Q-learning algorithm : An overview

By Shaun Paraplammoottil Cheriyan

Student Number :- 7924357

COMP – 3190

John Anderson

University of Manitoba

7th December 2022

Abstract.

Q-learning is an off-policy reinforcement learning algorithm which enables agents to learn how to act in an optimal manner in any state in a controlled Markovian domain even with the absence of a model for the system dynamics [Watkins and Dayan, 1992]. It is basically an algorithm that goes through various states and actions in system and evaluates the value for each action at each state and eventually learns the best possible set of actions to take for a given state based on the values that were calculated [Watkins and Dayan, 1992].

This paper presents a brief introduction to the topic of Q-learning and reinforcement learning and it provides details on how the q-learning algorithm works, the various equations and knowledge involved in the implementation and how it is related to the reinforcement learning algorithms such the Monte Carlo methods and the temporal difference learning algorithm. It also discusses the use/advancements of this algorithm in different fields and in doing so eventually highlight the importance of this algorithm in modern day industries and scientific studies.

1. Introduction

Q-learning algorithm is a type of reinforcement learning algorithm that does not require any model of the data/environment to function [Watkins and Dayan, 1992]. Even without any model of the data or system dynamics, q-learning enable these agents to learn and carry out the optimal action required in any controlled Markovian domains by letting them experience the consequences of their actions (i.e., feedback from the environment) taken at a particular state [Watkins and Dayan, 1992]. Agents in the q-learning algorithm learn in a manner similar to that of temporal difference learning (TD learning) i.e., the agents at a given state tries to perform an

action and they evaluate the result of the actions based on the feedback which they get which is the instant reward or penalty it receives for taking that action in that particular state and "it's estimate of the value of the state to which it is taken" [Watkins and Dayan, 1992]. Q-learning algorithm is also an off-policy reinforcement algorithm which it updates its q-values and chooses the action required to be taken based on the Bellman equation and greed policy [Jang et al, 2019]. Based on these, the q-learning algorithm goes through every single action in every state iteratively and learns which set of actions at different states are the best. This final decision is made with the help of a long-term discounted reward [Watkins and Dayan, 1992]. This is quite similar to how humans take action/decision as we also consider the cost of the action/decision we are about to take (based on experiences) before we actually implement it. It is also similar in the manner of how we as humans when do not have the knowledge on a particular domain or problem, we learn it through trial and error in order to find a solution.

Q-learning algorithm is quite an important algorithm because in contrast to other reinforcement learning algorithms, it has q-functions which are very uncomplicated and therefore it has served as the fundamental foundation for many other reinforcement learning algorithms [Jang et al, 2019]. Q-learning algorithm in practice is a very significant algorithm and hence has been used in many fields [Jang et al, 2019]. There are also many new variants of the q-learning algorithm which has been developed in order to specialise in specific fields such as robotics, games, network management, airplane control etc. It also has been involved in various industrial applications [Jang et al, 2019]. All in all, q-learning algorithm has been a major development in the field of reinforcement learning algorithms and machine learning and we will see the various procedures involved with the implementation of the algorithm and developments/applications as we go along this paper.

2. Reinforcement learning: What it is and how it relates to q-learning?

2.1 Introduction

With its recent success in many different fields such as game theory, operations research, information theory, simulation-based optimization, control theory, statistics etc., reinforcement learning, which is a part of machine learning, has grown to become a major contributor in field of computer intelligence which encompasses the technique of computers making their own decisions in a particular environment without being given any previous data/knowledge about the environment it is placed in [Jang et al, 2019]. As the field of artificial intelligence continue to grow more and more with every new innovation, the opportunities for the future use of the reinforcement learning will grow immensely and many new variants will be developed with response to requirements in different fields [Jang et al, 2019]. Reinforcement learning algorithm is a powerful learning algorithm which even without a model for the environment, learns the optimal policy in that environment by using an agent to interact with the environment and based on the interactions and feedbacks, finds the optimal policy [Jang et al, 2019].

