**Assignment – Week 8**

**Reinforcement Learning**

**Mountain car example**

We have solved the Mountain Car problem using Q-Learning, a popular reinforcement learning algorithm.

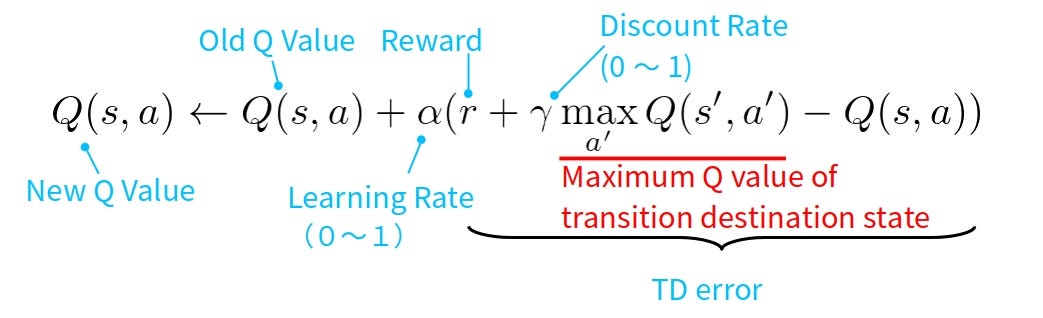
**1. Problem Overview**

The Mountain Car problem is a classic reinforcement learning challenge where a car is stuck in a valley between two hills. The goal is to drive the car up the hill on the right side. However, the driver (agent) doesn't know the suitable velocity and position so that they can climb the hill in a single pass. Instead, the car must learn to build momentum by moving back and forth to reach the goal eventually.

* **State Space:** The car's position and velocity.
* **Action Space:** The car can push left, push right, or do nothing.
* **Reward:** The car receives a reward of -1 for every time step until it reaches the goal.

**2. Q - Learning**

Q-learning is a model-free reinforcement learning algorithm that learns the value of actions in specific states. It uses a Q-table to store the expected future rewards for each state-action pair. The algorithm updates the Q-values iteratively using the Bellman equation.



The TBellman equation is an approach in which the Q value is updated based on the old Q value, action taken, learning rate, and reward received due to the action.

**Exploration vs. Exploitation**

* **Exploration:** The agent tries random actions to discover new states and rewards.
* **Exploitation:** The agent chooses the best-known action based on the Q-table.

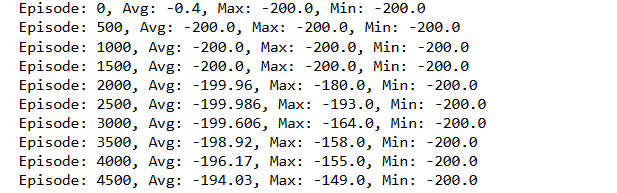
We use an epsilon-greedy strategy to balance exploration and exploitation. Initially, the agent explores more (high epsilon), but over time, it exploits more (epsilon decays).

**3. Q - Learning approach:**

It involves 4 main aspects, and the equation with all the suitable parameters and discretization was done to learn through reinforcement learning.

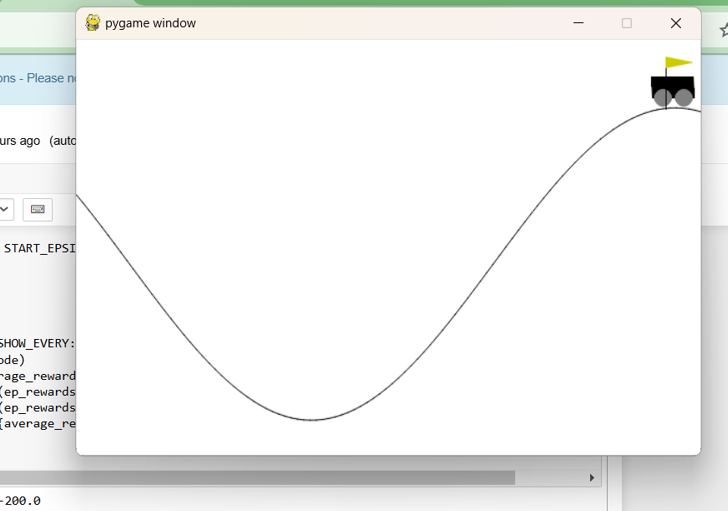
* **Discretization:** The state space (position and velocity) is continuous, so we discretize it into bins to make the Q-table manageable. This reduces the computational complexity.
* **Q-Table:** The Q-table stores the expected future rewards for each state-action pair. Initially, it is filled with random values, and it gets updated as the agent learns.
* **Epsilon-Greedy Strategy:** The agent balances exploration and exploitation by choosing random actions (exploration) or the best-known action (exploitation) based on a probability threshold.
* **Reward Tracking:** We track the rewards over episodes to monitor the agent's performance. The average, maximum, and minimum rewards are plotted to visualize the learning progress.

