SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF INFORMATION AND AUTOMATION ANTONIO RUBERTI

Master of Science of Engineering in Computer Science
Artificial Intelligence & Machine Learning

Professor:
Fabio Patrizi

Students:
Stefano Rucci

Academic Year 2022/2023

Contents

1 INTRODUCTION

The project aimed to address the challenge of training an agent to navigate through the "CliffWalking-v0" environment in Gym. The agent's task was to avoid falling off the "cliff" while maximizing its cumulative rewards.

"CliffWalking-v0" is a reinforcement learning environment provided by OpenAI Gym. In this 4x12 grid world, the agent starts at the top-left corner and aims to navigate to the bottom-right corner (state 47) while avoiding a hazardous area known as the 'cliff' (states 37-46). The agent can take four possible actions: move up, right, down, or left.

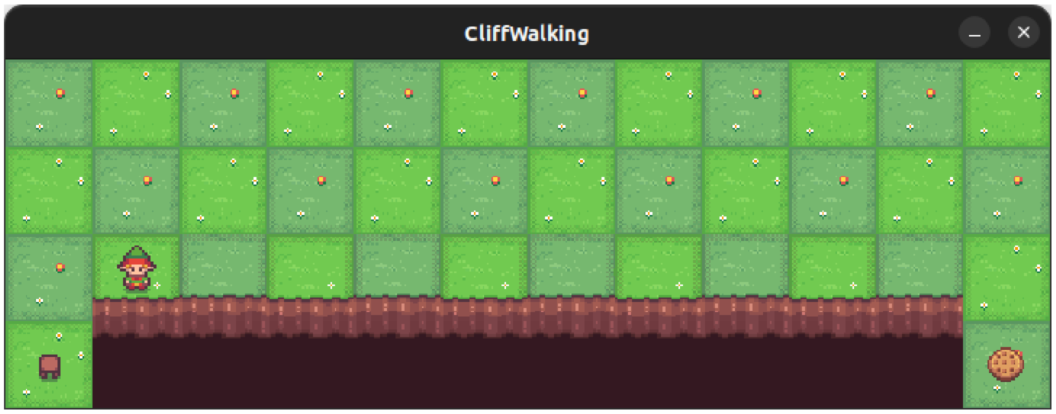

Figure 1.0.1: The environment

The rewards for each step are typically -1, but falling off the cliff incurs a penalty of -100. The environment resets after reaching the terminal state or falling off the cliff. The goal is for the agent to learn an optimal policy, finding the shortest path to the goal while avoiding the cliff.

2 Q-LEARNING ALGORITHM

In the initial phase, I employed the Q-Learning algorithm to facilitate the agent's learning of an optimal policy through iterative episodes.

Q-Learning is a machine learning algorithm used in the field of reinforcement learning to teach an agent to take actions in an environment in order to maximize a 'value function' called the Q-value. The Q-value reflects the advantage of taking specific actions in particular states, and the algorithm updates these values through exploration and exploitation, ultimately guiding the agent towards an optimal policy.

2.1 ARCHITECTURE OF THE SOLUTION

Our implementation begins by creating a Q-table, a table that tracks Q values for each State-Action pair. These Q values represent the utility of taking a particular action in a specific state. Initially, all values in the Q-table are initialized to zero.

The training process starts by iterating through a specified number of episodes. For each episode, the agent explores the environment, making decisions based on a strategy that balances exploration and exploitation. The agent decides whether to explore new actions or exploit actions with higher Q values. Exploration allows the agent to discover new information about the environment.

After each action, the Q-table is updated using the Q-Learning algorithm. The agent learns to associate rewards with actions and dynamically updates Q values to reflect this knowledge. The update equation includes key parameters such as the learning rate and discount rate, influencing how much the agent learns and how much weight is given to future rewards.

An interesting aspect of the code is the implementation of the exploration strategy. Initially, the agent explores with a higher probability to uncover the terrain. Over time, the probability of exploration decreases, allowing the agent to exploit acquired knowledge.

The code concludes by printing rewards for each episode. The agent's final strategy is reflected in the Q values associated with states and actions.

In conclusion, through this iterative process, the agent learns an optimal strategy to navigate the environment, avoiding the 'cliff' and maximizing total rewards. This example demonstrates how the Q-Learning algorithm can be successfully applied to address reinforcement learning problems.

