Q-Learning Algorithm: Convergence proof

Sagar Mukherjee

Supervised under: Dr. George Kerchev, Dr. Sergei Merkulov

May 28, 2020

Abstract

Reinforcement learning methods are used extensively for finding optimal policies for the agents to play in an environment. One such profound algorithm, the Q-Learning Algorithm is discussed here, alongwith proving it's convergence to the optimal solution in the given Markovian domain. Finally, certain applications of the algorithm are discussed, one of them being the prediction of future stock prices, given the data of available historic prices.

1 Introduction

This report intends to describe the *Q-learning algorithm* as proposed by Christopher Walkins in his doctoral dissertation[1].

Q-learning is a simple way for agents to learn the way to act optimally in any given controlled Markov Decision Processes. It follows an incremental method for dynamic programming thereby drastically limiting the computational demands. It works on the idea of exploration versus exploitation: it successively enhances its policy function, which determines of the quality of any particular actions at described states. We first need to formally describe what a Markov Decision Process. Certain key terms shall be defined along the way, to help us model the situation. This will lead us to define Value functions (Q-values) for given policies for an agent to move in the environment, and thus, our problem will finally reduce into an optimization task, where our objective would be to maximize the Q-values in a given Markov Decision Process. We shall also discuss a programmatic pseudo-code for the algorithm.

We then proceed with proving a convergence theorem for Q-learning sequence, in rigorous mathematical details, as described by Watkins and Peter Dayan[2], in 1989. The way that it will be done is that a proof device shall be developed, known as the Action-replay-process, which will help us in proving certain lemmas, which will then be used to prove our convergence theorem.

Finally, we shall conclude our discussion with some practical applications of the Q-Learning Algorithm, and explore its variety in usage.

2 Markov Decision Processes

Consider an agent $\mathscr{a}$, in an environment $\mathscr{E}$, restricted to the state-space $\mathscr{S}$, with an available action-space $\mathscr{A}$, and securing rewards from the reward-space $\mathscr{R}$.

Mathematically, we have:

$$\mathscr{S}, \mathscr{R} \in Ob(Set),\ \mathscr{E} = \mathscr{S} \times \mathscr{A},\ \mathscr{R} \in \mathbb{R}^{\dim(\mathscr{E})}$$

Let the transition probabilities, be denoted as:

$$P_{x,y}[a] \coloneqq \mathbb{P}[Y_n = y \mid X_n = x, A_n = a]$$

Definition 2.1. A *Markov Decision Process* $\mathscr{M}$, is defined[3] by the 4-tuple $(\mathscr{S}, \mathscr{A}, P, \mathscr{R})$, where, $\mathscr{S}, \mathscr{A}, P$ and $\mathscr{R}$ are described, as above.

· At $t = n$, let the agent register state $x_n \in \mathscr{S}$, choose an action $a_n \in \mathscr{A}$, and receive a probabilistic reward $r_n \in \mathscr{R}$, such that:

$$\mathbb{E}[r_n](x_n, a_n) \eqqcolon \rho_{x_n}$$

Let the discounting factor be $\gamma \in]0, 1[$.

· If $(R_t)_{t\in\mathbb{N}\cup\{0\}}$ describes the reward process and $(G_t)_{t\in\mathbb{N}\cup\{0\}}$ describes the cumulative reward process, then

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \infty \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots \infty) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Definition 2.2. Let $x \in \mathscr{S}$. A *policy* function $\pi : \mathscr{S} \to \mathscr{A}$ is defined as the solution for the Markov Decision Process $\mathscr{M}$.

Definition 2.3 (Values and Q-Values)**.**
(i) For a given policy π, the *value* of state x, under π, written $V^\pi : \mathscr{S} \to \mathbb{R}$, is defined as:

$$\begin{aligned} V^\pi(x) &= \mathbb{E}[G_t \mid S_t = x] \\ &= \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = x\Big] \end{aligned}$$

· $V_\pi(x)$ is the exactly the same as the expected reward at x, plus the discounted value at next step:

$$V^\pi(x) = \rho_x(\pi(x)) + \gamma \sum_{y\in\mathscr{S}(x)} P_{x,y}[\pi(x)] V^\pi(y)$$

(i) For a given policy π, the *Q-value* of state-action pair $(x, a) \in \mathscr{E}$, under π, written $Q^\pi : \mathscr{E} \to \mathbb{R}$, is defined as:

$$Q^\pi(x, a) = \rho_x(a) + \gamma \sum_{y\in\mathscr{S}(x)} P_{x,y}[\pi(x)] V^\pi(y)$$

3 Optimal policy

Our task, thus reduces to an optimization problem of finding an optimal policy.

