3-2019

Padawan to Jedi: Using Reinforcement Learning to Train an Agent to Play Mancala

Tiffanie Birrell

Padawan to Jedi

Using Reinforcement Learning to Train an Agent to Play Mancala

Tiffanie Birrell
SITC
Abilene Christian University
Abilene, TX USA
trb15a@acu.edu

ABSTRACT

This paper discusses the results of using reinforcement learning to train an agent to play Mancala. I trained the agent by having it play a certain number of games against itself, and at the end of each game, I rewarded each move depending on whether it won or lost. Each move was rewarded by varying amounts based on how close to the end of the game it occurred. See game code at github.com/trb15a/mancala

1. INTRODUCTION

1.1 Mancala: The Game

Mancala is a perfect knowledge game played by two players on a board. The board is a rectangular shape with one large hole at each end (henceforth referred to as “store” or “stores”) and six holes for each player running the length of the board (henceforth referred to as “hole” or “holes”). Each hole starts with 4 marbles in it, and each store starts with 0. Play alternates between the players, with each choosing one of their holes to move the marbles from, dropping a marble in each hole, moving clockwise around the board, and skipping the opponent’s store. If the last marble ends in the player’s store, the player may take another turn. Play ends when all the holes on one side of the board are empty, and the winner is the one with the most marbles in their store. If the number of marbles in each store is equal, the game is a tie. Assuming standard intelligence, a new player should be able to understand the game and begin playing strategically within just a few games. Figure 1 shows what a representation of what the game board looks like before any moves have been made. Figure 2 shows what the game board looks like after player 1 moved from hole 3. Note how hole 3 now has 0 marbles, and each of the 4 holes clockwise of hole 3 have increased by 1 marble.

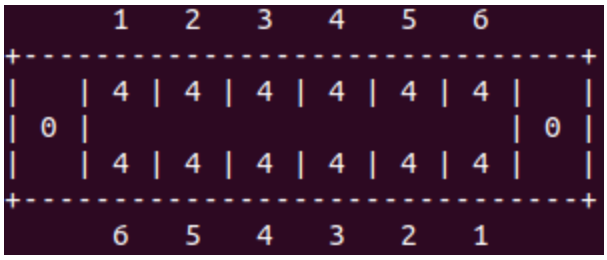

Figure 1: **Figure showing a representation of a Mancala game board**

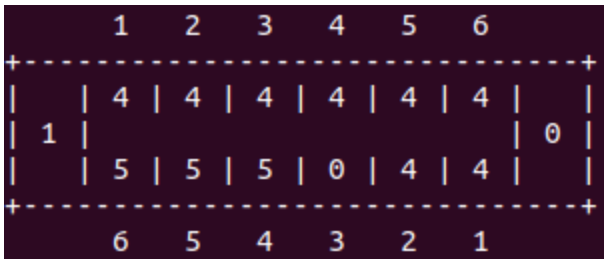

Figure 2: **Figure showing a representation of a Mancala game board after player 1 moved from hole 3.**

1.2 Expectations

Due to the turn-taking nature of the game, Mancala has a strong first player advantage, so that player 1 wins 70% of the time. I hoped to train the agent well enough to negate that advantage, and I hoped it would become good enough to beat a human player sometimes.

When I played the agent against itself, I expected the beginning games to be evenly divided close to

50-50, with a comparatively large number of ties; I expected the middle games to show a strong first player advantage close to 70-30, with a lower number of ties; and I expected the last games to even out close to 60-40, with a similar number of ties to the beginning games. I also expected that by the end of training, the difference in score would be 1 or 2 marbles. I did not think it is possible to entirely overcome the first player advantage in this basic form of the game.

2. Methodology

2.1 Training

For each turn, the agent chose which hole to move based on the probability it was a good choice based on the current state of the board. At the start of each turn, the agent saved the board state to a stack, and at the end of the game, went through each board state and adjusted the weight of the move. If the agent had won, the move weight was increased, so it became more likely to choose that move again. If the agent had lost, the move weight was decreased, so it became less likely to choose that move again. The amount that the weight is adjusted is largest for the moves at the end of the game and lessens for each preceding move. To see if it made a difference, I adjusted the weights by different amounts in different training runs.

