CS541 Final Project Report

Team members and contributions:

Travis McGowan: Monte Carlo code
Austin Britton: Minimax Agent Player Algorithm code
Jeremy Hamilton: Mancala Game code, Deep Q Learning Player code, Random Player code
James Bao: Testing and Trial Simulation
Vicky Haney: Max Agent Player, team organization

Introduction

Our team has produced five (5) different player (or agent) algorithms for the game, Mancala. We were able to conduct testing and trial simulation for all five algorithms and our measured outcome is shown below under Results. Mancala is a competitive 2 player game that is deterministic. The state space for the game can be upper bounded by about 12^{48} (12 dishes, 48 stones), so the game cannot be determined by searching the entire tree. So, the goal of this project is to compare different AI methods and compete the methods against each other.

Mancala Game rules

The Mancala game board consists of 14 dishes, two of which are score dishes and the other 12 are split between the two players. To begin the game four stones are placed in each of the 12 non-score dishes. A move is made by a player choosing one of their six dishes which contains stones. The stones in the chosen dish are all picked up and then each stone is placed one at a time in the next dish moving in a clockwise

motion. A stone can be placed in each of the twelve non-scoring dishes as well as the current player's scoring dish. The opponent's score dish will be skipped over. Play alternates between players. The game is over when one player scores 25 or more points [1]. There are two additional special rules:

1) When the last piece in your hand lands in your scoring dish, take another turn.
2) When the last piece in your hand lands in one of your own dishes, if that dish had been empty you get to keep all of the pieces in your opponent's dish on the opposite side. Put those captured pieces, as well as the last piece that you just played on your side, into the scoring dish.[2]

Previous Work

Mancala is a very popular game so several similar papers have been written on the topic and we sourced their information quite deeply. The main paper that we looked at to reflect some of the results found by DaValio and Langenborg[1]. We decided to replicate some of these results, but we also decided to add in a Deep Q Learning Agent in order to add some fresh data.

Code

Our code is available in GitHub via: https://github.com/flyingcircle/AI-project.

Player Agent Algorithms

Random Player, DeepQPlayer, Monte Carlo, Max Agent Player, MiniMax Player.

Agent Strategies

Random Agent

Our random agent strategy selects any randomly selected dish that is a non-empty dish. Dish vs stone vs AI ethics… oh what a choice to ponder

Max Agent

The Max agent strategy is to pick the dish that will ensure the player receives the highest amount of stones in its scoring dish. The strategy for Max agent contains three rules: 1) It is an advantage to end in our scoring dish so the player earns another turn. Hence, if any of the non empty dishes have exactly the number of stones to end our turn in our scoring dish, take that move. First we must select the dish closest to the bank. Dishes further away would overtake the closer dishes that meet the condition. This strategy is also used if any of the players' dishes contain enough stones to go around the entire game and return in the player's scoring dish. 2) If there are enough stones in a dish that allows the player to end its turn in an empty dish on the player's side, the player can collect the opponents stones in the opposite dish as well as the one piece in their side and place them in its scoring well. 3) If any of the non empty dishes have more than enough stones to get to our bank but continue on to the players side, take it.

Deep Q-Learning Agent

Deep Q Reinforcement Learning uses a neural network in place of a table that is used in typical Q learning. For this project, to avoid long training times, we stuck to a pretty small neural network of 3 layers using ReLu's as activation functions and a total of about 500 nodes. Considering the state space of Mancala, we didn't expect this neural net to compete with the likes of a real tree search. However, the advantage of using this method is that once it is trained, it is able to play quite quickly since it does not have to do any tree search on demand.

In a more rigorous environment, I would have trained separately for Player 1 and Player 2 since the game is not exactly symmetric for both players. However, the game is close enough that for Player 2, I simply passed it through the same trained model. This gave surprisingly excellent results.

Minimax

Minimax starts by making all possible moves for the selected player starting at depth 0. After each move is made, minimax is called recursively, adding one to the depth, and toggling the boolean value as a parameter to signify when the minimizer is being leveraged, or the maximizer, alternating between the two. At depth one, the minimizer makes all possible moves based on the board state passed in from the maximizer. The minimax algorithm has a base case depth of 2, where the state with the maximum number of stones is returned, as well as the index used to get that max value.

