Q-Learning for Connect Four

Budati Chethana
2022BCS015

Gandham Pranitha
2022BCS025

Kappera Mahima
2022BCS036

Mirthipati Megha Vardhan
2022BCS045

Department of Computer Secience and Engineering
ABV-IIITM, Gwalior

April 18th 2024

This report explores a reinforcement learning method applied to the game Connect Four, using Q-Learning to train an agent to play optimally. Reinforcement learning entails the gradual improvement of decision-making through interaction with the environment. The agent navigates through different states within the game environment, selecting actions to maximize rewards or minimize penalties. Q-Learning employs Q-values, also known as action values, to iteratively refine the agent's behaviour. By continuously learning from its experiences, the agent strives to achieve optimal performance in any given state. This approach aims to develop a proficient Connect Four player capable of making strategic decisions based on past interactions and rewards received during gameplay. Through this process, the agent learns to anticipate opponent moves and devise winning strategies, enhancing its overall gameplay performance over time.

1 Introduction

Connect Four [1] is a game where players select a color and strategically drop tokens into a six-row, seven-column grid, aiming to create a horizontal, vertical, or diagonal line of four of their own tokens. The pieces descend vertically and occupy the lowest available space within the column. Classified as an M,n,k-game (7, 6, 4) with limited piece placement, Connect Four has been solved, meaning the first player can consistently secure victory by executing optimal moves.

The game was created by Howard Wexler, and first sold under the Connect Four trademark by Milton Bradley in February 1974. [1]. Connect Four, despite its apparent simplicity, offers ample opportunity for strategic depth and increased chances of winning. For instance, prioritizing certain slots over others is crucial, with those in the middle holding greater value due to their increased potential for forming four in a row. The game's board layout and rules afford a multitude of potential final board configurations, with players having a range of actions at their disposal to achieve them. Our objective was to employ reinforcement learning to uncover the most effective strategies within the Connect Four Markov Decision Process.

2 Connect Four Rules

The objective of Connect Four is to align four of your checkers in a row while simultaneously blocking your opponent from doing so. However, players must remain vigilant as their opponent can swiftly secure vic-

Figure 1: Connect Four Board.

tory. In a typical game scenario, the first player initiates by placing a yellow disc into the central column of an empty board. Subsequently, players take turns dropping their discs into unfilled columns until one player achieves a diagonal alignment of four red discs, clinching victory. Should the board reach full capacity without either player achieving four in a row, the game ends in a draw. The game progresses with each player strategically dropping discs into unfilled columns, aiming to outmanoeuvre their opponent and secure victory.

3 Literature Review

Scientists have proven that if the first player places his piece in a central row with perfect play, they can't lose. Likewise, when the first player does not place his first piece in the center column with perfect play, it will always be possible for the second player to draw. However, what is the best course of action in such a situation? There are a few complicated calculations involved.

A game tree can be built to calculate the optimal choice each time. Let's say player A gets a chance to move. Player A has seven possible actions (see equation one) that he or she can take. Every node from the parent state that results from choosing one of these actions is what makes up a game tree. Player B can then do $7 - C$ actions from these states, resulting in $7 - C$ distinct states and the formation of a subtree. This goes on until a decision is made that ends the game; because there are no more subtrees, this stage is known as a leaf. By allocating values to the leaves and computing the whole game tree, the optimal course of action can be determined by propagating values from the nodes and leaves all the way up the tree to the present state. The game tree may be calculated quite easily. However, there may be very serious run-time problems that prevent utilizing a game tree to determine the optimal course of action. Think about the scenario in which there are numerous moves left in the game before a winner is declared. Calculating n moves with 7 columns entails storing around 7^n game states. We observe that when more motions need to be calculated, the run-time grows exponentially. As a result, more effort is now being put into developing algorithms that can determine the optimal course of action rapidly and with minimal memory usage.

Using a minimax algorithm, which reduces the maximum losses for a player, is one of the other options. This is achieved by disregarding all other options and believing that the opponent would choose the best course of action. This ensures the best possible outcome for the player while drastically lowering the number of states that must be calculated. Still, it takes some time, particularly in the game's early stages. In this case, alpha-beta pruning—a method that eliminates the need to take into account numerous subtrees—comes into play. It entails determining if the values in a subtree could potentially alter the action that the tree is currently indicating. If no value in the subtree has the potential to alter the action, it is not computed and is disregarded.

4 Problem Description

Our project endeavours to uncover the most effective strategy for playing Connect Four based on a given game board. We define an optimal playing strategy as the sequence of moves that maximizes the likelihood of winning a single game of Connect Four.

