From Board to Algorithm: A Study of Reinforcement Learning in Connect-4

Björn Thor Stefánsson

Kristján Sölvi Örnólfsson

Viktor Már Guðmundsson

Fall 2023

Abstract

In this project, we explored the potential of reinforcement learning (RL) techniques in the classic two-player game Connect-4. The objective was to investigate whether RL could effectively learn and apply strategic decision-making in a game that, while simple in appearance, demands complex strategic considerations. We employed two distinct RL methods: tabular Q-learning and linear function approximation. We initially experimented with smaller game boards to establish a foundational understanding and then expanded to the standard Connect-4 board size. Our study showcases the application of certain RL techniques in a turn-based environment, like Connect 4.

Final project in REI505M

Faculty of Industrial Engineering, Mechanical Engineering and Computer Science

1 Introduction

The field of artificial intelligence (AI) has seen significant advancements in recent years. Particularly, the domain of reinforcement learning (RL) has gained popularity for its ability to solve problems that were previously difficult to solve in fields such as healthcare, robotics, and board games. In layman's terms, RL is a machine-learning method where computer programs learn through trial and error. In our study, we focused on applying RL to Connect-4, a seemingly straightforward two-player game that turns out to be strategically rich when you delve into it. The game, shown in Figure 1, involves players alternating turns to drop colored discs into a rectangular grid, aiming to align four of their discs in a row, either horizontally, vertically, or diagonally.

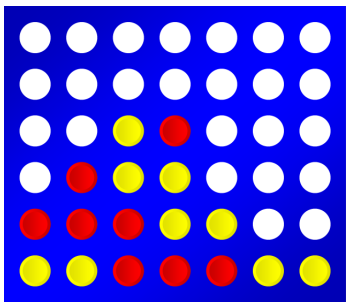

Figure 1: *Connect-4 board game*

Our project involved two main phases. In the first phase, we developed a digital version of Connect-4, ensuring it followed the game's standard rules and mechanics. In the second phase, we implemented two RL techniques - tabular Q-learning and linear function approximation. We compared our efforts with common baselines: a random agent and a minimax agent.

This report details our approach, methodology, and findings. It provides a comprehensive overview of how we applied RL to Connect-4, the challenges we faced, and how we dealt with them. The aim of our project is to study reinforcement learning in the context of a simple board game.

2 Prior Work

Connect-4 has been a subject of significant interest in artificial intelligence. The simple game mechanics and the strategic complexity of the game make it an ideal study for game theory and AI strategies. The game was solved, and it was shown that the first player could always win with perfect play. This was first reported separately by James D. Allen and Victor Allis, with detailed strategies in Allis's thesis Allis (1988). John Tromp's website is a valuable resource for understanding these strategies and exploring Connect-4 Tromp (nd).

Reinforcement Learning gained prominence in AI through successes in complex board games. Notably, AlphaGo applied RL to beat the human world champion in Go multiple times, a feat that seemed impossible just a few years earlier Silver et al. (2016). This success sparked the interest of many researchers around the world and reinforcement learning has been a rapidly growing field since.

Traditional Connect-4 research focused on search algorithms like minimax and alpha-beta pruning.

Recent work by Rijul Nasa in 'Alpha-Beta Pruning in Mini-Max Algorithm – An Optimized Approach for a Connect-4 Game' Nasa et al. (2018), delves into the Minimax algorithm's application in Connect-4, highlighting its effectiveness in optimal move determination but lacking in learning processes inherent in RL. Another intriguing approach is presented in 'A Robot that Learns Connect Four Using Game Theory and Demonstrations' by Ayub and Wagner (2020), further diversifying the Connect-4 AI research landscape.

The methods we use align more with the foundational work on Q-learning by Watkins (1989), a cornerstone in model-free learning in deterministic environments. Our project builds upon these principles, applying tabular Q-learning and linear function approximation to manage the extensive state spaces in Connect-4, aiming to extend our understanding of RL in the context of a solved game.