Theorem 3.1 (Bellman & Dreyfus[4], 1962)**.**
For a Markov Decision Process $\mathscr{M}$, described as above, $\exists \pi^*$, called the optimal policy, such that:

$$V^* := V^{\pi^*}(x) = \max_{a \in \mathscr{A}(x)} \left\{ \rho_x(a) + \gamma \sum_{y \in \mathscr{S}(x)} P_{x,y}[a] V^{\pi^*}(y) \right\} \qquad (\mathscr{B}.\mathscr{E}.)$$

The above equation $\mathscr{B}.\mathscr{E}.$, is famously known as the Bellman-equation.

Remark. A Bellman equation, named after Richard E. Bellman, is a necessary condition for optimality associated with the mathematical optimization method of dynamic programming. It writes the *value* of a decision problem at a certain point in time, in terms of the payoff from some initial choices and the *value* of the remaining decision problem that results from those initial choices. This breaks a dynamic optimization problem into a sequence of simpler sub-problems, as Bellman's "principle of optimality" prescribes.[5]

Our objective is to determine *Q-values* for an optimal policy:

$$\begin{aligned} Q^*(x,a) &:= Q^{\pi^*}(x,a), \quad \forall x \in \mathscr{S}, \forall a \in \mathscr{A} \\ \implies V^*(x) &= \max_{a \in \mathscr{A}(x)} \left\{ Q^*(x,a) \right\} \end{aligned}$$

Therefore, we shall get the optimal value V^*, corresponding to an optimal policy π^*, for each state-action pairs.

Remark. Although this might look undefined and unsolvable due to a circular equality, it is actually well defined.

Not only that, with the help of Dynamic Programming, we can have a number of methods for calculating the optimal values V^* and an optimal policy π^*, assuming of course, that rewards and transition probabilities are known. Unfortunately, that is not the case and the agent faces with the task to learn Q-values without this information.

There are certain methods to concurrently perform Dynamic programming, but all are highly computationally inefficient, due to the recurrence nature of definition.

We shall, therefore, have our focus on gradually reaching the correct answer through successive approximations, thereby adopting an incremental and iterative approach. They are not only conceptually simple, but also computationally efficient.

4 Q-learning Algorithm

Now, we shall discuss a method to arrive at the optimal policy and Q-values. Q-learning algorithm is a *Reinforcement learning technique*, used such that the agent learns optimal solution, on its own, in a given Markov Decision Process. The goal of Q-learning is to learn the optimal policy by iterating and adapting new Q-values, by using both- exploration and exploitation.

Definition 4.1. Consider a Markov Decision Process $\mathscr{M}$, described as above. Let $(\alpha_n)_{n\in\mathbb{N}\cup\{0\}}$ be a learning-rate sequence and $(r_n)_{n\in\mathbb{N}\cup\{0\}}$ be the reward sequence. We define the *Q-learning* sequence $(Q_n : \mathscr{E} \to \mathbb{R})_{n\in\mathbb{N}\cup\{0\}}$ as:

$$Q_n(x,a) = \begin{cases} (1-\alpha_n)Q_{n-1}(x,a) + \alpha_n(r_n + \gamma V_{n-1}(y_n)) & \text{if } (x,a) = (x_n, a_n) \\ Q_{n-1}(x,a) & \text{otherwise} \end{cases} \tag{1}$$

