Strategic Decision-Making in Connect 4
"A Comparative Analysis of Reinforcement Learning Techniques"

Sarthak Miglani

Introduction

The aim of this project is to explore and evaluate the effectiveness of various reinforcement learning techniques in the strategic game of Connect 4. Inspired by a study from Stanford University ("Reinforcement Learning Approaches to Connect Four" by E. Alderton, E. Wopat, and J. Koffman, accessible here), this investigation focuses on implementing and comparing Q-Learning, SARSA, and Deep Q Networks (DQN) to determine which method most effectively guides an AI to master this game.

Connect 4, a two-player connection game, involves players choosing a color and taking turns dropping colored discs into a vertically suspended grid. The goal is to be the first to form a horizontal, vertical, or diagonal line of four discs. Its simple rules yet complex strategic depth make Connect 4 an excellent model for studying AI in game-based learning environments. Through this project, I aim to deepen my understanding of how, each reinforcement learning technique impacts strategic decision-making in games that require foresight and planning. By examining how an AI optimizes its moves and counters its opponent within this grid structure, we can gain insights into the tactical capabilities of different learning strategies. This research is significant not only for its potential to illuminate the comparative efficiencies of established learning algorithms but also for setting a foundation for future explorations into how AIs can learn complex tasks governed by simple rules and feedback.

AI Concepts and Methods

Reinforcement Learning:

Reinforcement Learning (RL) is an area of machine learning where an agent learns to make decisions by executing actions within an environment to maximize some idea of cumulative reward. Receiving states from the environment, the agent takes actions, and receives rewards, consequently. The process comprises learning a policy: a mapping from apparent states of the environment to actions to be taken when in those states. Key components of any RL system include:

Agent: The learner or decision-maker.

Environment: The system within which the agent operates.

Reward: A feedback signal indicating the success of an action.

Policy: A strategy used by the agent to decide the next action based on the current state.

Value Function: A prediction of expected future rewards used to update policies.

Q-Learning:

Q-Learning is a model-free off-policy reinforcement learning algorithm that seeks to find the best action to take given the current state. It uses a Q-table, which is updated as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

where α is the learning rate, γ is the discount factor, r_{t+1} is the reward received after moving to the new state s_{t+1}, and a_t is the action taken at state s_t. The policy then becomes selecting the action with the highest Q-value at any state.

SARSA:

SARSA (State-Action-Reward-State-Action) is an on-policy reinforcement learning algorithm where the agent learns the action to take not only from the Q-value of the next state but also the action that it actually takes (as opposed to the best possible action in Q-Learning). Its update rule is given by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

SARSA directly learns the policy it follows, as opposed to Q-Learning which learns the optimal policy while following a possibly suboptimal one.

Deep Q Network (DQN):

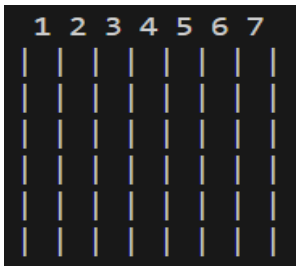
The Deep Q Network (DQN) approach extends Q-Learning by using a deep neural network to approximate the Q-value function. Instead of maintaining a table of Q-values, DQN uses a neural network to predict Q-values for all possible actions given a state. This approach is particularly suited for environments with high-dimensional observation spaces, like Connect 4, where the state can be represented as a grid. The use of experience replay and fixed Q-targets in DQN helps to stabilize the learning.

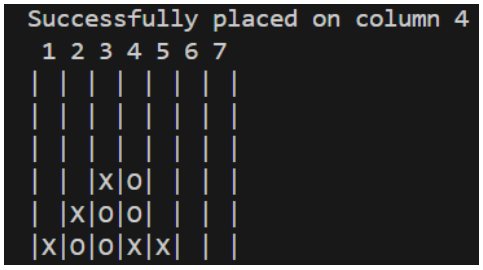
Project Code Overview

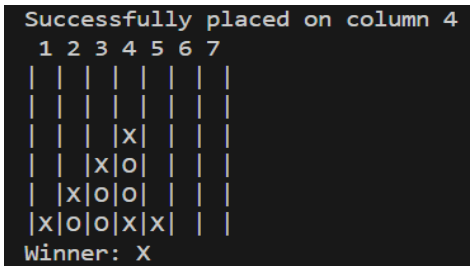
The codebase for the Connect 4 AI project is structured into several main modules, each handling a specific aspect of the project:

- ***agent_dqn.py, agent_q_learning.py, agent_sarsa.py:*** These modules contain the implementation of the respective reinforcement learning algorithms. Each agent module is designed to interact with the game environment, learn from it, and make decisions based on the current state of the board.
- ***config.py:*** This module stores configuration settings that dictate various parameters of the game and learning algorithms, such as learning rates, discount factors, and the number of episodes for training.
- ***connect_four.py:*** This module manages the Connect 4 game logic, including board representation, game rules, and win-check algorithms. It serves as the environment for the agents. The "test_connect_four.py" module verifies this logic with predefined move sequences.

