CSE 546

Reinforcement Learning

Assignment 1

Author: Sarveshwar Singhal(50418642)

Date of Submission: March 7, 2022

Introduction:

In this project I’ve defined two environments deterministic and stochastic.

Common parameters in both environment:

1. No. of timestamps = 4 \* no\_of\_states\_in\_env
2. Env Size: (4\*4) grid. There were in total 16 states defined as follows: {(0,0),(1,0),(2,0),(3,0),(0,1),(1,1),(2,1),(3,1),(0,2),(1,2),(2,2),(3,2),(0,3),(1,3),(2,3),(3,3)}
3. I’ve defined four set of actions: S: {Down, Up, Right, Left}
4. Following rewards were defined: {0,1,2,3,1,2,3,4,2,3,4,5,3,4,5,100}

**Deterministic environment**: These are those type of environments in which if an agent takes a step/move, the agent will definitely end up in that state.

The main objective of defining the deterministic environment is to train our agent in the ideal scenario. Generally deterministic environments are less complex as compared to a stochastic environment. When training our agent, agent training takes less time/computation in a deterministic environment as compared to stochastic (keeping the environment outline same)

If an agent is at state s, and takes an action a, if these two are valid combinations then agent position will be updated otherwise agent will remain at the same position.

If the agent is at state s and takes action a and reaches current position (defined below in table) then the agent will get the corresponding reward.

|  |  |  |
| --- | --- | --- |
| Previous state | Current Position | Reward |
| Any\* | (0,0) | 0 |
| Any\* | (1,0) | 0.5 |
| Any\* | (2,0) | 1 |
| Any\* | (3,0) | 1.5 |
| Any\* | (0,1) | 0.5 |
| Any\* | (1,1) | 1 |
| Any\* | (2,1) | 1.5 |
| Any\* | (3,1) | 2 |
| Any\* | (0,2) | 1 |
| Any\* | (1,2) | 1.5 |
| Any\* | (2,2) | 2 |
| Any\* | (3,2) | 2.5 |
| Any\* | (0,3) | 1.5 |
| Any\* | (1,3) | 2 |
| Any\* | (2,3) | 2.5 |
| Any\* | (3,3) | 3 |

Any\* describes the state s (from all possible 16 states)

**Stochastic Environment**: These are those type of environments in which if an agent takes a step/move, the agent may or may not end up in the desired state, the environment may move the agent in a different state as well.

The main objective of this type of environment is to train our agent on real world scenarios.

The Stochasticity in this environment is: if agent is at position (0,0), (1,1) … (i.i) then environment generates random probability (for these position only), if the probability is less than 10% then the agent will remain at the same position, otherwise the environment will allow the agent to move normally.

If the agent is at state s and takes action a and reaches current position (defined below in table) then the agent will get the corresponding reward.

|  |  |  |
| --- | --- | --- |
| Previous state | Current Position | Reward |
| Any\* | (0,0) | 0 |
| Any\* | (1,0) | 0.5 |
| Any\* | (2,0) | 1 |
| Any\* | (3,0) | 1.5 |
| Any\* | (0,1) | 0.5 |
| Any\* | (1,1) | 1 |
| Any\* | (2,1) | 1.5 |
| Any\* | (3,1) | 2 |
| Any\* | (0,2) | 1 |
| Any\* | (1,2) | 1.5 |
| Any\* | (2,2) | 2 |
| Any\* | (3,2) | 2.5 |
| Any\* | (0,3) | 1.5 |
| Any\* | (1,3) | 2 |
| Any\* | (2,3) | 2.5 |
| Any\* | (3,3) | 3 |

Any\* describes the state s (from all possible 16 states)

**Visualization of Environment**:

Deterministic Environment



Stochastic Environment:



**Stochastic environment** was defined based on the randomness in few states of the environment.

The Stochasticity in this environment is: if agent is at position (0,0), (1,1) … (i.i) then environment generates random probability (for these position only), if the probability is less than 10% then the agent will remain at the same position, otherwise the environment will allow the agent to move normally.

