ADHITHYAKRISHNA KOVAI SRINIVASAN

Department of computer science, University at Buffalo

UBIT NUMBER: 50317495

akovaisr@buffalo.edu

ABSTRACT

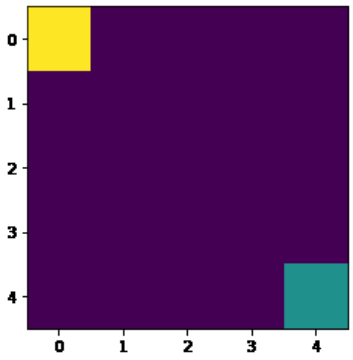
This project focuses on implementing a reinforcement learning methodology called Q-learning on 4X4 grid world environment. The grid navigates from a start point to end point avoiding obstacles.

INTRODUCTION

Reinforcement learning is the process of learning by interacting with an environment through positive feedback. Qlearning is a type of reinforcement learning that minimizes the behavior of a system through trial and error. Qlearning updates its policy i.e. the mapping of state and action based on reward. Q-learning is modeled as Markov Decision process MDP. We make use of Open AI Gym environment.

ENVIRONMENT:

The basic environment for our project consists of size of 4X4 grid that is compatible with Open Ai Gym environments. The start block is initialized to the index [0,0] and the destination is at [4,4]. It is expected that the steps taken to reach the final destination should be the least. A reward is assigned for whatever steps the agent takes. It will receive a reward of +1 for moving closer to the destination and -1 reward for moving away from the destination. The model is trained in such a way that the agent would be able to reach the destination within fair number of epochs. The start block can move in 4 direction top, bottom left and right.



Yellow block representing the start block and green representing the destination in a 4x4 grid environment

ARCHITECTURE

4.1 QLEARNING

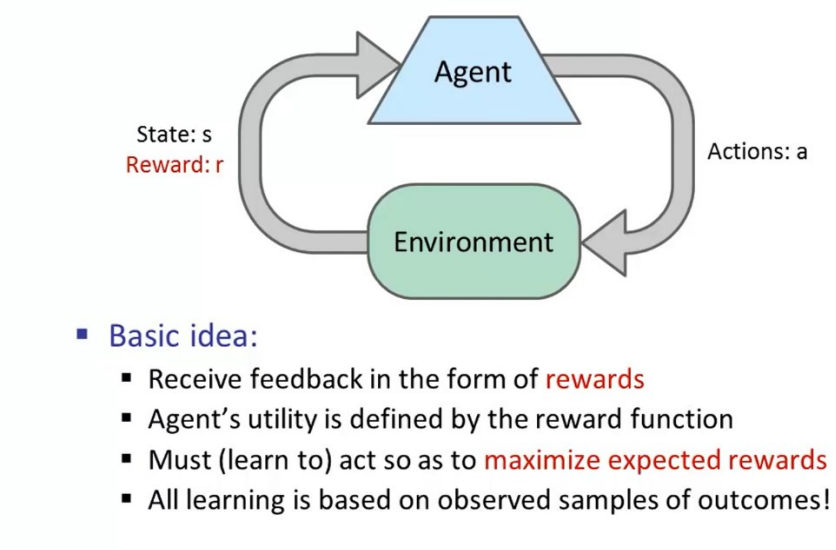


IMAGE REPRESENTING THE BASIC IDEA OF REINFORCEMENT LEARNING

Q-learning is a type of reinforcement learning that works on the basis of rewards. An agent can take an action at a given time stamp which might lead to a state transition 'S' and this change might in-turn lead to a part of the agent to change. A reward is then distributed to the agent and this process is repeated. The goal of the agent is to maximize the reward not just the immediate reward but also the cumulative reward.

4.2 PSEUDO CODE FOR Q-LEARNING

```
Initialize Q(s, a) arbitrarily
Repeat (for each episode):
   Initialize s
   Repeat (for each step of episode):
      Choose a from s using policy derived from Q
         (e.g., ε-greedy)
      Take action a, observe r, s'
      Q(s, a) <-- Q(s, a) + α [r + γ max_α,Q(s', a') - Q(s, a)]
      s <-- s';
   until s is terminal
```

Image representing the pseudo-code of Q-LEARNING

4.3 QLEARNING ALGORITHM

1) The Q-values tables are initialized to store **Q(s,a)** this is also called as state action pair. The Q-tables holds the values of states and their corresponding actions.
2) An action **a** for the steps are taken. The action selection policies include (ε -soft, ε -greedy or SoftMax).
3) The new rewards r as well as new states s' are observed. The cumulative reward received by the agent decides what decision it has to make next.
4) The Q-value is updated using the observed reward and the next possible maximum reward of the next state.
   The parameters used in Q-value update process are **α, γ** and $\mathbf{max_a}$**.** here **α** is the learning rate whose value lie between 0 and 1. **γ** is the discount factor and its values lie between 0 and 1 and these are used to give more weight to more immediate rewards. The discount rate will determine the present value of future reward. Finally, $\mathbf{max_a}$ which is the maximum reward that is attainable in the state following the present state.
5) A new state is set and the repeat the process (episodes) until a terminal state is reached.

