EARIN – LAB 06

Aayush gupta

Wypych Dawid

The task is to implement a reinforcement learning problem using the "Taxi-v3" environment from the Gymnasium library. The game involves a taxi that can pick up passengers from one of four destinations and drop them off at a different desired destination. The agent's actions result in rewards or penalties as follows:

- Each step taken by the agent incurs a penalty of -1 point.
- Each illegal drop-off of a passenger incurs a penalty of -10 points.
- Each successful drop-off of a passenger results in a reward of +20 points, which is the highest possible reward for an episode.

The task is to implement a reinforcement learning problem using the "Taxi-v3" environment from the Gymnasium library. The game involves a taxi that can pick up passengers from one of four destinations and drop them off at a different desired destination. The agent's actions result in rewards or penalties as follows:

- Each step taken by the agent incurs a penalty of -1 point.
- Each illegal drop-off of a passenger incurs a penalty of -10 points.
- Each successful drop-off of a passenger results in a reward of +20 points, which is the highest possible reward for an episode.

There are five possible actions that the driver can take: go up, go down, go left, go right, pick up, and drop off. With 500 possible states, the Q-table will have a size of 5x500. The Q-table determines the expected reward for each action in each state.

To train the agent, three adjustable hyperparameters are considered:

1. Learning rate(alpha),: It controls the impact of new information on the Q-values. A higher learning rate means the agent gives more importance to recent experiences.
2. Discount rate(gamma): It determines the agent's preference for immediate rewards over long-term rewards. A higher discount rate values future reward more.
3. Exploration rate(epsilon): It defines the likelihood of the agent exploring a new action instead of exploiting the currently known best action. A higher exploration rate promotes more exploration.

The task is to find optimal values for these hyperparameters through experimentation and fine-tuning to achieve the highest average reward over multiple episodes.

Execution:

The Q-Learning algorithm in the following steps:

1. Initialize the Q-table: Create a Q-table with dimensions (number of states) x (number of actions) and initialize all values to zero. The Q-table represents the expected rewards for each action in each state.
2. Select an action: Based on the current state, choose an action using an exploration-exploitation trade-off. This means either selecting the action with the highest Q-value (exploitation) or choosing a random action (exploration) based on an exploration rate.
3. Perform the selected action: Execute the chosen action in the environment and observe the next state and the received reward.
4. Update the Q-value: Update the Q-value of the previous state-action pair using the Q-learning equation:
5. Q(s, a) = (1 - learning_rate) * Q(s, a) + learning_rate * (reward + discount_factor * max(Q(next_state, all_actions))) Here, learning_rate is the learning rate hyperparameter, reward is the received reward, discount_factor is the discount rate hyperparameter, and max(Q(next_state, all_actions)) represents the maximum Q-value for the next state across all possible actions.
6. Repeat steps 2-4: Repeat steps 2 to 4 for a certain number of episodes or until the agent reaches a terminal state.
7. Decay exploration rate: After each episode or step, reduce the exploration rate to gradually decrease the agent's tendency to explore and encourage exploitation of the learned Q-values.
8. Repeat the process: Repeat steps 2 to 6 until the agent has converged to an optimal policy or a desired level of performance.

The Q-Learning algorithm allows the agent to learn the optimal Q-values through repeated exploration and exploitation. By updating the Q-values based on received rewards and maximizing future rewards, the agent gradually learns to make better decisions and achieve higher rewards in the given environment.

Results:

The defined hyperparameter are:

```
num_episodes = 10000

max_steps_per_episode = 100

alpha = 0.5

gamma = 0.6

epsilon = 1.0

exploration_decay_rate = 0.01
```

```
Episode: 0, Average Reward: -4.33
Episode: 100, Average Reward: -305.76
Episode: 200, Average Reward: -148.98
Episode: 300, Average Reward: -69.05
Episode: 400, Average Reward: -23.72
Episode: 500, Average Reward: -6.04
Episode: 600, Average Reward: 1.59
Episode: 700, Average Reward: 5.23
Episode: 800, Average Reward: 6.34
Episode: 900, Average Reward: 6.79
Episode: 1000, Average Reward: 6.84
Episode: 1100, Average Reward: 6.53
Episode: 1200, Average Reward: 7.38
Episode: 1300, Average Reward: 7.76
Episode: 1400, Average Reward: 7.22
Episode: 1500, Average Reward: 7.56
Episode: 1600, Average Reward: 8.24
Episode: 1700, Average Reward: 8.06
Episode: 1800, Average Reward: 7.61
Episode: 1900, Average Reward: 8.18
Episode: 2000, Average Reward: 8.15
Episode: 2100, Average Reward: 7.55
Episode: 2200, Average Reward: 8.11
Episode: 2300, Average Reward: 8.01
Episode: 2400, Average Reward: 7.62
Episode: 2500, Average Reward: 8.18
Episode: 2600, Average Reward: 7.69
Episode: 2700, Average Reward: 7.54
Episode: 2800, Average Reward: 8.25
Episode: 2900, Average Reward: 8.22
Episode: 3000, Average Reward: 7.95
```

```
Episode: 3100, Average Reward: 8.11

Episode: 3200, Average Reward: 7.93

Episode: 3300, Average Reward: 7.48

Episode: 3400, Average Reward: 7.75

Episode: 3500, Average Reward: 7.64

Episode: 3600, Average Reward: 8.3

Episode: 3700, Average Reward: 7.61

Episode: 3800, Average Reward: 8.12

Episode: 3900, Average Reward: 7.69

Episode: 4000, Average Reward: 7.81

Episode: 4100, Average Reward: 7.75

Episode: 4200, Average Reward: 8.01

Episode: 4300, Average Reward: 7.7

Episode: 4400, Average Reward: 7.74

Episode: 4500, Average Reward: 7.7

Episode: 4600, Average Reward: 8.06

Episode: 4700, Average Reward: 8.1

Episode: 4800, Average Reward: 7.85

Episode: 4900, Average Reward: 7.67

Episode: 5000, Average Reward: 7.55

Episode: 5100, Average Reward: 7.55

Episode: 5200, Average Reward: 8.24

Episode: 5300, Average Reward: 7.94

Episode: 5400, Average Reward: 7.9

Episode: 5500, Average Reward: 8.01

Episode: 5600, Average Reward: 7.85

Episode: 5700, Average Reward: 7.87

Episode: 5800, Average Reward: 7.92

Episode: 5900, Average Reward: 7.54

Episode: 6000, Average Reward: 7.73

Episode: 6100, Average Reward: 7.6

Episode: 6200, Average Reward: 7.94

Episode: 6300, Average Reward: 7.36

Episode: 6400, Average Reward: 7.47

Episode: 6500, Average Reward: 7.68

Episode: 6600, Average Reward: 7.65

Episode: 6700, Average Reward: 7.93

Episode: 6800, Average Reward: 7.65

Episode: 6900, Average Reward: 7.91

Episode: 7000, Average Reward: 8.16

Episode: 7100, Average Reward: 7.9
```

```
Episode: 7200, Average Reward: 7.76

Episode: 7300, Average Reward: 7.78

Episode: 7400, Average Reward: 8.13

Episode: 7500, Average Reward: 8.07

Episode: 7600, Average Reward: 8.2

Episode: 7700, Average Reward: 7.73

Episode: 7800, Average Reward: 7.55

Episode: 7900, Average Reward: 7.5

Episode: 8000, Average Reward: 8.39

Episode: 8100, Average Reward: 8.58

Episode: 8200, Average Reward: 7.74

Episode: 8300, Average Reward: 7.79

Episode: 8400, Average Reward: 7.65

Episode: 8500, Average Reward: 7.56

Episode: 8600, Average Reward: 7.51

Episode: 8700, Average Reward: 7.54

Episode: 8800, Average Reward: 7.75

Episode: 8900, Average Reward: 7.88

Episode: 9000, Average Reward: 7.64

Episode: 9100, Average Reward: 7.83

Episode: 9200, Average Reward: 7.62

Episode: 9300, Average Reward: 7.69

Episode: 9400, Average Reward: 7.77

Episode: 9500, Average Reward: 8.1

Episode: 9600, Average Reward: 8.27

Episode: 9700, Average Reward: 7.74

Episode: 9800, Average Reward: 7.81

Episode: 9900, Average Reward: 7.8

train_q_learning_agenting completed.

Score over time: 1.7219

Total Reward: 17219.
```

The QLearningAgent class is defined, which holds the Q-table and methods for updating the Q-table and implementing the epsilon-greedy policy.

