Neutreeko

André Gomes - 201806224
Gonçalo Teixeira - 201806562
Luís Recharte - 201806743

Specification of Reinforced Learning Problem

Game Logic

- Create Starting Condition
- Possible moves
- Manage player turns
- Perform a move and update the board
- Verify end game and tie conditions

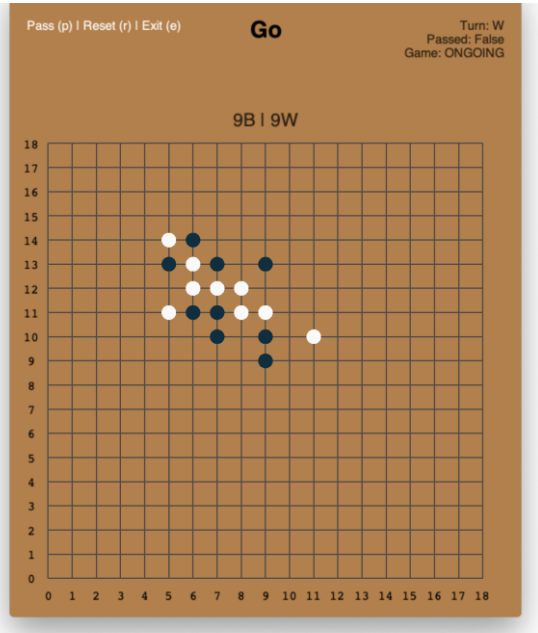
Environment

- Step – Apply the Agent action to the environment
- Reset – Reset the env
- Render – Print a representation of the env
- Close – Finish the episode
- Done – Check if episode is done

Agents

- Random choice Agent
- Best Choice Agent
- Q-learning Agent
- **SARSA Agent**
- **Monte Carlo Agent**

Related Work & References



Board games with OpenAI Gym:

- Abalone – https://github.com/towzeur/gym-abalone
- Go – https://github.com/aigagror/GymGo

Spaces' definition – https://github.com/openai/gym/tree/master/gym/spaces

Create an environment – https://github.com/openai/gym/blob/master/docs/creating-environments.md

Table of environments – https://github.com/openai/gym/wiki/Table-of-environments

States, Observation and Action Spaces in Reinforcement Learning - https://medium.com/swlh/states-observation-and-action-spaces-in-reinforcement-learning-569a30a8d2a1

Q-Learning Agent - https://medium.com/swlh/introduction-to-q-learning-with-openai-gym-2d794da10f3d

Tools and Algorithms

Anaconda environment – Python 3.8

JetBrains IntelliJ

Gym, Numpy

Agents Algorithms

Work already carried out

1. Started the structure of a OpenAI Gym project
2. Implemented the game logic
3. Implementation of the game loop and the environment. Random Agent to play the game
4. Define the rewards and their values. Implement the remaining agents
5. Transpose code to a Jupyter notebook and run tests