CSE 474/574: Introduction to Machine Learning

INTRODUCTION AND APPLICATION OF REINFORCEMENT LEARNING

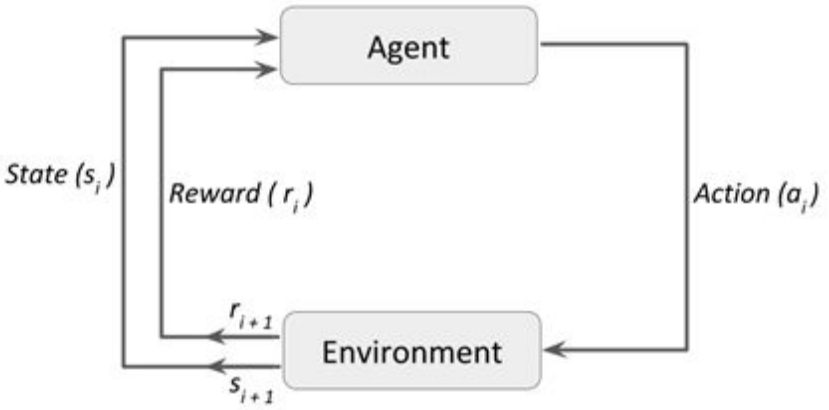
Instructor:- Sargur Srihari
University at Buffalo Buffalo, NY 14214
Student name: Makarand Mandolkar
UBIT person number: 50321880

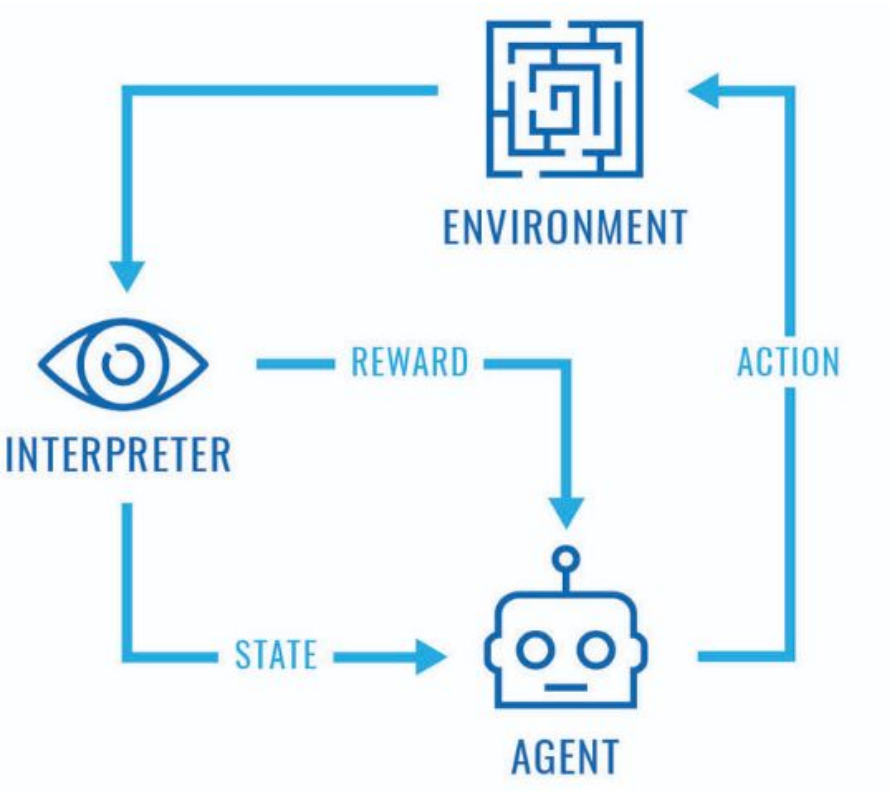
ABSTRACT

In this project, we are mainly supposed to carry out three tasks which are as follows:-
1)We have to build a reinforcement learning agent to navigate towards a given goal. We have already been given with a grid world environment of order 4X4. So our task is to build a policy function, which is further used to determine the actions of the agent in order to get a maximum reward while reaching the goal.
2) In the second task, we are supposed to update the Q-table, which is a matrix of the states and actions taken by the agent.
3) We are required to implement a training algorithm to train the agent so that it reaches the goal in an optimal cost by using the rewards and discount, to ultimately skew the results.

