COGS 182 Project 2

James Cor
A14951555

March 15th, 2021

Abstract

The goal of this project was to examine the performance of two different Reinforcement Algorithms on their ability to play a modified version of Checkers against a random player. The first algorithm, Q-learning, performed decently well, achieving an overall win rate of 51% and 64% when not counting ties. The second algorithm, Monte Carlo Tree Search, performed extraordinarily well, achieving an overall win rate of 84% and 90% when not counting ties.

1 Introduction

In Project 2, I wanted to apply the knowledge gained from COGS 182 and train a Reinforcement Learning agent to play the game Checkers. Originally, I wanted to train an agent to play Chess, but I quickly realized that this may require more computing power than I have available and switched to a modified version Checkers. Details on this version can be found in a later section.

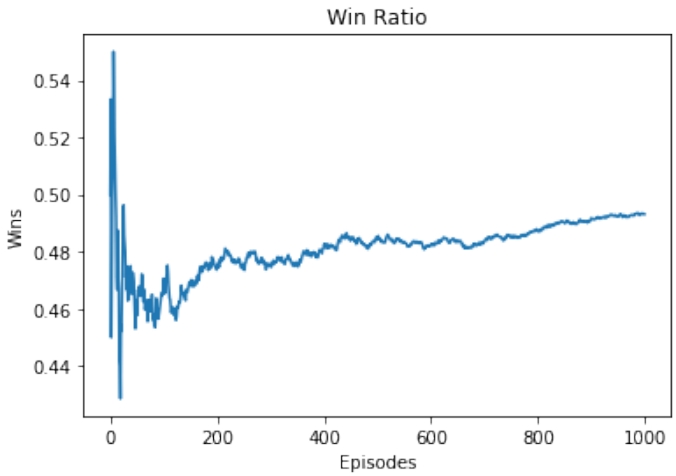
I chose to tackle this problem using Q-learning and Monte Carlo Tree Search. I chose these two algorithms because of both their simplicity (in terms of how they solve their problems) and how differently they evaluate values of states and/or action. Regardless, I would consider an agent that can beat a random agent more often than it loses, with minimal ties successful.

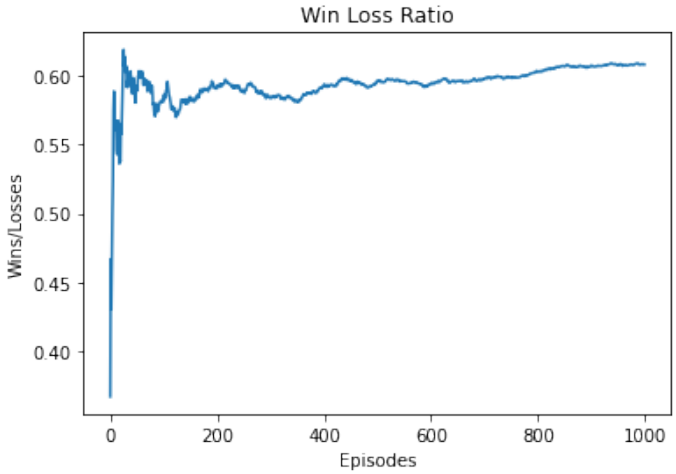
2 Environment

To reduce the branching factor of Checkers, I greatly simplified the game for our needs. Similar to the original version of Checkers, I have two colors of players (black and white) arranged on opposite sides of the 8-by-8 board on alternating squares. Unlike Checkers, the taking system is greatly simplified, in that players cannot chain jumps together. Additionally, when pieces make it to the opposite side, rather the being queened, the piece simply can't be moved as pieces can only move in one direction. As a result, the win condition is to have the most pieces once one of the players is out of moves.