**Safety in AI**:

Safety in AI is a major concern, especially when dealing with reinforcement learning agents (because in reinforcement learning we have to deal with uncertainty and new situations). We had to make sure that actions of our agent doesn’t tamper/damages the original environment or it doesn’t create any harm in real world.

In this environment we made sure the agent doesn’t go beyond the specified boundary limits, so by doing this we made sure that our agent is well within our limits and was training in a limited environment, so that it doesn’t harm the external world.

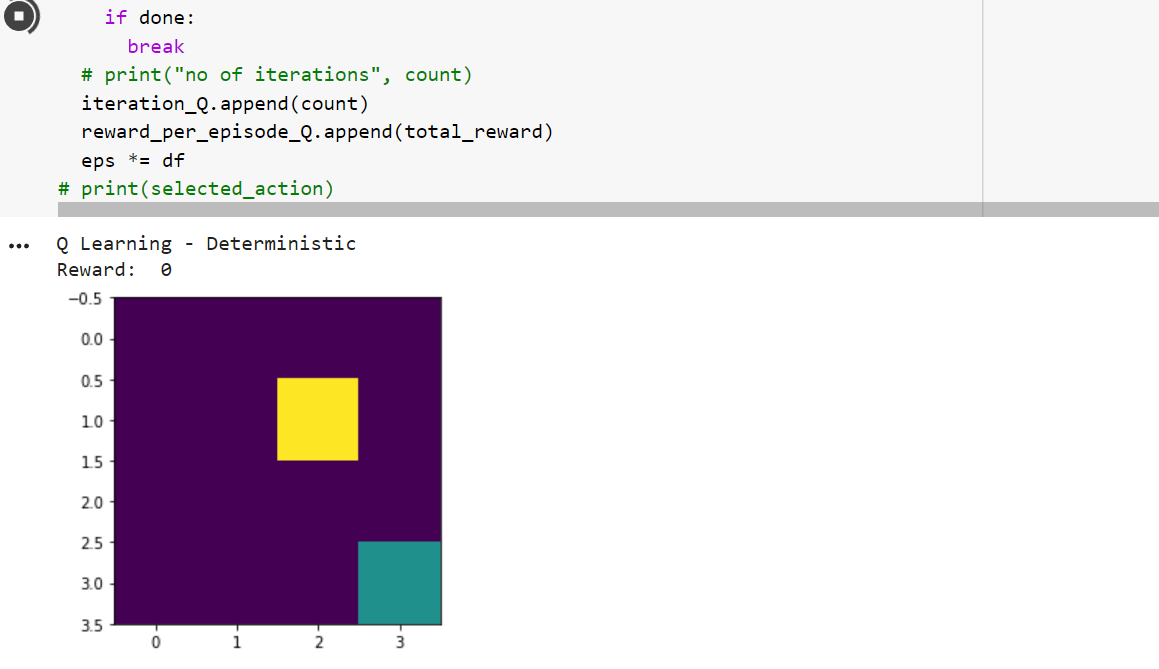
Phase 2 Report:

Intro:

As discussed previously in checkpoint 1, one variable ‘mode’ was defined in the environment, mode=0 means environment will behave deterministically and mode=1 means environment will behave stochastically.

Deterministic property of the environment was updated, previously certain states were defined as deterministic, now the deterministic property was extended to the entire environment. And Rewards were also updated so that algorithm convergence can be fast.

Applying Q-Learning to solve the deterministic environment



The above image shows one of timestamps during training of our agent

Graphical user interface, text

Description automatically generated

After Q-Learning algorithm converged, we can see as the episodes increased the no of steps starts converging towards 6. The agent actions were chosen based on epsilon greedy method. In few cases there were more than 6 iterations due to the gama. We have not completely reduced the gama to 0, due to which even after so many episodes the agent has little tendency to explore the environment.

As the no of episodes increased the agent actions changed from exploration to exploitation.

