2019].*

3 Convergence Analysis of Q-AMSGrad

In this section, we characterize the convergence guarantee for Q-learning under Adam-type updates.

3.1 Q-AMSGrad Algorithm

Although Adam has obtained great success as an optimizer in deep learning, it is well known that Adam by nature is non-convergent even for simple convex loss functions [Reddi *et al.*, 2018]. Instead, a slightly modified version called AMSGrad [Reddi *et al.*, 2018] is widely used to study the convergence property of Adam-type algorithms in conventional optimization. Here, we apply the update rule of AMSGrad to the Q-learning algorithm and refer to such an algorithm as Q-AMSGrad. Algorithm 1 describes Q-AMSGrad in detail.

More specifically, the iterations of Q-AMSGrad evolve by updating the exponentially decaying average of historical gradients (m_t) and squared historical gradients (v_t). The hyperparameters β_1, β_2 are used to exponentially decrease the rate of the moving averages. The difference between AMSGrad and Adam lies in the fact that AMSGrad makes the sequence $\hat{v}_{t,i}$ increasing along the time step t for each entry $i \in [d]$, whereas Adam does not guarantee such a property.

Algorithm 1 Q-AMSGrad

```
1:  Input: $\alpha, \lambda, \theta_1, \beta_1, \beta_2, m_0 = 0, \hat{v}_0 = 0$.
2:  for $t = 1, 2, \ldots, T$ do
3:      $\alpha_t = \frac{\alpha}{\sqrt{t}}, \beta_{1t} = \beta_1\lambda^t$
4:      Observe data $(s_t, a_t, s_{t+1})$ from policy $\pi$ and transition probability $P$
5:      $b_t = R(s_t,a_t) + \gamma \max_{a'} \phi^T(s_{t+1},a')\theta_t$
6:      $g_t = \left(\phi^T(s_t,a_t)\theta_t - b_t\right)\phi(s_t,a_t)$
7:      $m_t = (1-\beta_{1t})m_{t-1} + \beta_{1t}g_t$
8:      $v_t = (1-\beta_2)\hat{v}_{t-1} + \beta_2 g_t^2$
9:      $\hat{v}_t = \max(\hat{v}_{t-1}, v_t),\ \hat{V}_t = diag(\hat{v}_1, \ldots, \hat{v}_d)$
10:     $\theta_{t+1} = \Pi_{\mathcal{D},\hat{V}_t^{1/4}}(\theta_t - \alpha_t\hat{V}_t^{-\frac{1}{2}}m_t)$
        where $\Pi_{\mathcal{D},\hat{V}_t^{1/4}}(\theta') = \min_{\theta\in\mathcal{D}} \left\|\hat{V}_t^{1/4}(\theta' - \theta)\right\|$.
11: end for
12: Output: $\frac{1}{T}\sum_{t=1}^{T}\theta_t$
```

3.2 Convergence Result

Before stating the main theorem, we first introduce some technical assumptions and lemmas for our analysis.

Assumption 1. *For any state-action pair $(s,a) \in \mathcal{S} \times \mathcal{A}$, the kernel function ϕ is uniformly bounded and we have*

$$\|\phi(s,a)\| \le 1, \quad \forall(s,a). \tag{7}$$

This assumption is mild since we can normalize the kernel function if the kernel function is uniformly bounded. It is widely applied in the literature to simplify the analysis of RL algorithms with linear function approximation [Bhandari *et al.*, 2018; Chen *et al.*, 2019b].

Assumption 2. *[Chen* et al.*, 2019b, Lemma 6.7] At each iteration t, the noisy gradient is unbiased, i.e. $g_t = \bar{g}_t + \xi_t$ with $\mathbb{E}\xi_t = 0$ where $\bar{g}_t = \mathbb{E}[g_t]$. The equation $\bar{g}(\theta) = 0$ has a unique solution $\theta^\star$, and there exists a $c > 0$, such that for any $\theta \in \mathbb{R}^d$ we have*

$$(\theta - \theta^\star)^T\bar{g}(\theta) \ge c\left\|\theta - \theta^\star\right\|^2. \tag{8}$$

Assumption 2 has been proved as a key technical lemma in [Chen *et al.*, 2019b] under certain assumptions, which appears to be the weakest among the existing studies for establishing the convergence guarantee for Q-learning with linear function approximation. It is the standard assumption in the related literature to analyze the convergence Q-learning with linear function approximation [Zou *et al.*, 2019b; Chen *et al.*, 2019b; Xu and Gu, 2019].

