CSE 474/574: Introduction to Machine Learning (Fall 2019)
Reinforcement Learning using Q-Learning

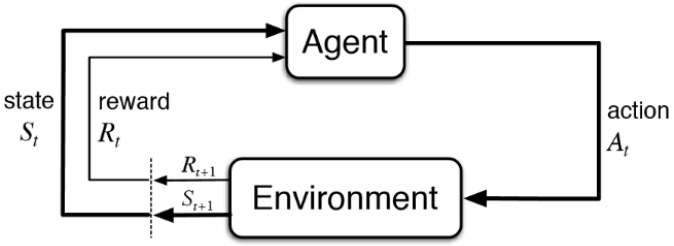
Sonali Naidu
Department of Computer Science
University at Buffalo
Buffalo, NY – 14226
sonaliye@buffalo.edu

Abstract

In this project we have worked on implementing reinforcement learning agent to navigate the 4x4 grid world environment. The environment used here is the OpenAI Gym environment and the agents used are random and heuristic agents. The agent learns the optimal policy through Q-Learning that allows it to take actions to reach the goal while avoiding obstacles.

Introduction

Reinforcement Learning is an Machine learning area which is about taking suitable action to maximize reward in a particular situation. It focuses on how agents can learn to take actions in response to the current state of an environment to maximize reward. It is modelled as Markov Decision Pocess(MDP). Below is the diagram of MDP.



We have an environment which represents the outside world to the agent and an agent that takes actions, receives observations from the environment that consists of a reward for his action and information of his new state. That reward informs the agent of how good or bad was the taken action, and the observation tells him what is his next state in the environment. The agent tries to figure out the best actions to take or the optimal way to behave in the environment in order to carry out his task in the best possible way.

An MDP is a 4-tuple (S, A, P, R), where

- S is the set of all possible states for the environment
- A is the set of all possible actions the agent can take
- P = P r(st+1 = s 0 |st = s, at = a) is the state transition probability function
- R : S × A × S → R is the reward function

Our task is find a policy π : S → A which our agent will use to take actions in the environment which maximize cumulative reward, i.e.,

$$\sum_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1})$$

where γ ∈ [0, 1] is a discounting factor (used to give more weight to more immediate rewards), st is the state at time step t, at is the action the agent took at time step t, and st+1 is the state which the environment transitioned to after the agent took the action.

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 a policy isn't needed. More specifically, q-learning seeks to learn a policy that maximizes the total reward.

Open AI Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like pong or pinball. Gym is an open source interface to reinforcement learning tasks. Gym provides an environment and its is upto the developer to implement any reinforcement learning algorithms. Developers can write agent using existing numerical computation library, such as TensorFlow or Theano.

Algorithm

Q Learning Algorithm:
The q in q-learning stands for quality. Quality represents how useful a given action is in gaining future reward.
We need an algorithm to learn(1) a policy (2) that will tell us how to interact(3) with an environment(4) under different circumstances(5) in such a way to maximize rewards(6).

1. **Create q-Table**
   When we perform q-learning we create a *q-table* or matrix that follows the shape of [state, action] and we initialize our values to zero. We then update and store our *q-values* after an episode. This q-table becomes a reference table for our agent to select the best action based on the q-value.
2. **Q-learning and making updates**
   The next step is simply for the agent to interact with the environment and make updates to the state action pairs in our q-table Q[state, action].

   Taking Action: Explore or Exploit (Agent interacts with environment in 1 of 2 ways)

   It first uses the q-table as a reference and views all possible actions for a given state. The agent then selects the action based on the max value of those actions. This is known as ***exploiting*** since we use the information we have available to us to make a decision.
   The second way to take action is to act randomly. This is called ***exploring*.**

   a. Agent starts in a state (s1) takes an action (a1) and receives a reward (r1)
   b. Agent selects action by referencing Q-table with highest value (max) OR by random (epsilon, ε)

$$\pi(s_t) = \underset{a \in A}{\operatorname{argmax}} Q_\theta(s_t, a)$$

   c. Update q-values

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

1. Learn: This implies we are not supposed to hand-code any particular strategy but the algorithm should learn by itself.

2. Policy: This is the result of the learning. Given a State of the Environment, the Policy will tell us how best to Interact with it so as to maximize the Rewards.
3. Interact: This is nothing but the “Actions” the algorithm should recommend we take under different circumstances.
4. Environment: This is the black box the algorithm interacts with. It is the game it is supposed to win. It’s the world we live in. It’s the universe and all the suns and the stars and everything else that can influence the environment and it’s reaction to the action taken.
5. Circumstances: These are the different “States” the environment can be in.
6. Rewards: This is the goal. The purpose of interacting with the environment. The purpose playing the game.

