Reinforcement Learning for Tic-Tac-Toe

Abstract

For our Reinforcement Learning project we decided to design, implement, and investigate how an agent can learn to play tic-tac-toe with reinforcement learning. We specifically wanted to determine if specific characteristics of agents might produce higher rates of winning. We decided to analyze how epsilon values influenced the training stage of agents, and ultimately if it correlates to that agent's winning rate. Additionally, we wanted to explore if there were other aspects of the game not related to agent training processes that influenced winning rates. Our results showed that agents learn unique policies based on if they are trained as the first or second player in the game, and the player who takes the first turn has a strong natural advantage. Additionally, the data indicates that lower epsilon values produce more effective agents, and that there is little benefit to significant exploration.

Members' contributions:

Kenan Anderson - Environment design, final report, debugging

Rodrigo Quijano-luna - Q-Learning agent, results visualization, debugging

Hafeeza Mughal - Object class for Player, final report, manual testing agents

Shuyang Ma - Epsilon value testing, creating data visualizations, debugging

Introduction

Reinforcement Learning is a branch of Machine Learning in which an agent's objective is to maximize its reward while it engages with a dynamic environment. For this project our dynamic environment is simulated through a tic tac toe game board. As the agent plays tic tac toe it observes the current state of the board and selects an action that would maximize its reward, ultimately increasing its chance of winning the game. Through each iteration the agent learns and improves its strategy until it produces an optimal policy (Shyalika). Our agent improves its strategy through q-learning, a value-based learning algorithm. During each iteration the agent updates the Q value function for state-actions, where given the current state and a specific action dictated by a policy the expected reward is the q-value.

The interesting component of training an agent with tic-tac-toe is the existence of an opponent. We modeled our code after existing ones from literature like Zhang's version of q-learning tic tac tac. Our algorithm simulates tic tac toe matches between two agents as well as an agent against a human player. The main hypothesis we created is that an advantage would be given to the agent executing the first move because the number of available states decreases after each action. Our secondary hypothesis is about the effect of the epsilon value, the rate of exploration, during the training stage of the agent. We hypothesized that lower epsilon values would result in more wins for an agent. Our results showed the 0.7 is the best epsilon value, and that players who took the first turn generally won more games.

Methods

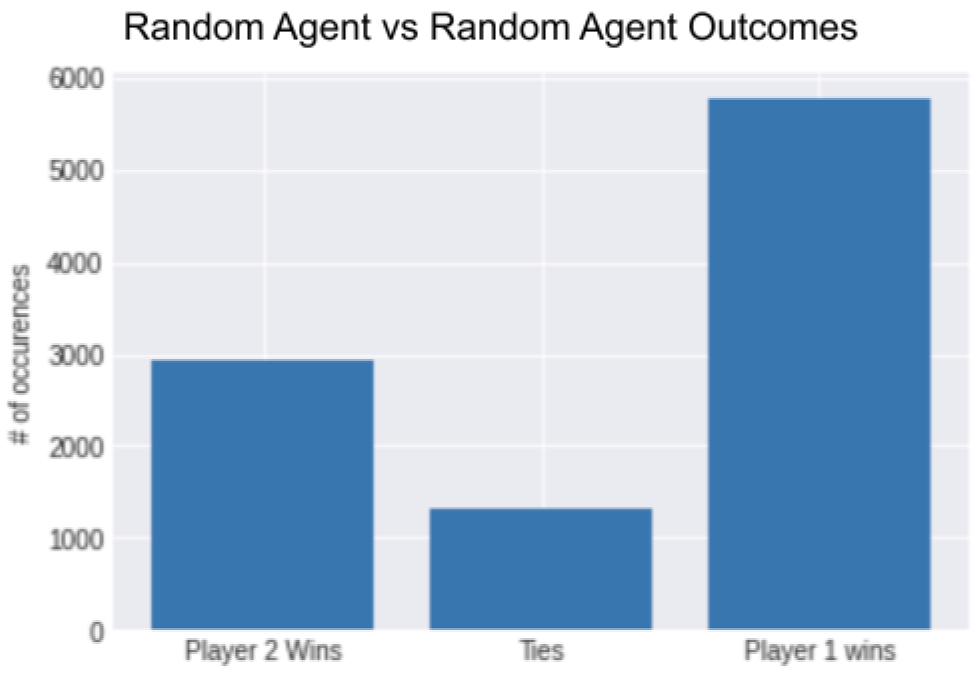
Referencing our class resources and existing literature we implemented a tic tac toe board that simulated agents playing against each other and an agent playing against a human. The algorithm consists of four classes: the State class that mainly manages the board, the QPlayer class that manages the q-agent's policy, the HumanPlayer class that accepts user input for a match, and a RandomPlayer class that randomly plays, i.e. chooses actions.

During the training (either against a random agent or another ‘Q-Agent’) the agent would create a saved “hash” of every state it observed. This was far more effective than trying to pre-populate some sort of table with every possible state. During each move, the agent would consider each possible move, and consult the moves existing value (if it had one) or give it a value of zero (if it hadn’t been seen before). At the end of the game the agent would either receive a reward of 1 if it won the game, 0 if it lost, or a partial reward if it was a draw.