Copy the script to run in terminal: ***python test_connect_four.py***

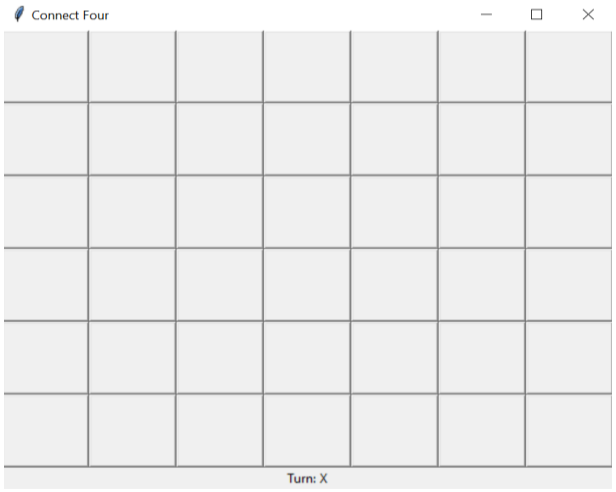


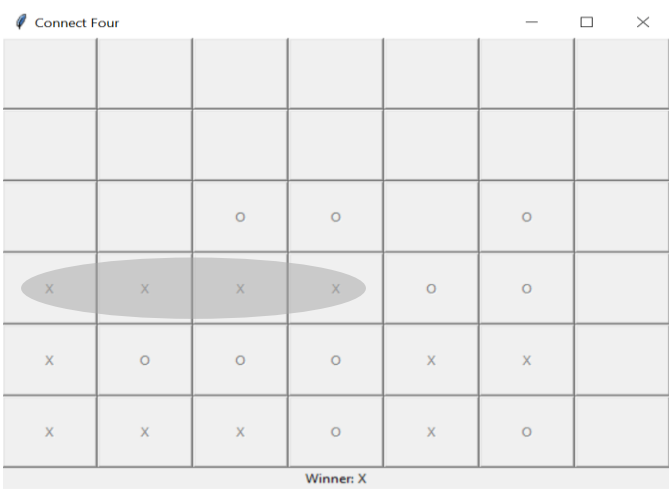




- ***connect_four_gui.py:*** This module provides a graphical user interface for the Connect 4 game, enhancing user interaction with visually appealing game play. It supports gameplay mode of human versus human, run the code and enjoy the game recreationally with friends. This interface ensures real-time updates and smooth interaction, facilitating an engaging gaming experience.

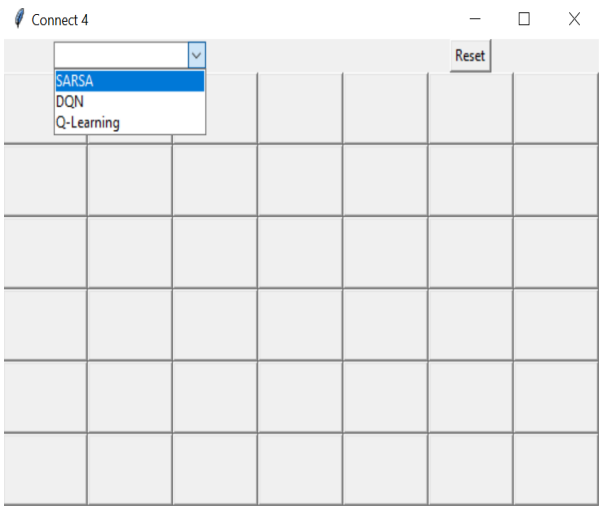
Copy the script to run in terminal and enjoy the game: ***python connect_four_gui.py***