2.2 RESULTS

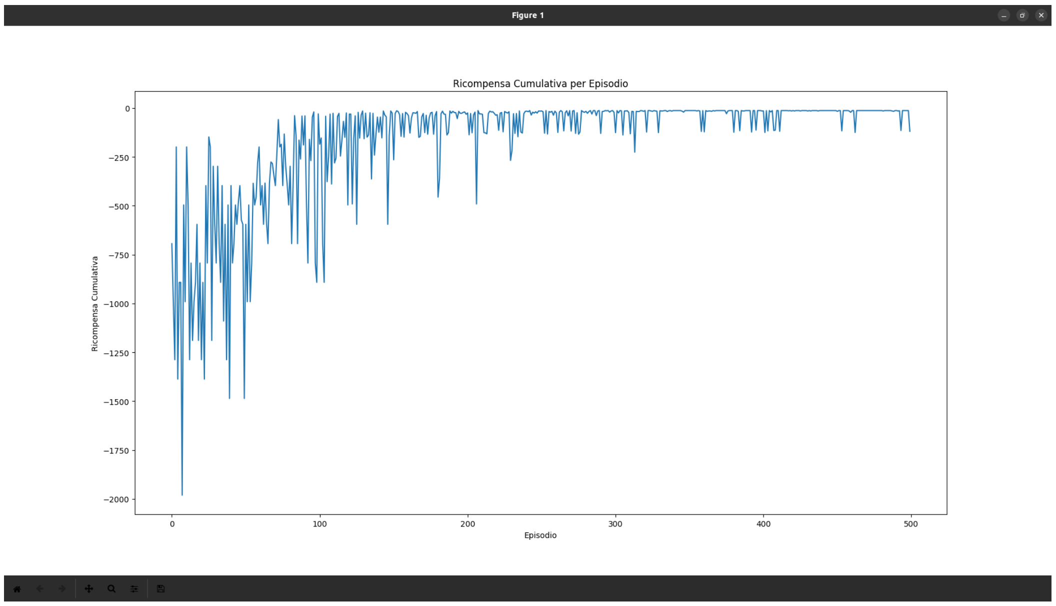

Figure 2.2.1: Cumulative Reward for each episode

3 DEEP Q-NETWORK (DQN)

Q-Learning is a method of learning for table-based models and may not be effective in environments with very large state and action spaces. In such cases, more advanced approaches like Deep Q-Networks may be preferable. This methodology extends the principles of Q-Learning, introducing innovative elements that make it more effective in complex environments.

Unlike traditional Q-Learning, DQN utilizes deep neural networks to approximate the Q-function, representing the quality of taking an action in a specific state. This proves particularly advantageous in environments with more complex state and action spaces, as neural networks are more adept at handling this complexity compared to the tables used in Q-Learning.

A key element of DQN is the introduction of "replay experiences" or replay buffer. This buffer stores past agent experiences, allowing for random sampling during training. This enhances the stability of learning by introducing greater diversity in the experiences considered.

3.1 ARCHITECTURE OF THE SOLUTION

Transitioning from Q-Learning, our Deep Q-Network (DQN) takes a step forward by integrating deep neural networks. This enhanced approach proves effective in tackling the complexities of dynamic environments.

Similar to our previous Q-table setup, the neural network, with its multiple layers, excels at capturing detailed state patterns. Unlike the static Q-table, it allows our agent to discern intricate relationships within the environment.

DQN introduces experience replay, a key element diversifying training. By periodically revisiting past experiences, the agent gains richer insights, fostering robust learning and adaptability.

Following the Q-Learning strategy, our DQN balances exploration and exploitation. Initially favoring exploration, the agent unravels environment intricacies. As knowledge accumulates, it gradually shifts towards exploitation, using acquired insights for optimal decision-making.

The neural network's dynamic learning involves continuous updates, linking rewards with actions. Key parameters like learning rate and discount rate shape learning depth and influence future reward prioritization.

A notable feature is the exploration strategy, starting with a higher probability and gradually diminishing. This empowers the agent to exploit acquired knowledge effectively.

In conclusion, the agent's learning journey manifests in cumulative rewards across episodes. The final strategy, embedded in learned Q values, showcases the agent's adeptness in navigating complexities and optimizing rewards. This shift exemplifies DQN's effectiveness in addressing reinforcement learning challenges.

3.2 RESULTS

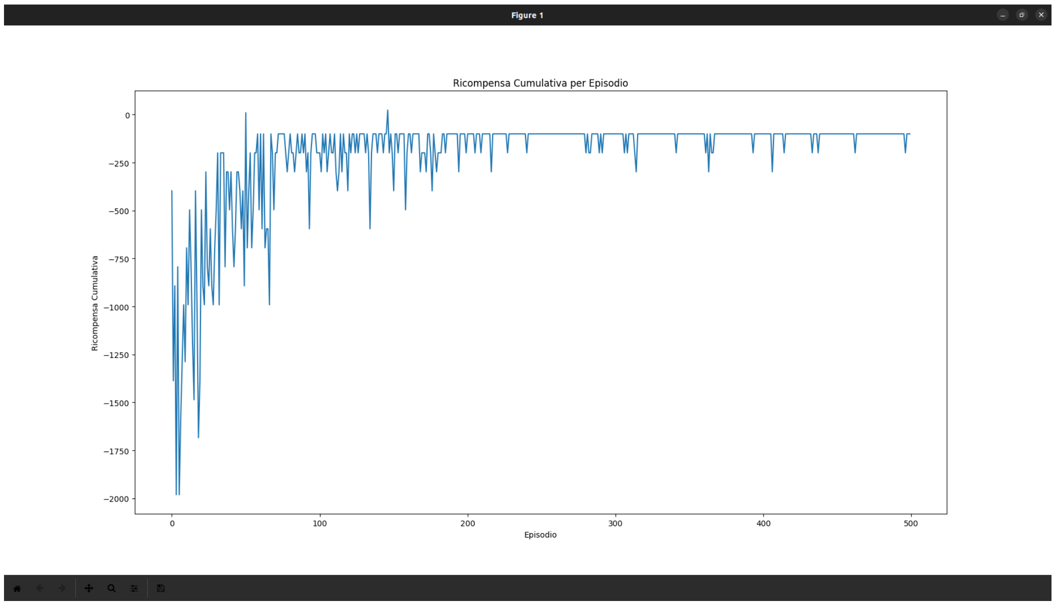

Figure 3.2.1: Cumulative Reward for each episode

4 CONCLUSION

In conclusion, our exploration of both Q-Learning and Deep Q-Network solutions demonstrates the adaptability of reinforcement learning techniques to address distinct challenges. While Q-Learning excels in simpler environments, the Deep Q-Network architecture proves invaluable in handling more complex scenarios. The choice between these methods depends on the specific characteristics of the environment, showcasing the importance of selecting an appropriate algorithmic approach tailored to the problem at hand. Through these endeavors, we gain valuable insights into the dynamic landscape of reinforcement learning and its applications in navigating challenging environments.