4.4 EXPLORATION VS EXPLOITATION – FINDING THE OPTIMAL POLICY

The agent chooses to perform an action that has highest rewards in the Q-table. Since the Q-table values are initially set to 0 for first action in the first episode the agent might not be able to find the correct action. For this problem, initially, the agent would select its action by exploration since epsilon value is set to 1. An epsilon decay is introduced to decrease the exploration by the agent.

We have to consider the trade off between exploration and exploitation which will explain how an agent takes an action in general. Exploration is the act of exploring the environment to find information about it. Exploitation is to make use of information that is already known by the environment and maximize it. A balance of both exploration and exploitation to choose an action would be optimal to solve the problem using Q-learning.

Initially the epsilon value is set to 1 indicating that the agent first performs its action by performing exploration. As the agent learns more about the environment, the concept of **Epsilon greedy strategy,** takes place in which the epsilon decays to some rate we set so that exploration likelihood becomes less as the agent learns more about the environment. The agent will then start "exploiting" the environment (making use of data from the Q table to perform the next action).

The new Q value is updated to the Q-table after each episode is update using the formula

$$q^{new}(s,a) = (1-\alpha)\underbrace{q(s,a)}_{\text{old value}} + \alpha\overbrace{\left(R_{t+1} + \gamma \max_{a'} q(s',a')\right)}^{\text{learned value}}$$

Where **q(s,a) –** is the value taken from the Q-table. **α** is the learning rate and R_t+1 is the reward received from performing an action. **γ max$_a$** is the discounted estimate of the optimal key value for the next state action pair **(s',a').**

5 RESULT

5.1 ITERATION 1

LEARNING RATE	DECAY RATE	NO. OF EPISODES
0.001	0.01	800

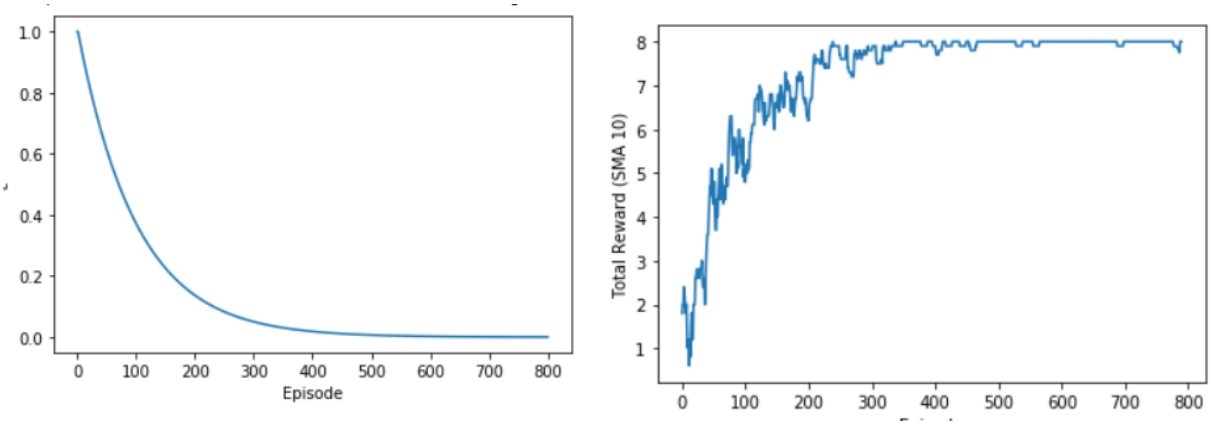
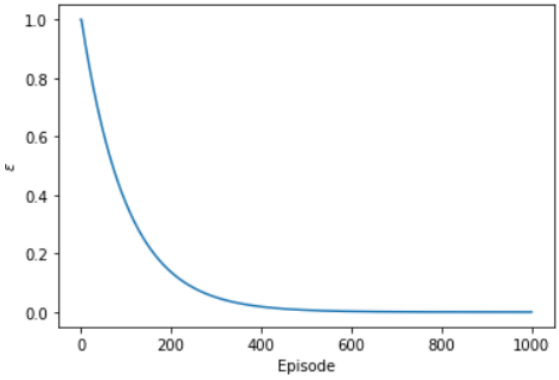
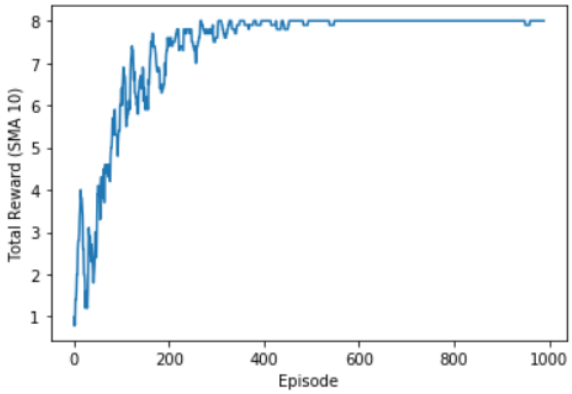


