Project Report: Computational and Statistical Challenges of DeepQ and Reinforcement Learning

Saptarashmi Bandyopadhyay
Department of Computer Science and Engineering
Pennsylvania State University, University Park,PA 16802
szb754@psu.edu

Bodhisatwa Chatterjee
Department of Computer Science and Engineering
Pennsylvania State University, University Park,PA 16802
bxc583@psu.edu

Abstract

In this project we have investigated the computational and statistical challenges of Deep Q Learning namely convergence, sample efficiency and over-estimation. We have discussed about the asymptotic convergence rate of Q Learning and have looked at how it can be improvised in polynomial time by the selection of appropriate learning rate. We have also discussed about the statistical convergence rate of Deep Q Learning and how it can be expressed as the sum of a statistical and an algorithmic error rate. The statistical error indicates the bias and variance of the function approximation of the Q-Value function by the deep neural networks, while the algorithmic error becomes zero for an infinite action space. Deep Q Networks also suffer from the challenge of over-estimation, whose lower bounds has been discussed in this report. Finally, we have seen that when Double Q-Learning is incorporated with large scale function approximation, the over-estimation problem is reduced.

1 Introduction

Ever since its inception in the mid-1960s, Reinforcement Learning has been a fundamental technique for understanding the problem of an agent interacting with some unknown environment. Its applications are in a wide domain of fields, ranging from *game theory, operations research, information theory*,etc. Deep Reinforcement Learning is now being widely used in *robotics* and other Artificial Intelligence applications.

In this paper, we have focused on the computational and statistical challenges of Reinforcement Learning and its variants, namely Q-Learning and Deep Q-Learning. In the first section, we have provided a background on Reinforcement Learning and Q-Learning and also pointed out a few fundamental challenges of Reinforcement Learning. In the successive sections, we have studied the asymptotic convergence rates of Q-Learning and contrasted that with the convergence rates of Reinforcement Learning defined by value functions. Then we considered how a simple trick in the adjustment of learning rate can cause the Q-Learning to converge in polynomial time. After that, the challenges behind the theoretical analysis of Deep Q Learning has been investigated. The convergence rate of Deep Q Learning has been provided with a proof sketch. After presenting this result, we have understood that Deep Q Learning is not sample-efficient and hence we have the motivation for alternatives to this sampling problem. Thereafter, the lower bound of over-estimation

Project Progress Report for Large Scale Machine Learning (CSE 597 Spring 2019), Pennsylvania State University, University Park, PA, USA.

of Deep Q Learning has been studied which is another major computational and statistical challenge. Double Q Learning has been proposed to solve the problem by reducing the lower bound on the over-estimation error to zero.

The whole report can be summarized in the following unified manner:

- Introduction to Reinforcement Learning(Background)
- Use of Value Function for achieving optimality (Alternative Approach to Q-Learning)
- Convergence Analysis of Q-Learning
- Deep-Q Network and Challenges for its theoretical analysis
- Convergence Analysis and Sample Efficiency of Deep Q Learning
- Over-Estimation Challenges of Deep Q Learning

2 Background Material

In this section, we will provide the background material of Reinforcement Learning required to grasp the challenges presented in this report. We will try to explain the process of RL in a simple and intuitive approach, focussing on its challenges at the same time. All the references to this section is from Li (2019)

2.1 Reinforcement Learning

In simple terms, Reinforcement Learning can be described as a problem involving an *agent* interacting with an *environment* which provides *numerical reward signal*. The goal of reinforcement learning is to perform actions which **maximizes** the numerical reward signal.As shown in Figure 1, the environment first provides the agent with some state s_t. Based on that state, the agent performs some action a_t and receives a *reward* r_t and a state s_{t+1} from the environment. This continues till the environment provides some terminal state to the agent.

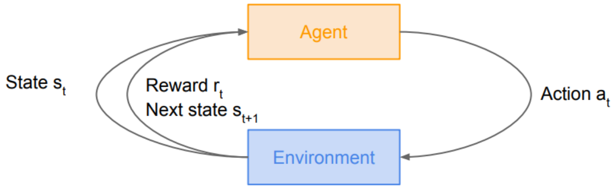

Figure 1: Illustration of Reinforcement Problem-Li (2019)

To formulate reinforcement learning problems mathematically, we use the concept of **Markov Decision Process**. It follows the *Markovian Property* which means that the current state completely characterizes the state of the world. The Markov Decision Process(MDP) formulation of Reinforcement Learning is defined by a 5-tuple set $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$, where:

- $\mathcal{S}$: set of possible states
- $\mathcal{A}$: set of possible action
- $\mathcal{R}$: distribution of reward given (state,action) pair
- $\mathbb{P}$: transition probability
- γ: discount factor

The discount factor γ represents how much we value rewards coming up soon/later (short-term/long-term rewards). Therefore, we can now formally represent the idea of reinforcement learning:

- At time step t = 0, environment samples initial state from the distribution of initial states, i.e $s_0 \sim p(s_0)$.

- Then, for t = 0 until done:
  - agent selects action a_t
  - environment samples reward $r_t \sim R(.|s_t, a_t)$
  - environment samples next state $s_{t+1} \sim P(.|s_t, a_t)$
  - agent receives reward r_t and also the next state s_{t+1}

A **policy π** is function which provides mapping from $\mathcal{S} \rightarrow \mathcal{A}$ that specifies what action to take in each in each state. Therefore, the objective in reinforcement learning can be stated as **find optimal policy π^* that maximizes cumulative discounted reward**. The *cumulative discounted reward* means the sum of rewards over all time steps($t > 0$) multiplied by the power of discount factor: $\sum\limits_{t>0} \gamma^t r_t$

2.2 Finding the optimal policy π^*

As we mentioned in the last section, we want to find an optimal policy π^* that maximizes the sum of rewards. But, to do this we would need information about the initial state, transition probabilities,...etc and many other factors, which are completely random. Therefore, to handle this element of randomness, we **maximize the expected sum of rewards** instead:

$$\pi^* = \arg\max_{\pi} \ \mathbb{E}\,[\sum_{t>0} \gamma^t r_t | \pi] \tag{1}$$

2.3 Value Function and Q-Value Function

Now, if the agent in reinforcement learning follows a policy, then it would lead to some trajectories(path) of states-actions-rewards like $s_0, a_0, r_0, s_1, a_1, r_1, ..., s_n, a_n, r_n$. Among all these possible states, we need to figure out how good is a particular state from all the other states. This is done by the **Value Function**. The *Value Function* at a state *s* is the expected cumulative reward from following reward from following the policy π from state *s*:

$$V^{\pi}(s) = \mathbb{E}\,[\sum_{t>0} \gamma^t r_t | s_0 = s, \pi] \tag{2}$$

Now, the next question which arises is that how can we distinguish between what particular action to take in a particular state. This is addressed by the **Q-Value Function**. The *Q-Value Function* at state *s* and action *a*, is the expected cumulative reward from taking action *a* in a state *s* and the following the policy π:

$$Q^{\pi}(s, a) = \mathbb{E}\,[\sum_{t>0} \gamma^t r_t | s_0 = s, a_0 = a, \pi] \tag{3}$$