**4. Results**

****

1. The agent starts with poor performance, often failing to reach the goal.
2. Over time, the agent learns to build momentum and reaches the goal more frequently.
3. The reward metrics show a clear improvement in performance as the number of episodes increases.
4. The initial reward of -200 implies that it never reached the goal, whereas from episode 2000, the reward reaches -180, which implies that the goal is reached occasionally.
5. Increasing the number of episodes and finetuning the parameters further can increase the consistency of achieving goal.

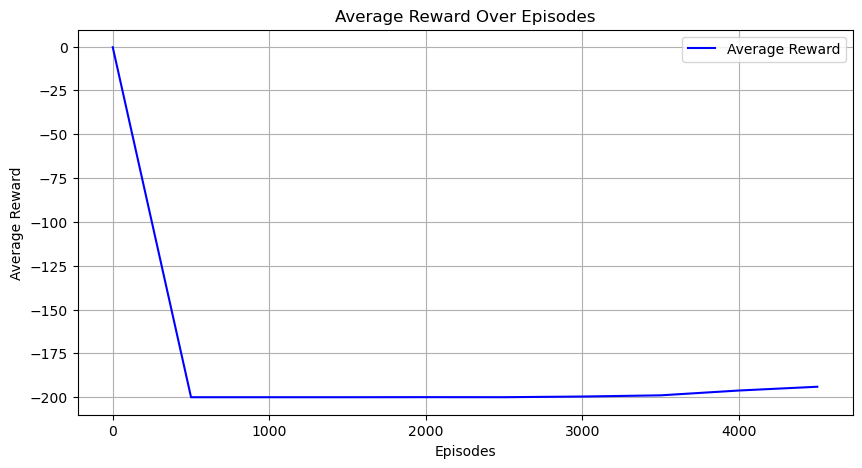
**5. Screen shot of goal achieved**



**6. Plots:**

**Graph 1: Average Reward Over Episodes**

The average reward obtained by the agent over a sliding window of episodes.

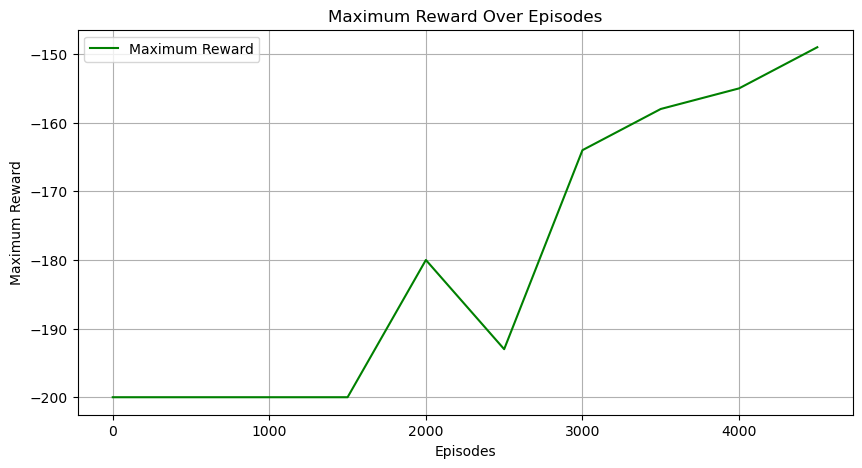


**Interpretation:**

* Initially, the average reward is close to -200, indicating that the agent fails to reach the goal.
* Over time, the average reward increases, showing that the agent is learning to solve the problem.
* By around 2000 episodes, the average reward stabilizes, suggesting that the agent has converged to a near-optimal policy.
* The final average reward is around -150, meaning the agent is still not consistently reaching the goal but is performing better than random exploration. it should be noted that the agent is reaching the goal but not consistently. we can improve further by fine tuning and increasing the number of episodes which again increases the runtime

**Graph 2: Maximum Reward Over Episodes**

The maximum reward achieved in each episode.