Policy, Update and Training

Q Table

q-table is a matrix that follows the shape of [state,action] that is used to store the q-values after an episode. This q-table becomes a reference table for our agent to select the best action based on the q-value.

```
Q_table: [[[ 5.24105233 -0.07368355  0.64976071 -0.14784104]
  [ 0.70078423 -0.1         0.19       -0.08209   ]
  [ 0.1         0.          0.1        -0.091     ]
  [ 0.1         0.          0.          0.        ]
  [ 0.          0.          0.          0.        ]]

 [[ 4.92914264 -0.27425622  0.22249    -0.06622731]
  [ 0.1        -0.11742083  0.62470279 -0.09019   ]
  [ 0.         -0.181       0.4339     -0.143659  ]
  [ 0.          0.          0.19       -0.22312   ]
  [ 0.1         0.         -0.1        -0.091     ]]

 [[ 0.1        -0.155971    4.52337252 -0.1       ]
  [ 0.1        -0.23851     4.01635747  0.00874144]
  [ 0.          0.          3.40751532 -0.091     ]
  [ 0.1         0.          2.70032043 -0.04045072]
  [ 1.89797151 -0.1        -0.19       -0.137098  ]]

 [[ 0.          0.          0.1        -0.19      ]
  [ 0.         -0.091       0.1        -0.1       ]
  [ 0.          0.          0.         -0.091     ]
  [ 0.          0.          0.         -0.1       ]
  [ 0.99975725  0.          0.         -0.1       ]]

 [[ 0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.        ]]]
```

Policy

Given a State of the Environment, the Policy will tell us how best to Interact with it so as to maximize the Rewards.

$$\pi\left(s_t\right) = \underset{a \in A}{\operatorname{argmax}} Q_\theta\left(s_t, a\right)$$

Update

Here we adjust our q-values based on the difference between the discounted new values and the old values. We discount the new values using gamma and we adjust our step size using learning rate (lr).

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

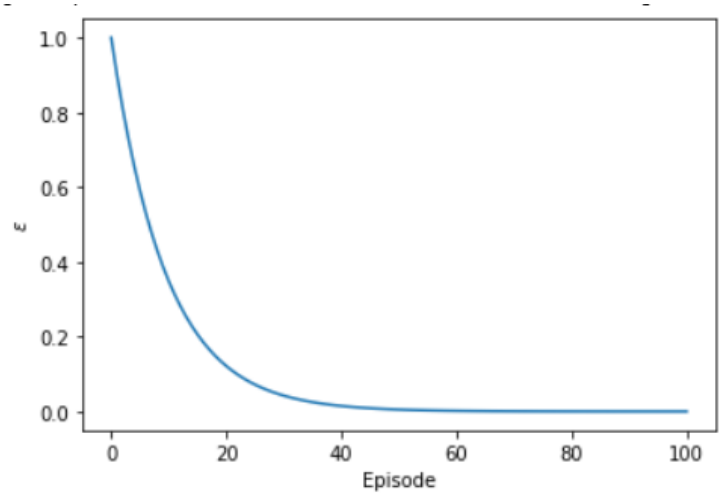
Hyperparameters

Learning Rate: lr or learning rate is defined as how much you accept the new value vs the old value. We take the difference between new and old and then multiply that value by the learning rate. This value then gets added to our previous q-value which essentially moves it in the direction of our latest update.
Gamma: It is used to balance immediate and future reward. Using he update rule above we apply the discount to the future reward. The value can range between 0.8 tp 0.99.
Reward: Reward is the value received after completing a certain action at a given state.
Max: np.max() uses the numpy library and is taking the maximum of the future reward and applying it to the reward for the current state. It impacts the current action by the possible future reward. We're allocating future reward to current actions to help the agent select the highest return action at any given state.

