Take The L - Reinforcement Learning

Artificial Intelligence

Bachelors in Informatics and Computer Engineering

Mário Travassos - up201905871@edu.fe.up.pt
Rita Mendes - up201907877@edu.fe.up.pt
Tiago Rodrigues - up201907021@edu.fe.up.pt

Group 37_2.1C

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Project Specification

The objective of this project is to implement a simplified version of the game Exactly One Mazes that we had developed in the previous project delivery, and create an agent that can figure out how to win the game

The objective of the game is to have the player cross from the bottom-left to the top right square, without crossing the same L shape more than one time

The Player can move in any direction, as long as the square he is in does not belong to an already visited L shape

Tools and Algorithms to be used

The assignment will make use of the following **tools**:

- **OpenAI Gym** - A framework that acts as a playground for testing agents, using controlled environments;
- **Numpy** - A library for processing data in arrays;
- **Matplotlib** - Data visualization libraries to draw plots and charts;
- **Jupyter Notebooks** - Interactive computing and development.

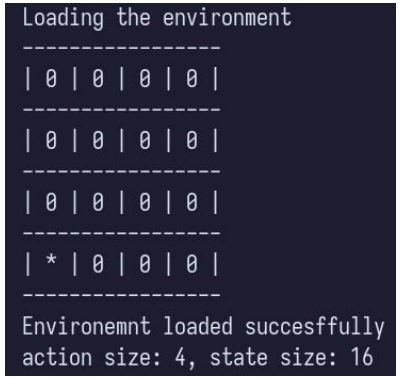
To teach the agent, the following **algorithms** will be employed:

- **Q-Learning** - State–action–reward–state;
- **SARSA** - State–action–reward–state–action.

Already Implemented Features

As of the first checkpoint, we have set up a simplified version of an OpenAI Gym environment where the agent will be trained.

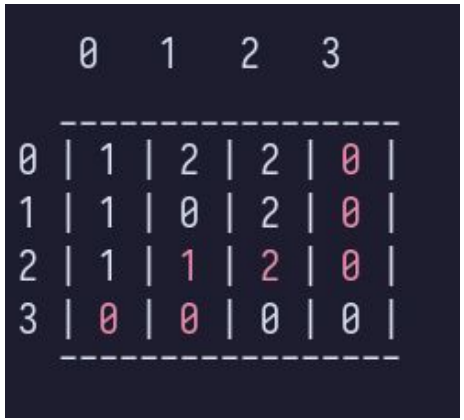
This includes the definition of the **action space**, the **observation space**, and a simple **reward function**.



There is also a stub for the agent to be developed.

Currently, the environment is being rendered on the command line only.

Simple Model



Hyperparameters used:

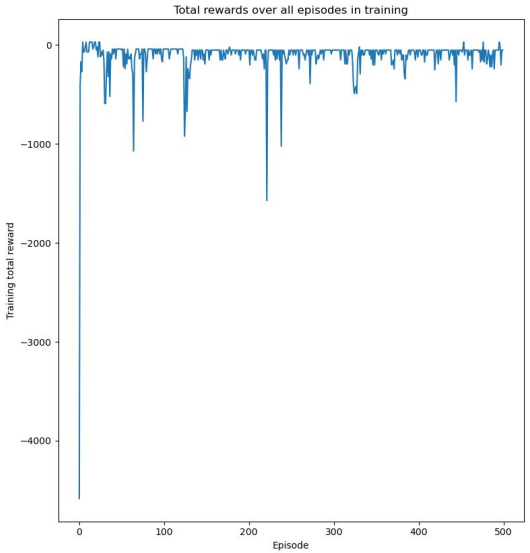
Learning rate: 0.5
Discount factor: 0.6
Epsilon: 0.1 and 0.01 decay rate
Total training episodes: 500 with 100 steps maximum

In the easy mode, a 4x4 board was used with two L shapes, which are close enough to be easy to figure out a path.

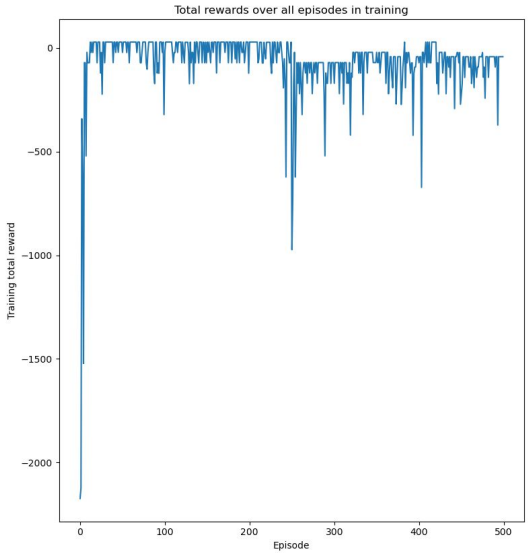
The rewards were distributed in such a way that the algorithm was severely punished for every bad move:

- If visiting an already visited cell, -50 points
- If visiting an L shape for the second time, -50 points
- If it reached the end without visiting all L shapes, -50 points
- If landing in a neutral (0) cell, -0.5 points
  - This was made so the algorithm would reach the end as fast as possible
- If visiting a newly discovered L shape, +10 points
- If it reached the end and visited all L shapes, +10 points

Simple Model - Analysis



Q-Learning with 35.2% success rate

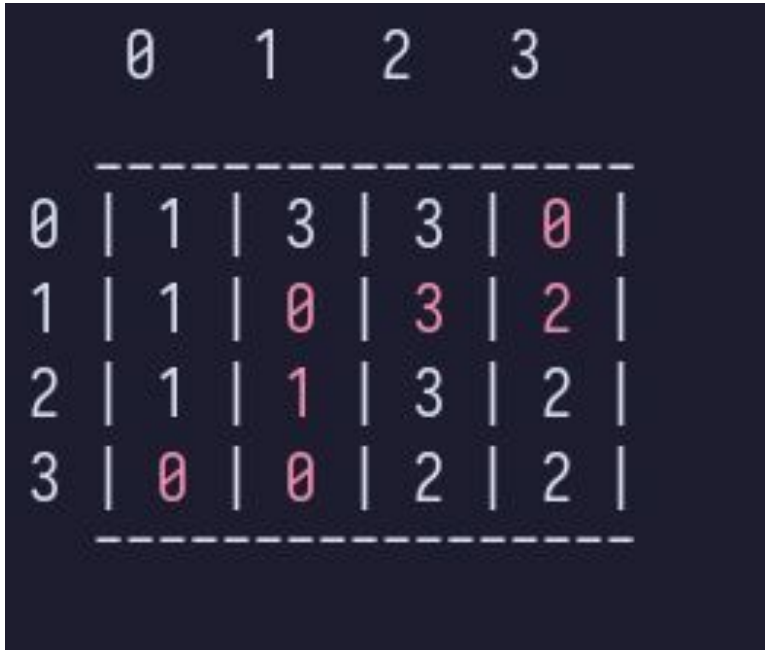


SARSA with 47.2% success rate

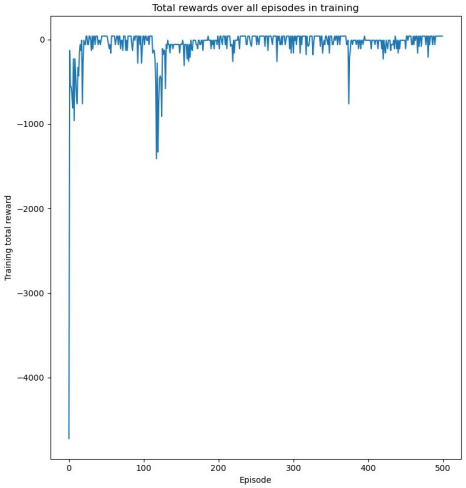
Complex Model

In the more complex model, the same hyperparameters were used, as they proved to function well for both cases

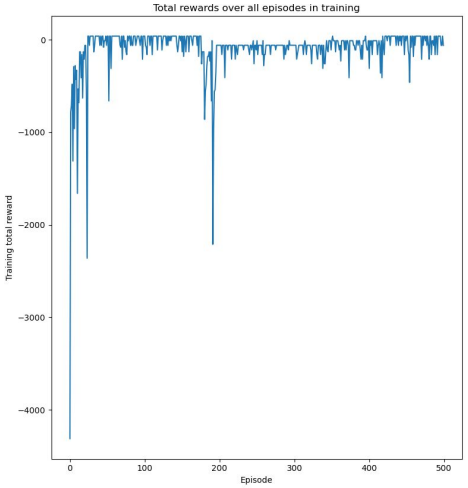
As the root of the problem didn’t change much, the rewards also stayed the same, only increasing the punishment for visiting a 0 cell to -0.75.



Complex Model - Analysis



Q-Learning with 48.4% success rate



SARSA with 46.6% success rate

References

Game References

Exactly One Mazes

Algorithms References

Q-Learning in Python

SARSA Learning

Monte Carlo Learning

Tools Used

OpenAI Gym Environments

Numpy Docs

Jupyter Docs