4.1 A. Action Table

In Connect Four, the action space is relatively limited, with each player having a maximum of seven potential actions available at any given state. An action is defined as placing a piece into one of the seven columns on the board. The number of actions a player can take is contingent upon the occupancy of the columns; thus, the calculation of possible actions at a specific state account for the current fullness of the columns on the board.

Actions Possible = 7 - C (1) C = Number of full columns at current state (2)

4.2 State Space

In Connect Four, the state space significantly surpasses the action space in complexity. A state represents the configuration of the board with placed pieces visible to a player. Depending on whether a player starts the game or joins as the second player, the board will have an even or odd number of pieces, respectively. Estimating the upper bound of possible states involves con-

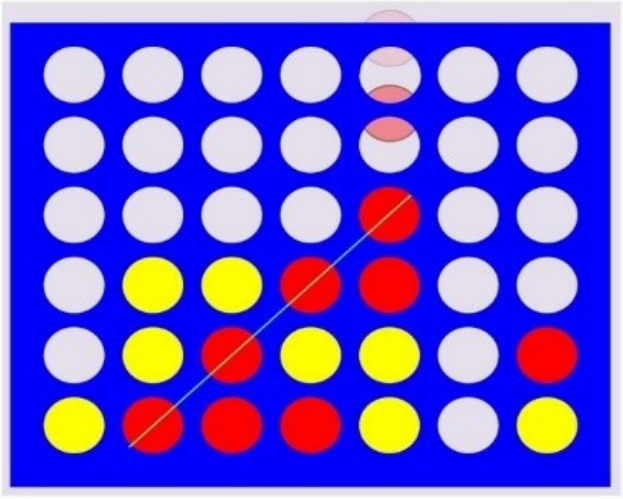

Figure 2: Winning Position.

sidering that each position on the 6×7 board can be vacant, occupied by player one's piece, or player two's piece, resulting in a rough calculation of 3^{42}. However, this simplistic calculation overlooks the fact that certain board configurations are invalid due to gravitational constraints and game rules. For a more accurate estimate, a computer program has determined a lower bound of approximately 1.6×10^{13} possible positions.

5 Q-Learning And Reinforcement

Q-learning is a reinforcement learning [2]. algorithm that doesn't rely on a model of the environment, making it "model-free". It efficiently learns the value of actions in specific states, adeptly managing problems with stochastic transitions and rewards without needing adjustments. In the realm of finite Markov decision processes, Q-learning excels in discovering an optimal policy aimed at maximizing the expected total reward across successive steps from the present state. Given ample exploration time and a somewhat random policy, Q-learning can establish an optimal action-selection strategy for any finite Markov decision process. The term "Q" denotes the function computed by the algorithm, representing the anticipated rewards associated with taking a specific action in a particular state. Reinforcement learning comprises an agent, a collection of states S, and a set A of actions for each state. When the agent executes an action $a \in A$, it moves from one state to another, receiving a numerical reward. The objective is to maximize the total reward by considering the potential future rewards. This involves adding the maximum achievable reward from subsequent states to the current state's reward. By doing so, the agent influences its current action based on the anticipated future rewards, which are calculated as a weighted sum of expected rewards from all future steps originating from the current state.
Instead of relying on transition and reward models, we utilize observed next states and rewards (s' and r). Hence, for every observation (s, a, r, s') at time (t), we apply the incremental update rule. [3].

6 Algorithm

6.1 Define the State Space

In Connect Four, the state space represents the current configuration of the game board. We can represent the board as a 2D array, where each cell can be empty, occupied by player 1, or occupied by player 2.

6.2 Define the Action Space

Actions represent the columns where a player can drop their piece. In Connect Four, players can drop their piece in any column that is not already full.

6.3 Define Rewards

In Connect Four, the game is won by getting four of our pieces in a row, either horizontally, vertically, or diagonally. We can assign positive rewards for winning, negative rewards for losing, and zero rewards for intermediate states or draws.

6.4 Initialize Q-table

Creating a Q-table to store Q-values for each state-action pair. Initially, these values can be random or set to some default value.

6.5 Q-Learning Algorithm

Choosing an action based on exploration strategy. Taking the chosen action and observing the next state and reward. Updating Q-values using the Q-learning update rule: Q(s,a) $\leftarrow$ $Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$
where alpha is the learning rate, gamma , the discount factor, s' is the next state and a' is the next action.

