REINFORCEMENT LEARNING

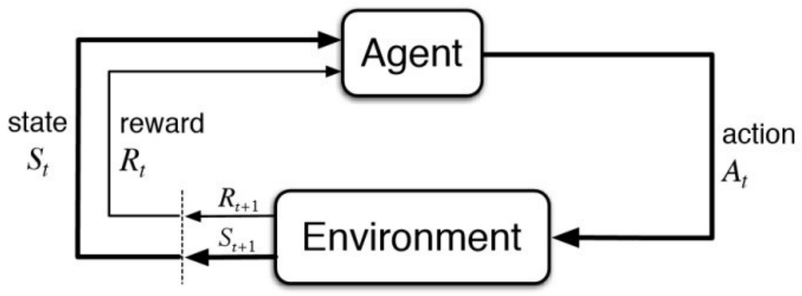
PRIFYSGOL BANGOR UNIVERSITY

Supervisor
Prof. Ludmila Kuncheva
l.i.kuncheva@bangor.ac.uk

Author
Sam Hennessey
sml18vly@bangor.ac.uk

Aims & Objectives

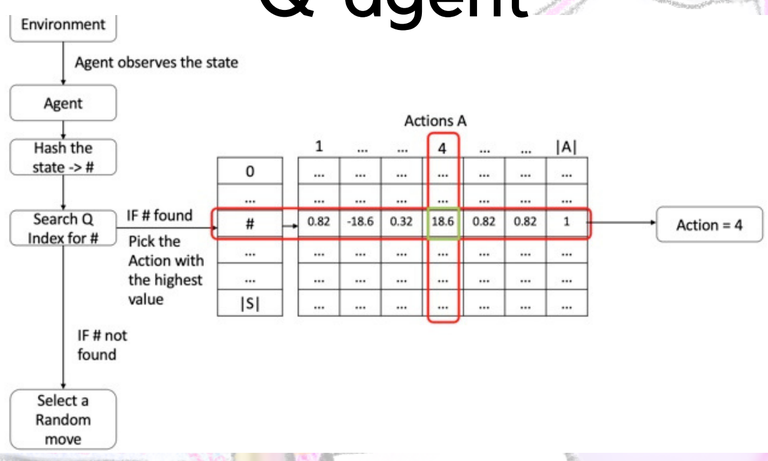
Aim: Create a reinforcement learning agent to play in Kaggles Connect-X competition



Objectives:

- Research ares of RL
- Describe basic RL algorithms and test them with examples
- Create a learning environment and train a RL agent
- Enter the agent into the Connect-X competition

Q-agent

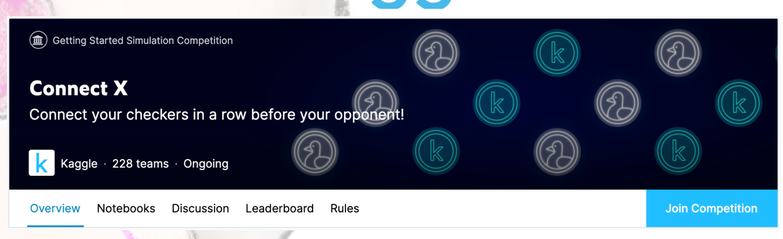


Agent Algorithm:

- Hash the board and look for it in the state index
- if found; Search the a table for the move values associated with the state and choose the one with the highest value
- else; Add the state to the state index and add a new row to the q-table to represent the action values (all 1,s) and choose a move at random

Problem

kaggle



Game Rules:

- Connect X tokens in a row to win (default = 4)
- Board dimensions can change (default = 6 x 7)

python

The agent must be submitted as a .py file.

Training

Training algorithm:

- While playing a game, each move made by the agent is recorded and given a reward depending on the outcome
- The reward is then used to update the (state,action) pair within the Q-table according to the Bellman equation (see below)
- Once all games have been played the Q-table is saved and ready for use by the agent.

Reward System:

- R = 20: for win
- R = -20: for loss
- R = 5: for draw
- R = -0.1: for non game ending moves

Training Parameteres:

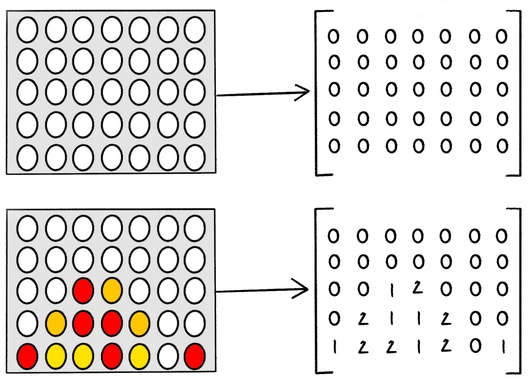
- alpha = 0.9
- gamma = 0.9

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)}^{\text{temporal difference}}$$

$$\text{new value (temporal difference target)}$$

Environment

Here is where the agent will interact and learn. An empty board is created, the players then take turns inputting counters until one of them wins or the board is full.



MATLAB

The environment was created using MATLAB

Results & Conclusion

MATLAB Testing: 100 games played.

Vs Random agent	Vs Rule Agent
• 96% - win	• 58% - win
• 4% - loss	• 39% - loss
• 0% - draw	• 3% - draw

kaggle

Leaderboard Position: 465 Total Agents

335	Sam Hennessey

Conclusion:

- The inability for a a-table to adapt to new rules such as new columns (i.e. new available moves) is a problem.
- The number of states for a connect 4 board are >4.5 trillion. Combined with all other possible states of all possible board configurations, it is unwise to find and store them all
- A better solution would be to use neural networks to find the optimum policy of the game.