1. INTRODUCTION

Reinforcement learning (RL) is learning by interacting with an environment. An RL agent learns from the consequences of its actions, rather than from being explicitly taught and it selects its actions on basis of its past experiences (exploitation) and also by new choices (exploration), which is essentially trial and error learning. Reinforcement learning makes optimal decisions for complex algorithm, where it uses a reward-based approach where the agent will get some of the reward when it performs as expected or say performs well and on the other hand it will get penalized when it doesn't perform as expected. Thus learning for its past mistakes, the agent learns and thus improves over time. By using the method of "Cause and Effect", unlike any other machine learning model, a reinforcement learning model communicates with the environment and improves.



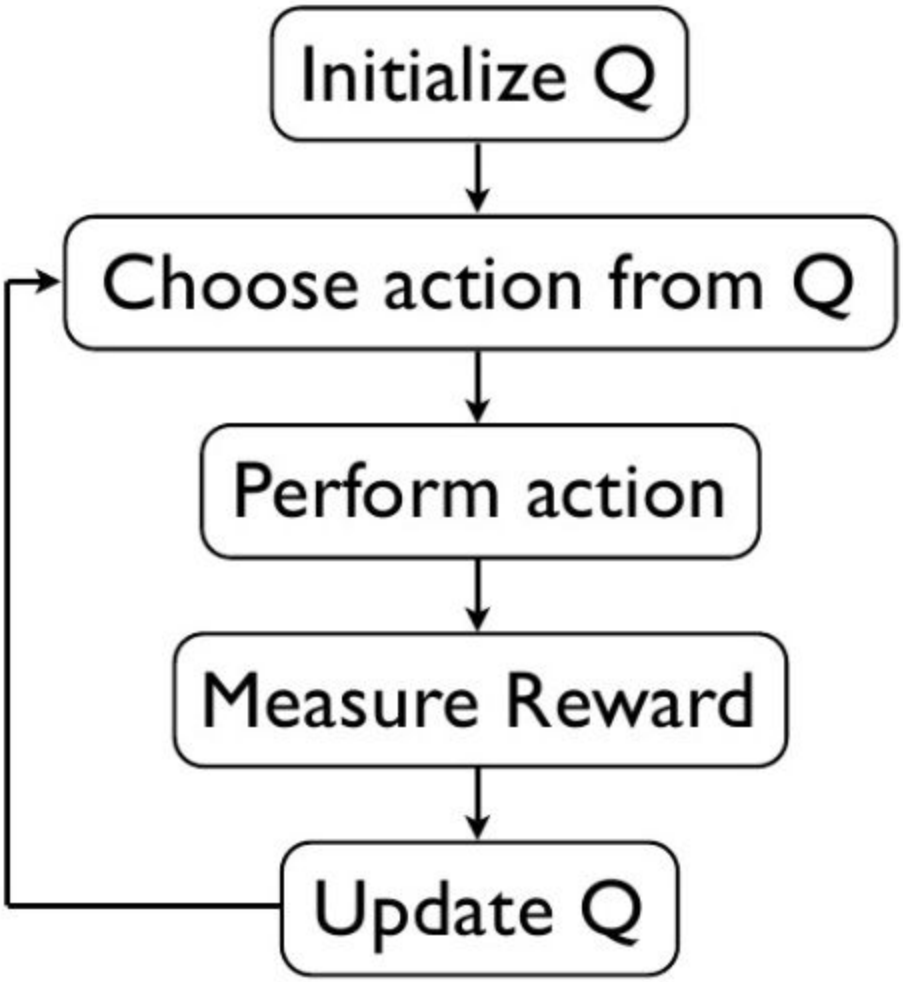


Reinforcement Learning

Q-learning is a model-free reinforcement learning algorithm. The goal of Q-learning is to learn a policy, which tells an agent what action to take under what circumstances. It does not require a model (hence the connotation "model-free") of the environment, and it can handle problems with stochastic transitions and rewards, without requiring adaptations.Q-Learning is a basic form of Reinforcement Learning which uses Q-values (also called action values) to iteratively improve the behaviour of the learning agent.