3 Dataset and Data Processing

The dataset for our reinforcement learning project was derived from the state space of the Connect-4 game. Even a relatively small board size results in a vast array of possible game states. For example, a standard 6x7 board has approximately $3^{42} \approx 1.1 \cdot 10^{20}$ potential states, since each cell can be in one of three states: empty, occupied by player one, or occupied by player two. However, not all these states are achievable in a legitimate game. John Tromp has calculated the exact number of feasible states for various board sizes, which is about $4.5 \cdot 10^{12}$ for the standard 6x7 board Tromp (nd). To make our task more manageable, we initially focused on a smaller 4x5 board, which reduced the state space to approximately 4 million states.

We encoded the game table as a string of ternary numbers, each digit corresponding to the state of a single spot on the board. In other words, the numbers tell us if a spot is empty, if it contains a disk from the first player or if it contains a disk from the second player. We made sure that only legal moves were possible to play at any given time and set up a function to check if the game has ended.

As our computer programs got better at the smaller board, we moved them to bigger boards. We also used a technique called reward shaping to help the programs learn faster. This means we didn't just reward them for winning, but also for making moves that improved their position in the game.

4 Methods

Our project's approach involved creating and testing algorithms specifically designed for playing Connect-4. We started by configuring the game rules and then created algorithms (agents), that could learn to play using two approaches: tabular Q-learning and linear function approximation.

4.1 Game Logic Implementation

The implementation of Connect-4's game logic was the first step. This phase involved designing and developing a range of functions for the game's operation, such as initializing the game board, placing tokens, and determining game outcomes like wins or draws. While we found inspiration in various open-

source models, our version was tailored to suit the specific requirements of our reinforcement learning agents.

We crafted a class, *Connect4_Game*, which encapsulated all game functionalities. Key methods within this class included *initialize_game( )* for setting up the game board, *place_token( )* for token placement in designated columns, and *check_winner( )* to evaluate whether the game had reached a conclusive state. The use of NumPy for array manipulation was essential for representing the game board and handling state transitions efficiently. This object-oriented structure provided a clear and organized framework for the game's logic, ensuring a realistic simulation of Connect-4's dynamics.

Moreover, the development process involved designing key functions like *calculate_reward( )* and *count_sequences( )*. Designing a robust game reward structure was essential to optimize our agent's learning process.

4.2 Tabular Q-Learning and Agent Construction

Our first learning agent utilizes Q-learning. A Q-function is what our agent uses to choose an optimal choice based on the enviroments reward structure. It is essentially the model that is being trained i.e. the function being approximated. This approach in reinforcement learning was centered on learning the value of executing specific actions in particular states, to maximize cumulative rewards Sutton and Barto (2018).

In our implementation the Q-table is organized as a state, action pair, $(state, action) := (s, a)$ tethered with a q-value in a python dictionary. Every state is paired with all possible valid actions in that state. Initially all q-values are set to zero.

Our approach used a temporal difference learning process with 1 step. The number of steps refer to how far we look into the future environment states when updating our q-values. Using one step will suffice as the game relatively simple. More steps will however possibly yield a better result. Temporal difference is based on Markov-chain Monte-Carlo learning which defines a total reward over episode ($R_{\sum}$).

$$R_{\sum} = \sum_{k=1}^{n} \gamma^k r_k$$

The Q-learning update is then formulated with:

$$\text{New } Q(s_k, a_k) = Q(s_k, a_k) + \frac{1}{n} \left[\underbrace{R_{\sum}}_{\substack{\text{Episode} \\ \text{Reward}}} - Q(s_k, a_k) \right] \forall k = 1, 2, \ldots, n$$

The Monte Carlo learning technique divides the sum of the rewards over a whole episode thus treating every action the same. This is very inefficient and has a much slower convergence than temporal difference. Temporal difference takes into account the "time" between the action and the reward and q-values are updated after each action. Q-learning is a specific version of termporal difference learning and uses

an off-policy learning method. The update of the Q-table values is governed by the following rule:

$$\text{New } Q(s,a) = Q(s,a) + \alpha \left[\underbrace{R(s,a)}_{\text{Reward}} + \gamma \cdot \overbrace{\max_{a'} Q(s',a')}^{\substack{\text{Maximum predicted reward for} \\ \text{the next state across all actions}}} - Q(s,a) \right]$$

