**Reinforcement learning to navigate a grid world**

# Aashna Mahajan

Department of Computer Science

University at Buffalo

Buffalo, NY 14214

[*aashnama@buffalo.edu*](mailto:aashnama@buffalo.edu)

# Abstract

The purpose of this project is to build a reinforcement learning agent to navigate the classic 4x4 grid-world environment. The agent learns an optimal policy through Q-Learning which allows it to reach the goal successfully, in an OpenAI Gym environment.

# 1 Introduction

## 1.1 Reinforcement Learning

Reinforcement learning is an area of machine learning which focuses on automating agents which can learn to take actions in response the states in the environment, in order to maximize the rewards. Reinforcement learning unlike supervised learning does not require any labelled input or output data. It also does not require any actions to be corrected explicitly. Instead it focuses on finding a balance between exploitation and exploration. RL is useful for game theory, control theory, multi-agent systems, etc.

## 1.2 Markov decision process (MDP)

It provides a mathematical framework for modelling decision making in various situations, where the outcomes could

vary. At each step, the process is in some state , and different actions can be chosen in the available state. In the next

step the process moves to a new state (s’) giving a reward R( s , s’ ).

This process of selecting an action from a given state, transitioning to a new state, and receiving a reward happens sequentially over and over again, which creates a trajectory that shows the sequences of states, actions and rewards. During this process the agent plans to maximize the total amount of received rewards (cumulative rewards not just the immediate rewards). The different components of an MDP are Agent, Environment, State, Action and Reward. An MDP is a 4-tuple (S; A; P; R), where S is the set of all possible states for the environment, State, Action and Reward.

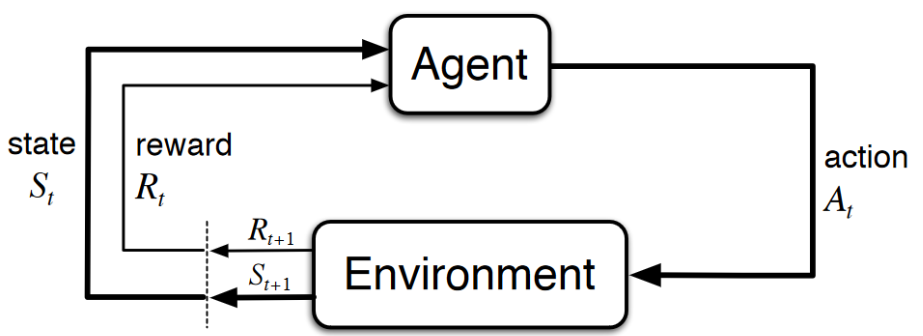


Figure 1: MDP Diagram

## 1.2.1 Policy

Policy is a function that maps a given state to probabilities of selecting each action from that state. At time ‘t’, under policy ‘π’, the probability of taking action ‘a’ in state ‘s’ is π(a|s). For each state s∈S, π is a probability distribution over a∈A(s).

## 1.2.2 Value functions

Value functions are functions of states, or state-action pairs, that help in estimate how good it is for an agent to be in a given state, or how good it is for the agent to perform a given action in a given state. This is measured in terms of the expected return.

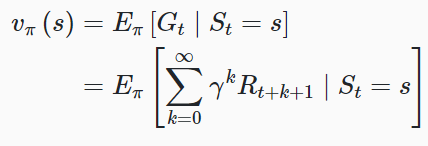
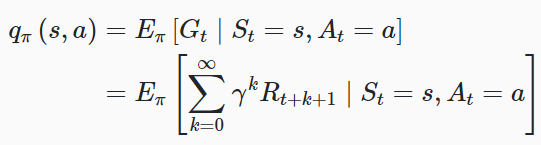
 

Figure 2: State-value function Figure 3: Action-value function

## 2 Challenges

* If the value of epsilon is chosen to be too low then it won't explore enough and would select the max value. Hence the agent would make the same choices again and again, since all the other options haven’t been explored.
* For the learning rate of 0.1, the convergence takes place quickly. It is possible to lose out on the other options that might lead to a better result. The higher the learning rate, the more quickly the agent will adopt the new Q-value.
* Modifying the number of iterations gives different results. Specify a max number of steps that our agent can take before the episode auto-terminates to avoid infinite loops.
* Using DQN, predicts the actions rather than saving the Q values.

## 3 Q-Learning Algorithm

Q-Learning is a reinforcement learning technique that is used for learning the optimal policy in a Markov Decision process. The objective of Q-Learning is to find the most optimal policy so that the maximum total reward is achieved over all successive steps. Thus, We need to learn the optimal Q-values for each state-action pair in order to find the optimal policy.