As an example of this probability-based move-choosing method, given holes 1, 2, 3, 4, 5, 6 with 1 marble, 3 marbles, 0 marbles, 7 marbles, 2 marbles, and 0 marbles respectively, the move weight each hole is assigned might be 60%, 19%, 0%, 19%, 2%, 0% respectively. The empty holes [3, 6] have 0% chance of being chosen, the hole that would earn an extra move [1] is given the largest weight because it is the most advantageous. The holes that would earn a point [2, 4] are given a smaller chance of being chosen, because they are less advantageous than moving from hole 1, but they are still given a fairly large chance of being chosen, because they are still winning a point, which makes either of them a good move. Hole 5 does not win another turn or a point, so it is given only a tiny chance of being chosen.

Each of the following 4 training sets began with an untrained agent that chose each move randomly. They started with each available move given an equal chance of being chosen, and at the end of each game (not the end of each training session), the move probability was adjusted.

2.2 Adjusting by 70%

Initially, the adjustment was by 70%. When the agent won, the winning move was rewarded with a 70% increase; when the agent lost, the final move was decreased by 70%. As in the aforementioned example, given holes [1, 2, 3, 4, 5, 6] with 1 marble, 3 marbles, 0 marbles, 7 marbles, 2 marbles, and 0 marbles respectively, the move weight each hole is assigned might be [60%, 19%, 0%, 19%, 2%, 0%] respectively. If hole 4 is chosen, then the game ends, the score is calculated, and the agent has won, the winning move is adjusted. The weight hole 4 is assigned is increased by 70% from 19% to 32%. The other holes must be decreased accordingly, so the move weight might look like [54%, 13%, 0%, 32%, 1%, 0%]. Now, if that exact same board state is encountered again, there is a greater chance hole 4 is chosen again, since it seems to be a great choice. It is of course true that either hole 1 or hole 2 would have been an equally good choice, and there is still a chance one of those will be chosen, which could also lead to a win, in which case, the weight assigned to that hole will be increased.

Suppose in this scenario, hole 4 was chosen, then the game ends, the score is calculated, and the agent lost. Then the weight hole 4 is assigned is decreased by 70%, and the other holes are adjusted accordingly, so the move weights might look like [65%, 24% 0%, 5%, 6%, 0%]. Now, if that exact same board state is encountered again, there is a greater chance hole 4 is not chosen again, and a greater chance a better choice is made.

So far, this discussion has only covered what happens to the final move made by the agent and not any of the preceding moves. It would not make sense for every single move in the entire game to be adjusted by the same amount, because earlier moves do not affect the outcome of the game as much as later moves. Therefore, each move was adjusted by 70% of the adjustment amount of the following move. The final move was adjusted by 70%; the second-to-last move was adjusted by 70%nof 70%, which is 49%; the third-to-last move was adjusted by 70% of 49%, which is 34%, and so on until the adjustment amount was less than 1%.

2.3 Adjusting by 50%

In addition to training the agent in the aforementioned way, I also trained it with an adjustment amount of 50%. I was interested to see whether adjusting by a lesser degree led to better or

faster training of the agent. This gives less of a reward to good moves, but it also leaves a greater chance of the agent exploring alternate moves.

Given an ending move with a 40% chance of being chosen, a win results in the move weight being raised 60%, and a loss results in the move weight being lowered to 20%. Given a second-to-last move with an 50% chance of being chosen, a win results in the move weight being raised to 62%, and a loss results in the move weight being lowered to 38%. And so on.

2.4 Adjusting by 90%-50%

In the two previous training methods, wins and losses are adjusted by the same amount. I wanted to see what would happen if wins and losses were not adjusted by the same amount, so I adjusted wins by 90% and losses by 50%. If a move resulted in the agent winning the game, the move weight of the final move was increased by 90%, the move weight of the second-to-last move was increased by 81%, the move weight of the third-to-last move was increased by 73%, and so on. If a move resulted in the agent losing the game, the move weight of the final move was decreased by 50%, the move weight of the second-to-last move was decreased by 25%, the move weight of the third-to-last move was decreased by 12%, and so on.

2.5 Adjusting by 50%-90%

The final method I used to train an agent is the reverse of the previous. The probabilities of winning moves being chosen again were increased by 50%, and the probabilities of losing moves being chosen again were decreased by 90%. I did the reverse of the previous method, rather than some other ratio, in order to have a direct comparison for my results.

3. Results

I trained each of the 4 methods on 150,500 games, and I trained using the data from both the winning agent and the losing agent of each game. I collected data on how many times player 1 won, how many times player 2 won, how many games were a tie, and the average difference in final score at each stage in training. For this last statistic, each of the four methods converged after 1000 games, after which the average difference in final score was, without exception, 3 points. I was expecting it to be 1 or 2 points, and I was not expecting it to converge so tightly and uniformly across the board.