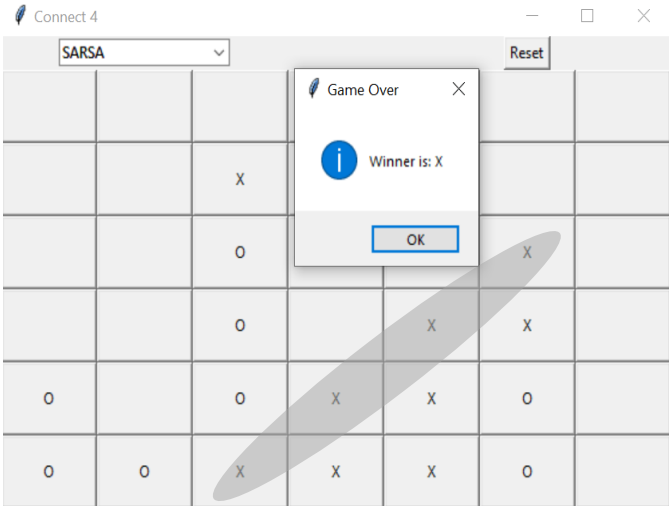




- ***main.py:*** This module is the application's entry point, allowing players to select an AI model (Q-Learning, SARSA, or DQN) for Connect 4 to play against. It manages game setup, execution, and restarts, toggling the first player between games to ensure fair and varied play.

Copy the script to run in terminal and all the best for the game: ***python main.py***





- ***game_dqn_vs_agent2.py:*** This script facilitates the match-up of the Deep Q Network (DQN) against both Q-Learning and SARSA agents, with DQN playing as Player 1. Upon execution, the script first displays the progression of the initial 10 games against each algorithm, providing a detailed view of the game dynamics. Following this, it runs simulations of 1000 games against each AI model and compiles the outcomes. The results are summarized in a bar plot that visually represents the win rates, offering a clear comparison of DQN's performance against Q-Learning and SARSA.
- ***game_q_vs_agent2.py:*** This script sets up matches where the Q-Learning agent plays as Player 1 against the SARSA and DQN agents. It starts by visually displaying the course of the first 10 games against each opponent, allowing observation of Q-Learning's strategy and decision-making process. The script then conducts an extensive test of 1000 games against each opponent, summarizing the results in a bar plot that illustrates the comparative effectiveness of Q-Learning.
- ***game_sarsa_vs_agent2.py:*** In this module, the SARSA agent is positioned as Player 1 to compete against the DQN and Q-Learning agents. The initial phase involves displaying the first 10 games against each opposing AI, giving insights into SARSA's gameplay strategies. The script subsequently performs a larger set of 1000 games against each AI, culminating in a bar plot that displays the win rates, providing a statistical analysis of SARSA's performance across a significant number of games.

Test Results

Experimental Setup

The experiments were designed to evaluate the performance of three reinforcement learning algorithms: Q-Learning, SARSA, and Deep Q Network (DQN), in the context of a Connect 4 game. The setup involved a series of matches where each algorithm competed against the others across a large number of games (20000 games for each agent as player 1 and 20000 games for each agent as player 2) to ensure statistical significance. The initial matchups showcased the outcomes of the first 10 games between each pair of AI agents in a GUI, offering a qualitative glimpse into their gameplay dynamics and strategies. This snapshot allowed for a preliminary assessment of each agent's decision-making process and adaptability in the early stages of competition. It is important to note that while individual experimental runs might show some variation in outcomes, the overall trends and conclusions remain consistent.

Performance Metrics

The primary metrics used to assess the performance of each AI agent in the Connect 4 game were as follows:

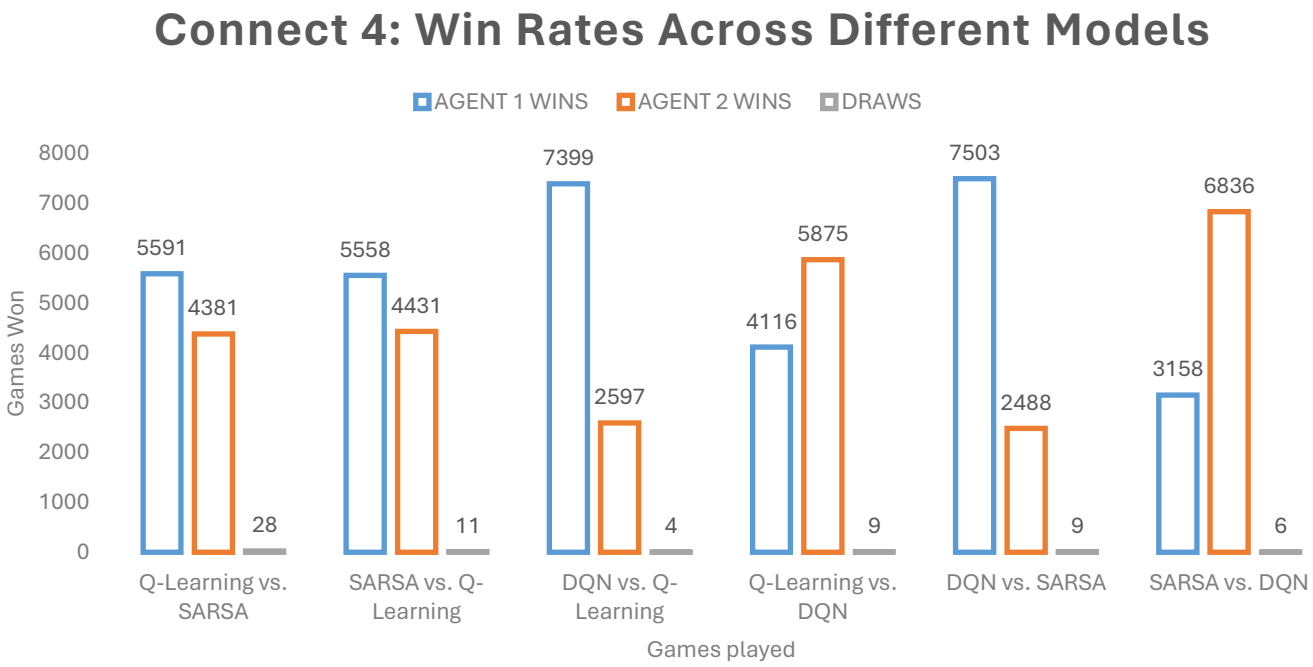
- *Win Rate:* This is the main metric, representing the percentage of games won by each agent. It directly measures the effectiveness of each AI's strategy under competitive conditions.