The QLearningAgent's update_q_table method updates the Q-value for a state-action pair based on the received reward and the maximum Q-value for the next state.

The epsilon_greedy_policy method selects an action based on an exploration-exploitation trade-off. It either chooses a random action with probability epsilon or selects the action with the highest Q-value for the current state.

The train_q_learning_agent method trains the agent by running episodes and steps within each episode. It updates the Q-table, accumulates episode rewards, and decays the exploration rate. It also prints the average reward every 100 episodes.

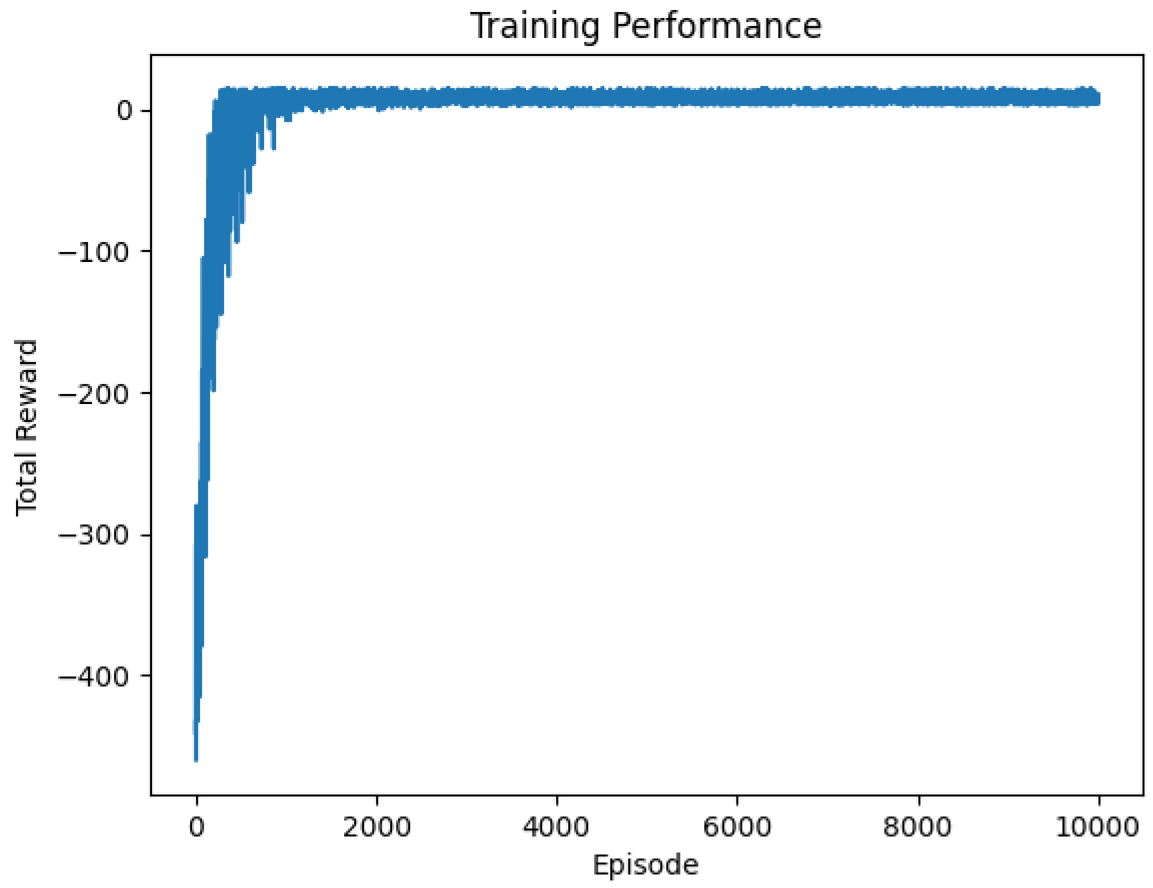
The hyperparameters are defined, including the number of episodes, maximum steps per episode, learning rate (alpha), discount rate (gamma), initial exploration rate (epsilon), and exploration rate decay rate.

The environment (Taxi-v3) and agent are created using the Gymnasium library.

The agent is trained using the train_q_learning_agent method, which returns the rewards and exploration rates over the training episodes.

The total reward is computed by summing the rewards obtained in each episode.

The graphs below show Training performance and exploration decay rate:



Training Performance Graph:

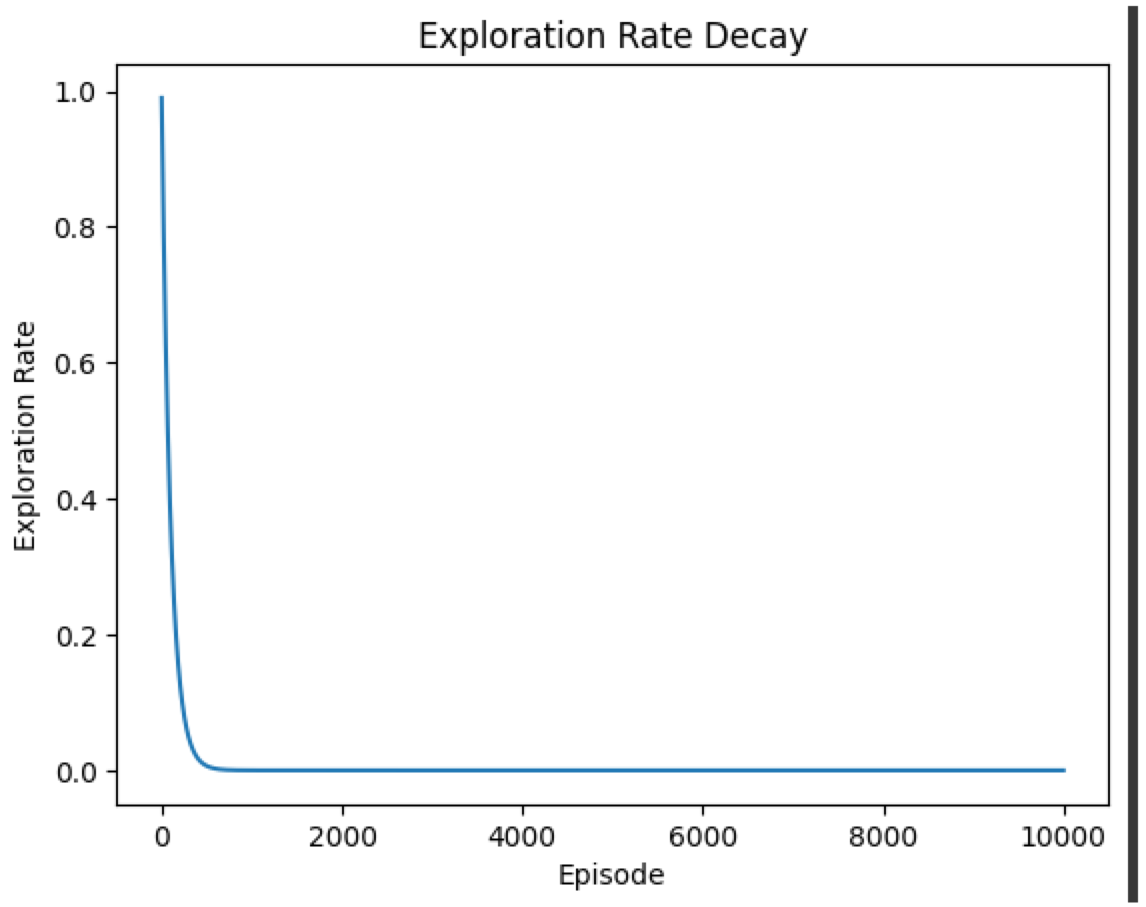
This graph shows the training performance of the agent over episodes.