3 Results

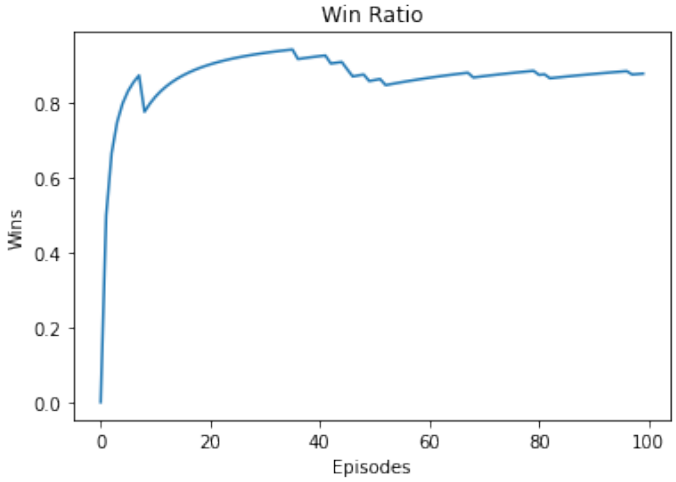
The results for the Q-learning agent were obtained by averaging 5 episodes of 1000 episodes with parameters: alpha = 0.1, gamma = 0.9, and epsilon = 0.05.
Here are the plots for Q-learning:

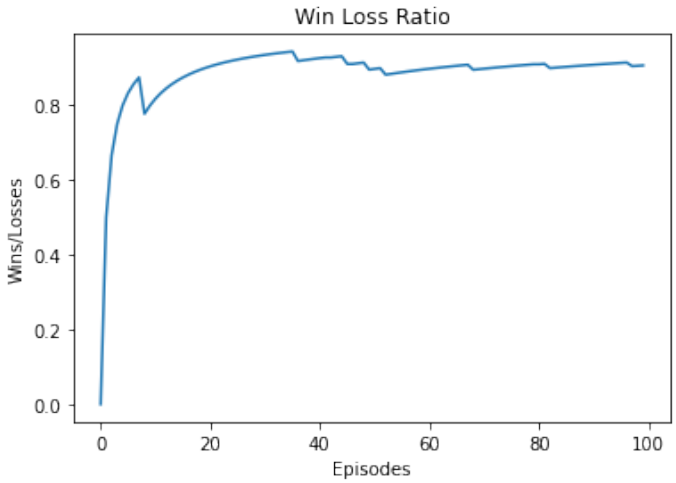




Total wins: 2468
Total loss: 1585
Total ties: 952

The results for the MCTS agent were obtained by running 100 episodes with parameters: num_trials = 5.
Here are the plots for Monte Carlo Tree Search:





Total wins: 88
Total loss: 8
Total ties: 4

A possibility for why Monte Carlo Tree Search out performed Q-learning by such a drastic amount is likely due to the amount of states each method has to experience. In the MCTS method, the agent tries every possible action, and estimates which one would be best using rollout. Meanwhile, the Q-learning agent followed an epsilon-greedy strategy which balances exploration and exploitation in a very simplistic manner. I believe that using larger epsilon would've led to the agent finding an optimal solution earlier, but it would struggle to perform optimally on average, as it would still explore consistently. Additionally, if a lower epsilon were used, the agent would take far too many episodes to come

across an optimal solution.

4 Deficiencies

A limitation of my testing is the amount of trials and number of experiments I was able to perform. Specifically for Monte Carlo Tree Search, each trial took too long to complete. With more experiments I would have a much more confident answer as to what the converged win rate would be. Additionally, I would be able to experiment with different parameters, such as number of trials, alpha, gamma, and number of episodes.

5 Conclusion

Overall, my goal was to train a Reinforcement Learning agent to play the game Checkers. I chose to tackled this problem using Q-learning and Monte Carlo Tree Search, with MCTS performing significantly better than Q-learning. Regardless, I both these agents were successful in earning a significant win ratio against a random agent.

6 Team Contribution

I worked on this alone, so I did everything.

7 Links

Here is the link to the YouTube video: https://youtu.be/f1_fO2E-l3k
Here is the link to the public GitHub repo: https://github.com/JCOR11599/cogs182_proj2