Tic Tac Toe with Reinforcement Learning

PROJECT REPORT

Rida Sohail | Machine Learning | May 21st, 2023

Table of Contents

Problem Description: 1

Team Member Roles / Goals: 1

State-of-the-art / Related Work: 1

Approach: 2

Why Q-network? 3

Experiments and Results: 3

Lessons Learned about ML topics: 4

Challenges Faced and Improvements: 5

Conclusion: 6

Github: https://github.com/ridasohail92/tictactoe_using_RL.git

Demo video: https://drive.google.com/file/d/10bzpSEFclf2NwAyJV9DheTpiugYLPQCF/view?usp=sharing

Paper for baseline model comparison: Ho, Jocelyn; Huang, Jeffrey; Chang, Benjamin; Liu, Allison; Liu, Zoe (2022): Reinforcement Learning: Playing Tic-Tac-Toe. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.20407575.v1

Problem Description:

This project aims to develop an AI agent capable of playing Tic-Tac-Toe through reinforcement learning. By training a deep neural network with the Q-learning algorithm, the agent will learn to make successful moves by receiving rewards for its decisions in the game environment. The project involves designing and implementing the neural network, collecting and preprocessing data, and training the model through various iterations to optimize its behavior. Once trained, the agent will be able to play Tic-Tac-Toe effectively and efficiently, demonstrating the capabilities of reinforcement learning in creating intelligent agents. The project will also include a comparative analysis of the agent's performance against other reinforcement learning approaches and baseline models. Overall, the project seeks to explore the potential of reinforcement learning in creating agents capable of learning and adapting to complex environments.

Team Member Roles / Goals:

The primary goal of this project is to develop an AI agent capable of playing Tic-Tac-Toe using reinforcement learning techniques. Specifically, the objectives include:

1. Designing and implementing a deep neural network to serve as the foundation for the AI agent.
2. Training the agent using the Q-learning algorithm to enable it to learn optimal moves based on rewards received in the game environment.
3. Evaluating the performance of the AI agent against other reinforcement learning approaches and baseline models.
4. Demonstrating the effectiveness and efficiency of reinforcement learning in creating intelligent agents capable of learning and adapting to complex environments.
5. Gaining insights into the challenges and lessons learned in applying machine learning techniques to the development of AI agents.

State-of-the-art / Related Work:

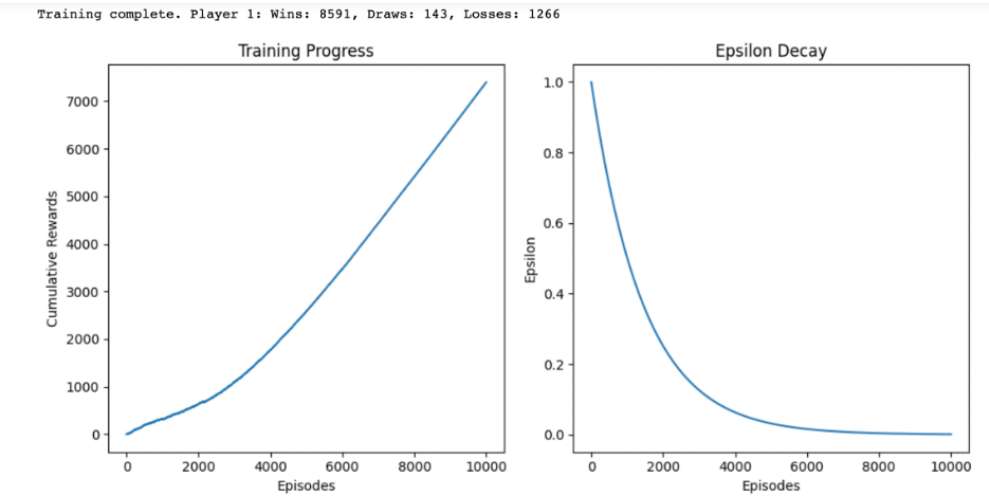
Several existing approaches have been explored in the field of Tic-Tac-Toe AI. Notably, the Monte Carlo Tree Search (MCTS) algorithm, minimax algorithm, and vanilla Q-learning have been widely studied. The MCTS algorithm uses