2.4 Bellman Equation and Solving for Optimal policy

The optimal Q-Value function Q^* is the maximum expected cumulative reward achievable from a given (state,action) pair:

$$Q^*(s, a) = \max_{\pi} \ \mathbb{E}\,[\sum_{t>0} \gamma^t r_t | s_0 = s, a_0 = a, \pi] \tag{4}$$

Q^* satisfies the following **Bellman Equation**:

$$Q^*(s, a) = \mathop{\mathbb{E}}_{S' \sim \varepsilon} [r + \gamma \max_{a'} Q^*(s', a') | s, a] \tag{5}$$

Intuition for Bellman Equation: if the optimal state-action values for the next time-step $Q^*(s', a')$ are known, then the optimal strategy is to take the action that maximizes the expected value of $r + \gamma\, Q^*(s', a')$.

The optimal policy π^* corresponds to taking the best action in any state as specified by Q^*.

2.5 Solving for Optimal policy: Q-Learning

To solve for the optimal policy, we use the **Value Iteration** Algorithm. The value iteration algorithm uses the Bellman Equation as an iterative update:

$$Q_{i+1}(s,a) = \mathbb{E}[r + \gamma \max_{a'} Q_i(s',a')|s,a] \tag{6}$$

At each step, we are going to refine our approximation of Q^* by trying to enforce the Bellman Equation. Q_i will converge to Q^* as $i \to \infty$

Fundamental Computational Challenge of Reinforcement Learning

The iterative update of value iteration algorithm for computation of optimal Q^* function is **Not Scalable**. It requires that we have to compute Q(s,a) for every state action pair, which is computationally infeasible to compute for entire state space!

The solution for this challenge is to use a **function approximator** to estimate Q(s,a) action-value function. This is called **Q-Learning**. If the function approximator is a deep neural network, then it is called **Deep Q-Learning**.

So, basically we parameterize the Q-value function with parameters θ, which represents the **weights** of deep neural network:

$$Q(s,a;\theta) \approx Q^*(s,a) \tag{7}$$

We define the Loss function as the difference between our approximated Q-Value function $Q(s,a;\theta_i)$ and y_i is the Bellman Equation $\mathbb{E}_{S' \sim \varepsilon}[r + \gamma \max_{a'} Q^*(s',a')|s,a]$:

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(.)}[(y_i - Q(s,a;\theta_i))^2] \tag{8}$$

Forward Pass: In the Forward Pass, we iteratively try to make the Q-value close to the target value (y_i) as it should have been, if the Q-function corresponds to the optimal Q^*.
Backward Pass: In Backward Pass, we do the Gradient Update (w.r.t Q-function parameters θ):

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(.);s'\sim\epsilon}[(r + \gamma \underset{max}{a'} Q(s',a';\theta_{i-1}) - Q(s',a';\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)] \tag{9}$$

3 Alternative Approaches for Achieving Optimality Criteria

Q-Learning is one of the approaches in which we could estimate the optimal Q-Value function for a reinforcement learning setting. However, the objective of the agent, which is to obtain a reward sequence as 'large' as possible, can also be achieved by computing the *optimal value function*, instead of computing the Q-Value function. The γ-discounted optimal value function $V^*_{\gamma,M}(s)$ for a state 's',infinite horizon MDP 'M', set of all policies Π and discount factor γ is defined by (Machandranath Kakade (2003)):

$$V^*_{\gamma,M}(s) = \sup_{\pi\in\Pi} V_{\pi,\gamma,M}(s) \tag{10}$$

The optimal value function also satisfies the Bellman Equation. There are two classes of algorithms which computes this optimal value function: *Exact Methods* and *Near-Optimal Methods*. The Exact Method is essentially the **Value Iteration** algorithm, similar to earlier Q-Value version, which also suffers the same problem of being intractable for entire state space. The Value Iteration Algorithm and its bounds is presented in Procedure 1.

Bounds for Discounted Value Iteration Algorithm
The bounds for this algorithm is (Proof in Appendix 10):

$$V_{\pi,\gamma}(s) \geq V^*_\gamma - 2\gamma^{T'} \tag{11}$$

Similar to the discounted value iteration algorithm, we also have γ-**Discounted Policy Iteration** where we compute an optimal policy from a value function in each iteration. This algorithm and its bounds is presented in Procedure 2.

Procedure 1 Discounted Value Iteration

Input: Infinite Horizon MDP **M**, Discount Factor $\boldsymbol{\gamma}$, total number of Epochs in MDP **T'**
Output: Optimal Deterministic Policy π^*

```
procedure DVI(M,$\gamma$,T')
    Initialize $J_0 = 0$, where $J_t \in \mathcal{R}^N$ is a vector
    for $t = 1, 2, 3....T'$ do
        Compute Backup Operator B :$[BJ](s) \equiv \max_{a\in\mathcal{A}}((1-\gamma)r(s,a)+\gamma\,\mathbb{E}_{s'\sim P(.|s,a)}[J_{t'}(s')])$
        Update $J_t = BJ_{t-1}$
    Return the Policy:
     $\pi(s) = \arg\max_{a\in\mathcal{A}}((1-\gamma)r(s,a)+\gamma\,\mathbb{E}_{s'\sim P(.|s,a)}[J_{T'}(s')])$
```

Procedure 2 γ-Discounted Policy Iteration

Input: Infinite Horizon MDP **M**, Discount Factor $\boldsymbol{\gamma}$, total number of Epochs in MDP **T'**
Output: Optimal Deterministic Policy π^*

```
procedure DPI(M,$\gamma$,T')
    Set the initial policy $\pi_0$ randomly .
    for $t = 1, 2, 3....T'$ do
        Update $J_t(s) = V_{\pi_{t-1},\gamma}(s)$
        Compute the Policy $\pi_t(s) = \arg\max_{a\in\mathcal{A}}((1-\gamma)r(s,a)+\gamma\,\mathbb{E}_{s'\sim P(.|s,a)}[J_{T'}(s')])$
    Return the Policy $\pi_{T'}$
```