The x-axis represents the episode number, and the y-axis represents the total reward obtained in each episode.

Each data point on the graph represents the total reward achieved in a single episode.

The line connecting the data points helps visualize the trend and progression of the agent's performance over time.

The graph allows you to analyze how the agent's performance improves or changes throughout the training process



Exploration Rate Decay Graph:

This graph shows the decay of the exploration rate over episodes.

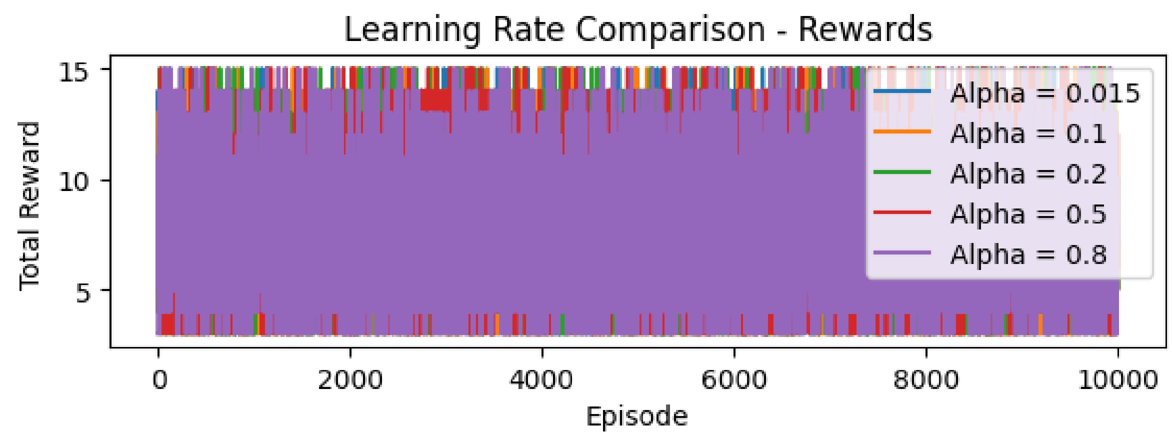
The x-axis represents the episode number, and the y-axis represents the exploration rate.

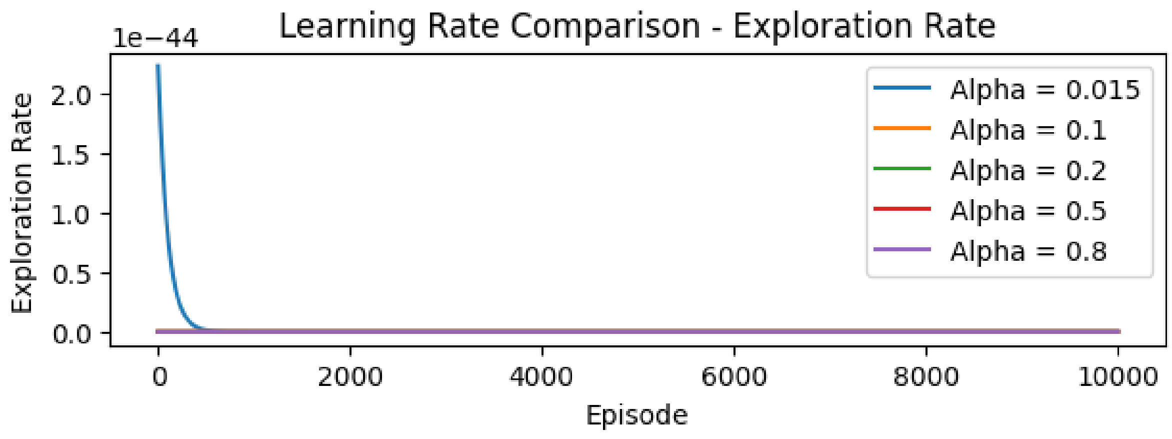
Each data point on the graph represents the exploration rate at a specific episode.

The line connecting the data points helps visualize the rate at which the exploration rate decreases over time.

The graph allows you to observe how the agent's exploration rate changes during training and how it adapts to the environment.

Learning step(Alpha) checking for different values.