Assumption 3. *The domain $\mathcal{D} \subset \mathbb{R}^d$ of approximation parameters is a ball originating at $\theta = 0$ with bounded diameter containing $\theta^\star$. That is, there exists D_∞, such that $\|\theta_m - \theta_n\| < D_\infty, \forall \theta_m, \theta_n \in \mathcal{D}$, and $\theta^\star \in \mathcal{D}$.*

This Assumption can be easily satisfied when we apply a projected algorithm, and is standard in the theoretical analysis of Adam-type algorithms [Chen *et al.*, 2019a; Zhou *et al.*, 2018].

Based on the above assumptions, we can immediately obtain the bounded property of the gradient, which is stated in the following lemma.

Lemma 1. *Under Assumptions 1 and 3, at each iteration t, the gradient estimator g_t in Q-AMSGrad is uniformly bounded. That is,*

$$\|g_t\|_\infty \le \|g_t\| \le R_{\max} + (1+\gamma)D_\infty, \quad \forall t. \tag{9}$$

In addition, we denote $G_\infty = R_{\max} + (1+\gamma)D_\infty$ and let $\{m_t, \hat{v}_t\}$ for $t = 1, 2, \dots$ be sequences generated by Algorithm 2. Then we have

$$\|\bar{g}_t\| \le G_\infty, \|m_t\| \le G_\infty, \|\hat{v}_t\| \le G_\infty^2.$$

We next provide the non-asymptotic convergence of Q-AMSGrad in the following theorem.

Theorem 1. *(Convergence of Q-AMSGrad) Suppose $\alpha_t = \frac{\alpha}{\sqrt{t}}, \beta_{1t} = \beta_1\lambda^t$ and $\delta = \beta_1/\beta_2$ with $\delta, \lambda \in (0,1)$ for $t = 1, 2, \dots$ in Algorithm 1. Given Assumptions 1 $\sim$ 3, the output of Q-AMSGrad satisfies:*

$$\mathbb{E}\|\theta_{out} - \theta^\star\| \le \frac{B_1}{T} + \frac{B_2}{\sqrt{T}} + \frac{B_3\sqrt{1+\log T}}{T}\sum_{i=1}^{d}\mathbb{E}\|g_{1:T,i}\|, \tag{10}$$

where $B_1 = \frac{G_\infty D_\infty^2}{2\alpha_2 c(1-\beta_1)} + \frac{\beta_1 G_\infty D_\infty^2}{2\alpha c(1-\beta_1)(1-\lambda)^2} + \|\theta_1 - \theta^\star\|^2$, $B_2 = \frac{dG_\infty D_\infty^2}{2\alpha c(1-\beta_1)}$, and $B_3 = \frac{\alpha(1+\beta_1)}{2c(1-\beta_1)^2(1-\delta)\sqrt{1-\beta_2}}$.

In Theorem 1, B_1, B_2, B_3 in the bound in Equation (10) are constants and independent of time. Therefore, under the choice of the stepsize and hyper-parameters in Algorithm 1, Q-AMSGrad achieves a convergence rate of $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ when $\sum_{i=1}^{d}\|g_{1:T,i}\| << \sqrt{T}$ [Reddi *et al.*, 2018].

Remark 2. *Our proof of convergence here has two major differences from that for AMSGrad in [Reddi* et al.*, 2018] in conventional optimization: (a) The two algorithms are quite different. Q-AMSGrad is a Q-learning algorithm alternatively finding the best policy with a moving target, whereas AMSGrad is an optimizer for conventional optimization and does not have alternating nature. (b) Our analysis is on the convergence rate whereas [Reddi* et al.*, 2018] provides regret bound. In fact, a slight modification of our proof also provides the convergence rate of AMSGrad for conventional strongly convex optimization, which can be of independent interest. Moreover, our proof avoids the theoretical error in the proof in [Reddi* et al.*, 2018] as pointed out by [Tran and others, 2019].*

4 Convergence Analysis of Q-AMSGradR

In this section, we propose to incorporate a momentum restart technique to Q-AMSGrad in order to improve its performance. We first introduce the algorithm and then provide the convergence analysis for such an algorithm. We demonstrate its desired experimental performance in Section 5.

4.1 Q-AMSGrad Algorithm with Momentum Restart

We introduce the restart technique to Q-AMSGrad and propose Q-AMSGradR as shown in Algorithm 2. Q-AMSGradR applies the same update rule as Algorithm 1, but periodically resets $m_t, \hat{v}_t$ with a period of r, i.e., $m_t = 0, \hat{v}_t = 0, \forall t = kr, k = 1, 2, \cdots$. We explain such an idea further as follows.