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.

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 the 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.



2.ARCHITECTURE
The Q-Learning Policy:

Q-learning is an off-policy reinforcement learning algorithm that seeks to find the best action to take given the current state. It's considered off-policy because the q-learning function learns from actions that are outside the current policy, like taking random actions, and therefore the policy isn't needed. More specifically, q-learning seeks to learn a policy that maximizes the total reward. Its a simple algorithm for comparing the expected utility for a set of actions given the current state. We only need to consider the most recent state when making a decision because we assume the environmental to be a Markov chain. This simply means that the current state completely characterizes the process.

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

Update rule for Q-Learning

The execution of this project is divided into the following stages:
STAGE 1:-
Policy function implementation
In the beginning, the agent starts at random, the search of a pattern, as it is not aware of the rewards. Further, the agent will move in the direction of rewards and thus solving the complex problem. Here, the agent will perform as per set policy, which is a function that maps a given state to probabilities of selecting each possible action from that state.

$$\pi^*(s) = \arg\max_a Q^*(s, a) \quad \forall s \in \mathbb{S}$$

Where, π = *policy*, s = *state* and a= *action.*
For example, if an agent follows policy π at time t, then $\pi(a|s)$ is the probability that At=a if St=s. This means that, at time t, under policy π, the probability of taking action a in state s is $\pi(a|s)$. Note that, for each state $s \in S$, π is a probability distribution over $a \in A(s)$.

When the agent has found a pattern to the problem it will select an action which will yield the highest reward possible.

STAGE 2:-

Q-table updation

There is an iterative process of updating the values. As we start to explore the environment, the Q-function gives us better and better approximations by continuously updating the Q-values in the table. A Q-table basically consists of state and action as the parameters. For every episode, the Q-table gets the new action and state until it reaches the goal. It does so by increasing its rewards and hence the results.

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

Here the Q-table is updated after each episode and this updated Q-table is further used.