In this equation, $Q(s,a)$ represents the current value for taking action a in the state s, while New $Q(s,a)$ is the updated value. The term $R(s,a)$ denotes the immediate reward received after taking action a in the state s, and γ is the discount factor that weighs the importance of future rewards. The expression $\max_{a'} Q(s',a')$ calculates the maximum predicted reward for the next state s', considering all possible actions a'. This maximum reward is then scaled by γ and combined with the immediate reward to update the current Q-value. The learning rate α controls the extent to which the newly acquired information overrides the old value. Through this iterative process, the agent incrementally refines the values in the Q-table, which guides its policy towards more rewarding strategies over time.

Using an agent as the model's "ambassador" to interact with the environment, it continually refined the Q-table values based on the rewards it received, thus learning from its experiences. The learning process balances between exploiting known strategies and exploring new strategies. An epsilon-greedy strategy was used. An epsilon greedy agent with an epsilon decay is forced to initially explore a broad spectrum of actions, gradually shifting its focus to exploiting the most effective strategies as it garnered more knowledge.

Efficiency was paramount in the development of our agent. We equipped it with the capability to recognize and exploit environmental symmetries, enhancing its learning efficiency. Furthermore, the epsilon value, a crucial element in determining the balance between exploration and exploitation, was programmed to decay over time, as discussed in the work by Sutton and Barto (2018). This managed decay marked the agent's transition from a phase of exploration to a more exploitation-focused approach.

Additionally, our agent was designed with functionalities for saving and loading the Q-table. This feature ensured continuity in the learning process allowing the agent to train on more difficult opponents such as the min-imax algorithm. These combined elements were vital in developing an agent capable of adeptly navigating and mastering its environment, reflecting a comprehensive and effective implementation of the tabular Q-learning algorithm in our project.

4.3 Linear Function Approximation

In Q-learning, function approximation emerges as a powerful technique for estimating Q-values when dealing with extensive state spaces. This approach is particularly useful for environments where a tabular representation of states is impractical due to their sheer volume and variability. In our project, we delved into this concept by employing linear function approximation, a method well-suited for environments characterized by a broad range of possible states Sutton and Barto (2018).

This strategy entailed depicting Q-values as a combination of weighted features, each derived from the game state. Concretely, for a state s and an action a, we approximated the Q-value, $Q(s,a)$, using a linear aggregate of features. Each feature, $f_i(s,a)$ corresponded to a specific weight, w_i^a, leading to the

equation:

$$Q(s,a) \approx f_1(s,a) \cdot w_1^a + \cdots + f_n(s,a) \cdot w_n^a.$$

During feature selection, we honed in on indicators likely to herald successful plays, particularly emphasizing the importance of open-ended sequences. These are lines of tokens extending to three or two in a row, which are not capped by the opponent's pieces or the board's edges. Such open-ended sequences are pivotal as they present the potential for wins or crucial defensive setups, unlike blocked sequences that are sealed on both ends, rendering them strategically inert. These patterns were critical in predicting game-turning moves or essential defensive setups. Utilizing this approach, our agents were better equipped to generalize across varied states, refining their tactics to navigate an expansive spectrum of game situations. The integration of linear function approximation represented a pivotal enhancement to our system's capacity to handle complex state spaces, transcending the limitations of conventional tabular approaches.

4.4 Evaluation and Model Selection

In our project, assessing the performance of our learning agents was a crucial aspect of the methodology. We recognized that simply testing our agents against a random-play opponent wouldn't provide a comprehensive understanding of their capabilities. To gain a more insightful evaluation, we utilized two distinct types of opponents: a basic agent that played randomly and a more advanced agent based on the minimax algorithm, a well-known approach in game theory and artificial intelligence.