Algorithm 2 Q-AMSGradR

```
 1: Input: $\alpha, \lambda, \theta_1, \beta_1, \beta_2, m_0 = 0, \hat{v}_0 = 0$.
 2: for $t = 1, 2, \dots, T$ do
 3:     if $\mathrm{mod}(t, r) = 0$ then
 4:         $m_t = 0, \hat{v}_t = 0$
 5:     end if
 6:     $\alpha_t = \frac{\alpha}{\sqrt{t}}, \beta_{1t} = \beta_1\lambda^t$
 7:     Observe data $(s_t, a_t, s_{t+1})$ from policy $\pi$ and transition probability $P$
 8:     $b_t = R(s_t, a_t) + \gamma \max_{a'} \phi^T(s_{t+1}, a')\theta_t$
 9:     $g_t = \left(\phi^T(s_t, a_t)\theta_t - b_t\right)\phi(s_t, a_t)$
10:     $m_t = (1-\beta_{1t})m_{t-1} + \beta_{1t}g_t$
11:     $v_t = (1-\beta_2)\hat{v}_{t-1} + \beta_2 g_t^2$
12:     $\hat{v}_t = \max(\hat{v}_{t-1}, v_t),\ \hat{V}_t = diag(\hat{v}_1, \dots, \hat{v}_d)$
13:     $\theta_{t+1} = \Pi_{\mathcal{D}, \hat{V}_t^{1/4}}\left(\theta_t - \alpha_t \hat{V}_t^{-\frac{1}{2}} m_t\right)$
14: end for
15: Output: $\frac{1}{T}\sum_{t=1}^{T}\theta_t$
```

Traditional momentum-based algorithms largely depend on the historical gradient direction. When part of the historical information is incorrect, the estimation error tends to accumulate. The restart technique can be employed to deal with such an issue. One way to restart the momentum-based methods is to initialize the momentum at some restart iteration. That is, at restart iteration r, we reset m_r, v_r, i.e., $m_r = 0, v_r = 0$, which yields $\theta_{r+1} = \theta_r$. It is an intuitive implementation technique to adjust the trajectory from time to time, and can usually help mitigate the aforementioned problem while keeping fast convergence property. For the implementation, we execute the restart periodically with a period r. It turns out that the restart technique can significantly improve the numerical performance, which can be seen in Section 5.

4.2 Convergence Result

In the following theorem, we provide the non-asymptotic convergence for Q-AMSGradR.

Theorem 2. *(Convergence of Q-AMSGradR) Suppose $\alpha_t = \frac{\alpha}{\sqrt{t}}, \beta_{1t} = \beta_1\lambda^t$ and $\delta = \beta_1/\beta_2$ with $\delta, \lambda \in (0,1)$ for $t = 1, 2, \dots$ in Algorithm 2. Given Assumptions 1 $\sim$ 3, the output of Q-AMSGradR satisfies:*

$$\begin{aligned}\mathbb{E}\|\theta_{out} - \theta^\star\| &\le \frac{B_1}{T} + \frac{B_2\sqrt{1+\log T}}{T}\sum_{i=1}^{d}\mathbb{E}\|g_{1:T,i}\| \\ &+ \frac{B_3}{T}\left(\sqrt{T} + \sum_{k=1}^{\lfloor T/r \rfloor}\sqrt{kr-1}\right) \\ &+ \frac{1}{T}\sum_{k=0}^{\lfloor T/r \rfloor}\left(\frac{G_\infty D_\infty^2}{\alpha}\sqrt{kr+2} + B_4\mathbb{E}\|\theta_{kr} - \theta^\star\|^2\right),\end{aligned} \tag{11}$$

where $B_1 = \frac{\beta_1 D_\infty^2 G_\infty}{2\alpha c(1-\beta_1)(1-\lambda)^2}$, $B_2 = \frac{\alpha(1+\beta_1)}{2c(1-\beta_1)^2(1-\delta)\sqrt{1-\beta_2}}$, $B_3 = \frac{dG_\infty D_\infty^2}{2\alpha c(1-\beta_1)}$, *and* $B_4 = 4c(1-\beta_1)$.