Here, $V_{n-1}(y)$ is the maximum value, the agent thinks it can achieve, after it arrives at state y:

$$V_{n-1}(y) = \max_{b\in\mathscr{A}(x)} \left\{Q_{n-1}(y,b)\right\} \tag{2}$$

The reader can use the following pseudo-code to implement the Q-learning algorithm:

```
Data:
    $\mathscr{S}$: State-space
    $\mathscr{A}$: Action-space
    $\mathscr{R}$: Reward-space
    $\mathscr{T} : \mathscr{S} \times \mathscr{A} \to \mathscr{S}$: Probabilistic-transition function
    $\gamma$: Discounting factor
    $\alpha$: Learning rate
Result: Estimate Q-values for $\pi \approx \pi^*$
Initialize $Q : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$ arbitrarily
while $Q$ is not converged do
    Start in state $x \in \mathscr{S}$
    while $x$ is not terminal do
        $\pi(x) \leftarrow \operatorname{argmax}_a Q(x,a)$
        $a \leftarrow \pi(x)$
        $r \leftarrow \mathscr{R}(x,a)$
        $y \leftarrow \mathscr{T}(x,a)$
        $Q(y,a) \leftarrow (1-\alpha) * Q(x,a) + \alpha * (r + \gamma * \max_{b\in\mathscr{A}(x)} Q(y,b))$
        $x \leftarrow y$
    end
end
```

Algorithm 1: Q-Learning Pseudocode[6]

Remark.
(i) The learning-rates α_n here, determines how much we retain the previous Q-values, and how much we update it to reflect the new Q-values. If α is closer to 1, then the agent quickly adapts to the new Q-values, whereas, if α is closer to 0, the agent sticks more to the old values. This conundrum leads to a balance between exploration versus exploitation. Thus, the agent initially starts with full-learning mode, but eventually retain its values.
(ii)Here we shall employ a short-hand notation:
Define $n^i(x,a)$: index of the i^{th}-time, action a is tried in the state x.
Set $I := \{n^i : i \in \mathbb{N}\}$, the set containing all the indexes.

Theorem $\mathscr{C}$ (Convergence Theorem)**.**
Given bounded rewards-sequence $|r_n| \leq R$*, and learning-rates* $0 \leq \alpha_n < 1$*, with* $(\alpha_n)_{n\in I} \notin \ell^1(I)$ *and* $(\alpha_n)_{n\in I} \in \ell^2(I)$*.*
(Recall that this means: $\sum_{i=1}^{\infty} \alpha_{n^i} = \infty$ *and* $\sum_{i=1}^{\infty} \alpha_{n^i}^2 < \infty$*)*
Then, the Q-learning algorithm converges to the optimal Q-values:

$$Q_n(x,a) \xrightarrow{n\to\infty} Q^*(x,a), \ \ a.s.$$

Remark. We shall be proving $\mathscr{C}$, with the below strategy:
• **Step 1**: Description of the Action-replay process $\mathbb{A}$
• **Step 2**: Prove certain lemmas:
· lemma A: Q_n optimal for $\mathbb{A}$
· lemma B1-B3: Preparatory lemmas and $\mathbb{A} \to$ real process
• **Step 3**: Use lemmas A-B3, to prove $\mathscr{C}$.

Action-Replay-Process

Consider the state-space $\{< x, n >| \ x \in \mathscr{S}\}$, and the action-space $\mathscr{A}$. Let the discounting factor $\gamma \in]0,1[$ for $\mathbb{A}$, be the same as well. Recall from above that, $n^i(x,a)$ is the index of the i^{th}-time, action a is tried in the state x.

Also, define i_* as:

$$i_* = \begin{cases} \text{argmax}_i\{n^i(x,a) < n\} & \text{if } (x,a) \text{ expected before } n \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

This means that n^{i_*} is the last time before episode n, such that (x,a) was expected.

Now,
· if $i_* = 0$, then the reward is set as $Q_0(x,a)$
$\implies$ The Action-replay-process $\mathbb{A}$ absorbs.

· Otherwise, let i_e be the index of episode replayed probabilistically as follows:

$$i_e = \begin{cases} i_* & \text{with probability } \alpha_{n^{i_*}} \\ i_* - 1 & \text{with probability } (1 - \alpha_{n^{i_*}})\alpha_{n^{i_*-1}} \\ i_* - 2 & \text{with probability } (1 - \alpha_{n^{i_*}})(1 - \alpha_{n^{i_*-1}})\alpha_{n^{i_*-2}} \\ & \vdots \\ 0 & \text{with probability } \prod_{i=1}^{i_*}(1 - \alpha_{n^i}) \end{cases} \tag{4}$$

The way to imagine an action-replay-process is to think of it in terms of a card game. Imagine each episode $(x_t, a_t, y_t, r_t, a_t,)$ is written on a card. All the cards together form an infinite deck, with the first episode-card next-to-bottom as well as stretching upwards infinitely, in order.