Bounds for γ-Discounted Policy Iteration Algorithm

The bound for this algorithm is given by (Proof in Appendix 10):

$$\left\| V_{\pi_{T'},\gamma} - V_\gamma^* \right\|_\infty \leq \gamma^{T'} \tag{12}$$

Now, we will end this section by giving a result on the rate of convergence of these two exact methods. From the bounds on convergence rates, we can see that both algorithms gives us policies which are $O(\gamma^{T'})$ close to the optimal solution after T' iterations. Now, we can choose T' as:

$$T' = \log_\gamma \epsilon = O(\frac{-\log \epsilon}{1-\gamma}) \tag{13}$$

However, in this project we will not focus on achieving the optimality criteria using value function. Instead we will analyse Q-Learning and its variants. The $Q(s,a)$ function gives a *better idea* about the reward with respect to the state and the action space while the V(s) function determines only the goodness of the state irrespective of the action space to get the reward.

4 Asymptotic convergence rate for Q Learning

We will now incorporate the concept of learning rate in estimating the optimal Q-Value Function. Since we have used MDP to mathematically formulate the RL problem, we can use the following algorithm to estimate the value of Q-Value functions for a state-action pair(Szepesvári (1998)):

$$Q_{t+1}(s,a) = (1-\alpha_t(s,a))Q_t(s,a) + \alpha_t(s,a)(r_t(s,a) + \gamma \max_{b\in\mathcal{A}} Q_t(y_t,b)) \tag{14}$$

As we know, $s \in \mathcal{S}$ and $a \in \mathcal{A}$ are states and actions and both $\mathcal{S}$ and $\mathcal{A}$ are finite. Also, $\alpha_t(s,a)$ is the **learning rate** associated with state-action pair (s,a) at time *t* and its value lies in $0 \leq \alpha_t(s,a) \leq 1$.

Conditions for Convergence of Q-Learning

The following constraints related to the learning rate $\alpha_t(s,a)$ should be satisfied for the convergence of Q-Learning:

$$\sum_{t=1}^{\infty} \alpha_t(s,a) = \infty \text{ and } \sum_{t=1}^{\infty} \alpha_t^2(s,a) \leq \infty \tag{15}$$

Another assumption which we need to make to get asymptotic convergence rate of Q-Learning, is that we have to assume that the learning rates takes the specific form:

$$\alpha_t(s,a) = \begin{cases} \frac{1}{S_t(s,a)} & \text{if } (s,a) = (s_t,a) \\ 0 & \text{Otherwise} \end{cases} \tag{16}$$

In the above equation, $S_t(s,a)$ represents the number of times the state-action pair was visited by the process $(s_s.a_s)$ before time-step t plus one,i.e $S_t(s,a) = 1 + \#(s_s,a_s) = (s,a), 1 \leq s \leq t$ If the above two conditions hold true, then the Q-Learning is guaranteed to converge to a fixed point Q^*, which is defined by the operator $T : \mathcal{R}^{\mathcal{S}\times\mathcal{A}} \to \mathcal{R}^{\mathcal{S}\times\mathcal{A}}$, which is defined by:

$$(TQ)(s,a) = R(s,a) + \gamma \sum_{y\in\mathcal{S}} P(s,a,y) \max_b Q(y,b) \tag{17}$$

Now if the above conditions hold true, then we have the following **asymptotic convergence rate with probability 1** (Proof in Appendix 10):

$$|Q_t(s,a) - Q^*(s,a)| \leq \frac{B}{t^{R(1-\gamma)}} \tag{18}$$

$$\text{and} \tag{19}$$

$$|Q_t(s,a) - Q^*(s,a)| \leq B\sqrt{\frac{\log\log t}{t}} \tag{20}$$

In the above equations, $B > 0$ is some suitable constant. $R = p_{min}/p_{max}$, where $p_{min} = \min_{(s,a)} p(s,a)$ and $p_{max} = \max_{(s,a)p(s,a)}$, where $p(s,a)$ is the *sampling probability* of (s,a).

5 Convergence of Q Learning in Polynomial Time with Appropriate Learning Rate

Learning rate can be suitably selected to obtain polynomial convergence rate. For time t, in case of polynomial learning rate $\frac{1}{t^\omega}$, where $\omega\epsilon(\frac{1}{2},1)$, the convergence rate is polynomial in $\frac{1}{1-\gamma}$, where γ is the discount factor. In case of linear learning rate $\frac{1}{t}$, the convergence rate is exponential in $\frac{1}{1-\gamma}$. The following theorems have been devised to show the polynomial and exponential convergence rate corresponding to the selection of polynomial and linear learning rates, respectively.

Let Q_t be the value of the synchronous Q learning algorithm using polynomial learning rate at time T.Then with probability 1-δ, for $||Q_T - Q_*|| \leq \epsilon$, $\beta = \frac{1-\gamma}{2}$ and $V_{max} = \frac{R_{max}}{1-\gamma}$,

$$T = \Omega((\frac{V_{max}^2 ln(\frac{|S||A|V_{max}}{\delta\beta\epsilon})}{\beta^2\epsilon^2})^{\frac{1}{\omega}} + (\frac{1}{\beta} ln \frac{V_{max}}{\epsilon})^{\frac{1}{1-\omega}}) \tag{21}$$

The detailed proof sketch has been illustrated in Theorem 4 in Appendix. The theorems for synchronous Q learning with linear learning rate and asynchronous Q learning with polynomial and linear learning rate are stated, below. The three theorems have been proved in a method similar to the above proof sketch in the paper Even-Dar and Mansour (2004).

Let Q_t be the value of the synchronous Q learning algorithm using linear learning rate at time T.Then for any positive constant Ψ with probability 1-δ, for $||Q_T - Q_*|| \leq \epsilon$,

$$T = \Omega((2+\Psi)^{\frac{1}{\beta} ln \frac{V_{max}}{\epsilon}} (\frac{V_{max}^2 ln(\frac{|S||A|V_{max}}{\delta\beta\epsilon})}{\beta^2\epsilon^2})) \tag{22}$$

Asynchronous Learning updates the value for only one state action pair at a time-step while Synchronous Learning updates the value for all state action pairs in a single time-step. To ensure, that the occurrence of the state-action pairs is enough for the update in t e algorithm to continue, the concept of covering time L has been introduced in the theorems below. The meaning of each iteration of the algorithm having a covering time L is that from any start state, in L time-steps, all state action pairs are executed. For the purpose of proving the theorems of convergence of asynchronous learning, the definition of the covering time is relaxed such that the condition of covering time occurs with probability $\frac{1}{2}$ and with high probability the covering time is LlogT over T iterations. The sequence of state-action pairs can be arbitrary.