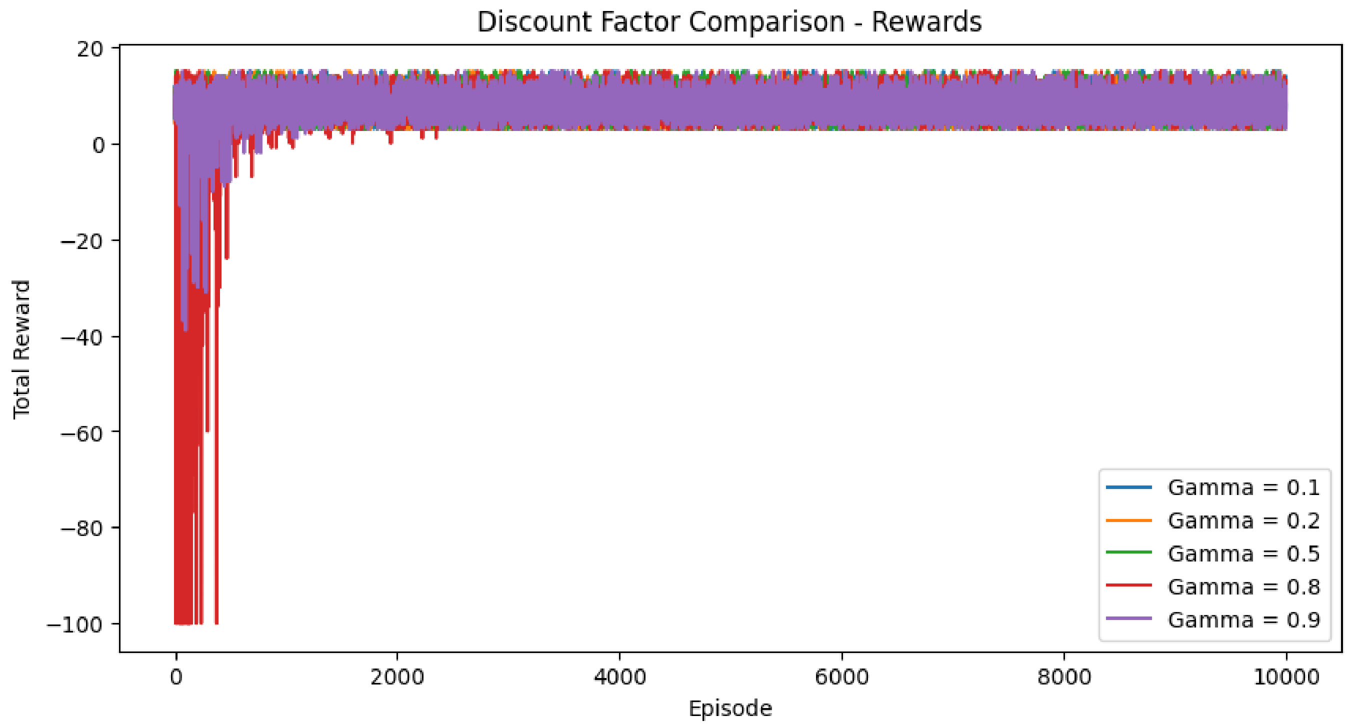




- As we can observe from the generated graphs, we experimented with different learning rates ranging from 0.02 to 0.8. Increasing the learning rate has a significant impact on the learning process of our Q-learning agent. The learning rate determines how much weight is given to newly observed information when updating the Q-table values.
- By adjusting the learning rate, we can control the rate at which the agent incorporates new knowledge into its decision-making process. A higher learning rate enables the agent to learn faster by giving more weight to recent experiences, which can lead to more rapid convergence to an optimal policy. However, setting a learning rate that is too high may result in the agent being overly influenced by noisy or irrelevant information, leading to instability or suboptimal performance.

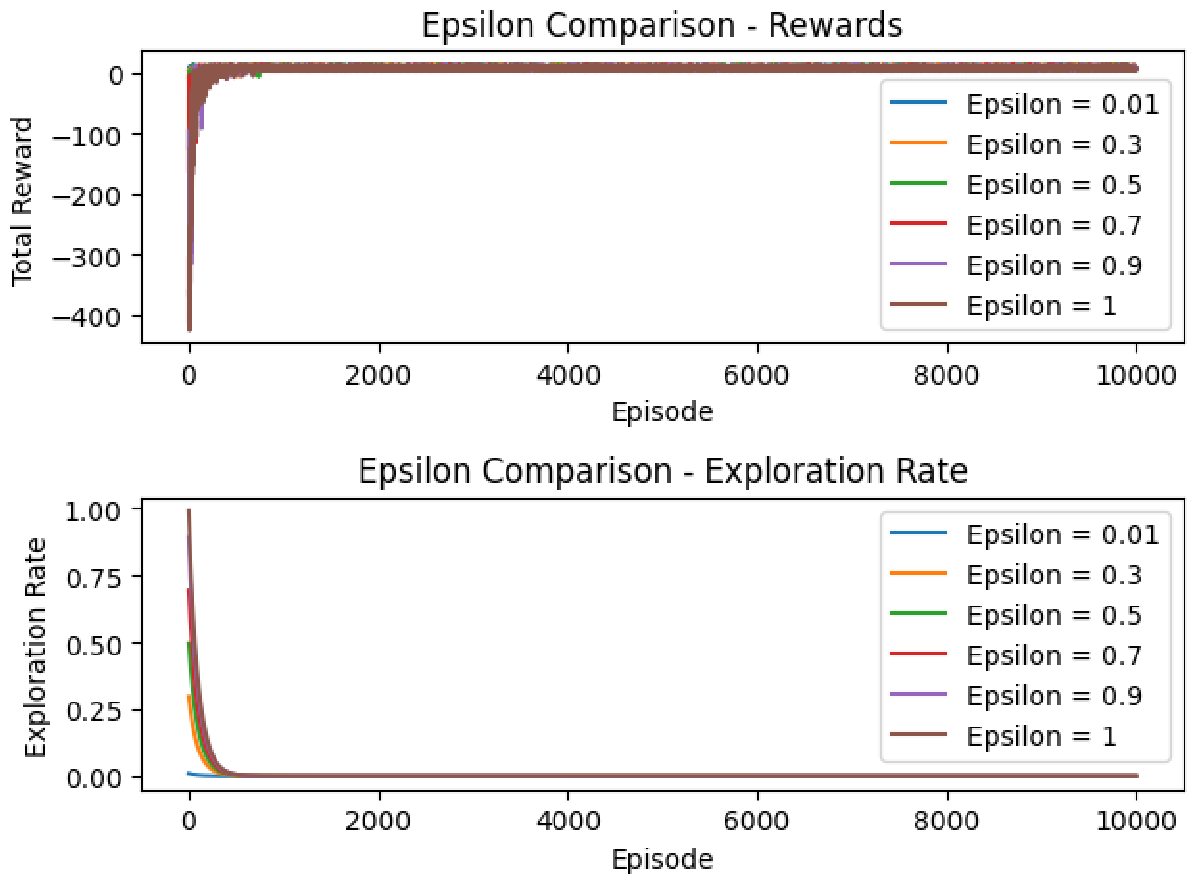
- The comparison of learning rates in the graph allows us to visualize the impact of different learning rates on the agent's performance. The rewards graph shows the total rewards obtained by the agent over the episodes for each learning rate. We can observe that different learning rates yield different convergence rates and levels of performance. The exploration rate graph displays the exploration rate decay over episodes for each learning rate, showing how the agent's exploration decreases over time as it learns.
- These graphs provide valuable insights into the relationship between learning rates and the agent's learning and exploration behaviors. They help us understand the trade-off between exploration and exploitation in reinforcement learning and assist in selecting an appropriate learning rate for achieving desired learning outcomes.

Discount rate (gamma_values) checking for diiffrent values.



- By analyzing the graph, we can observe the impact of the discount factor on the agent's learning and performance. The lines represent the trend of the total rewards over the course of the training episodes.
- A higher discount factor tends to prioritize long-term rewards, as it values future rewards more strongly. Conversely, a lower discount factor places less emphasis on future rewards and focuses more on immediate rewards.
- The graph allows us to compare the performance of the agent under different discount factors. We can observe which discount factor leads to higher total rewards and analyze the convergence and effectiveness of the Q-learning algorithm for different settings.

Exploration step (epsilons)checking for diiffrent values.



