**Experiment :- 06**

|  |
| --- |
| **Title:**  **Write a program to implement Q-Learning algorithm** |

# Objective:

# Books/ Journals/ Websites referred:

* https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/
* https://www.analyticsvidhya.com/blog/2021/04/q-learning-algorithm-with-step-by-step-implementation-using-python/

# Resources used:

# Python,

# Numpy,

# Gym (Gym is a toolkit for developing and comparing reinforcement learning algorithms.)

# Theory:

# Reinforcement Learning briefly is a paradigm of Learning Process in which a learning agent learns, overtime, to behave optimally in a certain environment by interacting continuously in the environment. The agent during its course of learning experience various different situations in the environment it is in. These are called states. The agent while being in that state may choose from a set of allowable actions which may fetch different rewards(or penalties). The learning agent overtime learns to maximize these rewards so as to behave optimally at any given state it is in. Q-Learning is a basic form of Reinforcement Learning which uses Q-values (also called action values) to iteratively improve the behavior of the learning agent.

# 1) Q-Values or Action-Values: Q-values are defined for states and actions. Q(S, A) is an estimation of how good is it to take the action A at the state S . This estimation of Q(S, A) will be iteratively computed using the TD- Update rule which we will see in the upcoming sections.

# 2)Rewards and Episodes: An agent over the course of its lifetime starts from a start state, makes a number of transitions from its current state to a next state based on its choice of action and also the environment the agent is interacting in. At every step of transition, the agent from a state takes an action, observes a reward from the environment, and then transits to another state. If at any point of time the agent ends up in one of the terminating states that means there are no further transition possible. This is said to be the completion of an episode.

# 3) Temporal Difference or TD-Update: The Temporal Difference or TD-Update rule can be represented as follows : This update rule to estimate the value of Q is applied at every time step of the agents interaction with the environment. The terms used are explained below. :

# S : Current State of the agent.

# A : Current Action Picked according to some policy.

# S' : Next State where the agent ends up.

# A' : Next best action to be picked using current Q-value estimation, i.e. pick the action with the maximum Q-value in the next state.

# R : Current Reward observed from the environment in Response of current action.

# $\gamma$ (>0 and <=1) : Discounting Factor for Future Rewards. Future rewards are less valuable than current rewards so they must be discounted. Since Q-value is an estimation of expected rewards from a state, discounting rule applies here as well.

# $\alpha$ : Step length taken to update the estimation of Q(S, A).

# 4)Choosing the Action to take using

# -greedy policy: $\epsilon$ -greedy policy of is a very simple policy of choosing actions using the current Q-value estimations. It goes as follows :

# 1) With probability (1-$\epsilon$) choose the action which has the highest Q-value.

# 2) With probability ($\epsilon$) choose any action at random.

# Pros:

# 1) Long-term outcomes, which are exceedingly challenging to accomplish, are best achieved with this strategy.

# 2) This learning paradigm closely resembles how people learn. Consequently, it is almost ideal.

# 3) The model has the ability to fix mistakes made during training.

# 4) Once a model has fixed a mistake, there is virtually little probability that it will happen again.

# 5) It can produce the ideal model to address a certain issue.

# Cons:

# 1) drawback of using actual samples. Think about the situation of robot learning, for instance. The hardware for robots is typically quite expensive, subject to deterioration, and in 2) need of meticulous upkeep. The expense of fixing a robot system is high.

# 3) Instead of abandoning reinforcement learning altogether, we can combine it with other techniques to alleviate many of its difficulties. Deep learning and reinforcement learning are one common combo.

# Implementation (Code):

# Kindly find the code and output in E6\_Q\_learning.ipynb

# Output Screenshots:

# Conclusion (Students should write in their own words):

# Q-learning is a reinforcement learning algorithm that aims to find an optimal policy for an agent in a Markov Decision Process (MDP). It is an off-policy algorithm, which means that it learns the value of the optimal policy, even if it is not the policy currently being followed.

# Q-learning works by estimating the Q-values (the expected cumulative reward) of each state-action pair in the MDP. It updates these Q-values using the reward received by the agent and the maximum Q-value of the next state. The algorithm uses an epsilon-greedy policy to balance exploration and exploitation.

# Q-learning has several advantages, such as its ability to converge to the optimal policy with a high degree of accuracy and its ability to handle large and complex MDPs. It is also relatively simple to implement and can be applied to a wide range of problems.

# However, Q-learning also has some limitations. It can be slow to converge in some cases, particularly in large or complex MDPs. It also requires a significant amount of computational resources and can be sensitive to the choice of hyperparameters.

# Overall, Q-learning is a powerful algorithm for learning optimal policies in reinforcement learning problems, particularly in large and complex MDPs. Its effectiveness can be improved by combining it with other techniques such as function approximation, eligibility traces, or experience replay.

# Applications:

# Q-learning (a type of reinforcement learning) has several applications in different fields, including:

# 1) Robotics: Q-learning can be used to teach robots to perform specific tasks or navigate through environments. For example, Q-learning can be used to train robots to autonomously explore an environment, avoid obstacles, and find objects.

# 2) Game AI: Q-learning can be used to create intelligent agents that can play games such as chess, checkers, or poker. By learning from experience, these agents can develop strategies to win against human players or other agents.

# 3) Control systems: Q-learning can be used to optimize control systems such as power grids, traffic control systems, or HVAC systems. By learning from feedback and adapting to changing conditions, these systems can be made more efficient and reliable.

# 4) Finance: Q-learning can be used to develop trading strategies for financial markets. By learning from historical data and adapting to changing market conditions, these strategies can be optimized to maximize profits and minimize risk.

# 5) Healthcare: Q-learning can be used to optimize treatment plans for patients with chronic diseases such as diabetes or hypertension. By learning from patient data and adapting to individual needs, these plans can be personalized and made more effective.