The bottom card (numbered 0) has written on it the initial values of agent $Q_0(x, a)$ for all the state-action pairs (x, a). A state of the $\mathbb{A}, < x, n >$, consists of a card number (or level) n, combined with a state x from the real process. The actions permitted in the Action Replay Process are the same as those permitted in the real process.

The next state of the $\mathbb{A}$, given current state $< x, n >$ and action a, is determined as follows:

First, all the cards for episodes coming after n are eliminated, leaving just a finite deck with us. Cards are then removed one at a time from top of this deck and examined until one is found whose starting state and action match x and a, say at episode t. Then a biased coin is flipped, with probability α_t, of coming out heads, and $1 - \alpha_t$, of tails.

If the coin turns up heads, the episode recorded on this card is replayed, a process described below; if the coin turns up tails, this card too is thrown away and the search continues for another card matching x and a. If the bottom card is reached, the game stops in a special, absorbing, state, and just provides the reward written on this card for x, a, namely $Q_0(x, a)$.

Replaying the episode on card t consists of emitting the reward, r_t , written on the card, and then moving to the next state $< y_t, t - 1 >$ in the $\mathbb{A}$, where y_t is the state to which the real process went on that episode. Card t itself is thrown away. The next state transition of the Action-Replay-Process $\mathbb{A}$ will be taken based on just the remaining deck.

Remark. Here, we shall adopt the following notation:
Let $P_{\mathbb{A}_{<x,n>,<y,m>}}[a]$ be the transition probability matrix of the action replay process and its expected rewards be $\rho_x^n(a)$. Then, we define the probabilities that, for each x, n and a, executing action a at state $< x, n >$ in the Action-Replay-Process leads to state y of the real process at some lower level in the deck:

$$P_{x,y}^n[a] := \sum_{m=1}^{n-1} P_{\mathbb{A}_{<x,n>,<y,m>}}[a]$$

We shall now see certain lemmas in the remaining discussion. The first goes on to prove that the Q-learning sequence is optimal for $\mathbb{A}$.

Lemma A (Q_n optimal for $\mathbb{A}$)**.**
Let Q_n be the Q-values obtained from the Q-learning algorithm and $Q_{\mathbb{A}}$ be the Q-values for $\mathbb{A}$. Then, $Q_n(x,a)$ are the optimal action-values for $\mathbb{A}$, with state $< x, n >$ and action a:

$$Q_n(x,a) = Q^*_{\mathbb{A}}(< x,n >, a), \;\; \forall (a,x) \in \mathscr{E}, \; \forall n \geq 0$$

Proof. From the construction of the action replay process $\mathbb{A}$, we have the following equation:

$$Q_0(x,a) = Q^*_{\mathbb{A}}(< x,0 >, a)$$

Hence, the theorem holds for $n = 0$.
Suppose that the Q_{n-1}-values, obtained from the Q-learning sequence are optimal for $\mathbb{A}$:

$$\begin{aligned} \forall (a,x) \in \mathscr{E} : \;\; Q_{n-1}(x,a) &= Q^*_{\mathbb{A}}(< x, n-1 >, a) \\ \implies V^*(< x, n-1 >) &= V_{n-1}(x) \\ &= \max_{a \in \mathscr{A}(x)} \left\{ Q_{n-1}(x,a) \right\} \end{aligned}$$

We now encounter the following two cases:
Case 1: $(x,a) \neq (x_n, a_n)$:
If (x,a) are not the same as (x_n, a_n), then this is the same as performing action a in $< x, n-1 >$, and $Q_n(x,a) = Q_{n-1}(x,a)$:

$$\begin{aligned} Q_n(x,a) &= Q_{n-1}(x,a) \\ &= Q^*_{\mathbb{A}}(< x, n-1 >, a) \\ &= Q^*_{\mathbb{A}}(< x, n >, a) \end{aligned}$$