Let Q_t be the value of the asynchronous Q learning algorithm using polynomial learning rate at time T.Then with probability 1-δ, for $||Q_T - Q_*|| \le \epsilon$,

$$T = \Omega((L^{1+3\omega}\frac{V_{max}^2 ln(\frac{|S||A|V_{max}}{\delta\beta\epsilon})}{\beta^2\epsilon^2})^{\frac{1}{\omega}} + (\frac{L}{\beta}ln\frac{V_{max}}{\epsilon})^{\frac{1}{1-\omega}}) \tag{23}$$

Let Q_t be the value of the synchronous Q learning algorithm using linear learning rate at time T.Then for any positive constant Ψ with probability 1-δ, for $||Q_T - Q_*|| \le \epsilon$,

$$T = \Omega((L + \Psi L + 1)^{\frac{1}{\beta}ln\frac{V_{max}}{\epsilon}}(\frac{V_{max}^2 ln(\frac{|S||A|V_{max}}{\delta\beta\epsilon})}{\beta^2\epsilon^2})) \tag{24}$$

6 Challenges to Theoretical Analysis of Deep Q Networks

Reinforcement Learning becomes intractable while using non-linear function approximators like deep neural networks. This is because it accounts to a highly non-convex statistical optimization problem, which is difficult to solve. A theoretical study of Deep Q Network (DQN) is challenging primarily due to two reasons. One is that reinforcement learning is unstable when general non-linear or even linear function approximators are directly implemented. Experience replay is used with DQN to stabilize Deep Q Learning by removal of the temporal dependency of observations that are used while training the neural network. Another challenge is that DQN uses a target network other than the Q network to estimate the mean-squared Bellman error in an unbiased way while training the neural network. The target network and Q network are coupled with each other after a set of iterations.

For the purpose of theoretical investigation of DQN, experience replay is simplified with an independence assumption that independent and identically distributed (i.i.d) observations $(S_i, A_i)_{i\epsilon[n]}$ are sampled from a fixed distribution $\sigma\epsilon P(SxA)$. Deep neural network with rectified linear units (ReLU) and large batch size has been used as the function approximator. The resulting algorithm is reduced to the neural fitted Q-iteration algorithm. The algorithm is as follows:

Procedure 3 Neural Fitted Q-Iteration

Inputs: $MDP(S, A, P, R, \gamma)$, function class F, sampling distribution σ, number of iterations K, number of samples n, initial estimator Q_0, where S is the state space, A is the action space, P is the transition probability distribution, R is the reward function and γ is the discount factor

Output: Estimator $\widetilde{Q}_K$ and policy π_K for Q^*

```
for $k = 0, 1, 2, .....K-1$ do
    i.i.d. observations {$(S_i, A_i)$, i $\epsilon$ n } are sampled from the $\sigma$ distribution
    $R_i$ is calculated from $R(.|S_i, A_i)$ and $S^{'}$ is obtained from $P(.|S_i, A_i)$
    The action-value function Q is updated. $\widetilde{Q}_{K+1}$ = arg $min_{f\epsilon F}$ $\frac{1}{n}\sum_{i=1}{}^{n}[Y_i - f(S_i, A_i)]^2$
$\pi_K$ policy is a greedy policy w.r.t $\widetilde{Q}_K$
```

7 Statistical Convergence Rate of Deep Q Learning

The following section establishes the convergence rate for Deep Q Learning in Mnih et al. (2013), using the concepts of Lipschitz continuous, Hölder Smooth Function and other theoretical tools. $Q^{\pi k}$ is the action-value function for policy π_k, $(k\epsilon N)$ in the Procedure 3

$$||Q^* - Q^{\pi k}||_{1,\mu} \le C.\frac{\phi_{\mu,\sigma}.\gamma}{(1-\gamma)^2}.|A|.(logn)^{\xi^*}.n^{\alpha^*-1} + 4\frac{\gamma^{K+1}}{(1-\gamma)^2}.R_{max} \tag{25}$$

where $F_0 = \{f : SxA \mapsto \mathbb{R} : f(.|a)\epsilon F(L, \{d_j\}_{j=0}^{L+1}, s)\forall a\epsilon A\}$, $F(L, \{d_j\}_{j=0}^{L+1}, s)$ is the family of sparse ReLU networks, $G_0 = \{f : SxA \mapsto \mathbb{R} : f(.|a)\epsilon G(\{\rho_t, t_j, \beta_j, H_j\}_{j\epsilon[q]})\forall a\epsilon A\}$, $G(\{\rho_t, t_j, \beta_j, H_j\}_{j\epsilon[q]})$ is a set of composition of Hölder Smooth Functions. $f(.|a)$ is Lipschitz continuous $\forall a\epsilon A$. Therefore for $s_1, s_2\epsilon S$, $\max f(s, a)$ is Lipschitz continuous as illustrated below.

$$|max_{a'\epsilon A}f(s_1, a^{'}) - max_{a'\epsilon A}f(s_2, a^{'})| \leq max_{a'\epsilon A}|f(s_1, a^{'}) - f(s_2, a^{'})| \tag{26}$$

Thus for any $f\epsilon F$ and $a\epsilon A$, $(Tf)(s, a)$ can be represented as composition of of Hölder Smooth Functions where T^{π} is the Bellman operator which can be defined as $(T^{\pi}Q)(s, a) = r(s, a) + \gamma.P^{\pi}Q(s, a)$, r(s,a) is the expected reward at state s while taking the action a. The constraints to equation 25 are:

$$(1-\gamma)^2.\sum_{m\geq 1}\gamma^{m-1}.m.k(m; \mu, \sigma) \leq \phi_{\mu,\sigma} < \infty \tag{27}$$

where σ is the sampling distribution in Procedure 3,μ is the fixed distribution on $S \times A$ and the m-th **concentration coefficient** can be defined as:

$$k(m; \mu, \sigma) = sup_{\pi_1,\pi_2...\pi_m}\sqrt{[\mathbb{E}_{\sigma}\,|\frac{dP^{\pi_m}P^{\pi_{m-1}}...P^{\pi_1}\mu}{d\sigma}|^2]} \tag{28}$$

where μ and σ are the probability measures which are continuous w.r.t Lebesgue measure $S \times A$. $\{\pi_t\}_{t\geq 1}$ is a sequence of policies based on which action A_t is taken. Initial states (S_0, A_0) has the distribution μ and $P^{\pi_m}P^{\pi_{m-1}}...P^{\pi_1}\mu$ is the distribution of (S_m, A_m) for any integer m.