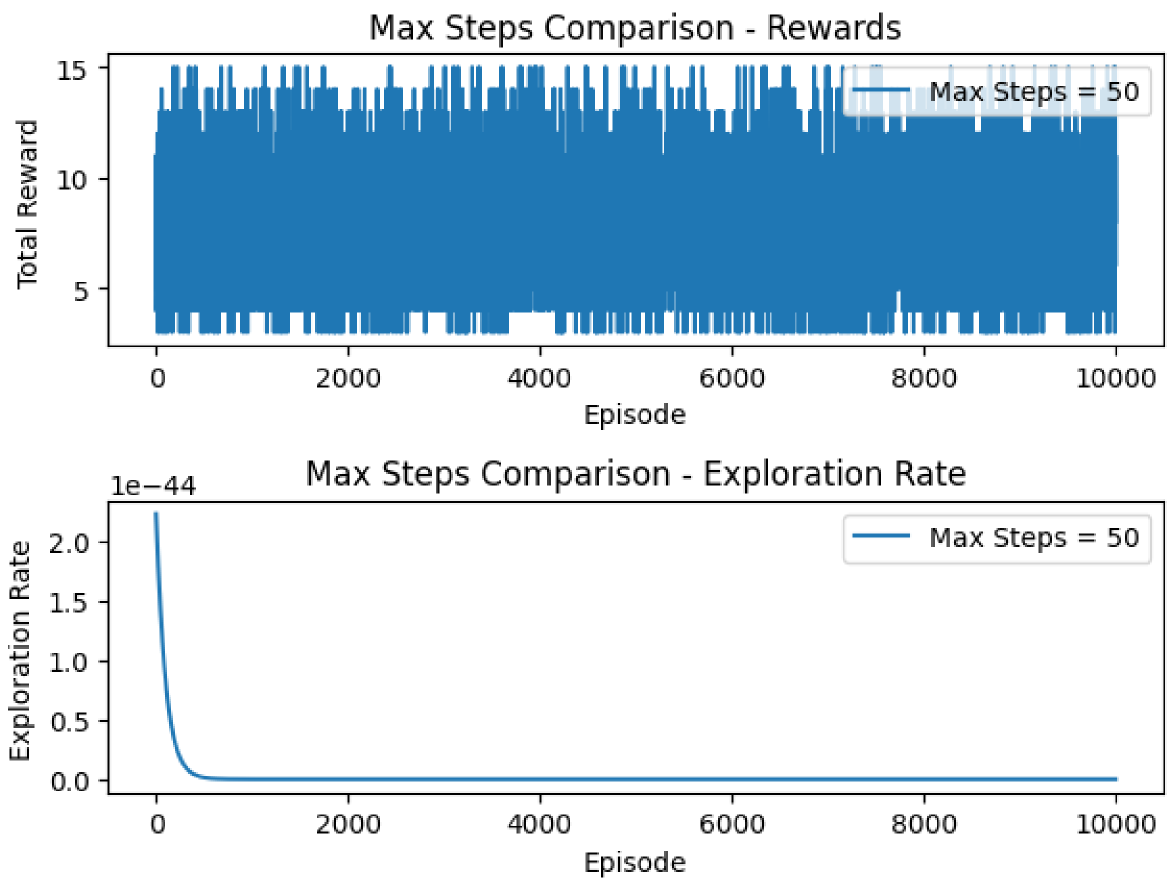
- compares the rewards and exploration rates for different epsilon values in the Q-Learning algorithm. Epsilon is a parameter that determines the exploration-exploitation trade-off in reinforcement learning.
- For each epsilon value, the code trains the Q-Learning agent using the specified epsilon value and other fixed hyperparameters (such as gamma and alpha). It then records the rewards and exploration rates during the training process.
- The rewards are plotted in the top graph, where each line represents a different epsilon value. The x-axis represents the number of episodes, and the y-axis represents the total reward accumulated by the agent in each episode. The graph allows us to compare the performance of the agent for different epsilon values over the training period.

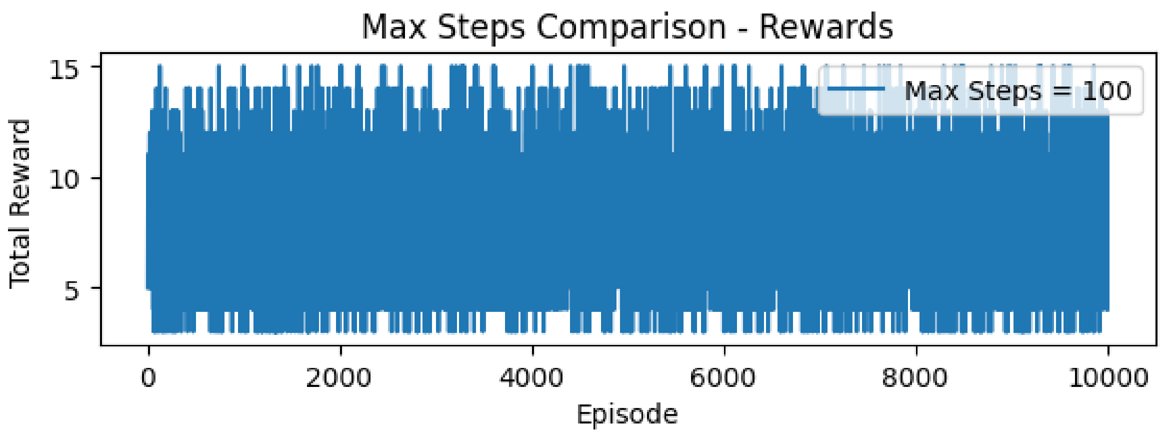
- The exploration rates are plotted in the bottom graph, again with each line representing a different epsilon value. The x-axis represents the number of episodes, and the y-axis represents the exploration rate, which indicates the likelihood of the agent choosing to explore new actions instead of exploiting the known ones. The graph provides insights into how the exploration rate changes over time for each epsilon value.

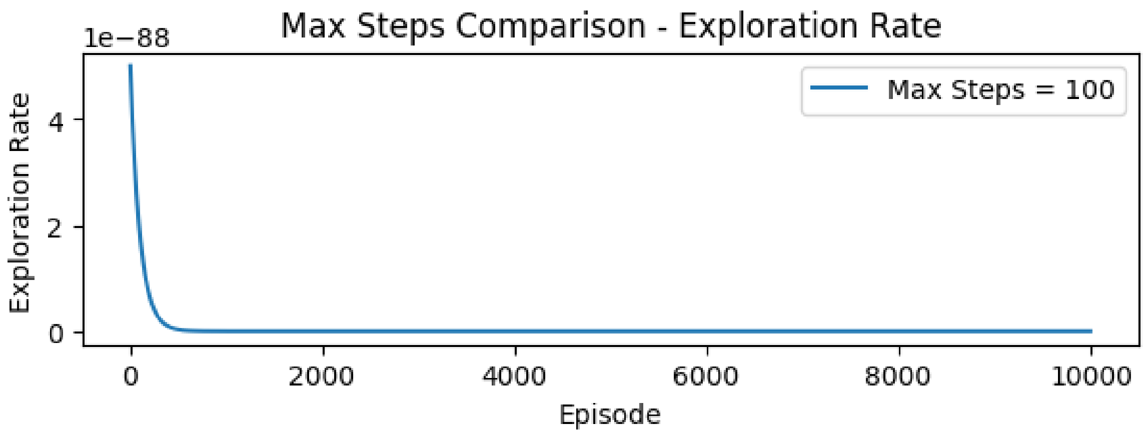
In our case, we observed that an epsilon value of 1 (maximum exploration) was found to be the optimal choice for our Q-Learning algorithm. This indicates that during the training process, the agent heavily focused on exploring new possibilities rather than exploiting known actions.