We calculate the function recursively to match our policy in order to calculate the discounted cumulative total reward.

Q-Learning can be done in a tabular form with rows denoting the states and columns denoting the actions taken. Initially the values in this q\_table are set to 0. Then on each time step in each episode, we choose an action, observe the reward and the next state. And then we update the q-value function. A value iteration update algorithm to update our Q-values as we explore the environment’s states.

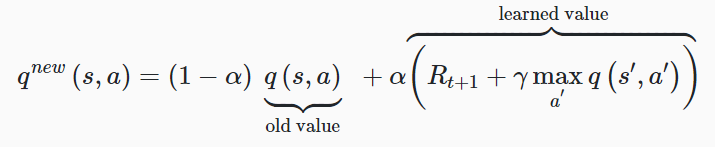


Figure 4: Q-Learning Updation

We use an epsilon greedy strategy, to get this balance between exploitation and exploration. Initially the exploration rate ϵ is set to 1. This exploration rate is the probability that the agent explores the environment rather than exploit it.

To determine whether the agent will choose exploration or exploitation at each time step, we generate a random number between 0 and 1. If this number is greater than epsilon, then the agent will choose its next action via exploitation, i.e. it will choose the action with the highest Q-value for its current state from the Q-table. Otherwise, its next action will be chosen via exploration, i.e. randomly choosing its action and exploring what happens in the environment.

## 4 Environment

Reinforcement learning environments can be of different forms. Some examples being video games, physical simulations, stock market simulations, etc. The Reinforcement learning community i.e. the openAI has developed a standard for all the environments. OpenAI’s Gym library facilitates these environments. Following is the environment for the problem that we are handling. This is the initial state of the grid-world environment.

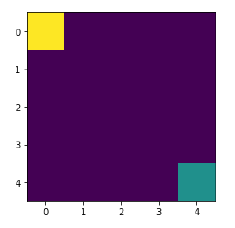


Figure 5: 4\*4 grid environment

The agent is supposed to reach the goal. The environment’s state space can be described as an n\*n matrix with real values in the interval [0,1] to designate different features and their positions. The four actions that the agent can take are: up, down, left and right. At each step, the agent takes an action then moves in the described direction, after deciding if it needs to go for exploration or exploitation. And the agent then receives a reward.

# 5 Training Model

Different hyper-parameters are used and modified in order to obtain the best results.

## 5.1 Episodes

1000 episodes have been used in the project. The number of episodes define the total number of times we want the agent to play during training.

## 5.2 Exploration rate

In the project we have used minimum 0.001 and maximum as 1 and decay as 0.005. Minimum and maximum rates

are the bounds of the exploration while decay rate is the rate at which the exploration decays.

The Exploration rate ϵ (epsilon) is initially set to 1. This exploration rate is the probability that the agent will explore

The environment rather than exploit it. With ϵ=1, it is 100% certain that the agent will start out by exploring the

environment. By changing the values of the epsilon rate, we infer whether the agent will take an action to explore or

exploit.

## 5.3 Learning rate

The learning rate is a number between 0 and 1, which can be thought of as how quickly the agent abandons the previous Q-value in the Q-table for a given state-action pair for the new Q-value. For our project we have used it as 0.1.

## 5.4 Discount rate

We have used it as 0.9 in this project. Discount rate tells how important are the rewards for the reinforcement learning

agent, in the distant future related to the immediate future. Discount factor is a value between 0 and 1.

# 6 Results

## 

|  |  |  |  |
| --- | --- | --- | --- |
| **Learning Rate** | **Episodes** | **Decay exploration rate** | **Discount Rate** |
| 0.1 | 1000 | 0.005 | 0.9 |

These hyper-parameters give high accuracy for the model to give correct results. Using 0.1 as the learning rate, 1000 episodes and 0.005 decay exploration rate. Using minimum exploration rate as 0.001 and maximum exploration rate as 1. The agent successfully reached the goal in 8 steps.

