Multi-agent Learning

Environments

- **Matching pennies game**

Matching pennies is classic zero-sum game, where:

- Both players A and B choose an action, Heads or Tails, at the beginning of the game
- Player A wins if it matches the action chosen by player B (i.e., H-H or T-T)
- Player B wins if it chooses a different action than player A (i.e., H-T or T-H)

	B → H	B → T
A → H	1 / -1	-1 / 1
A → T	-1 / 1	1 / -1

This game can be extended to three players in the following manner:

- Player A wins when matching the choice of player B
- Player B wins when matching the choice of player C
- Player C wins when mismatching the choice of player A

Payoff matrices:

	B → H	B → T
A → H	1	0
A → T	0	1

	C → H	C → T
B → H	1	0
B → T	0	1

	A → H	A → T
C → H	0	1
C → T	1	0

- **Climbing game**

The climbing game is an iconic coordination games. Two agents, each having 3 possible actions will receive a high penalty for miscoordination, having in the end to find one single optimal (coordinated) joint action.

Payoff matrix:

	a	b	c
a	11	-30	0
b	-30	7	0
c	0	6	5

We extended the game to a 3 player setting in the following manner:

- Each player will select an action that is matched against the other 2 players, according to the above payoff matrix.
- Their reward is then the average payoff of those two games.

Setup

In the *mal.py* file you can find the two environments described above, while in *main.py* you can find an outline in which you can plug your implementation of any desired learning algorithm. Alternatively, you also have the *ACAI_mal.ipynb* for anyone wishing to use Jupyter Notebooks.

Assignments

1. Implement the simple learning algorithm Independent Q-learning for both players in the Matching Pennies game. What is the policy they end up with. Does it converge? Investigate what happens if you add another player.
2. Use the algorithm in the previous assignment on the Climbing game and investigate the policy of the players in the same settings as above (2 players and 3 players).