Graphical user interface, text, application

Description automatically generated

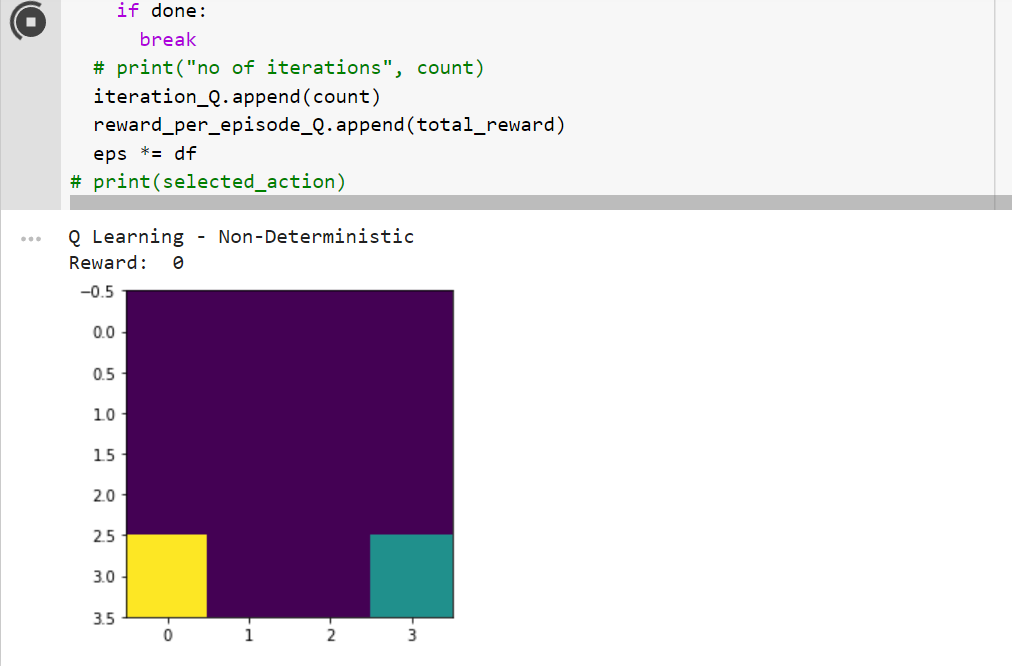
The above graph shows the epsilon decay graph, as the episodes increased epsilon decayed slowly but didn’t touch 0 (the bottom limit was 0.01).

Graphical user interface, text, application

Description automatically generated

We can see as the episodes increased, our agent was following the greedy policy, and due to exploitation the total rewards per episodes became constant (basically max reward in min steps).

**Applying Q-Learning to solve the Non-Deterministic environment**



The above image shows one of steps during training of our agent

Graphical user interface, text, application

Description automatically generated

After Q-Learning algorithm converged, we can see as the episodes increased the no of steps starts converging towards 6-8. Agent action was chosen based on epsilon greedy algorithm. In few cases there were more than 6 steps due to the stochasticity and gama. We have not completely reduced the gama to 0, due to this even after so many episodes the agent has little tendency to explore the environment.

As the no of episodes increased the agent actions changed from exploration to exploitation.

The below graph shows the last 200 episodes of the training

Graphical user interface, text, application

Description automatically generated

So we can see, in the last 200 episodes, no of steps were majorly between 6-8 (which is close to optimal policy, optimal steps were 6)