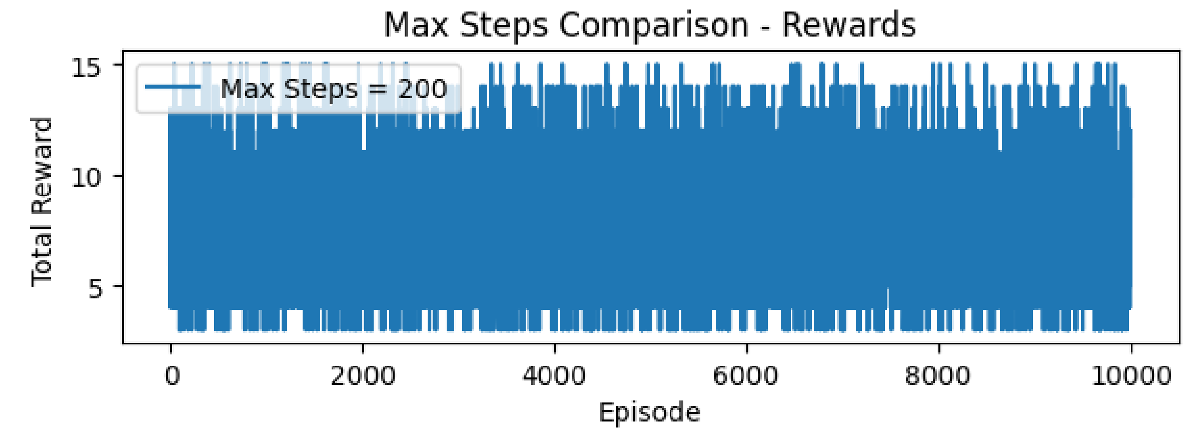
By setting epsilon to 1, the agent explored a wide range of states and actions, which allowed it to discover more optimal paths and maximize the potential rewards. This exploration phase helped the agent gather valuable information about the environment and learn from the feedback received through rewards and penalties.

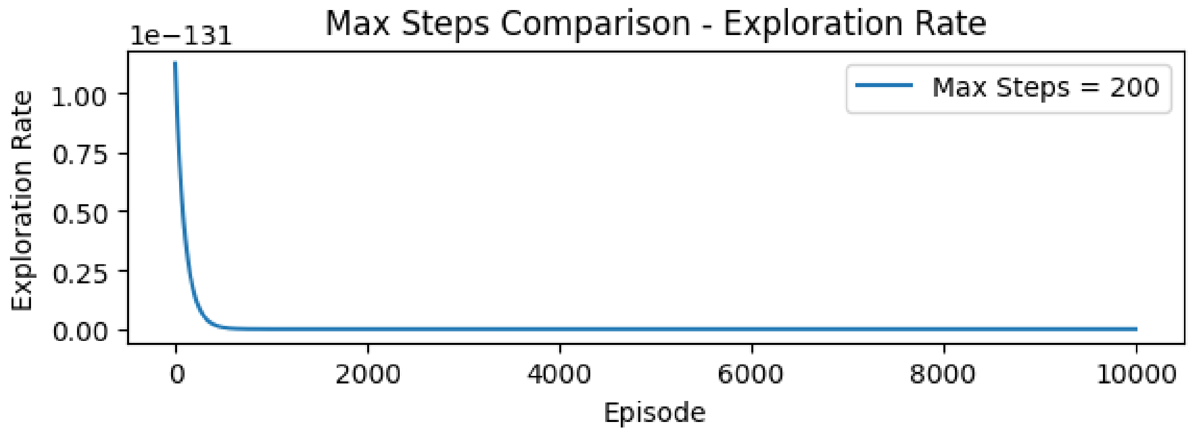
max_steps_values checking for different values

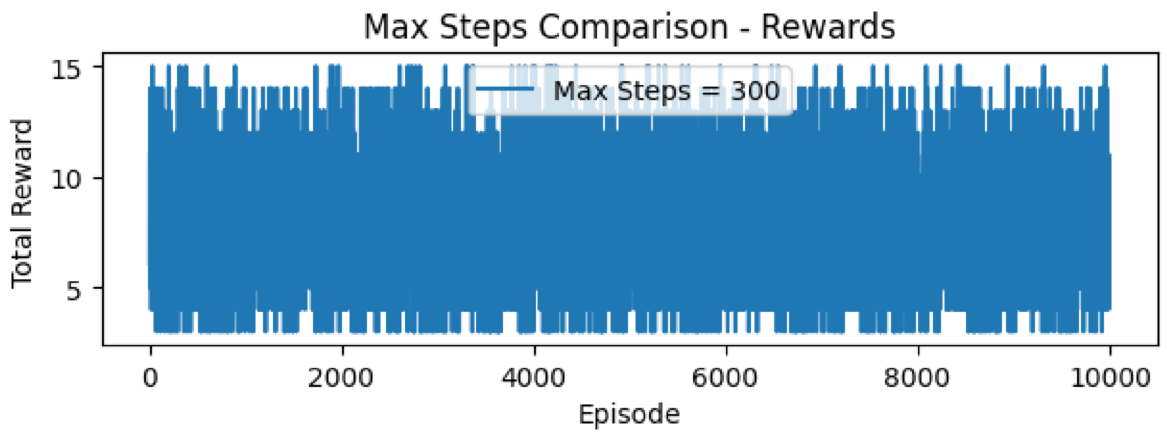


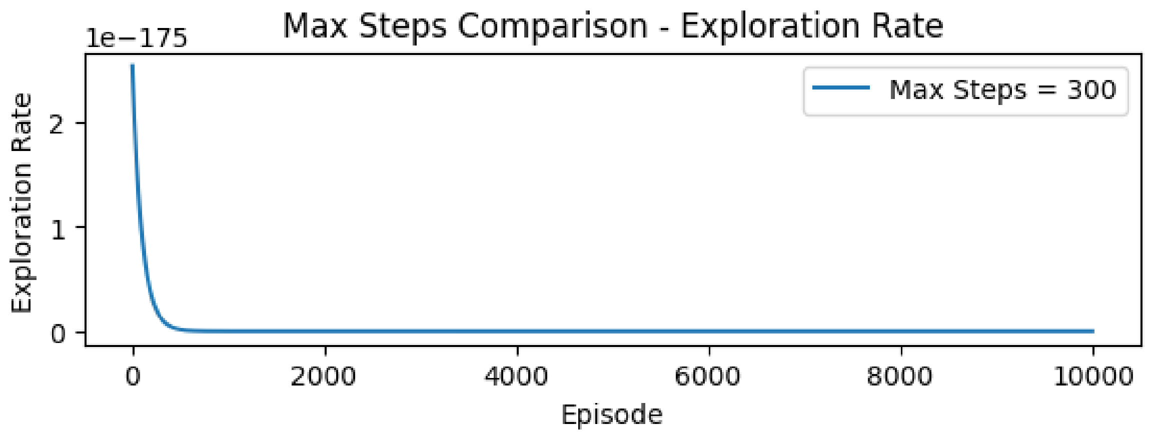

Max Steps Comparison - Rewards
Total Reward
15
10
5
0
2000
4000
6000
8000
10000
Episode
Max Steps = 100

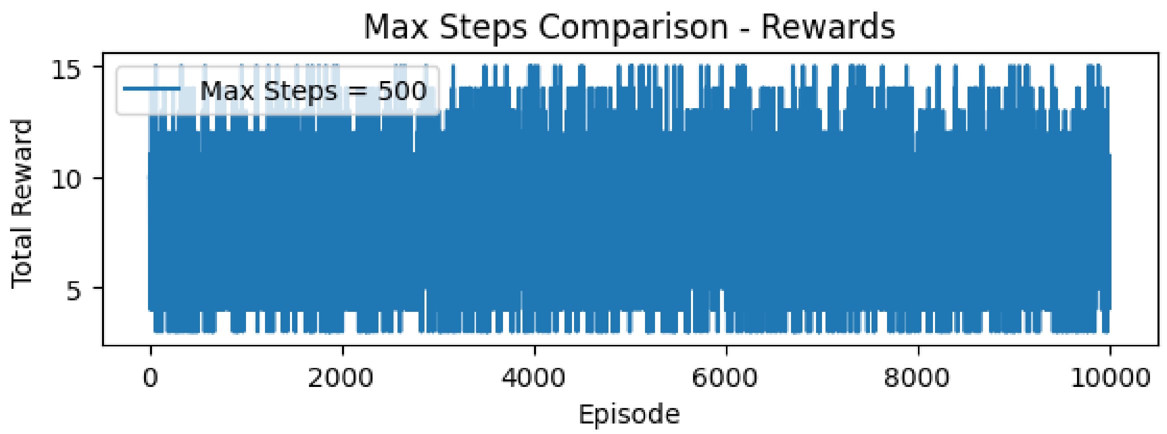

Max Steps Comparison - Exploration Rate
1e−88
Exploration Rate
4
2
0
0
2000
4000
6000
8000
10000
Episode
Max Steps = 100