Theorem 2 indicates that for Q-AMSGradR to enjoy a convergence rate of $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$, the restart period r needs to be sufficiently large and $\sum_{i=1}^{d} \|g_{1:T,i}\| << \sqrt{T}$. In practice as demonstrated by the experiments in Section 5, Q-AMSGradR typically performs well, not necessarily under the theoretical conditions.

5 Experimental Performance

In this section, we empirically evaluate the Q-learning algorithms studied in this paper. We first study the linear quadratic regulator (LQR) problem, which serves as a direct numerical demonstration of the convergence analysis under linear function approximation. We then use the Atari 2600 games [Brockman *et al.*, 2016], a classic benchmark for DQN evaluations, to demonstrate the effectiveness of the Q-learning algorithms for complicated tasks. Note that our experiments test the performance of Q-Adam and Q-AdamR, which serve as practical versions of Q-AMSGrad and Q-AMSGradR by adopting Adam for Q-learning and keeping all major properties of Q-AMSGrad and Q-AMSGradR including the alternating update between Q-function update and parameter fitting.

Our main focus here lies in: (a) comparison between vanilla Q-Adam and that with momemtum restart (Q-AdamR), (b) comparison between Q-Adam/Q-AdamR and vanilla Q-learning (through LQR), and (c) comparison between Q-Adam/Q-AdamR and DQN (through Atari 2600 games). Both of our experiments show that Q-AdamR outperforms Q-Adam, vanilla Q-learning and DQN in terms of convergence speed and variance reduction. Compared with DQN in the empirical experiments of Atari games, under the same hyper-parameter settings, Q-Adam and Q-AdamR improve the performance of DQN by 50% on average.

5.1 DQN Algorithm

As DQN is also included in this work for performance comparison. We recall the update of DQN in the following as reference. Differently from the vanilla Q-learning, DQN updates the parameters in a nested loop. Within the t-th inner loop, DQN first obtains the target Q-function as in Equation (12), and then uses a neural network to fit the target Q-function by running Y steps of a certain optimization algorithm as Equation (13). The update rule of DQN is given as follows.

$$T\hat{Q}(s,a;\theta_t^0) = R(s,a) + \gamma \max_{a' \in U(s')} \hat{Q}(s',a';\theta_t^0), \quad (12)$$

$$\theta_t^Y = Optimizer(\theta_t^0, T\hat{Q}(s,a;\theta_t^0)), \quad (13)$$

where $Optimizer$ can be SGD or Adam for example, and Equation (13) is thus a supervised learning process with $T\hat{Q}(s,a;\theta_t^0))$ as the "supervisor". At the t-th outer loop, DQN performs the so-called target update as

$$\theta_{t+1}^0 = (1-\tau)\theta_t^0 + \tau\theta_t^Y. \quad (14)$$

In practice, when one of the momentum-based optimizers is adopted for Equation (13), such as Adam, it is only initialized once at the beginning of the first inner loop. The historical gradient terms then accumulate throughout multiple inner loops with different targets.

Step size	$\tilde{\tau}$	Adam β_1	Adam β_2
0.0001	0.01	0.9	0.999
Restart period r	Stop criterion		γ
100	$\|K_i - K^\star\|_2 \le 10^{-4}$		1

Table 1: Hyper-parameters for LQR experiments.

Generally speaking, the difference between Q-Adam/Q-AdamR and DQN mainly lies in Q-Adam/Q-AdamR takes one-step Q-function update and one-step model parameter fitting alternatively, whereas DQN takes one-step Q-function update followed by a sufficient large number of steps for model parameter fitting (towards a target Q-function).

5.2 Linear Quadratic Regulator

We numerically validate the performance of Q-Adam and Q-AdamR through an infinite-horizon discrete-time LQR problem. A typical model-based solution (with known dynamics), known as the discrete-time algebraic Riccati equation (DARE), is adopted to derive the optimal policy $u_t^\star = -K^\star x_t$. The performance of the learning algorithm is then evaluated at each step of iterate t with the Euclidean norm $\|K_t - K^\star\|$. Given the problem nature of LQR, we also re-scale the loss term of $L(\theta_t) := \hat{Q}_t(s,a;\theta_t) - T\hat{Q}(s,a;\theta_t)$ in Equation (4) as $\tilde{L}(\theta_t) = \tilde{\tau}^2 L(\theta_t)$ with some scaling factor $\tilde{\tau} \in (0,1]$, which is beneficial for stabilizing the learning process. The performance result for each method is averaged over 10 trials with different random seeds. All algorithms share the same set of random seeds and are initialized with the same θ_0. The hyper-parameters of the learning settings are also consistent and further details are shown in Table 1. Note that for all the implementations, we also adopt the double Q-update [Van Hasselt *et al.*, 2016] to help prevent over-estimations of the Q-value. The performance results are provided in Figure 1. Here we highlight the main observations from the LQR experiments.