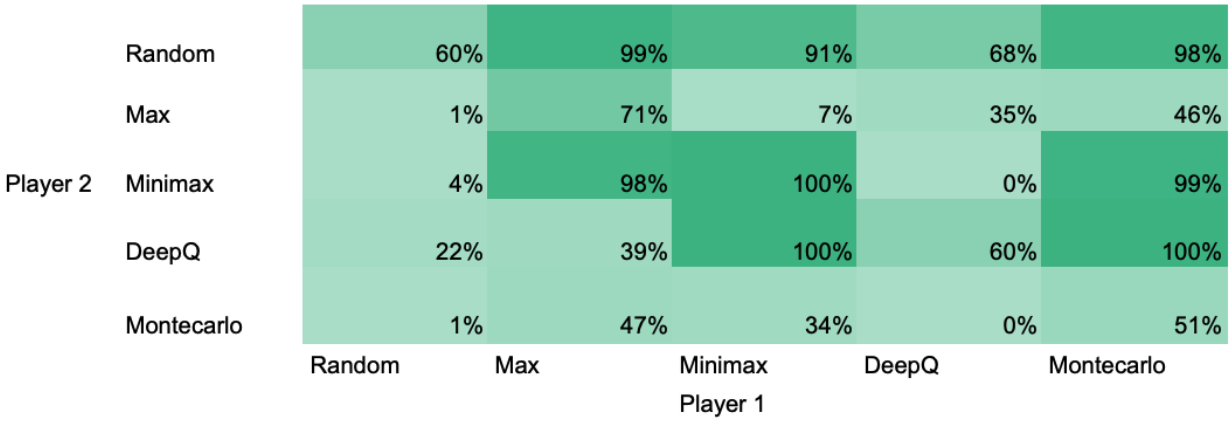
Once returned, the minimizer will collect what it thinks is the current minimum number of stones. Upon completion at each level, the minimum or maximum value is returned and minimax in general will return what it perceives as the best possible move, using total stones in the calling player's dish as a heuristic. It was decided that the difference between depth of 2 was close enough consistently to a depth of 4 that it was not worth the added computational time. It should be noted that the depth limit should always be depth % 2 == 0 such that the base case works to maximize the result.

Monte Carlo

For each play, the Monte Carlo algorithm runs a series of simulations. Simulations are stochastic in nature due to use of the random player algorithm for both players. Each simulation begins with the current board state, and ends when 5 plays have passed or the game has ended (whichever occurs first). When each simulation is complete a value is attributed to the first play in that series by way of a heuristic. That heuristic is the difference between the number of stones captured by the primary player and it’s opponent.

Results

The results show Player 1 (horizontal axis) versus Player 2 (vertical axis). Every combination of agents was tested for 100 games.



Conclusion

We were expecting that Random players would lose the most often which seems to be the case in our results. We also expected DeepQ to be more successful, especially against Monteclaro. This may be that our DeepQ model wasn't big enough to handle the complexity of the game.

Minimax as Player 1 won 100% of the time, but the study we compared these results to show that Minimax wins against itself 66% of the time. This may be due to a difference in the depth. Also recall that Minimax is optimal when playing against an optimal player. After reviewing the code one more time, we noticed that the Minmax player wasn't considering the possibility of multiple turns in a row. So we think our results for Minimax may be partially due to a coding error.

In future work, there is more room to expand on the DeepQ model. We feel that the DeepQ model was likely too small to encapsulate all of the nuances of the game. So expanding more on the model to make it bigger, more precision on finding better hyperparameters, or finding a better reward heuristic. There are many other models that

we could have tested, but weren't able to due to time constraints such as Asynchronous Advantage Actor - Critic Agent (A3C). A question that came to mind is why is Player 2 not handicapped? Every single result, no matter the actor, showed that Player 1 has a serious advantage in the game. So further work could be done in figuring out what a fair compensation for Player 2 could be in order to make the game more balanced.

References:

[1] https://towardsdatascience.com/the-ancient-game-and-the-ai-d7704bea280d

[2] https://harriscenter.org/wp-content/uploads/2020/03/mancala_rules.pdf