Case 2: $(x,a) = (x_n, a_n)$:
If (x,a) is the same as (x_n, a_n), then action a_n will be performed in $< x_n, n >$, with probability $(1 - \alpha_n)$ is exactly the same as performing the action a_n in $< x_n, n-1 >$, or with probability α_n yields immediate reward r_n and a new state $< y_n, n-1 >$:

$$\begin{aligned} Q^*_{\mathbb{A}}(< x_n, n >, a_n) &= (1-\alpha_n) Q^*_{\mathbb{A}}(< x_n, n-1 >, a_n) + \alpha_n (r_n + \gamma V^*(< y_n, n-1 >)) \\ &= (1-\alpha_n) Q_{n-1}(x_n, a_n) + \alpha_n (r_n + \gamma V_{n-1}(y_n)) \\ &= Q_n(x_n, a_n) \end{aligned}$$

The last two equalities are due to the induction hypothesis and the Q_n-iteration formula (1).

Hence, from the principle of mathematical induction, we have proved the lemma for all $n \in \mathbb{N} \cup \{0\}$.

$$\therefore Q_n(x,a) = Q^*_{\mathbb{A}}(< x,n >, a), \;\; \forall (a,x) \in \mathscr{E}, \; \forall n \geq 0$$

□

Lemma B.1 (Discounting infinite sequence)**.**
Consider a finite Markov process, with the discounting factor γ, bounded rewards ($|r_n| < R$), transition-probabilities $P_{x,y}[a]$. Let $\mathbb{X}_s \coloneqq (x_0, x_1, \ldots, x_s)$, be the fixed s-steps, starting from x_0, and ending at x_s. Then,

$$\forall \pi : \quad |V^\pi(\mathbb{X}_s) - V^\pi(x_0)| \xrightarrow{s\to\infty} 0$$

Proof. Ignoring the value of the $(s+1)^{th}$-state, we will incur a penalty δ, given by:

$$\delta \coloneqq \gamma^s \sum_{y_{s+1} \in \mathcal{S}(y_s)} P_{y_s, y_{s+1}}[a_s] V^\pi(y_{s+1})$$

Note that rewards are bounded and γ is less that 1.
$\because |r_n| < R,\ \gamma < 1,\ \forall n \in \mathbb{N} \cup \{0\}$:

$$\begin{aligned} V^\pi(y_{s+1}) &< R + \gamma R + \gamma^2 R + \cdots \infty \\ &= R\left(\frac{1}{1-\gamma}\right) \\ &= \frac{R}{1-\gamma} \\ \implies V^\pi(y_{s+1}) &< \frac{R}{1-\gamma} \end{aligned}$$

Therefore, delta is bounded as follows:

$$\begin{aligned} \therefore |\delta| &< \gamma^s \sum_{y_{s+1} \in \mathcal{S}(y_s)} P_{y_s, y_{s+1}}[a_s] \big(\frac{R}{1-\gamma}\big) \\ &= \gamma^s \big(\frac{R}{1-\gamma}\big)(1) \\ &= \frac{\gamma^s R}{1-\gamma} \\ \implies |\delta| &< \frac{\gamma^s R}{1-\gamma} \xrightarrow{s\to\infty} 0 \end{aligned}$$

Hence we have shown that the difference penalty converges to zero, as s goes to infinity.

$$\therefore \forall \pi : \quad |V^\pi(\mathbb{X}_s) - V^\pi(x_0)| \xrightarrow{s\to\infty} 0$$

□

Lemma B.2 (Rewards & transition-probabilities converge)**.**
The probabilities $P^{(n)}_{\mathbb{A}_{x,y}}[a]$ and expected rewards $\rho^{(n)}_{\mathbb{A}_x}[a]$ in the Action-replay-process converge and tend to the transition matrices and expected rewards in the real process as the level n increases to infinity.
$\forall a \in \mathscr{A}$:

$$P^{(n)}_{\mathbb{A}_{x,y}}[a] \xrightarrow{n\to\infty} P^{(n)}_{x,y}[a], \quad a.s.,$$
$$\rho^{(n)}_{\mathbb{A}_x}[a] \xrightarrow{n\to\infty} \rho^{(n)}_{x}[a], \quad a.s.$$

