# ConnectX AI Agents

Jocelyn Soto

Reinforcement learning techniques were employed to train various agents with the goal of playing the game ConnectX. I utilized two algorithms: Q-Learning and Sarsa($\lambda$).

Q-Learning is an algorithm that learns a Q-value function to make optimal decisions in a reward-based environment. The Q-Learning agent was trained by interacting with the ConnectX environment over multiple episodes, exploring the action space, and updating its Q-value estimations based on received rewards. Q-Learning was chosen due to its simplicity, effectiveness in gaming environments, and ability to learn independently.

On the other hand, Sarsa($\lambda$) combines TD($\lambda$) methods with the Sarsa algorithm, using a lambda parameter to weigh Q-value function updates. Sarsa($\lambda$) was trained similarly to the Q-Learning agent, interacting with the ConnectX environment over multiple episodes and updating its weights based on received rewards and Q-value estimations. Sarsa($\lambda$) was selected for its ability to handle problems with temporally discredited learning and offer greater control over the impact of rewards over time.

For the Q-Learning agent, after training, a Win Rate of 61.63% was achieved. When facing the 'random' opponent, the agent demonstrated an ability to win quite easily. However, against 'negamax', the Win Rate was only 2.25%, suggesting that battling this opponent was more challenging.

For the Sarsa($\lambda$) agent, a Win Rate of 69.39% was achieved against the 'random' opponent, showing a high success rate and an ability to win easily. However, against 'negamax', the Win Rate was less than 5%, although the agent still managed to win some matches.

When running a game between the Q-Learning and Sarsa($\lambda$) agents, it was observed that the Sarsa($\lambda$) agent outperformed the Q-Learning agent in terms of victories. In 10 games, the Q-Learning agent won 30% of the time, while the Sarsa($\lambda$) agent won 50%. This trend remained consistent in 100 and 1000 games, where the Sarsa($\lambda$) agent surpassed the Q-Learning agent in terms of victories.

In conclusion, the Sarsa($\lambda$) agent showed overall more robust performance and a better ability to adapt to different opponents compared to the Q-Learning agent. Although both agents were able to learn effective strategies for the ConnectX game, the Sarsa($\lambda$)-based approach proved to be more resilient and successful in a variety of situations.

The methodology used to train and evaluate the reinforcement learning agents in the ConnectX game was divided into several stages. First, the game environment was created using the Kaggle Environments library, setting parameters such as the number of rows, columns, and the number of consecutive chips needed to win. Then, two reinforcement learning algorithms, Q-Learning and Sarsa($\lambda$), were implemented, each with its own agent class.

For the Q-Learning agent, a class was defined that initialized a Q-table to store action values and implemented methods to select actions and update Q-values based on received rewards. This agent was trained by interacting with the environment over multiple episodes, during which it explored the action space and updated its Q-value estimates.

On the other hand, the Sarsa($\lambda$) agent was implemented using a class that initialized weights and the eligibility trace, and selected actions and updated weights based on received rewards and observations. Like the Q-Learning agent, this agent was trained by interacting with the environment over multiple episodes.

To evaluate the performance of the trained agents, tests were conducted by pitting them against different opponents, including random opponents and an opponent based on the negamax algorithm. Win rates were recorded, and the performance of each agent in different scenarios was analyzed.

Additionally, games were run between the Q-Learning and Sarsa($\lambda$) agents to directly compare their performances. Wins for each agent were recorded in different numbers of games to get a more complete picture of their relative performance.

In summary, the methodology included creating the agents, their interactive training with the environment, evaluating their performance against different opponents, and comparing their results. This allowed for a detailed understanding of how the reinforcement learning algorithms tackled the challenge of playing ConnectX and how they compared to each other in terms of effectiveness and robustness.