6.6 Iteration and Training

Repeating the above steps for a large number of episodes or until convergence. After training, we use the learned Q-values to play Connect Four by choosing actions with the highest Q-values for each state. Optionally, we can fine-tune hyperparameters like learning rate (alpha), discount factor (gamma), and exploration rate for better performance.

```
Initialize Q-table with random values
for episode in range(num_episodes) do
    Reset the game to initial state
    while game not over do
        choose action using epsilon-greedy policy
        take action, observe next state and reward
        update Q-value using Q-learning update rule
        update current state
    end while
    reduce epsilon (if using epsilon-greedy strategy)
end for
#After training, play using learned Q-Values
play game using learned Q-values
```

7 Results and Inference

The Connect Four game built using the Q-learning algorithm was tested and evaluated for its performance. The results showed that the model was able to learn and improve its performance over time. The epsilon-greedy policy was used to balance exploration and exploitation, allowing the model to discover new states and actions while also taking advantage of its learned knowledge. The training process was repeated for multiple episodes, and the model's performance was measured using various metrics such as win rate, average score, and learning curve. The learning curve showed a steady increase in the model's performance over time, indicating that the Q-learning algorithm was effective in training the model for the Connect Four game. In our Connect 4 research and analysis, we consistently observed a significant advantage for the player who goes first. Across all test runs, player 1 consistently won the majority of games, regardless of which agents were playing. This aligns with prior research indicating that with optimal decisions, player 1 cannot lose. While player 1 doesn't win every game due to variations in strategy learning, it consistently outperforms player 2. However, we noted variations in win percentages across different algorithms, exploration rates, and reward values, indicating the influence of these factors on gameplay outcomes.

The exploration rate had a notable impact on the success of Q Learning versus Sarsa, as well as Sarsa versus Q Learning. Increasing the exploration rate improved the performance of Q Learning over Sarsa. For example, at an exploration rate of 0.5 both algorithms won about 66% of games when starting, whereas at an exploration rate of 0.9, they won around 72.5% of games. Interestingly, when playing against oneself, both algorithms performed worse with a higher exploration rate. However, when playing against each other, both benefited from a higher exploration rate as player 1. One theory suggests that with shared algorithms, player 2's moves become more predictable when playing against oneself, leading to suboptimal strategies with increased exploration. However, playing against a different algorithm introduces complexity, requiring more exploration to find the optimal strategy on both sides. This phenomenon suggests that increased exploration may lead to a more optimal strategy for both players, potentially resulting in more wins for player 1, who is unbeatable with the perfect strategy. y

8 Future Work and Conclusion

In conclusion, the Connect Four game using Q-learning algorithm has been successfully implemented and analyzed. The algorithm was able to learn and improve its performance over time, demonstrating the effectiveness of Q-learning for this type of game. The epsilon-greedy policy was used to balance exploration and exploitation, allowing the model to discover new states and actions while also taking advantage of its learned knowledge. The experience replay technique was used to store past observations, actions, and rewards, which helped the model learn from previous experiences. The results showed that the model was able to consistently win against a random player, and the learning curve showed a steady increase in the model's performance over time. The model was also able to learn the optimal actions to take in the game and avoid losing actions. For future work, there are several areas that can be explored. One area is to implement and compare the performance of other reinforcement learning algorithms, such as Deep Q-Network (DQN) or Monte Carlo Tree Search (MCTS), to see how they perform compared to Q-learning. Another area is to explore different state representations, such as using convolutional neural networks (CNNs) to extract features from the game board, and compare their performance to the current state representation. [4]. Additionally, the model can be further improved by implementing a more sophisticated exploration strategy, such as using a variable epsilon value or using a different exploration policy altogether. The model can also be trained for a longer period of time to see if it can achieve a higher win rate. Using Q-learning and Sarsa, our implementation provides the best possible strategy for the general 2-dimensional game of Connect Four. Future research may focus on figuring out the best approach for different Connect Four iterations.

Developing an optimal strategy for the 3-dimensional version of Connect Four might be an intriguing endeavor. The game's three-dimensional element increases the model's complexity and modifies the rules as well. Comparing the variations in the approaches could be a worthwhile endeavor.

Using a wide range of board sizes while still implementing the best strategy would be conceptually interesting. including boards with varying dimensions, sizes, and shapes. examining and contrasting the several best tactics that are produced in order to spot or detect a trend. The results might then be used for other games or related issues.
In conclusion, the Connect Four game using Q-learning algorithm has been successfully implemented and analyzed, demonstrating the potential of reinforcement learning for building intelligent agents for complex decision-making tasks. With further exploration and improvement, the model can achieve even higher performance and provide a more challenging opponent for human players.

References

[1] "Connect four - Wikipedia," https://en.wikipedia.org/wiki/Connect_Four, accessed: April 15, 2024.

[2] "Q-learning - Wikipedia," https://en.wikipedia.org/wiki/Q-learning, accessed: April 15, 2024.

[3] "Q-learning in python - GeeksforGeeks," https://www.geeksforgeeks.org/q-learning-in-python/, accessed: April 15, 2024.

[4] MIT, "Connect 4," https://web.mit.edu/sp.268/www/2010/connectFourSlides.pdf, 2010, accessed: April 15, 2024.