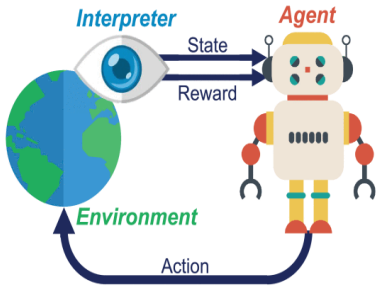

Figure 1. "Reinforcement Learning Framework" [Spano et al, 2019]

As you can see from figure 1, the agent performs an action in the environment which is being observed by an interpreter [Spano et al, 2019]. The interpreter observes and then lets the agent know the feedback from the environment which is the current state of the environment and the

reward obtained for that action which is either positive or negative number which the represents the quality figure for the action that was made [Spano et al, 2019]. By iterating through this process continuously, the agent eventually learns the optimal policy/method to perform the specific task [Spano et al, 2019]. Therefore, through reinforcement learning, the agent is learning by itself through the interactions and feedback from the environment. This is the same way in which the biological thought processes occur. It is similar to how humans and animals perform certain actions, observe the result of the action, and learn from it [Naeem et al, 2020].

2.2 Required terms and knowledge

The basic terms and knowledge that needs to be known is given in this section.

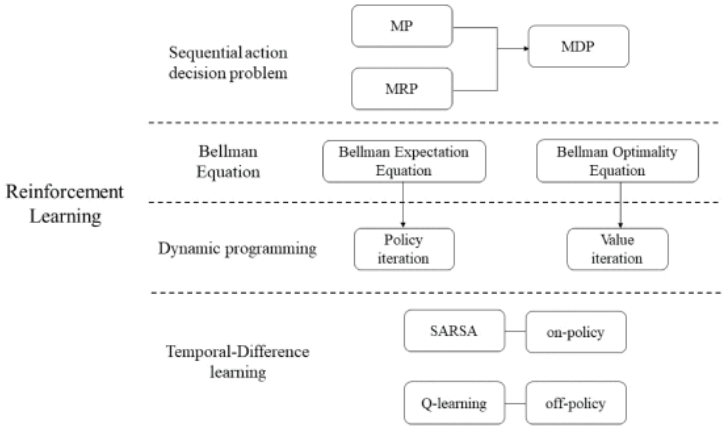

Figure 2: "The flow of reinforcement learning" [Jang et al, 2019]

The figure 2 represents the evolution/flow of reinforcement learning algorithm [Jang et al, 2019]. As shown in the figure, the basis of reinforcement learning is the sequential action decision problem. This problem is defined by the Markov decision process (MDP) [Jang et al, 2019]. The Markov decision process "describes an agent that introduces the concept of the value function for learning, and the value function is linked to the Bellman equation" [Jang et al, 2019]. After the Bellman equation as been constructed with the help of Markov decision process and the value

function, we can use the q-learning algorithm is used to solve the Bellman equation problem [Jang et al, 2019]. Thus, the basic description/understanding of the Markov learning process, value function and Bellman equation is required to understand q-learning algorithm and hence is given below in their respective sections.

A) Markov Decision Process

Markov Decision Process is a decision-making process that is used in an environment which is completely random (which means that it cannot be completely determined by the agent) with the aim of finding the optimal action for every state in that environment [Naeem et al, 2020; Jang et al, 2019]. Its is also the "mathematical definition of the sequential action decision problem" [Jang et al, 2019]. In the Markov Decision process, the choice of actions that can be performed at a certain state is determined by rules which are called as policies [Jang et al, 2019; Even-Dar and Mansour, 2001].

A.1) State

In a given stochastic environment/world, an agent, by performing an action, can reach or visit certain states, positions or places and these states, positions or places is known as a state [Naeem et al, 2020]. For example, in a game of chess, a set of states for an agent (i.e., a chess piece) is the tile/tiles to which it can move to.

A.2) Action

In reinforcement algorithms, an action can be defined as " anything which an agent or a robot can imagine or allowed to do in an environment" [Naeem et al, 2020]. For example, in a game of chess, a set of actions for an agent (i.e., a chess piece) is the possible movements to different tiles it can make.