Graphical user interface, text, application

Description automatically generated

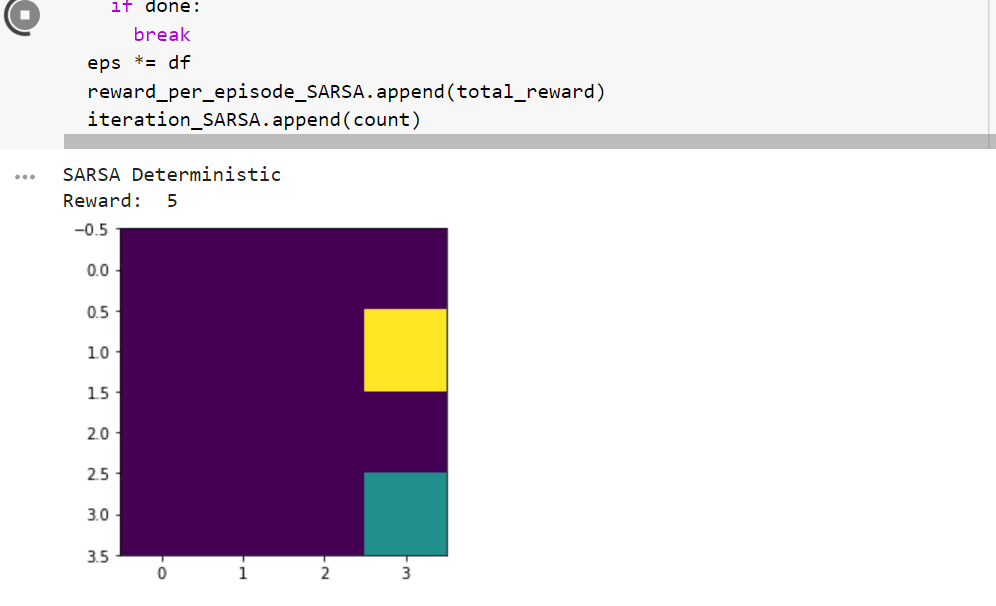
The above graph shows the epsilon decay graph, as the episodes increased epsilon was decayed slowly.

Graphical user interface, text, application

Description automatically generated

We can see as the episodes increased, our agent was following the greedy policy, and due to exploitation the total rewards per episodes became constant (basically max reward in min steps).

**Applying SARSA Algorithm to solve the Deterministic environment**



The above image shows one of timestamps during training of our agent

Graphical user interface, text, application, email

Description automatically generated

After SARSA algorithm converged, we can see as the episodes increased the no of steps starts converging towards 6. The agent actions were chosen based on epsilon greedy method. In few cases there were more than 6 iterations due to the gama. We have not completely reduced the gama to 0, due to which even after so many episodes the agent has little tendency to explore the environment.

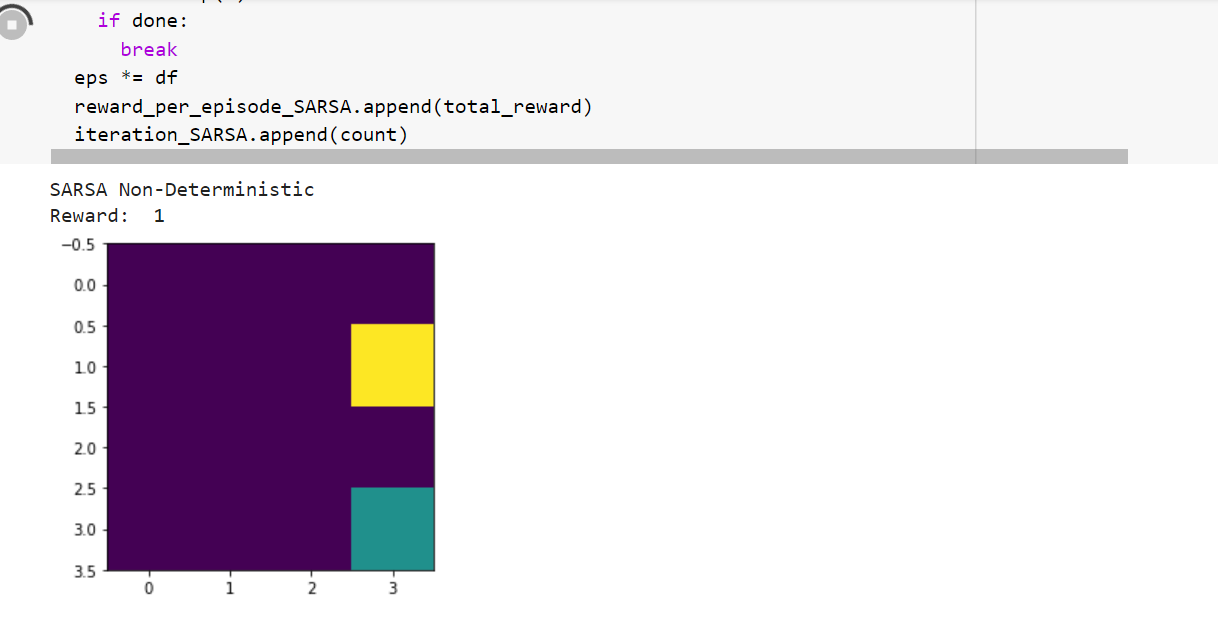
As the no of episodes increased the agent actions changed from exploration to exploitation.