Image representing episodes vs epsilon value on the left. Image representing episode vs rewards on the right. We can observe from the image that the epsilon value decreases rapidly for each episode and the reward received by the agent is not consistent. Thus, we increase the learning rate to 0.01 and set the number of episodes to 1000.

5.2 ITERATION 2

LEARNING RATE	DECAY RATE	NO. OF EPISODES
0.01	0.01	1000

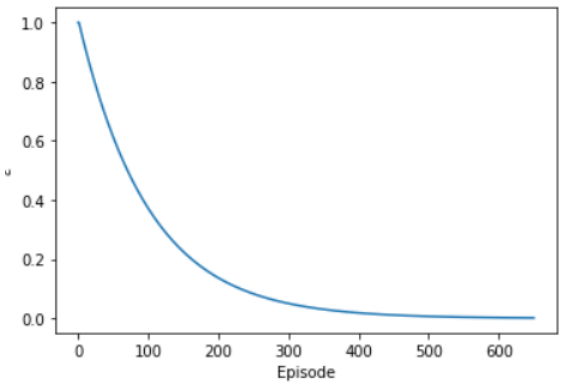


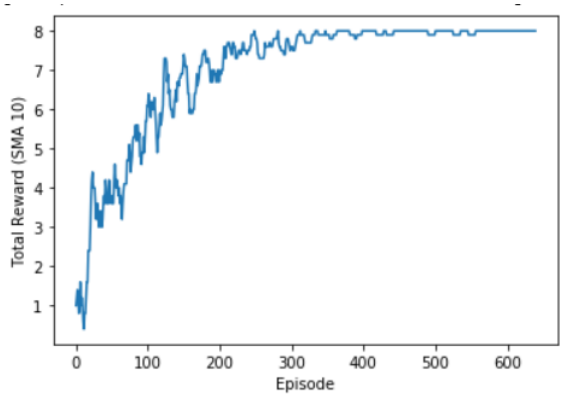


The epsilon value decreases consistently and reaches the minimum value at around 450th episode and also can observe the rewards reached its maximum during the 650'th episode. We try to reduce the number of episode and run it several times to check if the model is able to learn within 650 episodes.

5.3 ITERATION 3

LEARNING RATE	DECAY RATE	NO. OF EPISODES
0.01	0.01	650

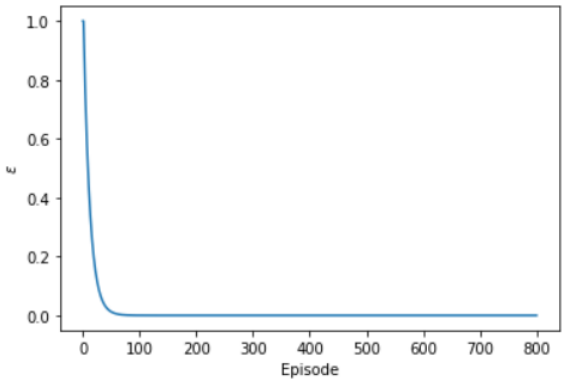


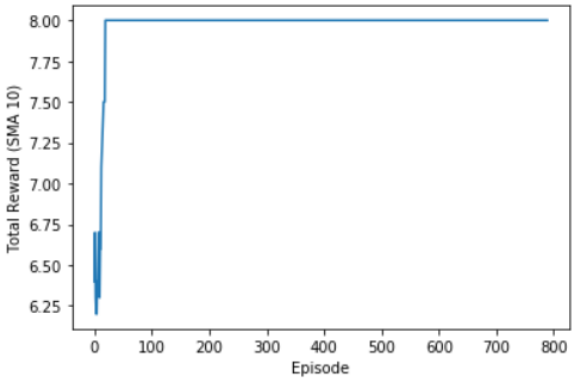


We can observe from the images that reducing the number of episodes from 1000 to 650 still lets model train and perform consistently. For the next observation the decay rate is increased to check if the agent is able to strike balance between exploration and exploitation as quick as possible.