The statistical rate of convergence is the sum of a statistical and an algorithmic error. The algorithmic error converges to 0 with increasing number of iterations. The statistical error represents the bias and the variance due to approximation of the action value function with the neural network which gives an idea about the difficulty of the problem. When the number of iterations is $K > C^{'}.\frac{[log|A|+(1-\alpha^*).logn]}{log\frac{1}{\gamma}}$, where $C^{'}$ is sufficiently large, the algorithmic error is over-shadowed by the statistical error. If the poly-algorithmic term is ignored and γ and $\phi_{\mu,\sigma}$ are treated as constants, then the error rate is $|A|.n^{\alpha^*-1} = |A|.n^{-\frac{2\beta*}{2\beta*+t_j}}$ which scales linearly to the size of the action space and becomes 0 when $n \mapsto \infty$.

Proof Sketch: The theorem for the convergence rate for DQN has been proven by using the following 2 theorems and 2 lemmas:

Error propagation theorem: It is based under the conditions in equations 27 and 28 where the maximum one-step approximation error $\epsilon_{max} = max_{k\epsilon[K]}||T\hat{Q}_{k+1} - \hat{Q}_k||_{\sigma}$ and T^{π} is the Bellman operator.

$$||Q^* - Q^{\pi k}||_{1,\mu} \leq 2.\frac{\phi_{\mu,\sigma}.\gamma}{(1-\gamma)^2}.\epsilon_{max} + 4\frac{\gamma^{K+1}}{(1-\gamma)^2}.R_{max} \tag{29}$$

The theorem illustrates that ϵ_{max} is the statistical error of the DQN in one step and how the one-step error propagates as the time-step iterations of the algorithms increases. The detailed proof is in the paper Yang et al. (2019).

One-step approximation error theorem: This theorem is essential to identify the bias and variance while estimating the Q function by the deep ReLU neural networks. The theorem constructs a family of ReLU networks F_0 to approximate the functions f(.la) in G_0 and establish an upper bound on the approximation error. The l_{∞} error for approximating the functions have been established in equation 30.$N[\delta, F(L, \{d_j\}_{j=0}^{L+1}, s)$ is the minimal δ covering of F w.r.t l_{∞} norm and and N_{δ} is its cardinality.

$$||\hat{Q}^* - TQ^{\pi k}||_{\sigma}^2 \leq (1+\epsilon)^2.\omega(F) + \frac{C.V_{max}^2}{n.\epsilon}.logN_{\delta} + C^{'}.V_{max}.\delta \tag{30}$$

$$\omega(F_0) = sup_{f'\epsilon G_0}inf_{f\epsilon F_0}||f - f^{'}||_{\sigma}^2 \leq sup_{f'\epsilon G_0}inf_{f\epsilon F_0}||f - f^{'}||_{\infty}^2 \tag{31}$$

$\omega(F_0)$is the bias incurred while approximating functions in G_0 with functions in F_O.The detailed proof is in the paper Yang et al. (2019).

Approximation of Hölder Smooth Function lemma: The functions that are being approximated in G_0 is the family of Hölder Smooth Function lemma. Now for a ReLU network $f \epsilon F(L, \{d_j\}_{j=0}^{L+1}, s, V_{max})$ and $g \epsilon C_r([0,1]^r, \beta, H)$ is a family of Hölder Smooth Function lemma. The detailed definitions and constraints are described in the paper Yang et al. (2019).

$$||f - g||_\infty \leq (2H+1).3^{r+1}.N.2^{-m} + H.2^\beta.N^{-\frac{\beta}{R}} \tag{32}$$

This lemma has been used to prove the following bound, the details of which are in the paper Yang et al. (2019).

$$\omega(F_0) \leq n^{\alpha^*-1} \tag{33}$$

Covering number of ReLU network lemma: The covering number N_δ has been used in the equation 30 to describe the one-step approximation error theorem.

$$log|N[\delta, F(L, \{d_j\}_{j=0}^{L+1}, s, V_{max}, ||.||_\infty]| \leq (s+1).log[2\delta^{-1}.(L+1).D^2] \tag{34}$$

The lemma is used to determine the following bound, the details of which are in the paper Yang et al. (2019).

$$logN_0 \leq |A|.log|N_\delta| \lessapprox n^{\alpha^*}.(logn)^{1+2\xi'} \tag{35}$$

Equations in 29, 30,33,35 have been used to obtain the theorem for the convergence rate for Deep Q Learning in equation 25.

8 Sample Efficiency of Deep Q Learning

The theoretical question of whether the algorithms in Reinforcement Learning can be made sample-efficient has remained unsolved even with finitely large number of states and actions. Chebyshev's inequality has been applied on i.i.d data X_i where $S_n = \sum_{i=1}^n X_i$ and $\mathbb{E}[X_i] = \mu$. Therefore,

$$\forall \epsilon > 0, P(|\frac{S_n}{n} - \mu| > \epsilon) \leq \frac{\sigma}{n\epsilon^2} \tag{36}$$

The inequality shows that the size of the dataset has to be large enough to achieve convergence in the expectation in Rocke (2017). The convergence is also dependent on the variance of the data.

9 Over-Estimation Challenge in Deep Q Learning

Deep Q Learning suffers from the problem of over-estimation of the action value parameters under certain conditions Hasselt et al. (2016). Over-estimation poses a computational and a statistical challenge in DQN which may adversely impact the outcome of the Deep Q Learning if the over-estimations are not uniformly concentrated across the states. Sometimes extremely high action values are learned due to a maximization step over estimated action values while updating the objective function, thus being biased to over-estimated action values. The over-estimations can be due to inaccurate action values which is possible as the true values are unknown. Another reason for over-estimation can be due to inflexible function approximation with irreducible asymptotic errors.