The minimax algorithm is a decision rule used for minimizing the possible loss in a worst-case scenario. In the context of games like Connect-4, it operates by constructing a game tree using depth-first search (DFS) and evaluating the game states at various depths. Each level of the tree represents a possible move, and the algorithm iteratively deepens its analysis to a specified depth, which acts as a hyperparameter. By assessing the potential outcomes of moves and counter-moves, the minimax agent strategically selects the move that maximizes its chance of winning while minimizing the opponent's opportunity to win.

Our evaluation protocol entailed extensive testing against these opponents, with our agents participating in hundreds to thousands of game iterations. This comprehensive approach allowed for the statistical smoothing of performance metrics, providing a solid basis for comparative analysis.

Model selection was an iterative and experimental process. We adjusted various hyperparameters, such as the learning rate and exploration rate, to optimize our agents' performance. Our approach was methodical: we altered one parameter at a time while keeping others constant. This strategy helped us isolate the effects of individual parameters on the agents' learning curves and overall performance. By systematically tweaking these settings, we were able to refine our models and select the configurations that yielded the most proficient agents in the context of our project.

5 Results

This section presents the findings from our reinforcement learning project, focused on evaluating the performance of developed agents in Connect-4. We analyzed the effectiveness of our methodologies and

the learning capabilities of our agents through a series of experiments.

5.1 Agent Evaluations

The performance of evaluation agents was assessed to establish a baseline for our learning agents. We pitted Random and MiniMax agents against each other in multiple games. The following table 1 summarizes the outcomes of these matchups:

Matchup (100,000 games)	**Wins (%)**	**Losses (%)**	**Draws (%)**
Random vs. Random	55.48	44.27	0.25
Random vs. MiniMax	11.40	88.60	0.00
MiniMax vs. Random	94.66	5.33	0.00
MiniMax vs. MiniMax	100.00	0.00	0.00

Table 1: *Performance outcomes of evaluation agent matchups. The results indicate the advantage of strategic play and the first-move advantage in Connect-4.*

These results illustrate the significant advantage of strategic play inherent in the MiniMax algorithm and highlight the first-mover advantage in Connect-4. They provide a critical benchmark against which the performance of our reinforcement learning agents is measured.

5.2 Tabular Q-Learning Performance

We began by using tabular Q-learning on a smaller, 4x5 board, to test if the agent would learn quicker on a smaller board. In figure 2, you can see the Q-table size increasing over time, with the x-axis showing every 5000 rounds. We noticed that the Q-table's growth slowed down after a while. Despite this method, we didn't achieve great results on the 4x5 board; the best agent was only about 60% better than playing randomly.

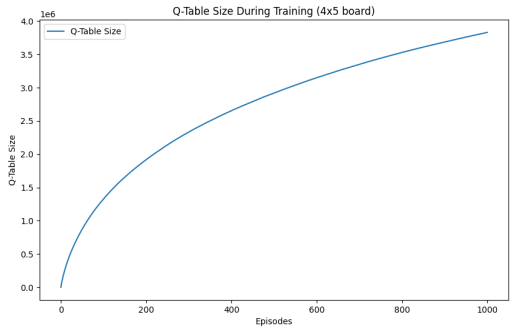

Figure 2: *Q-table size over time.*

.

Figure 3: *Performance of Q-Learning agent against the Random agent.*

After experimenting with the 4x5 board for a while, we decided to try a bigger 6x7 board. We wanted to see how the tabular Q-learner would perform on this larger board. Table 2 shows how the agent's performance improved against a random opponent when we gave it more training episodes.

Episodes	Win Rate (%)	Loss Rate (%)	Draws (%)
0	55.2	44.4	0.4
1.000	57.8	42.0	0.2
10.000	61.2	38.4	0.4
100.000	70.5	29.5	0.0
1.000.000	73.0	26.8	0.2
4.000.000	77.1	22.7	0.2

Table 2: *Win and loss rates of the Q-learning agent against a random opponent across training episodes.*

After some fine-tuning on the learning rate we concluded that a learning rate of 0.3 seemed to yield the best results. In figure 4 one can see how the agent wins over time.

