CartPole-v1

Q-Learning

Q-learning is a reinforcement learning algorithm that helps an agent learn the optimal action based on q values.
For the cartpole environment, the actions are moving to the right or left, based on the positon, angle, velocity and angular velocity of the cart.

Implementation

1. Define the environment

   Define the gymnasium environment.

```
env = gym.make('CartPole-v1', render_mode='human' if render
else None
```

2. Initialize the arrays

   Initialize the array using the limits within which the pole should be for position, velocity, angular velocity and angle.

```
pos_space = np.linspace(-2.4, 2.4, 10)
    vel_space = np.linspace(-4, 4, 10)
    ang_space = np.linspace(-.2095, .2095, 10)
    ang_vel_space = np.linspace(-4, 4, 10)
```

3. Set up hyperparameters

```
learning_rate_a = 0.1
    discount_factor_g = 0.99
    epsilon = 1
    epsilon_decay_rate = 0.00001
```

4. Set up q table

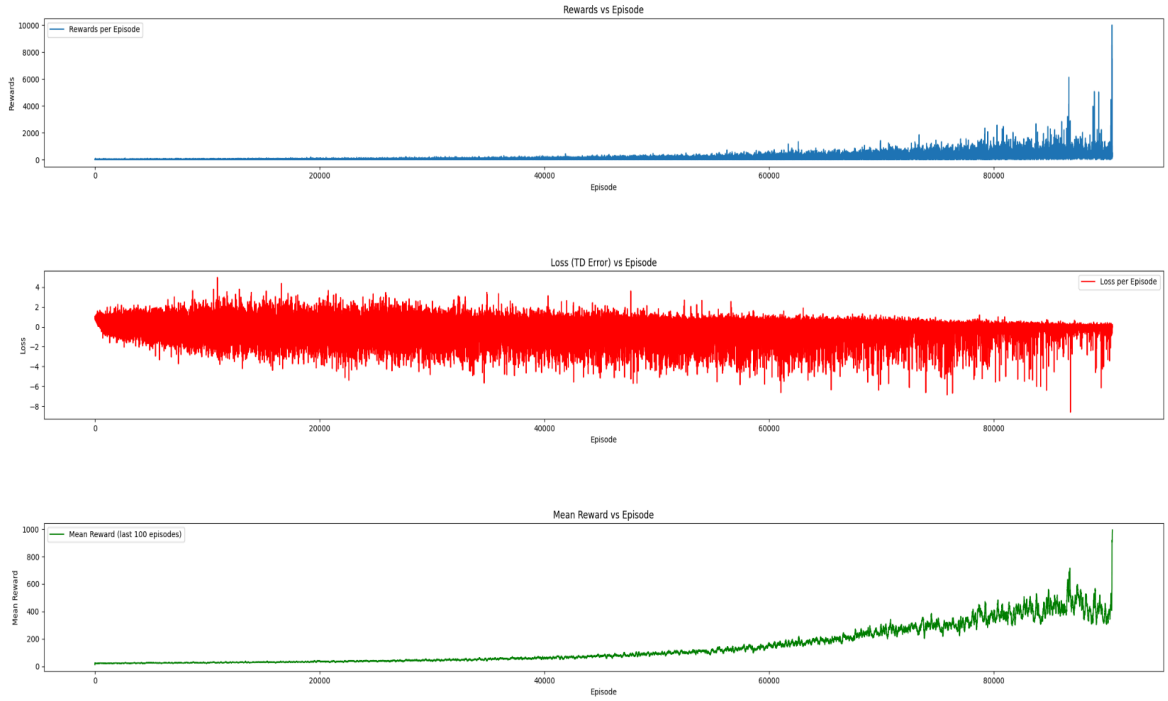
```
state = env.reset()[0]
        state_p = np.digitize(state[0], pos_space)
        state_v = np.digitize(state[1], vel_space)
        state_a = np.digitize(state[2], ang_space)
        state_av = np.digitize(state[3], ang_vel_space)
```

5. Implement Algorithm

6. Terminate

Results

The following graphs show the trends of rewards, loss and mean rewards differing with respect to episodes.