9.1 Lower Bound of Over-Estimation in Deep Q Learning

Let a state S be considered where the optimal action values are equal at $Q_*(s,a) = V_*(s)$ for some $V^*(s)$. Let Q_t be unbiased and arbitrary value estimates such that $\sum_a(Q_t(s,a) - V_*(s)) = 0$ but not in all cases when $\sum_a(Q_t(s,a) - V_*(s))^2 = C$ where $C > 0$. $m \geq 2$ is the number of actions in s.

$$max_a Q_t(s,a) \geq V_*(s) + \sqrt{\frac{C}{m-1}} \tag{37}$$

Over-optimism increases with the number of actions.

Proof Sketch: The error for each action a is $\epsilon_a = Q_t(s,a) - V_*(s)$. The set of error values are divided into 2 sets of positive error values ϵ_i^+ of size n and negative error values ϵ_i^- of size m-n. Let $\{\epsilon_a\ \}$

satisfies the following inequality $max_a\epsilon_a \le \sqrt{\frac{C}{m-1}}$ and $\sum_a {\epsilon_a}^2 = mC$, from the condition of the theorem. Therefore $n \neq m$, as otherwise $\sum_a \epsilon_a = 0$ contradicts the condition. Thus $n \le m-1$.

$$\sum_{i=1}^{n} \epsilon_i^+ \le n max_i \epsilon_i^+ \le n\sqrt{\frac{C}{m-1}} \tag{38}$$

Again, $\sum_{j=1}^{m-n} \epsilon_j^- \le n\sqrt{\frac{C}{m-1}}$ and $max_j|\epsilon_j^-| \le n\sqrt{\frac{C}{m-1}}$. Therefore, by applying Hölder's inequality,

$$\sum_{j=1}^{m-n} {\epsilon_j^-}^2 \le \sum_{j=1}^{m-n} \epsilon_j^- max_j|\epsilon_j^-| \le n^2\frac{C}{m-1} \tag{39}$$

Therefore from equations 38 and 39.

$$\sum_{j=1}^{m-n} {\epsilon_j}^2 \approx \sum_{j=1}^{m-n} {\epsilon_j^+}^2 + \sum_{j=1}^{m-n} {\epsilon_j^-}^2 \le \frac{C(n(n+1))}{m-1} \le mC \tag{40}$$

This contradicts the condition in the theorem, therefore $max_a\epsilon_a \ge \sqrt{\frac{C}{m-1}}$. The theorem holds true.

9.2 Double Q Learning

The Double Q Learning algorithm has been proposed in the paper Hasselt (2010) to address the problem of over-estimation. It can be generalized for large scale function approximation for Deep Reinforcement Learning as proposed in the paper Hasselt et al. (2016). It has been proposed that the over-estimation can be reduced by breaking up the maximization operation in the target function into action selection and action evaluation. Target network gives the second value function without additional networks.Greedy policy is evaluated according to online network but target network is used to estimate its value. The Double Q Learning algorithm can be represented as follows:

Procedure 4 Double Q Learning

```
Inputs: Starting state $s\epsilon S$, $Q_A$ and $Q_B$ are initialized at a certain timestep.
Output: Does not return any variable. Maximum reward is obtained at the end of the algorithm.
  while $s\epsilon S$ do
      a is selected based on $Q_A(s,.)$ and $Q_B(s,.)$. r and $s^{'}$ are observed.
      UPDATE(A) or UPDATE(B) is selected randomly.
      if UPDATE(A) then
          $a^* = argmax_a Q^A(s^{'}, a)$
          $Q^A(s,a) = Q^A(s,a) + \alpha(s,a)(r + \gamma Q^B(s^{'}, a^*) - Q^A(s,a)$
      else
          if UPDATE(B) then
              $a^* = argmax_a Q^B(s^{'}, a)$
              $Q^B(s,a) = Q^B(s,a) + \alpha(s,a)(r + \gamma Q^A(s^{'}, a^*) - Q^B(s,a)$
      $s = s^{'}$
```

The two Q function Q_A and Q_B for Double Q Learning are updated with a value from the other Q function from the next state. Instead of $Q_A(s^{'}, a^*)$, $Q_B(s^{'}, a^*)$ is used to update Q_A and vice-versa. Q_B is considered to be an unbiased estimate of the action value for Q_A as it is updated in the same algorithm but on a different set of experience samples. Similar reasoning can be applied for Q_A. Both Q functions are learning from separate sets of experience samples. But they can use both value functions to select an action. Therefore, **Double Q Learning is not less data-efficient than Deep Q Learning**.

9.3 Lower Bound of Over-Estimation in Double Q Learning

The bound on the absolute error $|Q_t^{'}(s, argmax_a Q_t(s,a)) - V_*(s)|$ decreases to 0 for Double Q Learning. In case of double Q learning.

$$max_a Q_t(s, a_1) \geq V_*(S) + \sqrt{C\frac{m-1}{m}} \tag{41}$$

$$max_a Q_t(s, a_i) \geq V_*(S) + \sqrt{C\frac{1}{m(m-1)}}, i \leq 1 \tag{42}$$

Then from the conditions of the theorem for the lower bound of over-estimation of DQN, if we have $Q_t^{'}(s, a_1) = V_*(s)$, then the error becomes 0.

9.4 Summary of Double Q Learning

Double Q-Learning reduces the over-optimism leading to more stable and reliable learning. It has been demonstrated with Atari games that better policies can be learned by this learning mechanism without any additional neural network. Double Q-Learning is more robust than Deep Q-Learning and provides higher mean and median scores for the Atari games. The theorem that demonstrates the convergence of the Double Q Learning to the Optimal Policy has been demonstrated in the Theorem 1 in Appendix.

10 Conclusion

The statistical convergence rate of Deep Q Learning is the sum of an algorithmic and a statistical error, under the independent sampling assumption on experience replay and by using ReLU in deep neural networks, that captures the main features of DQN. The algorithmic error converges to 0 for infinite action space while the statistical error represent the bias and variance of the function approximation. The asymptotic convergence rate of Q Learning has also been studied in the project and it has been observed that appropriate selection of the learning rate leads to convergence of Q Learning in polynomial time. Sample efficieny is a challenge for Deep Q Learning. Over-estimation is another major computational and statistical challenge of Deep Q Learning. The lower bound of the over-estimation in Deep Q Learning has been investigated in the project. The mechanism of Double Q Learning has been used for large scale function approximation to solve over-estimation and ensure stable and reliable learning. Future work involves computation of the convergence rate to enable a comparative analysis of the Deep Q Learning and Deep Reinforcement Learning with Double Q Learning with a theoretical perspective.