A.3) State transition probability matrix

The state transition probability represents what will happen when we or an agent move from a state S to a state S´ by performing an action A and it is represented by a value in the form of numbers [Jang et al, 2019; Naeem et al, 2020]. In the Markov decision Process, this probability does not depend on past states or actions and is completely determined by the current state and actions [Jang et al, 2019; Naeem et al, 2020]. "The probability is:

$$P^{a}_{SS'} = P\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$$

where $P^{a}_{SS'}$ is the probability contained in the matrix P of moving to state s´ when action a is performed in state s, and t denotes the time" [Jang et al, 2019].

A.4) Reward

A reward is the feedback that the agent receives from the environment after it moves to a state by performing an action and it is represented by a numerical value [Jang et al, 2019; Naeem et al, 2020]. The agent then can use this information to learn [Jang et al, 2019]. "When the state is s and the action is a at time t, the reward that the agent receives is:

$$R^{a}_{SS'} = E\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

where $R^{a}_{SS'}$ is the definition of the reward function. t is the time, and E is the expected value for the reward to be given as action a occurs when it moves from a state to s´" [Jang et al, 2019]. The reason why we represent reward in terms of an expected value is because depending on the environment, the same action taken in the same state can result in a different feedback from the environment [Jang et al, 2019]. Also, the reason we represent the reward to be received by the agent as R_{t+1} is because when an agent moves from state S to state S´ by performing an action A,

the environment will only inform the agent of the reward it will receive at time t + 1 [Jang et al, 2019].

A.5) Discount Factor

Discount factor was a concept that was introduced as a response to discovery of the problem of value of the reward decreasing over time [Jang et al, 2019]. Discount factor is a value between 0 and 1 and "it defines how much the agent has to take into account long-run rewards instead of immediate ones" [Spano et al, 2019].

A.6) Policy

A policy can be defined as a rule by which an agent can select the actions it can perform [Jang et al, 2019]. The policy is

$$\text{“ } \pi (a|s) = P[A_t = a| S_t = s]$$

where π is the probability of policy that the agent chooses a in state at time t" [Jang et al, 2019]. Eventually, through learning better policies through trial and error, the reinforcement learning algorithm will reach the optimal policy [Jang et al, 2019].

B) Value Function

In order for the reinforcement learning algorithms to learn and find the optimal policy, it needs to choose better policies, and this is done so with help of the value function which gives the total reward that is expected to be received when we follow a certain policy in a specific state [Jang et al, 2019]. The value function is "as follows:

$$v_\pi (s) = E_\pi [R_{t+1} + \gamma\, v_\pi (S_{t+1})|S_t = s)$$

where the expectation equation V_π (s) is the expected value E_π, R_{t+1} is the reward value to be

awarded next and l is the discount factor" [Jang et al, 2019]. The above equation helps the agent to choose the better state based on a given state s and policy π and hence is known as a state value function [Jang et al, 2019; Naeem et al, 2020]. Now, an agent also needs to use choose the better action at a state and it does so with the help of the Q- function which is given as :

"$q_\pi(s, a)=E_\pi[R_{t+1}+\gamma\, q_\pi(S_{t+1}, A_{t+1}|S_t =s, A_{t+1} =a)$" [Jang et al, 2019].

C) Bellman Equation

C.1) Bellman Expectation Equation

The value function gives us the total reward that the agent is expected to receive when it moves from a state S to and state S´ by performing an action A under a certain policy and the bellman equation is used to show us the relationship between the value function of state S and the value function of state S´ under that policy [Jang et al, 2019]. The Bellman expectation equation is given by

$$"v_{\pi'}(s) = \sum_{a\epsilon A} \pi(a|s)(R_{t+1} + \gamma \sum_{s'\epsilon S} P^{a}_{ss'} v_\pi(s'))"$$

where "$\sum_{a\epsilon A} \pi(a|s)$is the probability policy to do the action" and "$\sum_{s'\epsilon S} P^{a}_{ss'}$is the state transition probability matrix" [Jang et al, 2019]. Also, just as in previous equations, "R_{t+1} is the reward and γ is the discount factor" [Jang et al, 2019].