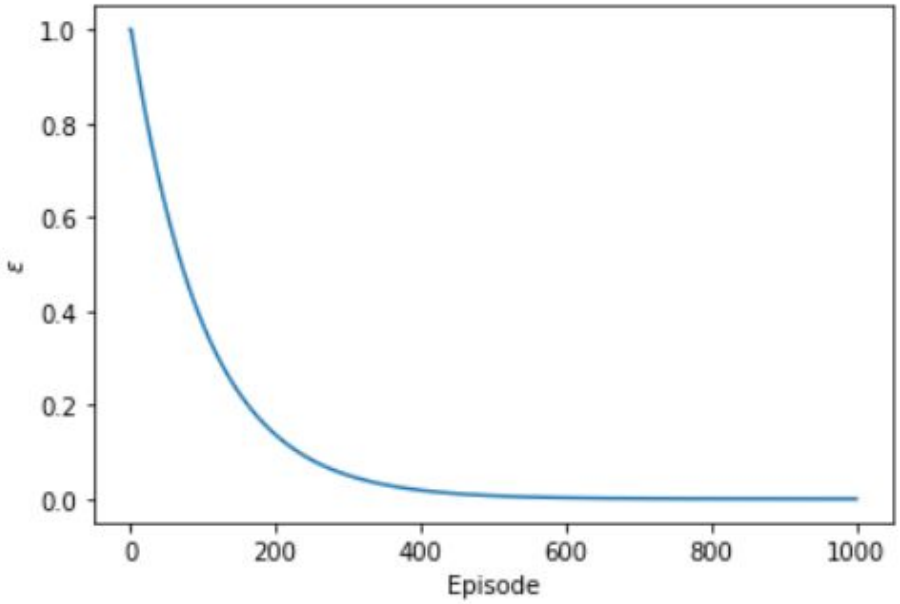
STAGE 3:-

Here the environment is similar to OpenAI's gym environment. The environment will be making use of three menthids namely reset, step and render. They help in the functioning of the process. Here, the reset menthid initializes the environment with a fresh episode. The new state is returned by the step method, which takes an action as a parameter, processes it and thus returns the new state, with a boolean indicating the run is over and the reward for performing the action. The agent will be trained for a fixed number of episodes and update the state, action, reward, and next state for each episode. Here, it makes use of exploration rate of epsilon so that at first we'll let our agent to try all kinds of things before it starts to see the patterns. When the action is not picked randomly. Reward value will be picked by the agent based on the current state, followed by picking the action that will give highest reward. By introducing an exponential decay epsilon, we reduce the number of random action and allow are an agent to explore the environment.

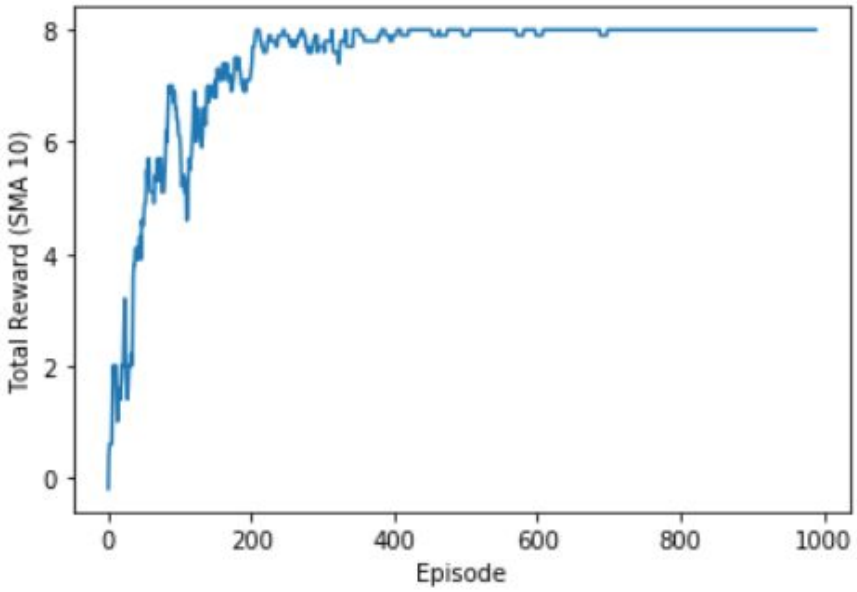
The example in the template code given us information about Random agent, which executes random action so that the robot can move in the environment and Heuristic agent, which as the name suggests, calculates the heuristic distance between the goal and current position, followed by trying to minimise the distance between them.

3.RESULTS

Below graph gives information about number of episodes vs the epsilon value



Below graph give us information about the number of episodes vs total rewards per episode



State of the agent of our basic grid-world environment on Q-table updation

(please find the below images)