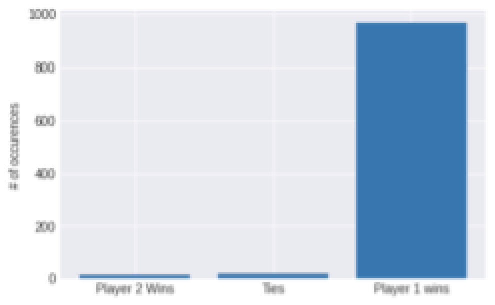
After developing a general method for training, we decided to attempt to test various epsilon values, to see which produced the most effective trained agents. Our q-agents were trained against a random player across 10000 games to produce a policy. After being trained with a certain epsilon value, the agent is then played against another pre-trained agent that, and a record is kept of it’s wins and losses. This process was repeated for each epsilon value. We tested epsilon values at regular intervals both between 0 and 1, as well as between 0 and 0.1, to try and get as detailed a view as possible about the optimal value. We found that, while there was a lot of randomness involved due to the training by random agents, in general, lower epsilon values tended to produce both better testing and training outcomes.

Results

Overall, the results were pretty effective. We began by testing random agents against each other to measure a baseline to compare the trained agents too. We found that random agents have a fairly diverse set of outcomes but with a heavy skew towards whichever agent moves first winning.

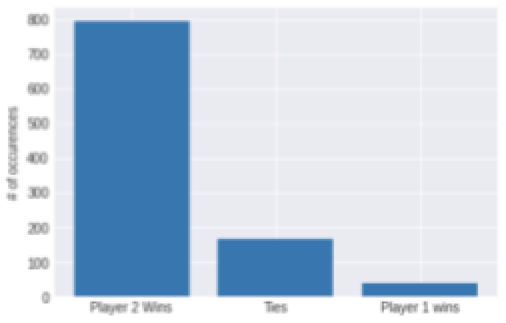


However, after this we were initially concerned that our agents were not performing optimally. When played against each other, they failed to frequently produce draws as you'd expect two optimal players to do. Additionally, their win-rates against random agents, while large, were disappointing. However, we realized that it was due to an error where both the first and second player Q-Agents were loading the first player policy. This would cause the second player to play sub-optimally, as the strategy it had developed for going first was different than for when it was playing second. However, after this issue was resolved, the agents performed as expected. When playing each other, they played to a draw every single game, and had nearly perfect win-rates against random-agents.

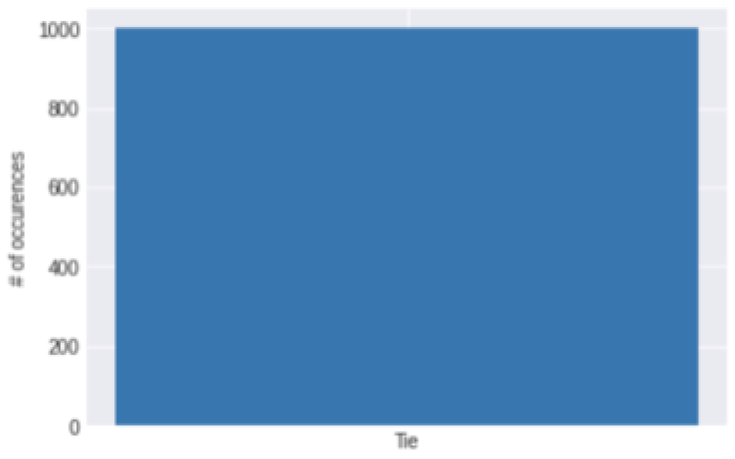


(To Left)
Outcomes for a
Q-Agent (p1) vs
RandomAgent(p2)

(To Right)
Outcomes for a
RandomAgent (p1)
vs Q-Agent(p2)



(Below)
Results for Q-Agent vs Q-Agent



Additionally, we tested them against a human player and found that, when the appropriate policy was loaded, they were able to strategically counter some of the human player's moves . But, the human player was still able to win. In our limited human testing, the frequency of errors increased when the agent loaded the incorrect policy (for example, they were player 1, but loaded the policy for player 2). Finally, we found that the trained Q-Learning agents could defeat random agents, but still had a much higher win-rate as the first player (> 90%) than as the second player (≈ 70%) . We believe this is just due to the large advantage that the first player in tic-tac-toe has in being able to force draws. The final aspect of our results worth discussion was the epsilon value tests. While the random nature of the competing agents added a lot of variance to our results, we found that in general, lower epsilon values were optimal for training. We believe this is because the strategy for tic-tac-toe is fairly straight forward, and there are limited benefits to exploration or trying to "innovate" a new strategy.

Discussion

The largest limitation we faced in this project was that while tic-tac-toe may seem to be a symmetrical game, whether an agent plays first or second actually had a large effect on the Q-learning algorithms ability to perform effectively. For example, while a Q-Learning agent that was trained as player one was able to beat a random agent nearly 100% of the time while playing first, it would frequently draw or lose games if it attempted the same strategy as the second player. This is not hard to address temporarily, as you could create a "super-agent" that loaded both the player one and player two policies, and then used the appropriate one based on its playing order. However, there is still an open question as to how effectively the agent would learn if it actively switched sides during the training process, as opposed to having two separate policies generated by two separate agents. Additionally, due to time and resource constraints, we were unable to try testing a game against a human, but with additional resources, that might produce more optimal and fine-tuned results than testing against both random and previously trained Q-Agents.

References:

1. Shyalika, Chathurangi. “A Beginners Guide to Q-Learning.” *Medium*, Towards Data Science, 16 Nov. 2019, towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c.
2. Zhang, Jeremy. “Reinforcement Learning - Implement TicTacToe.” *Medium*, Towards Data Science, 24 Aug. 2019, towardsdatascience.com/reinforcement-learning-implement-tictactoe-189582bea542.