Proof. Stating one of the standard theorems[7] provided by Kushner & Clark, 1978:
"If $(X_n)_{\mathbb{N}}$ is a sequence, updated as:

$$X_{n+1} = X_n + \beta_n(\xi_n - X_n),$$

with the following conditions:
· $0 \leq \beta_n < 1$,
· $(\beta_n) \notin \ell^1$,
· $\xi_n$bounded, and
· $\mathbb{E}[\xi_n] = \Xi$, then

$$X_n \xrightarrow{n\to\infty} \Xi, \quad a.s."$$

In our case:

$$\rho_{<x,n^{i+1}>}(a) = \rho_{<x,n^i>}(a) + \alpha_{n^{i+1}}(r_{n^{i+1}} - \rho_{<x,n^i>}(a)),$$
$$P^{(n^{i+1})}_{\mathbb{A}_{x,y}}[a] = P^{(n^i)}_{\mathbb{A}_{x,y}}[a] + \alpha_{n^{i+1}}(\mathbb{1}_{\{y_n=y\}} - P^{(n^i)}_{\mathbb{A}_{x,y}}[a])$$

such that,

$$\mathbb{E}[r_n] = \rho_x(a),\ \mathbb{E}[\mathbb{1}_{\{y_n=y\}}] = P_{x,y}[a]$$

$$\therefore P^{(n)}_{\mathbb{A}_{x,y}}[a] \xrightarrow{n\to\infty} P^{(n)}_{x,y}[a], \quad a.s.,$$
$$\rho^{(n)}_{\mathbb{A}_x}[a] \xrightarrow{n\to\infty} \rho^{(n)}_{x}[a], \quad a.s.$$

Hence, the theorem directly applies to both, and so we have proved our lemma. Also, since there is only a finite number of states and actions, the convergence is uniform. □

The Action-Replay-Process effectively estimates the mean rewards and transitions of the real process over all the episodes. Since its raw data are unbiased, the conditions on the sums and sums of squares of the learning rates $\alpha_{n^i(x,a)}$ ensure the convergence with probability one.

Lemma B.3 (Close rewards and probabilities imply close values).
Consider an s-step Markov chain, formed according to the probabilities $(P^n_{x_n,x_{n+1}}[a_n])_{n\in\{1,\dots,s\}}$.
Let $\eta > 0$ *be given. Set:*
$Q'(x, a_1, a_2, \dots, a_s)$*: the expected reward for the Markov chain, till s-actions,*
$Q(x, a_1, a_2, \dots, a_s)$*: the expected reward for the real process, till s-actions.*
If we have the following bounds: $\forall a, x, y, \forall i \in \{1, \dots, s\}$*:*

$$|P_{x,y}[a] - P^i[a]| < \frac{\eta}{R},$$
$$|\rho_x(a) - \rho^i_x(a)| < \eta$$

then we have:

$$|Q'(x, a_1, a_2, \dots, a_s) - Q(x, a_1, a_2, \dots, a_s)| < \frac{s(s+1)}{2}\eta$$

Proof. We shall proving the lemma for $s = 2$. The similar proof holds for all s, with more number of terms.
We have:

$$Q(x, a_1, a_2) = \rho_x(a_1) + \gamma \sum_{y\in\mathscr{S}(x)} P_{x,y}\rho_y(a_2),$$
$$Q'(x, a_1, a_2) = \rho^1_x(a_1) + \gamma \sum_{y\in\mathscr{S}(x)} P^1_{x,y}\rho^2_y(a_2)$$

Taking the difference, we see:

$$\begin{aligned}\implies |Q'(x, a_1, a_2) - Q(x, a_1, a_2)| &\le |\rho^1_x(a_1) - \rho_x(a_1)| \\ &\quad + \gamma\Big|\sum_{y\in\mathscr{S}(x)} P^1_{x,y}(\rho^2_y(a_2) - \rho_y(a_2))\Big| \\ &\quad + \gamma\Big|\sum_{y\in\mathscr{S}(x)} \rho_y(a_2)(P^1_{x_y} - P_{x,y})\Big| \\ &\le \eta + \gamma\cdot\eta\cdot 1 + \gamma\cdot R\cdot\frac{\eta}{R} \\ &= 3\eta\end{aligned}$$

Similarly, for the s-step chain, we get $\frac{s(s+1)}{2}$ terms and we get the result. □

This lemma says that if the probabilities $P^n_{x_y}[a]$ and expected rewards ρ^n_x at appropriate levels of the Action-Replay-Process for each of the actions, are close to $P_{x,y}[a]$ and $\rho_x(a)$ respectively, then the value of the series of actions in the Action-Replay-Process will be close to its value in the real process.