- *First Player Advantage:* Special attention was given to which agent played first in each game. This parameter is crucial as it often influences the outcome in strategic games like Connect 4, where the first player can have a theoretical advantage due to the opportunity to control the board early.

Results Overview

The results from the experimental matchups were as follows:

GAMES	AGENT 1 WINS	AGENT 2 WINS	DRAWS
Models: Q-Learning and SARSA			
Q-Learning vs. SARSA	55.91%	43.81%	0.28%
SARSA vs. Q-Learning	55.58%	44.31%	0.11%
Models: DQN and Q-Learning			
DQN vs. Q-Learning	73.99%	25.97%	0.04%
Q-Learning vs. DQN	41.16%	58.75%	0.09%
Models: SARSA and DQN			
DQN vs. SARSA	75.03%	24.88%	0.09%
SARSA vs. DQN	31.58%	68.36%	0.06%



Discussion and Analysis

Result Analysis

The experimental outcomes highlight significant variations in performance among the AI models. The Deep Q Network (DQN) consistently outperformed both Q-Learning and SARSA in direct matchups, achieving win rates of approximately 74% and 75% respectively when playing as Agent 1. This suggests that the neural network-based

approach of DQN provides a more robust strategy in Connect 4, possibly due to its ability to generalize from past experiences more effectively than the table-based methods used by Q-Learning and SARSA. Although results can vary slightly with each execution, the aggregate findings consistently demonstrate DQN's superior performance.

Impact of Player Order

The data also shows a notable impact of player order on outcomes. For instance, when Q-Learning faced SARSA, the first player won around 56% of the games, regardless of the algorithm used. This pattern holds less sway in matches involving DQN, where the strength of the neural network model overshadows the advantage conferred by playing first. This indicates that the starting advantage in Connect 4 can be mitigated by more advanced AI strategies.

Implications of Findings

The superior performance of DQN suggests that deep reinforcement learning could be more effective in board games that require complex strategic planning and adaptability. The findings imply that developing AIs with the capability to learn and generalize from vast amounts of data can significantly enhance performance, not only in traditional games but also in applications requiring nuanced decision-making, such as robotics and financial strategy.

Future Work

Moving forward with this project, a promising direction involves incorporating search algorithms into the existing reinforcement learning framework to enhance strategic decision-making capabilities. Specifically, integrating Monte Carlo Tree Search (MCTS) as an opponent in main.py could provide a diverse challenge for the learning-based models. MCTS, known for its success in complex games, dynamically explores potential moves based on random sampling and could offer a robust test against the predictable early-game weaknesses of learning models.

Conclusion

This project was designed to evaluate and compare the performance of three distinct reinforcement learning strategies—Q-Learning, SARSA, and Deep Q Network (DQN)—within the context of the strategic board game Connect 4. The experimental results compellingly highlighted the superior decision-making capabilities of DQN, which consistently outperformed the other two models in head-to-head competitions. Additionally, the study explored the strategic advantage traditionally conferred to the first player in Connect 4. The findings revealed that while starting first generally offers a tactical upper hand, the implementation of more advanced AI techniques, such as those embodied by DQN, can significantly mitigate this advantage. Despite the potential for minor variations in specific game outcomes, repeated runs support the same overarching conclusions.