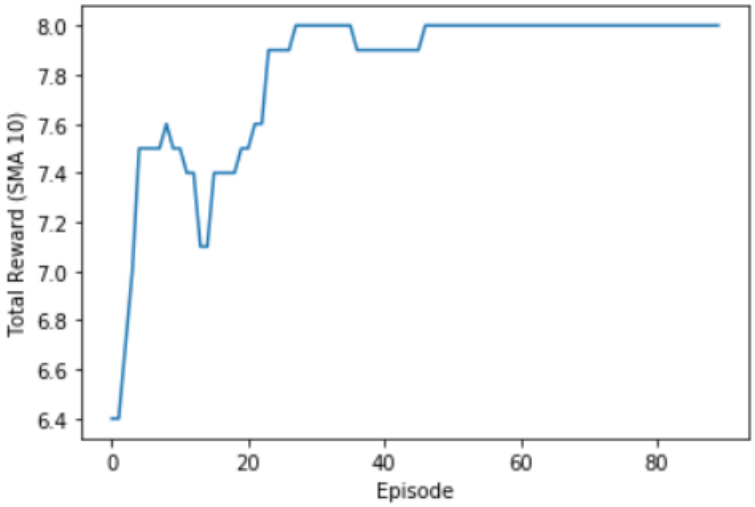
Results:

Graphs representing

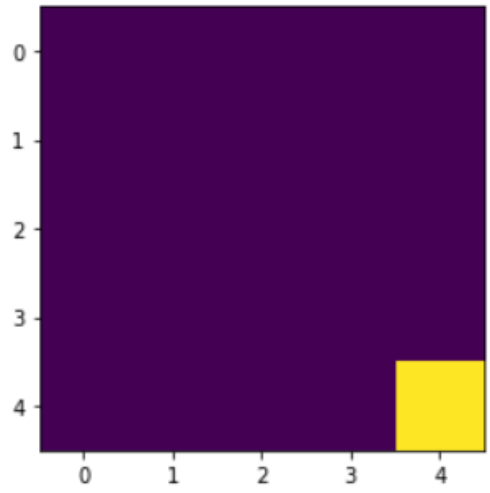
Number of Episode versus Epsilon



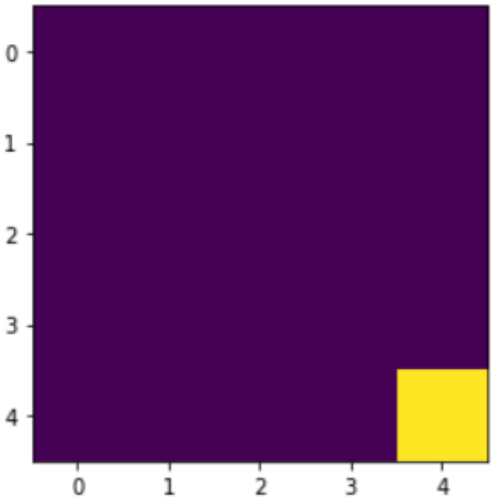
Number of Episode versus Total Reward



Below we can see the Epsilon value after 50 and 99 episodes

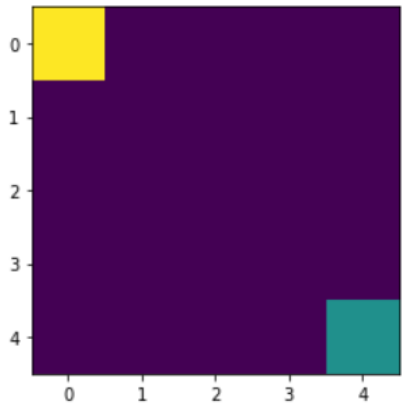
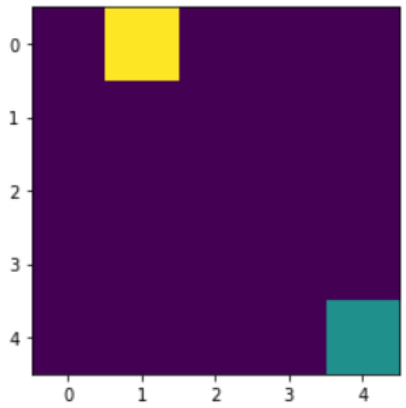
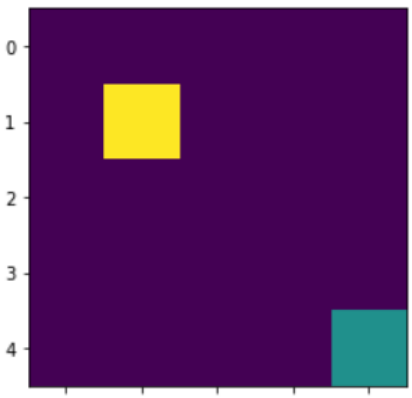
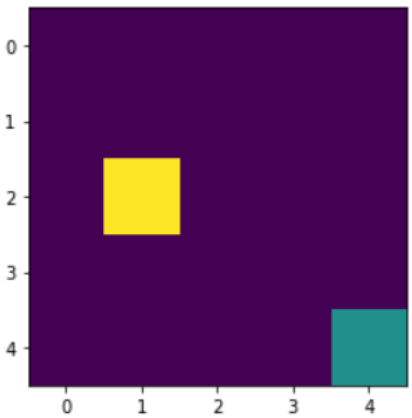