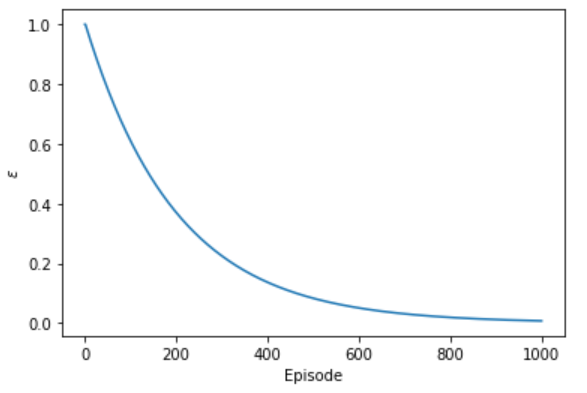
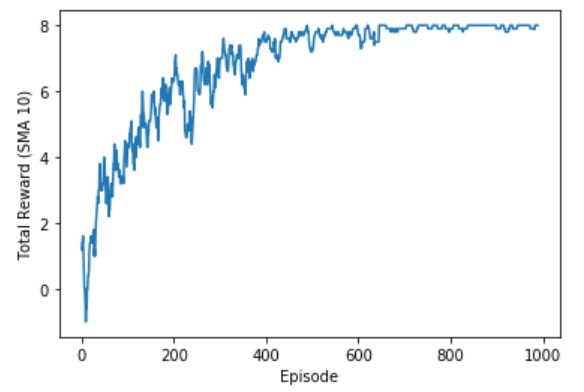
 

Figure 6: Epsilon graph Figure 7: Reward graph

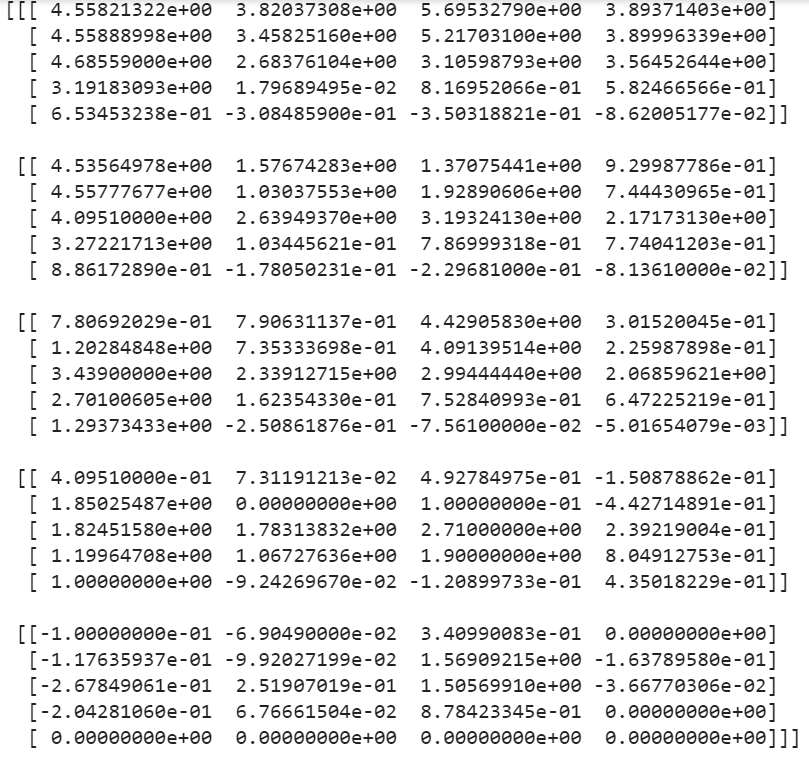


Figure 6: Q\_table

# 6 Conclusions

Reinforcement learning is a highly efficient to be used for video games. The agent successfully reached the goal in 8 steps. Using minimum exploration rate as 0.001 and maximum exploration rate as 1. Hence we efficiently implemented reinforcement learning in this project.

## References

1. <https://www.youtube.com/watch?v=nyjbcRQ-uQ8&list=PLZbbT5o_s2xoWNVdDudn51XM8lOuZ_Njv&index=1>
2. <https://en.wikipedia.org/wiki/Markov_decision_process>
3. <https://deeplizard.com/learn/video/qhRNvCVVJaA>
4. <https://deeplizard.com/learn/video/nyjbcRQ-uQ8>
5. <https://ublearns.buffalo.edu/bbcswebdav/pid-5242352-dt-content-rid-27556999_1/courses/2199_23170_COMB/Project_Description.pdf>
6. <https://www.youtube.com/watch?v=qhRNvCVVJaA&list=PLZbbT5o_s2xoWNVdDudn51XM8lOuZ_Njv&index=6>
7. <https://towardsdatascience.com/introduction-to-reinforcement-learning-markov-decision-process-44c533ebf8da>
8. <https://towardsdatascience.com/reinforcement-learning-demystified-markov-decision-processes-part-1-bf00dda41690>