C.2) Bellman Optimality Equation

Reinforcement learning algorithms is used learn the policy, determined by the value function, which gives us the greatest expected reward (i.e., the optimal policy) in a sequential action decision problem which is defined by the Markov Decision Process and this policy, which

receives the best possible value with the help of the value function is known as a Bellman optimality equation [Jang et al, 2019]. The Bellman optimality equation is as follows:

$$\text{“} \qquad v_*(s) = \max_a E_\pi \left[R_{t+1} + \gamma\, v_*(S_{t+1}) | S_t = s\right]$$

where $\max_a E_\pi$ he maximum expected value among the policies that agents can receive” [Jang et al, 2019].

2.3) From reinforcement learning to Q-learning

A) Monte Carlo methods

Monte Carlo methods are methods learns through trial and error (i.e., model free) and it learns after the agent reaches the terminal state i.e., the end of the episode by the using the concept of average return [Naeem et al, 2020]. It also uses the concept of randomness which means that the start state of the problems the methods is used on should always be random [Naeem et al, 2020]. The Monte Carlo method can be divided into two types of approaches with the first being the First-visit Monte Carlo methods, which as the name implies gives the mean results based on the “first visit to a state s over a period of episodes” and the second being the Every-visit Monte Carlo method, which as the name implies gives the mean results based on “all visits to a state s” [Naeem et al, 2020]. An advantage of Monte Carlo methods is that it will always find the optimal policy even with the lack of a model but, unfortunately, a major drawback with the Monte Carlo methods is that since it learns only at the end of an episode, it only works if there is a terminal state and also we have to wait to wait till the agent reaches that terminal state in order to update the value function [Naeem et al, 2020]. The solution that was made to overcome this drawback was to update the value function after each step, which is the basic concept behind temporal difference algorithm [Naeem et al, 2020].

B) Temporal Difference Methods

As we have seen from the previous section, temporal difference methods are made to update the value function after each step and hence removes the limitation of requiring a terminal state (i.e., must be a complete sequence) and needing to wait until its completed to update the value function [Naeem et al, 2020]. The temporal methods all follow a basic pattern which is as follows:

$$\text{New estimate} \leftarrow \text{Old estimate} + \alpha[\text{Target} - \text{Old estimate}]$$

where α is the learning rate (value between 0 and 1), target is the actual value and the difference between the target and the old estimate gives the estimation error δ [Naeem et al, 2020]. Our aim while using temporal difference methods is to reduce the estimation error δ as much as possible [Naeem et al, 2020]. If we got the estimation error δ to be equal to 0, it would mean that there is no difference between the target and the old estimate i.e., the old estimate is the actual value. The basic value function and equation of the calculation of target is as follows:

" $V(s) = \mathbb{E} \sum_{t=0 \to \infty} \gamma^t R(S_t)$

$\text{Target} = \mathbb{E} \sum_{t=0 \to \infty} \gamma^t R(S_t)$ " [Naeem et al, 2020]

In temporal difference learning we have to update the value function after each step, However, we do not information about all the rewards or the values before hand (except for the rewards at t + 1), so we use the old estimates to calculate the new estimates. This concept is known as bootstrapping [Naeem et al, 2020]. The calculation of the target is then as follows :

" $Target = \mathbb{E}\,[r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \ldots. \gamma^{k-1} r_{t+k+1})]$

$Target = \mathbb{E}\,[r_{t+1} + \gamma U(s_{t+1})]$ " [Naeem et al, 2020]

Based on the upper equations, we can right the final update as follows :

“ $$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \quad (1)$$

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$ ” [Naeem et al, 2020]

From the above two equations, we can rewrite the value function equation as follows

“ $$V(s_t) = V(s_t) + \alpha\delta_t e_t(s_t)$$ ” [Naeem et al, 2020]

In the temporal difference methods, we need to keep track of the set of states that have been visited previously and it is done so with the help of a short-term memory technique know as eligibility traces [Naeem et al, 2020]. The eligibility traces can be defined using the following equations:

“ $e_t(s) = \gamma\lambda et - 1(s)$ if $s \neq s_t$

$\gamma\lambda et - 1(s) + 1$ if $s = s_t$ ”

where λ (which is a value between 0 and 1) is the decay parameter and can be used to determine in what manner that the states are updated [Naeem et al, 2020]. The temporal difference method combines the techniques of both dynamic programming and Monte Carlo methods, which are used to solve the bellman equations [Naeem et al, 2020; Jang et al, 2019]

C) Q-learning

Q-learning is one of the famous temporal difference learning algorithms which follows an off-policy approach i.e., it learns by observation, to find an optimal policy [Naeem et al, 2020]. The update rule for q-learning is as follows :

“ $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \sigma[r_{t+1} + \gamma \max_a Q(s_t, a_t)]$$ ” [Naeem et al, 2020].

Q-learning algorithms select future actions which has the greatest q-value of the next state i.e., it follows a greedy policy to choose an action until it reaches convergence [Naeem et al, 2020]. Since q-learning algorithm is a temporal difference algorithm, it will have low variance and some bias unlike Monte Carlo methods which have high variance and no bias [Naeem et al, 2020].

3. Applications/Advancements of Q-learning algorithm

One of the fields in which q-learning algorithms have been successful on is the field of artificial intelligence. One of the technologies in which q-learning has been implemented was in quadrotor control [Jang et al, 2019]. Quadrotor is a technology which has had an increase in research for commercial by major companies like Google and Amazon and it has been used for different purposes such as unmanned mail delivery, navigation, surveillance etc. [Jang et al, 2019]. Many different techniques for the control of quadrotors have been studied and developed by using q-learning algorithms as an algorithm for the control [Jang et al, 2019]. One of the major advancements in the field of quadrotors has the development of the drone technology in which the part played by q-learning algorithms has been immense [Jang et al, 2019]. Other areas of artificial intelligence in which q-learning has played a major role have been in image classification, pattern recognition, natural language processing etc. [Jang et al, 2019]. Outside of artificial intelligence, q-learning algorithms and its variations have also had a major impact in the fields of robotics, computer networking, control of industrial processes, operations research and game theory [Jang et al, 2019].

4. Conclusion

Q-learning algorithm is a model free, off policy temporal difference learning algorithm that iterates through various states and actions for these in order to find the optimal policy [Jang et al, 2019; Watkins and Dayan, 1992]. In this paper, we have seen the basic processes present in any reinforcement algorithm such as the Markov Decision Process, value function and Bellman equations [Jang et al, 2019]. This paper has also shown how q-learning algorithm is a temporal difference learning algorithm and it combine various aspects of both dynamic programming and Monte Carlo methods [Naeem et al, 2020; Jang et al, 2019]. Towards the end of the paper, it was shown the variety of fields in which the q-learning algorithms were employed [Jang et al, 2019]. All in all, through this research, it is clearly shown that in the future, q-learning algorithms and its variations will without a doubt become an indispensable part of future technologies.

References

- [Watkins and Dayan, 1992] Watkins, C. J. C. . & Dayan, P. (1992) Technical Note: Q-Learning. Machine learning. [Online] 8 (3), 279–.
- [Jang et al, 2019] Jang, B. et al. (2019) Q-learning Algorithms: A Comprehensive Classification and Applications. IEEE access. [Online] 71–1.
- [Spano et al, 2019] Spano, S. et al. (2019) An Efficient Hardware Implementation of Reinforcement Learning: The Q-Learning Algorithm. IEEE access. [Online] 7186340–186351.
- [Naeem et al, 2020] Naeem, M. et al. (2020) A Gentle Introduction to Reinforcement Learning and its Application in Different Fields. IEEE access. [Online] 8209320–209344.
- [Even-Dar and Mansour, 2001] Even-Dar, E. & Mansour, Y. (2001) 'Learning Rates for Q-Learning', in Computational Learning Theory. [Online]. 2001 Berlin, Heidelberg: Springer Berlin Heidelberg. pp. 589–604.