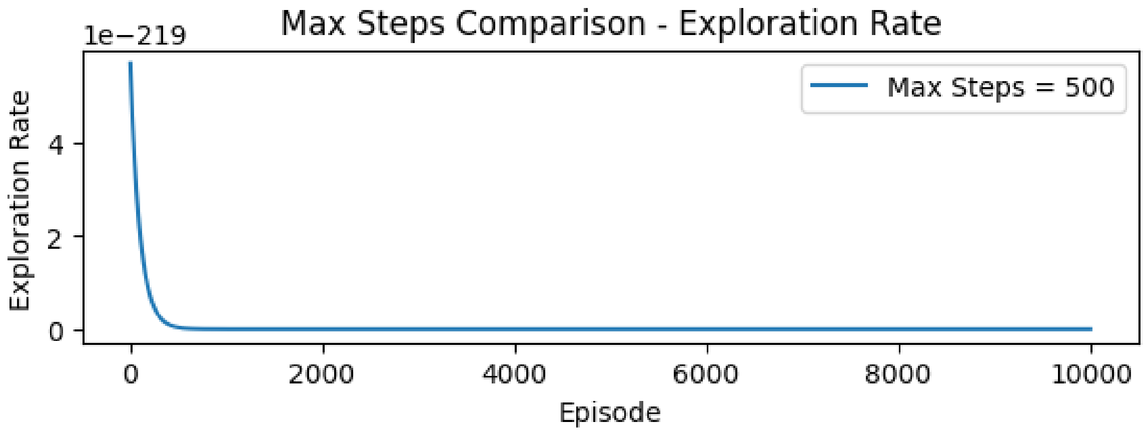

Max Steps Comparison - Rewards
Max Steps = 200
Total Reward
15
10
5
0
2000
4000
6000
8000
10000
Episode


Max Steps Comparison - Exploration Rate
1e−131
Max Steps = 200
Exploration Rate
1.00
0.75
0.50
0.25
0.00
0
2000
4000
6000
8000
10000
Episode


Max Steps Comparison - Rewards
Total Reward
15
10
5
Max Steps = 300
0
2000
4000
6000
8000
10000
Episode


Max Steps Comparison - Exploration Rate
1e−175
Exploration Rate
2
1
0
Max Steps = 300
0
2000
4000
6000
8000
10000
Episode

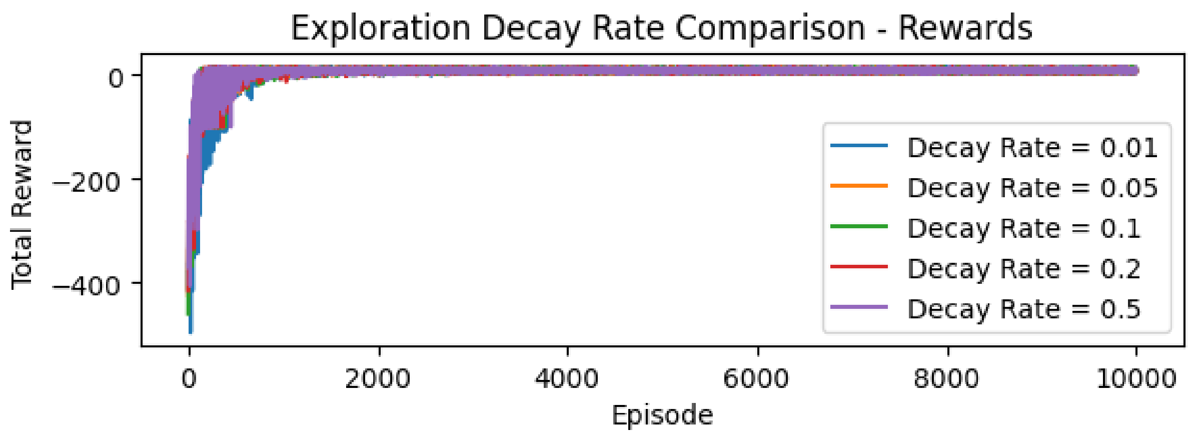


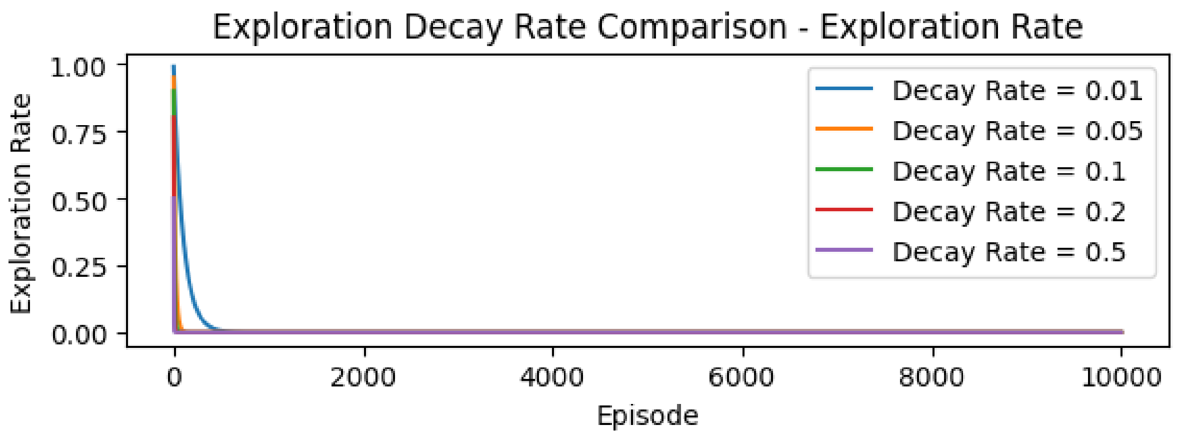


The graph shows the comparison of rewards and exploration rates for different maximum steps per episode values: [50, 100, 200, 300, 500].

- We observe that a higher maximum steps value allows the agent to explore the environment for a longer duration in each episode. This can be beneficial in complex environments where reaching the goal state may require more steps. However, we should consider the trade-off between the number of episodes required for convergence and the time taken per episode.
- In our specific environment, we find that the maximum steps values of [50, 100, 200, 300, 500] provide a good balance. The agent is able to converge to the optimal solution within a reasonable number of episodes, and it is able to reach the goal state within the maximum steps limit.
- It is important to note that if the maximum steps per episode is too low, the agent may not have enough steps to reach the goal state, leading to suboptimal performance. On the other hand, if the maximum steps per episode is too high, the training process may take longer without significant improvement in performance.
- Based on the results, we can conclude that the maximum steps per episode values [50, 100, 200, 300, 500] are suitable for our environment. However, if the environment becomes more complex, we may need to adjust this parameter accordingly.