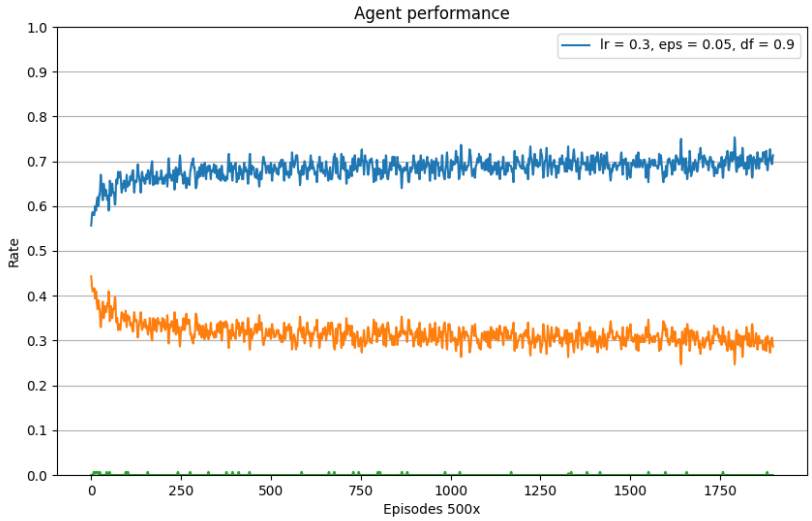

Figure 4: *Comparison of agent and random bot*

.

We compared our findings to a paper by John Persson (2011) and seemed to get very similar results. The highest win rate we got for our agent was 78%.

5.3 Linear Function Approximation Results

The linear function approximation agent's training and performance evaluation were conducted against both Random and MiniMax agents. The evolution of this agent's capabilities is depicted through its win, loss, and draw throughout training episodes.

5.3.1 Training and Evaluation against Random agent

The Approx agent was first trained against Random agent, where it developed some fundamental game-play strategies. Figure 5 shows the agent's performance improvement over time. Notably, the agent's

win rate as both Player 1 and Player 2 increased significantly, which is a testament to the robustness of the learning algorithm.

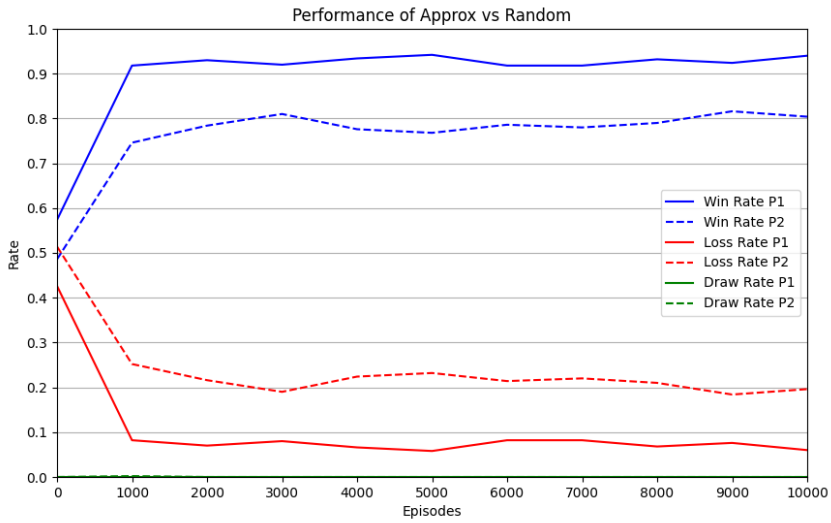

Figure 5: *Win, loss, and draw rates of the Approx agent against the Random agent across training episodes.*

The weights obtained from this initial training phase were preserved and served as a starting point for subsequent training against the more strategic MiniMax agent.