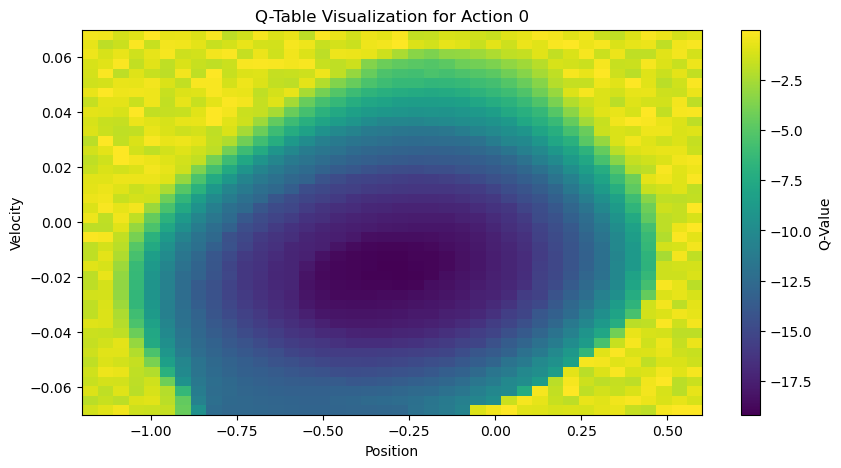
**Interpretation:**

* Initially, the maximum reward is close to -200, indicating that the agent rarely reaches the goal.
* Over time, the maximum reward improves, reaching around -150 by the end of training.
* This improvement shows that the agent is occasionally achieving better results, but it is still not consistently reaching the goal.

**Plot 3: Q-Table Visualization**

Q Graph represents the car's position in the environment (X-axis) and car's velocity, ranging from negative values (moving left) to positive values (moving right) (Y -axis).

* The color represents the Q-value for Action 0 (e.g., pushing left) at each state.
* Darker colors (e.g., blue) indicate lower Q-values (poorer actions).
* Lighter colors (e.g., yellow) indicate higher Q-values (better actions).



**Interpretation:**

* Darker regions corresponding to lower Q value, These regions correspond to states where Action 0 (pushing left) is not effective.

For example when the car is on the right side of the valley (high position) and moving right (positive velocity), pushing left is counterproductive. When the car is stuck in the bottom of the valley (low position) with low velocity, pushing left does not help build momentum.

* lighter regions corresponding to higher Q value, These regions correspond to states where Action 0 (pushing left) is effective.

For example when the car is on the left side of the valley (low position) and moving left (negative velocity), pushing left helps build momentum to eventually reach the goal.

* When the car has high velocity (either left or right), pushing left can help maintain or increase momentum.
* Transition Regions: In some regions, the Q-values are moderate, indicating that Action 0 is neither highly effective nor completely ineffective. These regions often correspond to states where the car is transitioning between building momentum and reaching the goal.

**General Analysis**

* Effectiveness of Action 0: The graph shows that Action 0 is most effective when the car is moving left on the left side of the valley.
* This makes sense because pushing left in these states helps the car build momentum to eventually climb the hill.
* In states where the car is on the right side of the valley or has low velocity, Action 0 is less effective.
* This highlights the difficulty of the Mountain Car problem, where the agent must learn to balance actions to build momentum.

**Learning Progress:**

* The Q-values reflect the agent's learning progress. Over time, the agent has learned to associate certain states with higher Q-values for Action 0.
* However, the agent still struggles in some regions, indicating that further training or tuning may be needed.

**7. Challenges Faced by the Agent**

* **Sparse Rewards:** The agent receives a reward of -1 for every time step until it reaches the goal. This makes it difficult for the agent to learn, as it only receives positive feedback when it reaches the goal.
* **Difficulty Building Momentum:** The car's engine is not powerful enough to climb the hill in a single pass.The agent must learn to move back and forth to build enough momentum to reach the goal.
* **Exploration vs. Exploitation:** Balancing exploration and exploitation is challenging, especially in the early stages of training when the agent knows little about the environment.

**8. Summary and Conclusions**

**Summary**

* The agent starts with poor performance, often failing to reach the goal.
* Over time, the agent learns to build momentum and occasionally reaches the goal.
* The average reward increases, indicating that the agent is learning to solve the problem.
* The maximum reward also improves, but the agent is still not consistently reaching the goal.

**Conclusions**

* The agent successfully learns to improve its performance in the Mountain Car environment using Q-Learning.
* However, the agent does not consistently reach the goal, suggesting that further tuning or more advanced algorithms (e.g., Deep Q-Learning) may be needed.
* The results demonstrate the effectiveness of Q-Learning for solving reinforcement learning problems like the Mountain Car, but they also highlight the challenges of learning in environments with sparse rewards.

**Recommendations for Improvement**

* **Increase Training Episodes:** Train the agent for more episodes to allow it to explore the environment further and improve its policy. (How ever time intensive)
* **Tune Hyperparameters:** Adjust the learning rate, discount factor, and exploration rate to improve learning efficiency.
* **Use Advanced Algorithms:** Consider using Deep Q-Learning (DQN) or other advanced reinforcement learning algorithms to handle the sparse reward problem more effectively.
* **Improve State Discretization:** Use a finer discretization of the state space to allow the agent to learn more precise policies.

All these things can improve the model but increase the learning time. We can also improve the learning by using techniques such as transfer learning, deep q-learning, reward shaping, etc.