Acknowledgments

We are highly grateful to Dr. Mehrdad Mahdavi for his continuous guidance and motivation in exploring the exciting domains of Deep Reinforcement Learning.

References

F.-F. Li, "Cs231n: Convolutional neural networks for visual recognition," in *Stanford University*, 2019, http://cs231n.stanford.edu/.

S. Machandranath Kakade, "On the sample complexity of reinforcement learning," 01 2003.

C. Szepesvári, "The asymptotic convergence-rate of q-learning," in *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10*, ser. NIPS '97. Cambridge, MA, USA: MIT Press, 1998, pp. 1064–1070. [Online]. Available: http://dl.acm.org/citation.cfm?id=302528.302898

E. Even-Dar and Y. Mansour, "Learning rates for q-learning," *J. Mach. Learn. Res.*, vol. 5, pp. 1–25, Dec. 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1005332.1005333

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

Z. Yang, Y. Xie, and Z. Wang, “A theoretical analysis of deep q-learning,” *CoRR*, vol. abs/1901.00137, 2019. [Online]. Available: http://arxiv.org/abs/1901.00137

A. Rocke, “Theoritical limitations of dqn,” in *Online Reinforcement Learning Blog*, 2017, https://paulispace.com/inference/2017/08/29/dqn.html.

H. v. Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI’16. AAAI Press, 2016, pp. 2094–2100. [Online]. Available: http://dl.acm.org/citation.cfm?id=3016100.3016191

H. V. Hasselt, “Double q-learning,” in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 2613–2621. [Online]. Available: http://papers.nips.cc/paper/3964-double-q-learning.pdf

Appendix

Lemma 1: Assume that $\eta_1, \eta_2, ..., \eta_t, ..$ are independent random variables with a common underlying distribution $\mathcal{P}(\eta_t = i) = p_i > 0$. Then the process x_t defined by :

$$x_{t+1}(i) = \begin{cases} (1 - \frac{1}{s_t(i)})x_t(i) + \frac{\gamma}{s_t(i)} \left\| x_t \right\| & \text{if } \eta_t = i \\ x_t(i), & \text{if } \eta_t \neq i \end{cases} \tag{43}$$

satisfies

$$\left\| x_t \right\| = O(\frac{1}{t^{R(1-\gamma)}}) \tag{44}$$

with probability 1, where $R = (\frac{\min_i p_i}{\max_i p_i})$

Proof Sketch

Let $T_0 = 0$ and $T_{k+1} = \min\{t \geq T_k | \forall i = 1, ..., n, \exists s = s(i) : \eta_s = i\}$, i.e T_{k+1} is the smallest time after time T_k such that during the time interval $[T_k + 1, T_{k+1}]$, all the components of $x_t(.)$ are updated in (33), atleast once. Then we can say:

$$x_{T_{k+1}+1}(i) \leq (1 - \frac{1-\gamma}{S_k}) \left\| s_{T_k+1} \right\| \tag{45}$$

In the above equation, $S_k = \max_i S_k(i)$. This inequality holds since if $t_k(i)$ is the last time in $[T_k + 1, T_{k+1}]$ when the i^{th} component is updated, then we have:

$$x_{T_{k+1}+1}(i) = x_{t_k(i)+1}(i) = (1 - 1/S_{t_k(i)})x_{t_k(i)}(i) + \gamma/S_{t_{k(i)}} \left\| x_{t_k(i)}(.) \right\| \tag{46}$$

$$\leq (1 - 1/S_{t_k(i)}) \left\| x_{t_k(i)}(.) \right\| + \gamma/S_{t_{k(i)}} \left\| x_{t_k(i)}(.) \right\| \tag{47}$$

$$= (1 - \frac{1-\gamma}{S_{t_{k(i)}}}) \left\| x_{t_k(i)}(.) \right\| \tag{48}$$

$$\leq (1 - \frac{1-\gamma}{S_k}) \left\| x_{t_k(i)}(.) \right\| \tag{49}$$

In the above set of equations, the fact that $\left\| x_t \right\|$ is decreasing was used. Now, if we use reverse iterations in (35), we get:

$$x_{T_k+1}(.) \leq \left\| x_0 \right\| \prod_{j=0}^{k-1} (1 - \frac{1-\gamma}{S_j}) \tag{50}$$

Now, if we consider the approximations: $T_k \approx Ck$, where $C \geq 1/p_{min}$, $S_k \approx p_{max}T_{k+1} \approx p_{max}/p_{min}(k+1) \approx (k+1)R_0$, where $R_0 = 1/Cp_{max}$. We can then use Large Deviation Theory:

$$\prod_{j=0}^{k-1} (1 - \frac{1-\gamma}{S_j}) \approx \prod_{j=0}^{k-1} (1 - \frac{R_0(1-\gamma)}{j+1}) \approx (\frac{1}{k})^{R_0(1-\gamma)} \tag{51}$$

The above equation holds with probability 1. Now, we define $s = T_k + 1$ to get $s/C \approx k$ and therefore we can write:

$$\left\| x_s \right\| = \left\| x_{T_k+1} \right\| \leq \left\| x_0 \right\| (\frac{1}{k})^{R_0(1-\gamma)} \approx \left\| x_0 \right\| (\frac{C}{s})^{R_0(1-\gamma)} \leq \left\| x_0 \right\| (\frac{C}{s})^{R(1-\gamma)} \tag{52}$$

The above equation holds because x_t and $1/k^{R_0(1-\gamma)}$ are monotonic and also since $R = p_{min}/p_{max} \leq R_0$

Theorem 1:Convergence of Double Q Learning to the Optimal Policy

In a given ergodic MDP, for both Q_A and Q_B as computed by Double Q Learning in Procedure 4 converges to the optimal state-action value function Q^* as given in the Bellman optimality equation 4 with probability if infinite number of experiences that can be expressed by rewards and state transitions for each state action pair can be obtained by using a proper learning policy π. The additional conditions that have to be satisfied are:
1) The MDP is finite. [SxA] < ∞. 2) $\gamma \epsilon [0,1)$ 3) The Q values have been stored in a lookup table.
4) Q_A and Q_B can receive infinite updates. 5) $\alpha_t(s,a) \epsilon [0,1]$, $\sum_t \alpha_t(s,a) = \infty$, $\sum_t (\alpha_t(s,a))^2 < \infty$ with probability 1 and $\forall (s,a) \neq (s_t, a_t)$, $\alpha_t(s,a) = 0$. 6) $\forall s, a, s^{'}$, variance$\{R_{sa}^{s^{'}}\} < \infty$.