In the following section, we now shall proceed with proving the convergence theorem $\mathscr{C}$, by using the above proved lemmas.

5 Proof of Q-learning convergence

We are now ready to prove the convergence theorem $\mathscr{C}$:

<u>*Proof of* $\mathscr{C}$</u>.
Putting the above proved lemma together, we get the following:
From lemma B.2:
For all action $a \in \mathscr{A}$:

$$P^{(n)}_{\mathbb{A}_{x,y}}[a] \xrightarrow{n\to\infty} P^{(n)}_{x,y}[a], \quad a.s.,$$
$$\rho^{(n)}_{\mathbb{A}_x}[a] \xrightarrow{n\to\infty} \rho^{(n)}_{x}[a], \quad a.s.$$

From lemma B.3:

$$|Q'(x,a_1,a_2,\dots,a_s) - Q(x,a_1,a_2,\dots,a_s)| < \frac{s(s+1)}{2}\eta$$
$$\implies Q^*_{\mathbb{A}}(x,a_1,a_2,\dots,a_s) \to Q^*(x,a_1,a_2,\dots,a_s)$$

Hence, from lemma B.1, we have Q-values of the Action-replay-process $\mathbb{A}$, to converge with the Q-values of the real process, in optimal case:

$$Q^*_{\mathbb{A}}(<x,n>,a) \xrightarrow{n\to\infty} Q^*(x,a)$$

Finally, from lemma A, we know that the Q_n-values, obtained from the Q-learning algorithm is optimal for the Action-replay-process $\mathbb{A}$:

$$Q_n(x,a) = Q^*_{\mathbb{A}}(<x,n>,a)$$
$$\therefore Q_n(x,a) \xrightarrow{n\to\infty} Q^*(x,a)$$

Hence, we have proved the convergence of the sequence obtained from the Q-learning process, to the optimal solution. □

6 Conclusions and practical applications

The aim of this project/seminar was to study the Q-learning algorithm, in a Markovian domain, and propose a theorem stating its convergence to the optimal policy. We then were able to prove it by, creating a proof-device called the Action-replay process and proving certain other supporting lemmas.

We shall now state certain practical applications of the Q-Learning algorithm, along with certain key areas to explore for the reader:

1. There are many Quantitative hedge-funds and trading firms which are leveraging reinforcement learning for evaluating trading strategies. Q-learning algorithms can be used in this case, by predicting future stock prices from the available historic price data. One way to implement this is to create a Q-learner module, and train it with several technical indicators (momentum, mean-reversion, moving-averages, etc.) as features.

2. There has been tremendous progress on applying reinforcement learning in Robotics, where robots are programmed to learn policies which map raw video images to robot's actions.

3. Q-learning is extensively used in the gaming industry. One such example is from an online multiplayer role-playing game, called *Defense of the Ancients*, or DotA, which is a very complex game in itself, and millions of human-players play it daily. With the advent of Q-learning, Artifical Intelligence bots have been created, which are able to beat the best DotA players in the world.

4. Google's *DeepMind* was able to create a machine, programmed with a modification of Q-learning (Deep Q-learning), which was able to beat the best chess players in the world.

Acknowledgments

I am very grateful to Dr. George Kerchev for providing me a possibility to discover such wonderful avenues in the realms of reinforcement learning. He guided me throughout and gave me a well-rounded experience in the subject, both in terms of its practical application in Financial trading, as well as mathematical foundation of the algorithm. I would also like to deeply thank Dr. Sergei Merkulov, for giving me the opportunity to give a seminar talk on this project to professors and fellow students.

References

[1] C. J. Watkins, *Learning From Delayed Rewards.* PhD thesis, King's College, London, 1989.

[2] C. J. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, 1992.

[3] R. Bellman, "A markovian decision process," *Indiana University Mathematics Journal*, vol. 6, 1957.

[4] R. Bellman and S. E. Dreyfus, *Applied Dynamic Programming.* Princeton University Press, 1962.

[5] Wikipedia contributors, "Bellman equation," 2020.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* Cambridge, Massachusetts: The MIT Press, 2018.

[7] H. J. Kushner and D. S. Clark, *Stochastic approximation methods for constrained and unconstrained systems.* New York: Springer-Verlag, 1978.