APPENDIX

Copy the script to run in terminal to see the results.

python game_dqn_vs_agent2.py	***python game_q_vs_agent2.py***	***python game_sarsa_vs_agent2.py***
Displaying the first 10 games (DQN vs. Q-Learning): *Game 1 Winner: DQN Agent* *Game 2 Winner: Q-Learning Agent* *Game 3 Winner: DQN Agent* *Game 4 Winner: DQN Agent* *Game 5 Winner: Q-Learning Agent* *Game 6 Winner: DQN Agent* *Game 7 Winner: DQN Agent* *Game 8 Winner: DQN Agent* *Game 9 Winner: Q-Learning Agent* *Game 10 Winner: DQN Agent* *Results of 10000 games:* *DQN Agent wins: 7399* *Q-learning Agent wins: 2597* *Draws: 4*	*Displaying the first 10 games (Q-Learning vs. DQN):* *Game 1 Winner: Q-Learning Agent* *Game 2 Winner: DQN Agent* *Game 3 Winner: Q-Learning Agent* *Game 4 Winner: Q-Learning Agent* *Game 5 Winner: Q-Learning Agent* *Game 6 Winner: DQN Agent* *Game 7 Winner: Q-Learning Agent* *Game 8 Winner: Q-Learning Agent* *Game 9 Winner: DQN Agent* *Game 10 Winner: DQN Agent* *Results of 10000 games:* *Q-Learning Agent wins: 4116* *DQN Agent wins: 5875* *Draws: 9*	*Displaying the first 10 games (SARSA vs. DQN):* *Game 1 Winner: SARSA Agent* *Game 2 Winner: DQN Agent* *Game 3 Winner: DQN Agent* *Game 4 Winner: SARSA Agent* *Game 5 Winner: SARSA Agent* *Game 6 Winner: DQN Agent* *Game 7 Winner: SARSA Agent* *Game 8 Winner: SARSA Agent* *Game 9 Winner: DQN Agent* *Game 10 Winner: DQN Agent* *Results of 10000 games:* *SARSA Agent wins: 3158* *DQN Agent wins: 6836* *Draws: 6*
Displaying the first 10 games (DQN vs. SARSA): *Game 1 Winner: DQN Agent* *Game 2 Winner: DQN Agent* *Game 3 Winner: DQN Agent* *Game 4 Winner: DQN Agent* *Game 5 Winner: DQN Agent* *Game 6 Winner: DQN Agent* *Game 7 Winner: DQN Agent* *Game 8 Winner: SARSA Agent* *Game 9 Winner: DQN Agent* *Game 10 Winner: SARSA Agent* *Results of 10000 games:* *DQN Agent wins: 7503* *SARSA Agent wins: 2488* *Draws: 9*	*Displaying the first 10 games (Q-learning vs SARSA):* *Game 1 Winner: Q-learning Agent* *Game 2 Winner: SARSA Agent* *Game 3 Winner: Q-learning Agent* *Game 4 Winner: SARSA Agent* *Game 5 Winner: SARSA Agent* *Game 6 Winner: SARSA Agent* *Game 7 Winner: Q-learning Agent* *Game 8 Winner: SARSA Agent* *Game 9 Winner: SARSA Agent* *Game 10 Winner: Q-learning Agent* *Results of 10000 games:* *Q-learning Agent wins: 5591* *SARSA Agent wins: 4381* *Draws: 28*	*Displaying the first 10 games (SARSA vs Q-learning):* *Game 1 Winner: SARSA Agent* *Game 2 Winner: Q-learning Agent* *Game 3 Winner: SARSA Agent* *Game 4 Winner: Q-learning Agent* *Game 5 Winner: SARSA Agent* *Game 6 Winner: SARSA Agent* *Game 7 Winner: SARSA Agent* *Game 8 Winner: Q-learning Agent* *Game 9 Winner: SARSA Agent* *Game 10 Winner: Q-learning Agent* *Results of 10000 games:* *SARSA Agent wins: 5558* *Q-learning Agent wins: 4431* *Draws: 11*

REFERENCES

[1] Alderton, E., Wopat, E., & Koffman, J. (2019). Reinforcement Learning for Connect Four. Stanford University. Available at: https://web.stanford.edu/class/aa228/reports/2019/final106.pdf

[2] Built In. (n.d.). SARSA Reinforcement Learning Algorithm. Retrieved from https://builtin.com/machine-learning/sarsa

[3] GeeksforGeeks. (n.d.). Deep Q-Learning. Retrieved from https://www.geeksforgeeks.org/deep-q-learning/