random simulations to build a search tree that represents possible moves and their outcomes. It dynamically expands and explores the tree to guide the agent's decision-making process. While effective, MCTS can be computationally expensive for larger game spaces. The minimax algorithm, on the other hand, exhaustively searches all possible game states to determine the optimal move. This approach guarantees an optimal outcome but suffers from scalability issues in more complex games. Vanilla Q-learning, a form of temporal difference learning, is a model-free approach that uses a tabular representation of states and actions. It updates the Q-values based on rewards received during gameplay. However, this method becomes infeasible for large state spaces due to the exponential growth of the table.

Approach:

In this project, the Deep Q-Network (DQN) approach is employed to overcome the limitations of the tabular Q-learning method. The DQN utilizes a deep neural network to approximate the Q-values, enabling it to handle larger state spaces more efficiently. The network takes the current game state as input and outputs the Q-values for each possible action. By training the DQN through iterations of 10,000 gameplay and updating the network's weights using a variant of the Q-learning algorithm, the agent learns to make optimal moves based on the received rewards. The learning algorithm aims to maximize cumulative rewards over multiple episodes where it receives a reward of 1 for winning, 0.5 for draw and no reward for losing.

After the end of the training, I plotted the cumulative rewards against the number of episodes to visualize the training progress of the agent. I also plotted the epsilon value against episodes to show the visualization of the epsilon decay function.



At the end of the training for 10,000 episodes, the agent won on average of 85 % games compared to the 75% win rate for an agent trained for 300,000 episodes as shown in the paper I used as a baseline comparison for the performance of my agent.

Why Q-network?

Using a Q-network (deep Q-learning) in Tic-Tac-Toe offers advantages such as scalability to complex environments, generalization of knowledge across similar states, automatic feature extraction, support for continuous learning, efficient memory usage, and potential transfer learning. These benefits enable efficient handling of large state spaces, more effective decision-making, elimination of manual feature engineering, adaptation and improvement over time, compact representation of knowledge, and leveraging previous task knowledge. Q-networks are powerful tools for complex and high-dimensional environments compared to traditional tabular Q-learning.

Experiments and Results:

To evaluate the performance of my agent, I conducted a series of tests in which it played 25 games against a human opponent. I recorded the number of wins, losses, and draws to gain a better understanding of the agent's capabilities. I compared the performance of three different agents in these tests:

Agents	**Wins**	**Lose**	**Draw**
Naive	1	21	3
Trained Agent	5	18	2
Trained Agent with hyperparameter tuning	15	1	9

The results demonstrate a clear improvement in the agent's performance as the training duration and optimization efforts increase. Agent 1, which was minimally trained, performed relatively poorly, winning only 1 out of 25 games. This indicates that the agent lacked sufficient training to make effective moves.

Agent 2, trained for 10,000 episodes without optimization, exhibited improvement. It won 5 games, lost 18, and drew 2, indicating a much stronger performance compared to Agent 1. This suggests that a higher number of training episodes allowed the agent to learn better strategies and make more successful moves.

Agent 3, trained for 10,000 episodes with hyperparameter tuning, showed the best performance among the three agents. It won 15 games, lost only 1, and drew 9. The optimization process likely helped fine-tune the agent's learning process, resulting in more optimal decision-making and significant improved performance. As per the results, the success rate (win/draw percentage) for my agent comes out to be 96 % compared to the 90 % success rate in the paper using plain Q-learning for training of their agent.