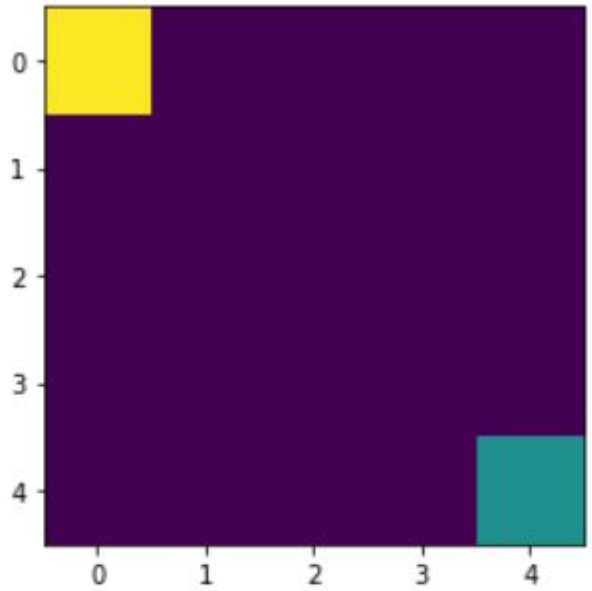


Fig.1

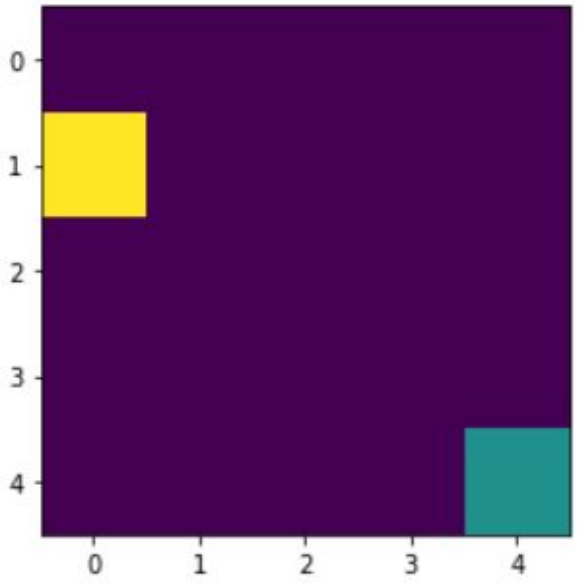


Fig. 2

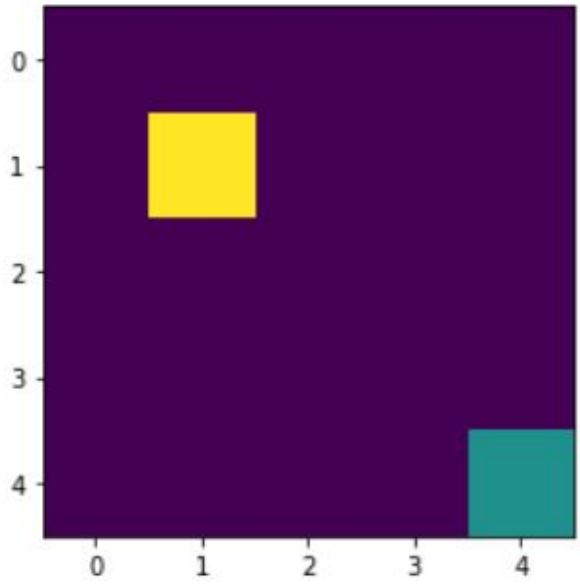


Fig. 3

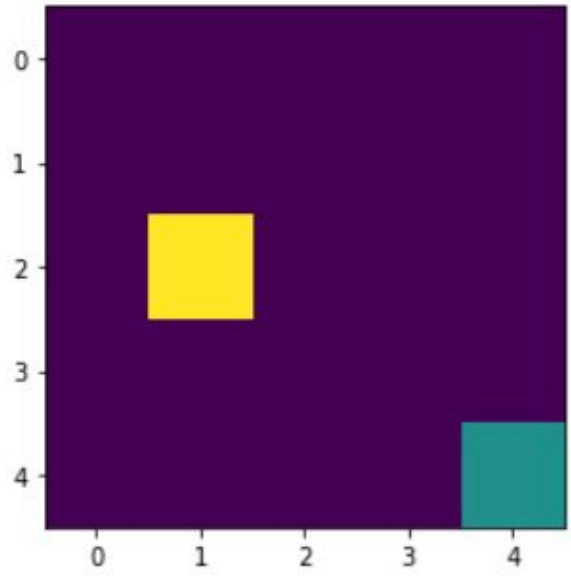


Fig. 4

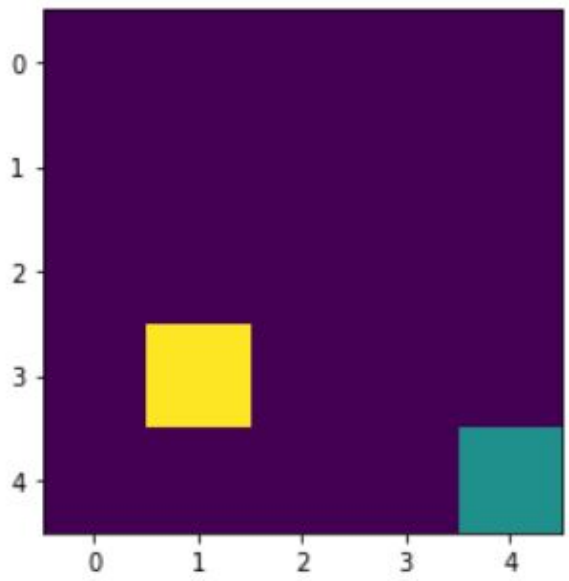


Fig.5

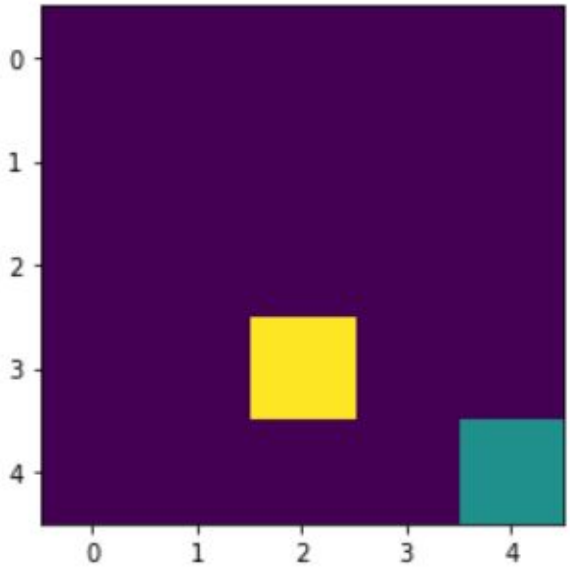


Fig.6

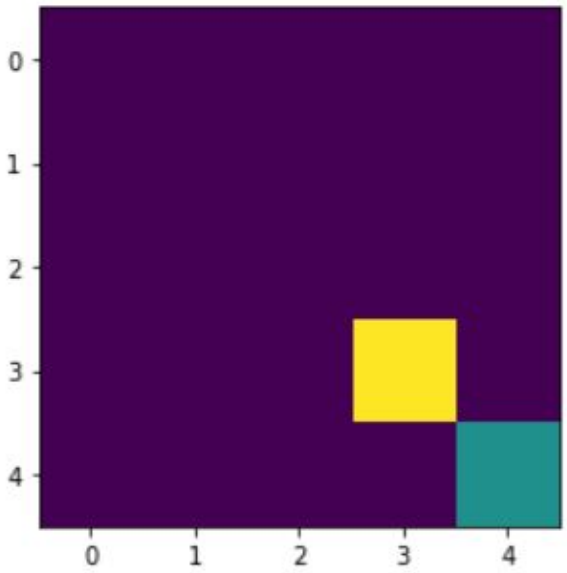


Fig.7

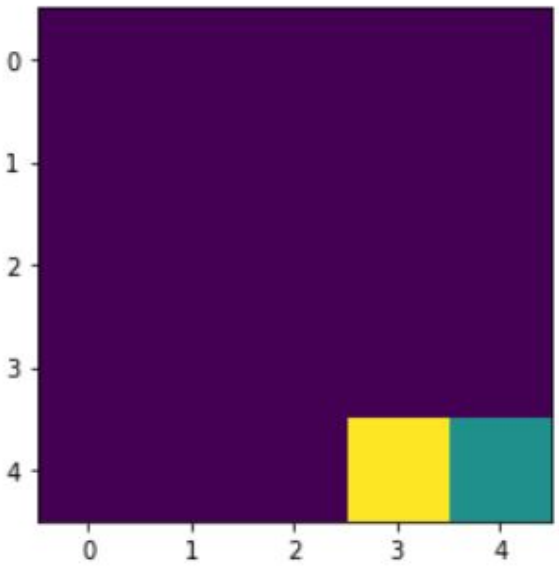


Fig. 8

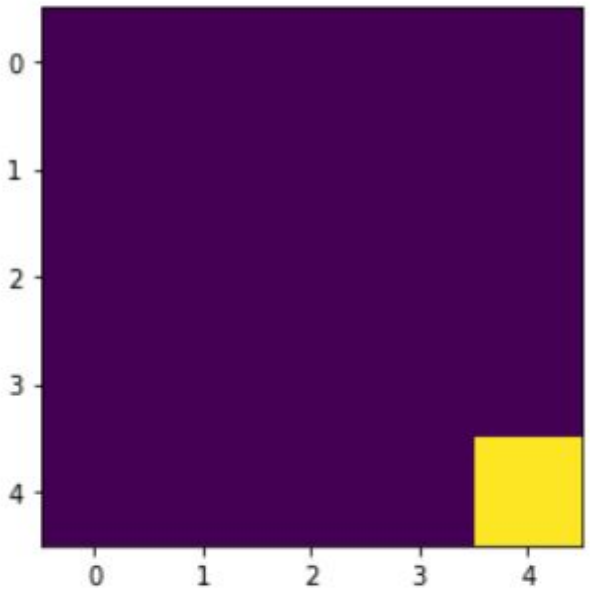


Fig. 9

Q Table

State	Action 0	Action 1	Action 2	Action 3
(0,0)	2.74916588e+00	2.59913976e+00	5.69532790e+00	2.82380140e+00
(0,1)	2.25513818e+00	2.57669280e+00	5.21703100e+00	2.91558371e+00
(0,2)	4.68559000e+00	1.75065427e+00	1.74004390e+00	1.38672664e+00
(0,3)	5.63232353e-01	-2.23479893e-03	1.69796535e+00	2.77237650e-01
(0,4)	1.69045667e+00	-8.80724478e-02	0.00000000e+00	-1.51324993e-01
(1,0)	2.53246436e+00	2.92522507e-01	1.10343148e+00	-1.13727106e-01