Q-AdamR outperforms DARE. In ideal cases where data sampling perfectly emulates the system dynamics and the target is accurately learned in each inner loop, DARE for LQR would become equivalent to the DQN-like update if the neural network is replaced with a parameterized linear function. In practice, such ideal conditions are difficult to satisfy, and hence the actual Q-learning with target update is usually far slower (in terms of the number of steps of target updates) than DARE. Note that Q-AdamR performs significantly well and even converges faster than DARE, and thus implies it is faster than the most well-performing Q-learning with target update.

Q-AdamR outperforms Q-Adam. Overall, under the same batch sampling scheme and restart period, Q-AdamR achieves a faster convergence and smaller variance than Q-Adam.

5.3 Atari Games

We apply the Q-Adam and Q-AdamR algorithms to the more challenging tasks of deep convolutional neural network playing a group of Atari 2600 games. The particular DQN we train

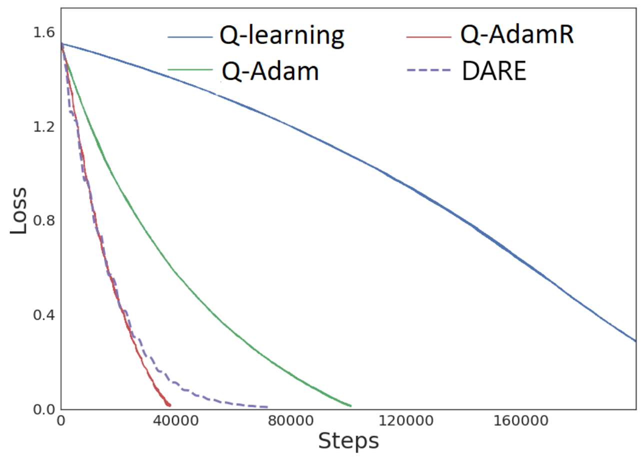

Figure 1: LQR experiments with performance evaluated in terms of policy loss $\|K_t - K^\star\|_2$.

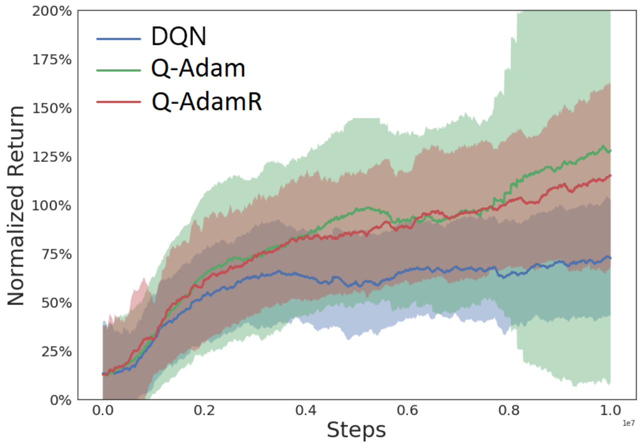

Figure 2: Atari game experiment with performance normalized and averaged over 23 games.

to compare against adopts the dueling network structure [Wang *et al.*, 2016], double Q-learning setup [Van Hasselt *et al.*, 2016], ϵ-greedy exploration and experience replay [Mnih *et al.*, 2015]. Adam is also adopted, without momentum restart, as the optimizer for the inner-loop supervised learning process. Q-Adam and Q-AdamR are implemented using the identical setup of network construction, exploration and sampling strategies.

We test all three algorithms with a batch of 23 Atari games. The choice of 10 million steps of iteration is a common setup for benchmark experiments with Atari games. Although this does not guarantee the best performance in comparison with more time-consuming training with 50 million steps or more, it is sufficient to illustrate different performances among the selected methods. The software infrastructure is based on the baseline implementation of OpenAI. Selections of the hyper-parameters are listed in Table 2. We summarize the results in Figure 2. The overall performance is illustrated by first normalizing the return of each method with respect to the results obtained from DQN, and then averaging the performance of

Step size	Scale factor $\tilde{\tau}$	Adam β_1	Adam β_2
0.0001	0.0001	0.9	0.999
r	Buffer size	γ	Batch size B
10^4	10^5	0.99	32
Total training steps K		Target update frequency	
10^7		10^4	

Table 2: Hyper-parameters for Atari games experiments of DQN, Q-Adam and Q-AdamR.

all 23 games to obtain the mean return and standard deviation. Considering we use a smaller buffer size than common practice, DQN is not consistently showing improved return over all tested games. Therefore, the self-normalized average return of DQN in Figure 2 is not strictly increasing from 0 to 100%.