5.3.2 Training and Evaluation against MiniMax agent

Leveraging the learned weights from the training against the Random agent with depth 1, the Approx agent underwent further training against a MiniMax agent. The transition to a more challenging opponent required the agent to refine and adapt its strategies. As demonstrated in Figure 7, the performance metrics indicate the agent's progression in adapting to the challenges posed by a more advanced opponent.

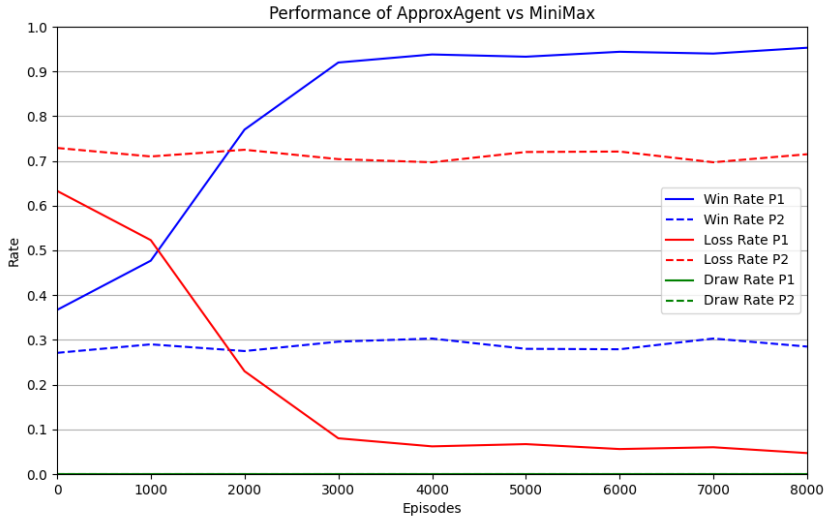

Figure 6: *Win, loss, and draw rates of the Approx agent against the MiniMax agent.*

The graphs show the agent's enhanced ability to compete against a MiniMax strategy, highlighting the effective transfer of learning from the Random agent scenario to a more complex strategic setting. The rate of improvement against the MiniMax agent, while more gradual, indicates a solid understanding and application of strategic concepts learned during the agent's training.

These results demonstrate the potential for transfer learning in reinforcement learning agents, where knowledge gained in one context can be successfully applied to more challenging scenarios.

5.4 Challenges and Limitations

We encountered challenges related to the sparse reward structure of Connect-4, which we mitigated through reward shaping. Additionally, training on larger boards significantly increased computational demands, impacting our model's optimization speed.

The main obstacle in the project was the delay of unit-testing. Along the way we stumbled on a few major bugs which delayed our project much more than we care to admit. Different code design was implemented by contributors making integration laborious. Unit-tests would have prevented these troubles.

One of the most significant challenges in developing the Linear function approximation model was selecting appropriate features for a given state. The Linear function approximation model is limited by the features that have been chosen as the model depends on the features to adapt to each game state.

6 Conclusions and Future Work

6.1 Conclusions

This project has successfully demonstrated the application of reinforcement learning techniques in strategic board games through the development and assessment of agents in the game Connect-4. By employing tabular Q-learning and linear function approximation, our agents not only grasped the fundamental rules of Connect-4 but also formulated strategic behaviors that improved their game play against both stochastic and heuristic-based opponents. However the agents developed strategies against certain opponents. It is highly likely that these strategies are overfitted or simply not effective against other opponents.

The use of temporal difference with more steps would have possibly yielded a more sophisticated tabular q-learning model.

The viability of reinforcement learning to excel in discrete, turn-based games was underscored by our results. Despite the inherent challenges of sparse rewards and the vastness of the state spaces, our agents displayed a commendable capacity for strategy development. The use of feature selection was pivotal in linear function approximation, directly influencing the agents' learning efficacy and their capability to extrapolate knowledge to new situations.

Furthermore, our investigation revealed substantial computational demands when scaling to larger state spaces, underscoring the necessity for algorithms that balance complexity with computational feasibility like linear function approximation or non-linear function approximation with neural networks.