Graphical user interface, text, application, email

Description automatically generated

We can see as the episodes increased, our agent was following the greedy policy, and due to exploitation the total rewards per episodes became constant (basically max reward in min steps).

**Applying SARSA Algorithm to solve the Non-Deterministic environment**



The above image shows one of steps during training of our agent

Graphical user interface, text

Description automatically generated

After SARSA Algorithm converged, we can see as the episodes increased the no of steps starts converging towards 6-8. Agent action was chosen based on epsilon greedy algorithm. In few cases there were more than 6 steps due to the stochasticity and gama. We have not completely reduced the gama to 0, due to this even after so many episodes the agent has little tendency to explore the environment.

As the no of episodes increased the agent actions changed from exploration to exploitation.

The below graph shows the last 200 episodes of the training

Graphical user interface, text, application

Description automatically generated

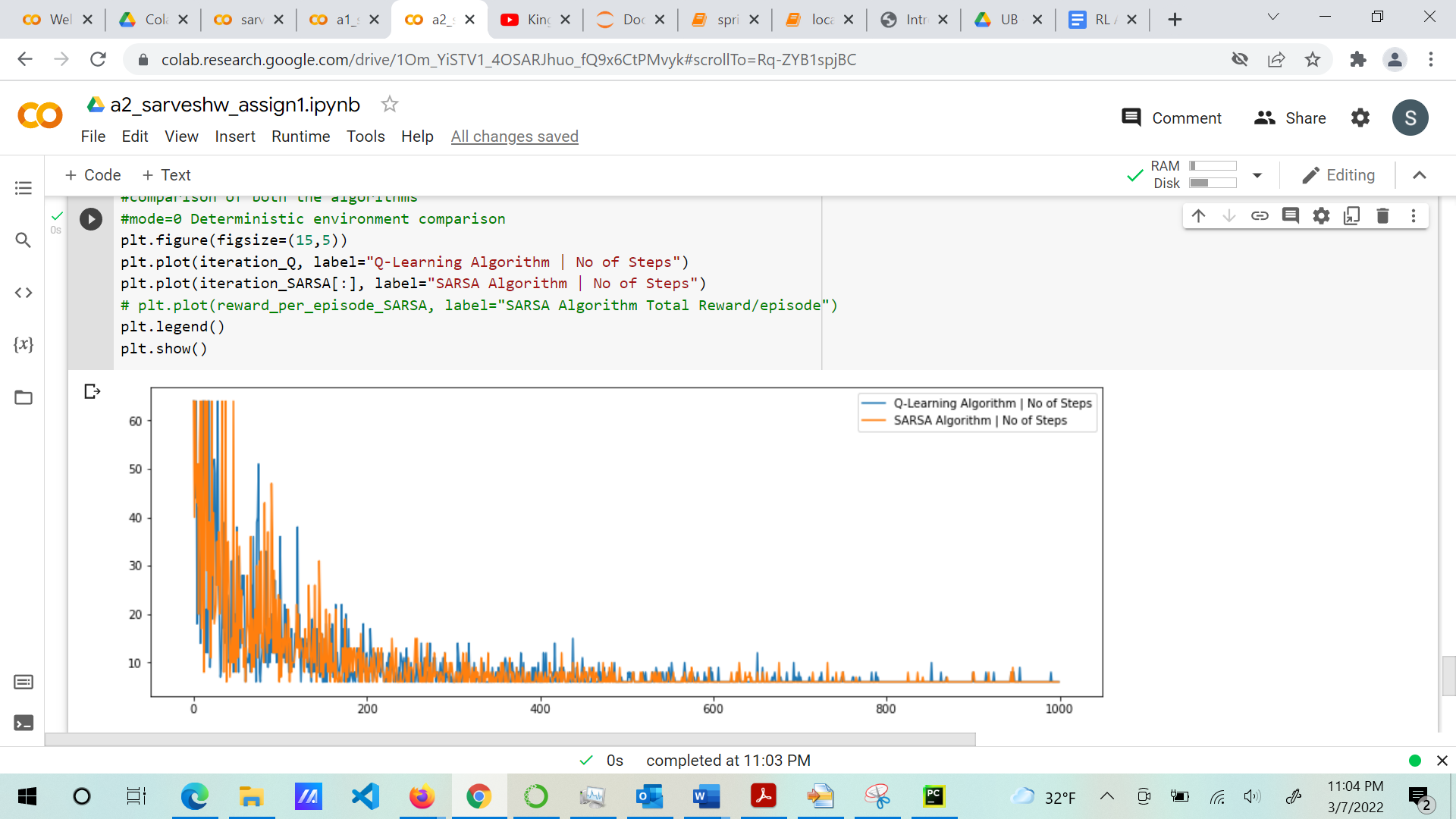
So we can see, in the last 200 episodes, no of steps were majorly between 6-8 (which is close to optimal policy, optimal steps were 6)