Theorem 2: Proof of Bounds for Discounted Value Iteration Algorithm

The **max norm** for the J vector can be defined as:

$$\|J\|_{\infty} \equiv \max_{a} \ |J(s)| \tag{53}$$

Applying the contraction property for two vectors J and J', we have:

$$\|BJ - BJ'\|_{\infty} \leq \gamma \|J - J'\|_{\infty} \tag{54}$$

Considering the first and last epoch, we get:

$$\|BJ_{T'} - J_{T'}\|_{\infty} \leq \gamma^{T'} \|BJ_0 - J_0\|_{\infty} \tag{55}$$

Now, we know that $J_0 = 0$ and $\|BJ_0\|_{\infty} \leq (1-\gamma)$, since our reward functions are normalized. Therefore, we can write:

$$\|BJ_{T'} - J_{T'}\|_{\infty} \leq (1-\gamma)\gamma^{T'} \tag{56}$$

Also, we can show that(proof is omitted for sake of conciseness):

$$V_{\pi,\gamma}(s) \geq V_{\gamma}^{*} - 2\gamma^{T'} \tag{57}$$

Theorem 3: Proof of Bounds for γ-Discounted Policy Iteration Algorithm
Similar to proof of Value Iteration,we can also apply the contraction property here:

$$\begin{aligned} \left\|V_{\pi_{T'},\gamma} - V_{\gamma}^{*}\right\|_{\infty} &\leq \gamma \left\|V_{\pi_{T'-1},\gamma} - V_{\gamma}^{*}\right\|_{\infty} && (58) \\ &\leq \gamma^{T'} \left\|V_{\pi_0,\gamma} - V_{\gamma}^{*}\right\|_{\infty} && (59) \\ &\leq \gamma^{T'} && (60) \end{aligned}$$

Theorem 4: Proof Sketch for Convergence Rate for Synchronous Q Learning with polynomial learning rate

The proof uses a sequence of values D_i such that $D_{k+1} = (1-\beta)D_k$ and $D_1 = V_{max}$ The theorem has been proved with the help of the following lemmas:

Lemma 2:For $m \geq \frac{1}{\beta} ln(\frac{V_{max}}{\epsilon})$, it can be proven that $D_m \leq \epsilon$

Proof Sketch: In order to find the m that satisfies $D_m = (1-\beta)D_m \leq \epsilon$, a logarithm has to be taken on both sides of the inequality.

Lemma 3:For synchronous Q-learning with a polynomial learning rate, for any $t \geq \tau_k$, let $Y_{t;\tau_k}(s,a) \leq D_k$, then for any $t \geq \tau_k + \tau_k^{\omega}$, let $Y_{t;\tau_k}(s,a) \leq D_k(\gamma + \frac{2}{e}\beta)$

Proof Sketch: Let $Y_{t;\tau_k}(s,a) = \gamma D_k + \rho_k$ where $\rho_k = (1-\gamma)D_k$. Then, $Y_{t;\tau_k}(s,a)$ is re-written as $Y_{t;\tau_k}(s,a) = \gamma D_k + (1 - (1 - \alpha_t^{\omega})\rho_t$. It has been shown in the paper Even-Dar and Mansour (2004) that $\rho_t \leq \frac{2}{e}\beta D_k$

Lemma 4:For synchronous Q-learning with a polynomial learning rate, with probability $1 - \delta$, for every iteration of $k\epsilon[1, m]$ and $t\epsilon[\tau_{k+1}, \tau_{k+2}]$, then, $W_{t;\tau_k} \leq (1 - \frac{2}{e})\beta D_k$, provided $\tau_0 = \Omega((\frac{V_{max}^2 ln(\frac{|S||A|V_{max}}{\delta\beta\epsilon})}{\beta^2\epsilon^2})^{\frac{1}{\omega}}$. The lemma has been proved in the paper Even-Dar and Mansour (2004)

Lemma 5: If $a_{k+1} = a_k + a_k^\omega = a_0 + \sum_{i=0}^{k} a_i^\omega$,
then for any constant $\omega\epsilon(0, 1)$, $a_k = O(a_0 + k^{\frac{1}{1-\omega}})$

Theorem 5:Proof Sketch of Asymptotic Convergence Rate of Q-Learning
The main idea behind this proof is to compare Q_t with a simpler process $\hat{Q}_t$, defined by:

$$\hat{Q}_{t+1}(s, a) = (1 - \alpha_t(s, a))\hat{Q}_t(s, a) + \alpha_t(s, a)(r_t(s, a) + \gamma \max_b Q^*(y_t, b)) \tag{61}$$

As we can see the only difference between $\hat{Q}_t$ and Q_t is the inclusion of Q^* in the definition of $\hat{Q}_t$. Because of this difference, we can see that $\hat{Q}_t$ converges to Q^*. Also, we know that the difference process $\Delta_{t+1}(s, a) = \left|Q_t(s, a) - \hat{Q}_t(s, a)\right|$ satisfies the following inequality:

$$\Delta_{t+1}(s, a) \leq (1 - \alpha_t(s, a))\Delta_t(s, a) + \gamma\ \alpha_t(s, a)(\|\Delta_t\| + \left\|\hat{Q}_t - Q^*\right\|) \tag{62}$$

The trick here is that now we can show the difference operator Δ_t converges to 0, which is equivalent to showing convergence of Q_t to Q^*. Now, we can simplify the above notations, by introducing an abstract process, whose update equation is given by:

$$x_{t+1}(i) = (1 - \frac{1}{S_t(i)})x_t(i) + \frac{\gamma}{S_t(i)}(\|x_t\| + \epsilon_t) \tag{63}$$

The intuition behind this update equation is that $i \in 1, 2, ...n$ can be identified with the state-action pairs, x_t with Δ_t, ϵ_t with $\hat{Q}_t - Q^*$. This process can be analysed in two steps: without the perturbation term ϵ_t and then with the perturbation term ϵ_t. Without the perturbation term, we can apply the lemma 1 (proved in 10) for this process. Now let us assume that discount factor $\gamma > 1/2$. From the extension of Law of Iterated Logarithm to stochastic approximation processes that the convergence rate of $\left\|\hat{Q}_t - Q^*\right\|$ is $O(\sqrt{\frac{\log\log t}{t}})$, where the uniform boundedness of the random reinforcement signals must be exploited.