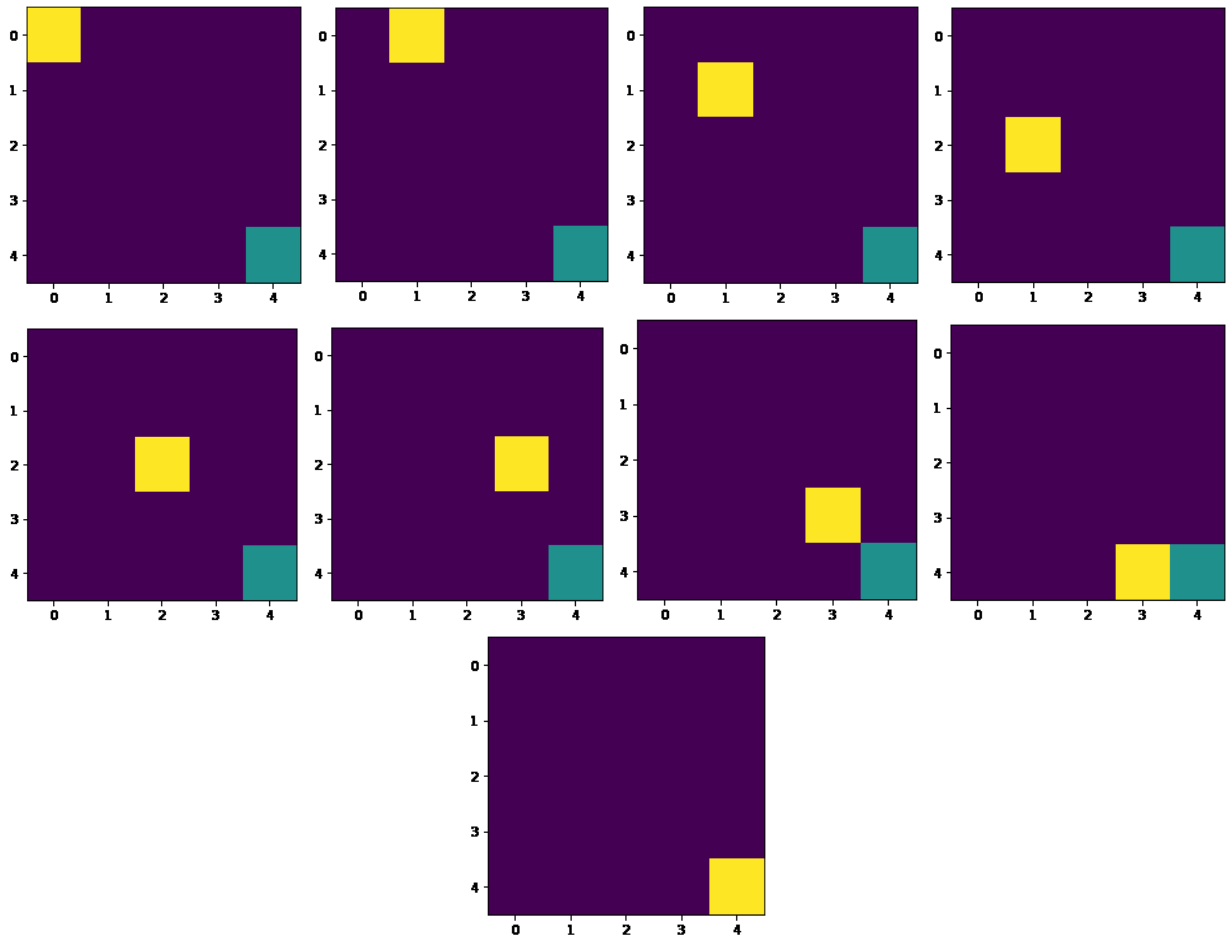
5.4 ITERATION 4

LEARNING RATE	DECAY RATE	NO. OF EPISODES
0.01	0.09	800





We can observe from the images that increasing the decay rate to 0.09 makes the agent to get maximum reward very quickly.



6 CONCLUSIONS

Thus, we were able to build a reinforcement learning agent that navigates a 4X4 grid environment. The greedy epsilon based policy that balances between exploration and exploitation is learned by agent and it is capable of navigating the environment to find the best path in least number of steps.

7 REFERENCES

- https://www.cse.unsw.edu.au/~cs9417ml/RL1/algorithms.html
- https://deeplizard.com/learn/video/HGeI30uATws