Exploration_decay_rate checking for different values





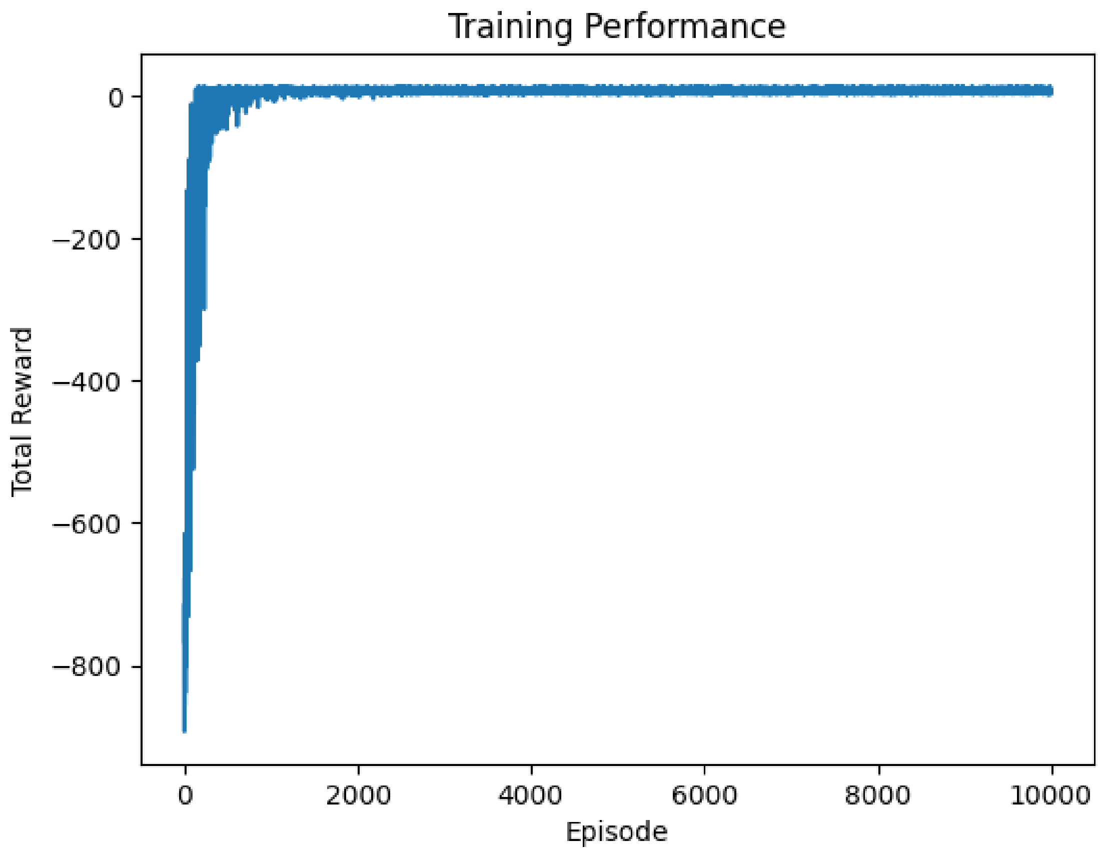
In the plotted graph, we can observe the impact of different exploration decay rates on the learning process. The default decay rate allows for a balanced exploration and exploitation trade-off, ensuring that the agent explores different states and actions while gradually shifting towards exploiting the learned knowledge.

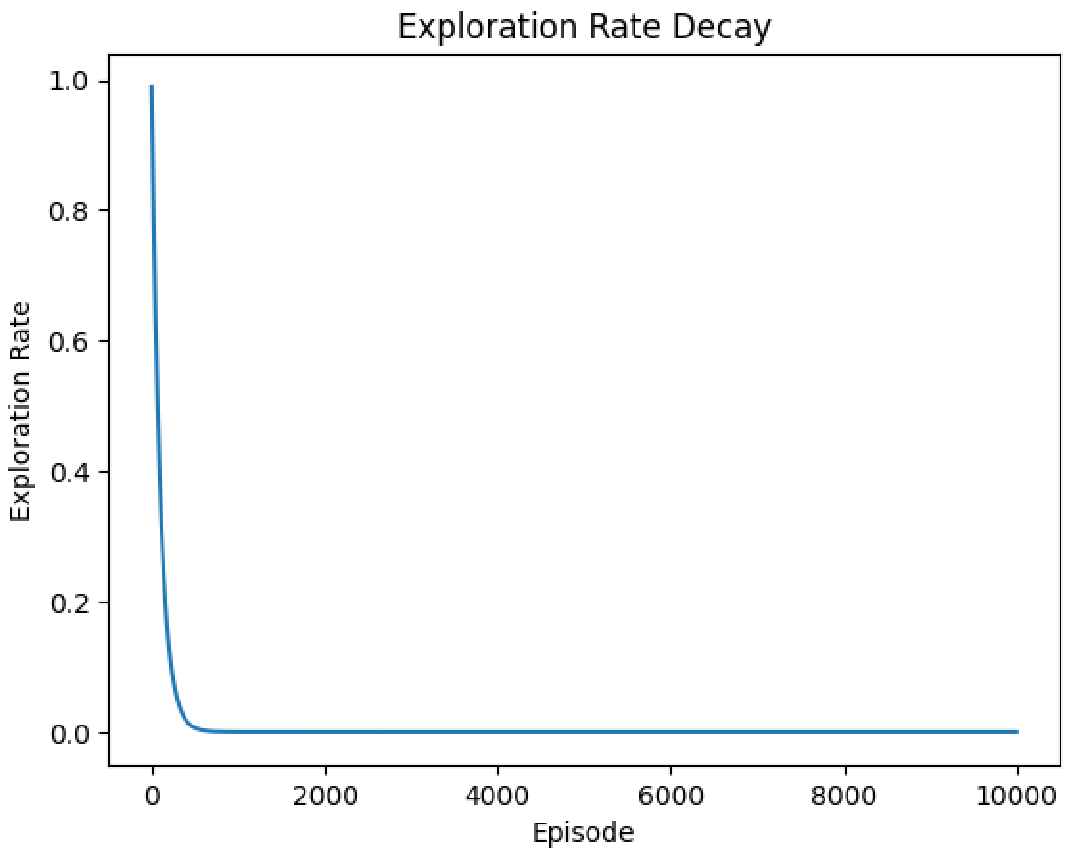
However, when the exploration decay rate is set to a higher value, such as 0.1, we can see that the agent quickly focuses on exploiting the known routes. As a result, the reward increases rapidly initially, giving the impression of faster convergence. However, this approach may not be optimal because the agent may not have explored all possible combinations of states due to the fast decay rate.

- if the exploration decay rate is set to a very high value, such as 0.5, the agent heavily prioritizes exploitation over exploration. This can lead to suboptimal performance as the agent may not discover optimal routes or fail to adapt to changing conditions.

Therefore, it is important to strike a balance when setting the exploration decay rate. It should not be too high, as it would prioritize exploitation too early in the learning process, limiting the agent's ability to discover new, potentially better routes. Finding the right decay rate ensures that the agent explores the environment adequately while gradually transitioning towards exploiting the learned knowledge.

Final Define hyperparameters





Exploration Rate Decay
Exploration Rate
Episode
1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000
8000
10000