These results highlight the importance of training duration and hyperparameter tuning in reinforcement learning. Increasing the number of training episodes allows the agent to gather more experience and refine its strategies, leading to better performance. Optimizing the agent's hyperparameters further enhances its learning and decision-making abilities, yielding even stronger performance. These findings suggest that with additional training and optimization, the agent's performance can continue to improve, potentially achieving even higher win rates and better gameplay.

Lessons Learned about ML topics:

Throughout this project, several key lessons were learned about various ML topics:

1. **Deep Reinforcement Learning**: The project provided a practical understanding of deep reinforcement learning, a powerful technique that combines deep neural networks with reinforcement learning algorithms. This approach allows the AI agent to learn and make decisions based on rewards and punishments received in the game environment.
2. **Q-learning Algorithm:** The Q-learning algorithm played a crucial role in training the AI agent. By assigning Q-values to different actions in each game state, the agent learned to make optimal decisions. The concepts of

the discount factor and learning rate were explored and fine-tuned to balance the trade-off between immediate rewards and long-term benefits.

3. **Iterative Refinement**: The training process of the AI agent involved iterative refinement. Starting with minimal training, the agent gradually improved its performance by playing numerous games and updating its neural network based on the Q-learning algorithm. This iterative approach allowed for continuous learning and adaptation, enhancing the agent's decision-making capabilities.
4. **Hyperparameter Tuning:** Fine-tuning the hyperparameters was an essential step in optimizing the agent's performance. Adjusting parameters such as the learning rate, exploration rate, and neural network architecture played a significant role in improving the agent's gameplay. By experimenting with different values and evaluating the results, the agent's learning process was further enhanced.
5. **Continuous Evaluation:** A crucial lesson was the importance of continuous evaluation throughout the project. Regularly analyzing the agent's performance, such as win rates, losses, and draws, provided valuable insights into its progress. By monitoring these metrics, it was possible to identify areas for improvement and guide further training and optimization efforts.
6. **Exploration vs. Exploitation**: The exploration-exploitation trade-off was a critical aspect of the project. Balancing the agent's exploration of new moves and exploiting its existing knowledge was essential for effective learning. The epsilon value, controlling the exploration rate, was fine-tuned to strike the right balance between exploring new strategies and exploiting known successful moves.

By gaining hands-on experience with these ML topics, a deeper understanding of their practical implications was achieved. The project highlighted the importance of iteratively refining the training process, fine-tuning hyperparameters, continuous evaluation, and finding the right balance between exploration and exploitation. These lessons can be applied to future ML projects, helping to improve the training and performance of AI agents in various domains.

Challenges Faced and Improvements:

During the course of this project, several challenges were encountered and subsequent improvements can be made to the project.

Training Time: One of the major challenges faced was the significant amount of time required for training the AI agent. Training deep neural networks with reinforcement learning algorithms can be computationally intensive, especially when dealing with large iterations. Several times during training the agent, either my system crashed or the kernel of jupyter's notebook died.

Key improvements to enhance the project:

1. **Comparative Analysis**: Benchmark the AI agent against other reinforcement learning algorithms like DQN, policy gradients, and MCTS to understand its strengths and weaknesses, performance trade-offs, and generalizability in Tic-Tac-Toe.
2. **User Interface Integration**: Incorporate a visually interactive interface for human players to compete against the AI agent, providing features like difficulty levels, game statistics, and an intuitive move selection interface for practical evaluation and user engagement.
3. **Training against a Smart Agent:** Enhance the agent's learning capabilities by training it against a pre-existing intelligent opponent, introducing novel strategies, and challenging its gameplay. Gradually increase the difficulty to help the agent adapt and generalize its knowledge effectively.

These improvements will validate the agent's performance, improve user experience, and advance the understanding and application of reinforcement learning techniques in game playing and beyond.

Conclusion:

In conclusion, through the utilization of reinforcement learning and the development of a Q-network-based AI agent, I successfully created an autonomous Tic-Tac-Toe player. The agent learned to play the game strategically and achieved competitive performance.