Output from last few episodes:

```
Episode: 89700 440.0  Epsilon: 0.10  Mean Rewards 526.8
Episode: 89800 415.0  Epsilon: 0.10  Mean Rewards 442.7
Episode: 89900 562.0  Epsilon: 0.10  Mean Rewards 444.2
Episode: 90000 265.0  Epsilon: 0.10  Mean Rewards 636.1
Episode: 90100 528.0  Epsilon: 0.10  Mean Rewards 685.7
Episode: 90200 779.0  Epsilon: 0.10  Mean Rewards 651.3
Episode: 90300 635.0  Epsilon: 0.10  Mean Rewards 553.5
Episode: 90400 686.0  Epsilon: 0.10  Mean Rewards 562.7
Episode: 90500 565.0  Epsilon: 0.10  Mean Rewards 738.6
Episode: 90600 262.0  Epsilon: 0.09  Mean Rewards 632.1
Episode: 90700 198.0  Epsilon: 0.09  Mean Rewards 479.8
Episode: 90800 1088.0  Epsilon: 0.09  Mean Rewards 544.5
Episode: 90900 848.0  Epsilon: 0.09  Mean Rewards 898.2
Episode: 91000 1207.0  Epsilon: 0.09  Mean Rewards 763.7
Episode: 91100 743.0  Epsilon: 0.09  Mean Rewards 763.7
```

Strengths

- The agent successfully learns to balance the pole through Q-learning. The increasing trend in the rewards demonstrates effective learning.
- Very similar to the way people learn. Very Ideal

- No probability that a mistake will be repeated.

Weaknesses

- As the problem becomes more complex, the q table becomes larger and larger hence it becomes difficult to store as well as makes it more computationally intensive.
- Can be practical only for small environments.

Deep Q Network

Q learning is a prerequisite of DQN. DQN uses a deep neural network to approximate the q values.

Implementation

1. Environment

```
env = gym.make('CartPole-v1')
```

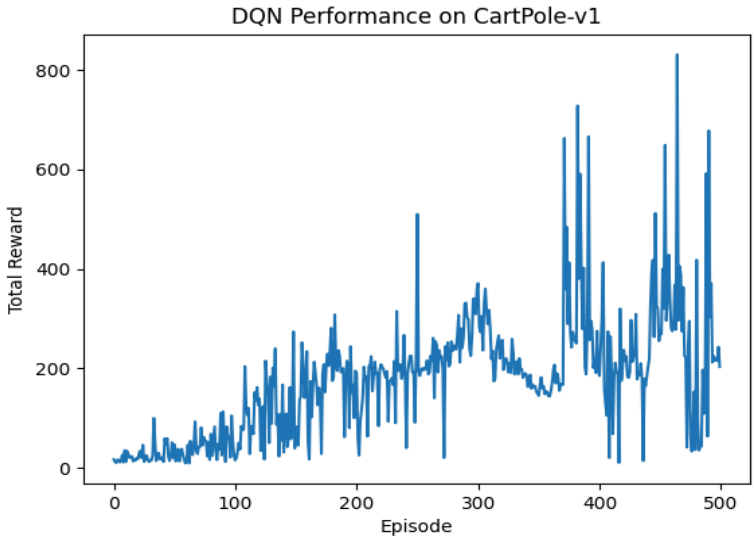
2. Hyperparameters

```
GAMMA = 0.99
LEARNING_RATE = 0.001
MEMORY_SIZE = 10000
BATCH_SIZE = 64
EPSILON_START = 1.0
EPSILON_END = 0.01
EPSILON_DECAY = 0.995
TARGET_UPDATE_FREQUENCY = 10
NUM_EPISODES = 500
```

3. Define model
4. Train
5. Terminate

Results

The following is the graph for total reward vs episode



This is the output for the last few episodes:

```
Episode 360, Total Reward: 154.0, Epsilon: 0.16
Episode 370, Total Reward: 167.0, Epsilon: 0.16
Episode 380, Total Reward: 256.0, Epsilon: 0.15
Episode 390, Total Reward: 284.0, Epsilon: 0.14
Episode 400, Total Reward: 185.0, Epsilon: 0.13
Episode 410, Total Reward: 188.0, Epsilon: 0.13
Episode 420, Total Reward: 237.0, Epsilon: 0.12
Episode 430, Total Reward: 309.0, Epsilon: 0.12
Episode 440, Total Reward: 203.0, Epsilon: 0.11
Episode 450, Total Reward: 274.0, Epsilon: 0.10
Episode 460, Total Reward: 275.0, Epsilon: 0.10
Episode 470, Total Reward: 225.0, Epsilon: 0.09
Episode 480, Total Reward: 418.0, Epsilon: 0.09
Episode 490, Total Reward: 678.0, Epsilon: 0.09
```

Strengths

- The agent shows improved performance over time, suggesting effective learning and convergence towards an optimal policy. The use of experience replay and target networks helps stabilize training.

Weaknesses

- Requires a large number of samples to learn efficiently.
- Slow learning in complex scenarios