(1,1)	3.04382474e+00	4.28340707e-01	9.57975110e-01	1.55143458e-01
(1,2)	4.09510000e+00	1.98808370e+00	2.38979274e+00	2.22040472e-01
(1,3)	3.15770010e+00	-1.83883643e-01	4.59649900e-01	2.44926536e-01
(1,4)	1.94772659e+00	-6.11911465e-03	-1.00000000e-01	-7.02528119e-03
(2,0)	8.61823518e-01	9.09438118e-02	2.67888315e+00	-1.68786823e-01
(2,1)	8.26950426e-01	-9.79951670e-02	3.67762564e+00	1.56288892e-02
(2,2)	1.01664193e+00	2.22083399e+00	3.43900000e+00	8.88586735e-01
(2,3)	1.03816010e+00	1.00803225e+00	2.71000000e+00	8.25837976e-01
(2,4)	1.90000000e+00	1.07987919e-01	-1.46063166e-01	6.79792145e-01
(3.0)	7.12990441e-01	-8.34861772e-02	0.00000000e+00	-2.17369000e-01
(3,1)	6.39215875e-01	-1.17429009e-02	0.00000000e+00	-1.57510000e-01
(3,2)	8.61260277e-01	-1.17649663e-02	1.00000000e-01	-1.81000000e-01
(3,3)	1.07750243e+00	0.00000000e+00	0.00000000e+00	-8.29000000e-02
(3,4)	1.00000000e+00	3.17003819e-01	-5.44973966e-02	-1.29897510e-01
(4,0)	-2.38510000e-01	-1.28938951e-01	3.43900000e-01	-1.64800000e-01
(4,1)	-1.00000000e-01	0.00000000e+00	6.37649604e-01	-7.56100000e-02
(4,2)	-2.30563900e-01	-9.10000000e-02	9.58947884e-01	0.00000000e+00
(4,3)	-1.00000000e-01	-8.29000000e-02	7.17570464e-01	0.00000000e+00
(4,4)	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00

5.Conclusion

From performing the above project, we got good understanding about application of reinforcement learning. This is project, in the task were we used random agents, we couldn't achieve the objective (Goal was not reached) as it couldn't learn. In the task where we made use of heuristic agents, the goal was reached. Whereas by using Q-table learning, through next state commands, we were able to reach the final goal, and thus the final goal was reached.

References

[1] Srihari, Sargur & Chauhan, Mihir. iml_Project3.pdf

[2] https://en.wikipedia.org/wiki/Reinforcement_learning