Graphical user interface, text, application, email

Description automatically generated

We can see as the episodes increased, our agent was following the greedy policy, and due to exploitation the total rewards per episodes became constant (basically max reward in min steps).

**Comparison of both algorithms in Deterministic Environment:**



Based on the above graph, given the environment reward and properties, in first look it’s hard to say which one performed better. But we can deduce some conclusions: we can say that SARSA is converging faster or requires less steps till 600 episodes and then after both the algorithms performed similar or we can say SARSA still has an edge over Q-Learning.

**Comparison of both algorithms in Non-Deterministic Environment:**

Graphical user interface, text, application

Description automatically generated

Again, based on the above graph, given the environment reward and properties, in first look it’s hard to say which one performed better. But we can say that SARSA is converging faster or requires less steps throughout the training. Or we can say SARSA still has an edge over Q-Learning.

Final Q table for Q-Learning

Graphical user interface, text, application

Description automatically generated

Final Q table for SARSA

Graphical user interface, text, application

Description automatically generated

**Tabular Methods Used**:

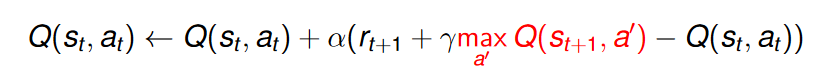
* Q-Learning Algorithm
* SARSA Algorithm

In Reinforcement learning we have various tabular methods to train our agent in an environment. These methods work on current rewards, future reward, learning rate, gama etc.

Q-Learning Algorithm:

Q-learning is a model-free algorithm to learn the value of an action in a particular state. It does not require a model of the environment (hence "model-free"),

The goal of Q-learning is to learn iteratively the optimal Q-value function using the Bellman equation. To do this we store all the Q value in a Q table and then update the new values in the same table against the state-action pair.



Here Q(s,a): is the current state Q value.

Alpha: learning rate

Gama: discount factor

Max a’ (Q(S(t+1), a’)): is the maximum Q value in the next state.

SARSA Algorithm

SARSA is an on-policy algorithm, where, in the current state S, action A is taken and then agent gets a reward R and end up in the state S1 and takes action A1 in S1. Therefore, it’s a sequence of (S, A, R, S1, A1) thus the acronyms make it SARSA algorithm.

In this algorithm, from one state we go to next state estimate the action value in that state and then come back to original state and update the original state value in Q table.



Here Q(s,a): current state

Alpha: learning rate

Gama: discount factor

Q(s’,a’): future state-action value

SARSA is same as Q-Learning algorithm, except in the SARSA action are chosen on-policy whereas in Q-learning max reward action is chosen.