6.2 Future Work

This project covers the application of the more simple methods that are popular in RL. Generally, it is good practice to implement simpler models before experimenting with complex models that are hard to explain. Potential directions include:

1. **Advanced Function Approximation Techniques**: The integration of sophisticated function approximation methods such as deep learning could further enhance agent performance. The application of convolutional neural networks could be particularly promising for recognizing spatial patterns inherent in Connect-4.

2. **Enhanced Feature Engineering**: Further research into feature engineering could refine the agent's perception of the game state, leading to more nuanced strategies and faster learning curves.

3. **Alternative Reinforcement Learning Algorithms**: The exploration of other reinforcement learning paradigms, such as deep Q-networks (DQN) and policy gradient methods, could provide more robust learning mechanisms and improved decision-making capabilities.

4. **Variants and Extensions of Connect-4**: Investigating different variations of Connect-4, with alternative board dimensions or rule alterations, would challenge the agents to adapt to new complexities and strategies.

5. **Optimizing Reward Structures**: An in-depth exploration of various reward structures and their effects on learning speed and efficacy could offer valuable insights, potentially leading to more efficient training protocols for reinforcement learning agents.

In retrospect, this project has enhanced our understanding of reinforcement learning's role in board games, providing a foundation for further exploration. It underscores the ongoing need for collaborative and innovative research in the application of artificial intelligence within this area.

References

Allis, V. (1988). *A Knowledge-Based Approach of Connect Four: The Game is Over, White to Move Wins*. M.sc. thesis, Vrije Universiteit Amsterdam.

Ayub, A. and Wagner, A. R. (2020). Teach me what you want to play: Learning variants of connect four through human-robot interaction. In *Social Robotics*, pages 349–358. Springer International Publishing.

John Persson, T. J. (2011). Self-learning game player connect-4 with q-learning. *KTH*.

Nasa, R., Didwania, R., Maji, S., and Kumar, V. (2018). Alpha-beta pruning in mini-max algorithm –an optimized approach for a connect-4 game. *International Research Journal of Engineering and Technology (IRJET)*, 5(04):1637.

Silver, D., Huang, A., Maddison, C. J., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, 2nd edition.

Tromp, J. (n.d.). John's connect four playground. `https://tromp.github.io/c4/c4.html`. Retrieved November 24, 2023.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, University of Cambridge.

7 Collaboration

The successful execution of our project, which centered on the application of reinforcement learning to Connect-4, was the result of a well-coordinated effort among our team members. Each member brought valuable expertise to the project, and while individual responsibilities were defined, the collaborative nature of our work meant that each member contributed to all aspects of the project.

- **Björn Thor Stefánsson** took charge of the Q-learning algorithm's development and served as the main programmer. His organizational skills ensured seamless integration of the various components of our project. Björn also conducted rigorous unit tests on the game logic, contributing to a solid foundation for our agent's learning capabilities.
- **Kristján Sölvi Örnólfsson** was a key player in both Q-learning algorithm enhancement and report writing. His analytical skills were critical in evaluating our models and synthesizing our findings into the final report, ensuring that our results were communicated clearly and effectively.
- **Viktor Már Guðmundsson** was instrumental in advancing our Linear Approximation approach and developing the MiniMax agent. His focus on feature selection and meticulous unit tests on agent training and evaluation provided the necessary rigor to our analysis and helped refine our agents' performance.

Through regular meetings and strategy sessions, our team ensured that every member was actively involved in decision-making and problem-solving, promoting a dynamic exchange of ideas. This collaborative environment was vital for the equal distribution of workload and for achieving a synergy that propelled our project forward. Collectively, we fostered an innovative approach to problem-solving, which is reflected in the depth and breadth of our comprehensive final report.

8 Acknowledgments

We want to extend our gratitude to Steinn Guðmundsson, our professor at the University of Iceland, whose guidance and expertise were invaluable throughout this project. Special thanks go to our peers in the REI505M Machine Learning course for their insightful feedback and suggestions. The code for this project is available on GitHub at github.com/viktormar123/FinalProjectREI505M.