Overall, both Q-Adam and Q-AdamR achieve significant improvement in comparison with the DQN results. In particular, AltQ-Adam increases the performance by over 100% in some of the tasks including Asterix, BeamRider, Enduro, Gopher, etc. However, it also illustrates certain instability with complete failure on Amidar and Assault. This is also capture by the higher variance illustrated on Figure 2. Periodic restart (Q-AdamR) resolves this issue efficiently with an on-par performance on average and far smaller variance. In terms of the maximum average return, Q-Adam and Q-AdamR perform no worse then DQN on 17 and 20 games respectively out of the 23 games being evaluated. Furthermore, if we consider having a final score that is smaller or equal to the start score as a learning failure, DQN fails the learning in 5 out of 23 games (Asteroids, DoubleDunk, Gravitar, Pitfall, Tennis), Q-Adam fails in 3 out of 23 games (Amidar, Assault, Asteroids) and Q-AdamR does not fail in any of the tasks. That is, Q-AdamR not only reduces the variance, but also provides a more consistent performance across the task domain. This implies that momentum restart effectively corrects the accumulated error and stabilizes the training process.

6 Conclusion

We study two Q-learning algorithms with Adam-type updates, and demonstrate their superior performance over the vanilla Q-learning and DQN algorithms through a linear quadratic regulator problem and a batch of 23 Atari 2600 games.

It is of considerable future interest to further investigate the potential of the restart scheme. One possible direction is to develop an adaptive restart mechanism with changing period determined by an appropriately defined signal of restart. This will potentially relieve the effort in hyper-parameter tuning of finding a good fixed period.

Acknowledgements

The work was supported in part by the U.S. National Science Foundation under Grants CCF-1761506, ECCS-1818904, CCF-1909291 and CCF-1900145, and the startup funding of the Southern University of Science and Technology (SUSTech), China.

References

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

[Bertsekas, 1995] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. 1995.

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

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[Chen *et al.*, 2019a] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of Adam-type algorithms for non-convex optimization. In *International Conference on Learning Representations*, 2019.

[Chen *et al.*, 2019b] 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.

[Du *et al.*, 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*, pages 8058–8068, 2019.

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

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[Knight and Lerner, 2018] Ethan Knight and Osher Lerner. Natural gradient deep Q-learning. *arXiv preprint arXiv:1803.07482*, 2018.

[Lewis and Vrabie, 2009] F. L. Lewis and D. Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9(3):32–50, Third 2009.

[Lu *et al.*, 2018] Tyler Lu, Dale Schuurmans, and Craig Boutilier. Non-delusional Q-learning and value-iteration. In *Proceedings of the Thirty-second Conference on Neural Information Processing Systems (NeurIPS-18)*, pages 9971–9981, Montreal, QC, 2018.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[O'donoghue and Candes, 2015] Brendan O'donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732, 2015.

[Reddi *et al.*, 2018] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *International Conference on Learning Representations*, 2018.

[Schaul *et al.*, 2015] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[Tran and others, 2019] Phuong Thi Tran et al. On the convergence proof of AMSGrad and a new version. *IEEE Access*, 7:61706–61716, 2019.

[Van Hasselt *et al.*, 2016] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[Wang *et al.*, 2016] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1995–2003, 2016.

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

[Xiong *et al.*, 2020] 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*, 2020.

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

[Yang *et al.*, 2019] Zhuora Yang, Yuchen Xie, and Zhaoran Wang. A theoretical analysis of deep Q-learning. *arXiv preprint arXiv:1901.00137*, 2019.

[Zhou *et al.*, 2018] Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *arXiv preprint arXiv:1808.05671*, 2018.

[Zou *et al.*, 2019a] Fangyu Zou, Li Shen, Zequn Jie, Weizhong Zhang, and Wei Liu. A sufficient condition for convergences of adam and rmsprop. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11127–11135, 2019.

[Zou *et al.*, 2019b] Shaofeng Zou, Tengyu Xu, and Yingbin Liang. Finite-sample analysis for sarsa with linear function approximation. In *Advances in Neural Information Processing Systems*, pages 8665–8675, 2019.