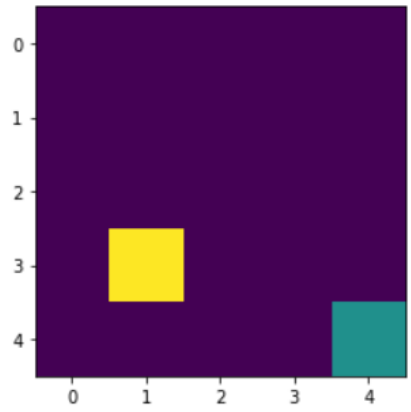
Episode: 50
Epsilon: 0.004638397686588107
Total Rewards: 8

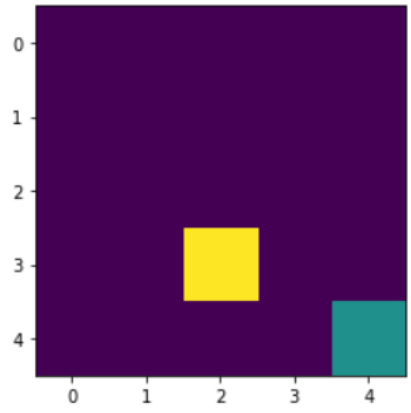


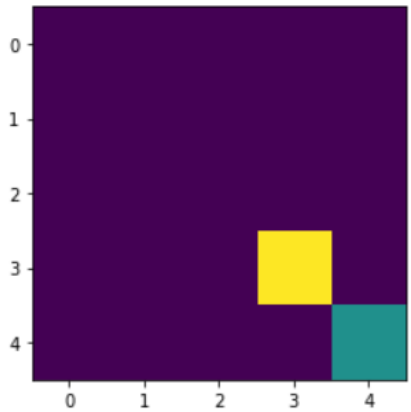
Episode: 99
Epsilon: 2.6561398887587544e-05
Total Rewards: 8

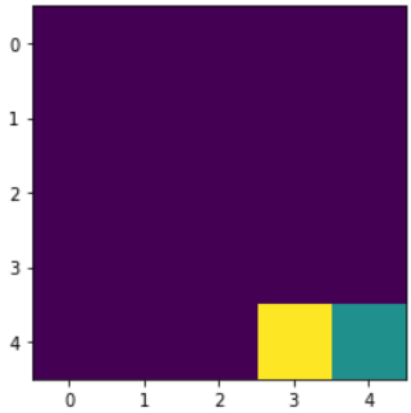
After training the agent with 100 episodes by continuous updation of q-tables following the calculation of rewards we see the below results.

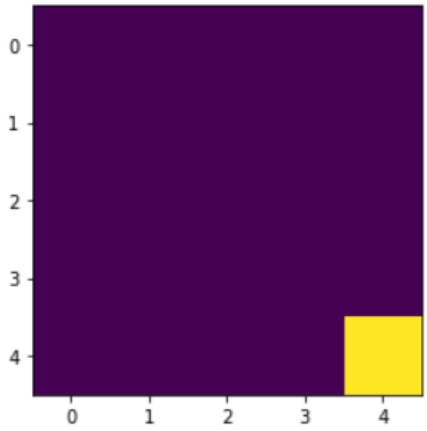






0
1
2
3
4
0
1
2
3
4


0
1
2
3
4
0
1
2
3
4


0
1
2
3
4
0
1
2
3
4


0
1
2
3
4
0
1
2
3
4



Conclusion

We see that the q learning is a off-policy reinforcement learning algorithm. We use the basic update rule for q learning. We also see that the q-learning uses future rewards to influence the current action given a state and therefore helps the agent select best actions that maximize total reward.

References:

https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56
https://www.geeksforgeeks.org/what-is-reinforcement-learning/
https://towardsdatascience.com/reinforcement-learning-demystified-36c39c11ec14
https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56
https://towardsdatascience.com/q-learning-54b841f3f9e4