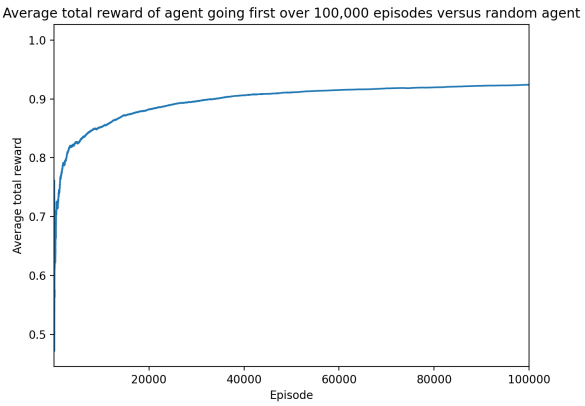
Reinforcement Learning with Tic-Tac-Toe Checkpoint

Jacob Baber Stephen Cain

For TD learning, Once I had established the environment and the agent, our initial experiment involved observing the behavior of the agent by playing against it. The agent was assigned rewards of -1, 0, and 1 for losing, tying, and winning, respectively. It also had alpha and epsilon values of 0.1. I played and won dozens of games against the agent, but eventually it managed to get some ties and even a few wins. The progress was slow but I could definitely see the agent's policy improving.

After playing against our agent for some time, I decided to create another agent that would always choose a random available action, and see the results as our agent played against it over thousands of games. I hypothesised that as the number of games went on, the average reward of the agent would increase as it adapts it's policy to beat the random action agent, eventually converging at a high average reward. This is exactly what happened, the average total reward increased as the number of games went on, and eventually converged at an average reward of about 0.9. I believe that the average reward against the random agent can converge at a higher rate, perhaps even .99, but the epsilon value of 0.1 for our agent became more harmful than helpful as it's policy became more optimized.



As far as the difficulties encountered for TD learning, the primary one was creating the value function correctly. The nature of Tic Tac Toe makes it so a reward can only be given once the game is over, but TD learning is designed to revalue a state after going to another state. I believed I implemented a correct value function of $v(s) = v(s) + \alpha(R - v(s))$ at a terminal state and $v(s) = v(s) + \alpha(\gamma \cdot v(s+1) - v(s))$ at nonterminal states.

The next step in regards to TD learning would be to make the agent play against itself for a large amount of games. The agent can only learn so much against a random action agent. After the average reward converges for the agent against itself, I'll adjust the epsilon value over a number of training sessions so the agent is no longer randomly taking and exploring actions and is rather using it's policy for each and every action. At this point the agent's policy should be the optimal policy for going first in tic tac toe.

For the overall project, once both the TD learning and Q learning agents' policies are optimized, we will compare the policies, and let the agents play against each other, to ultimately see which policy the between the two is optimal.