3.1 Results of Adjusting by 70%

I expected a strong player 1 advantage with a high number of tie games. In the first 30 games, player 1 won 18 times, player 2 won 12 times, and there were 0 ties. Player 1 won 66% of the games. In the first 50 games, player 1 won 26 times, player 2 won 23 times, and there was 1 tie. The score of 26-23 is much closer than 18-12. I was not expecting player 2 to catch up to player 1 so quickly. The score after 100 games was even more closely matched: 48-47 with 5 tie games. I expected player 1 to hold the advantage over player 2 for much longer, not for player 2 to become equally matched so quickly.

After player 2's record catches up to player 1's record after 100 games, the two players stay at the same level for the rest of the training games. The following 3 figures show a sampling throughout the training games to show the ratio of player 1 wins to player 2 wins. It is interesting that after 100 training games, the ratio of ties to player 1 and player 2 wins remains approximately the same.

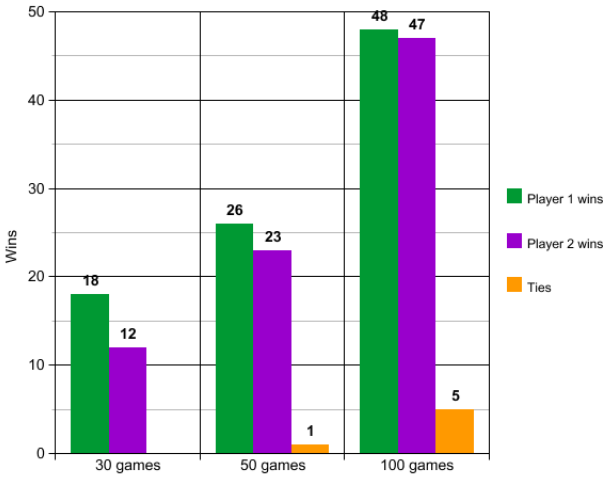

Figure 3: **Figure showing comparison of the win records for player 1 and player 2 and the tie games, for each of 30 games, 50 games, and 100 games.**

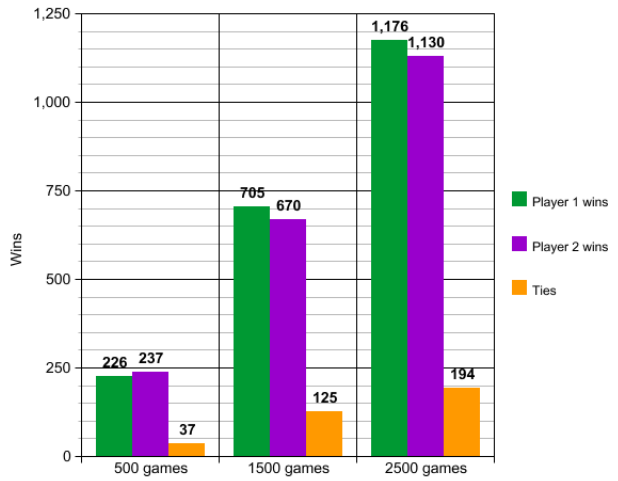

Figure 4: **Figure showing comparison of the win records for player 1 and player 2 and the tie games, for each of 500 games, 1500 games, and 2500 games.**

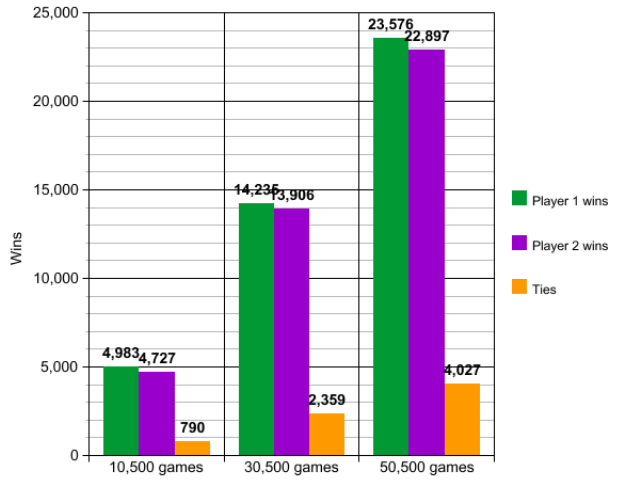

Figure 5: **Figure showing comparison of the win records for player 1 and player 2 and the tie games, for each of 10,500 games, 30,500 games, and 50,500 games.**

3.2 Results of Adjusting by 50%

Initially, I held the same expectations for the results from this testing. I expected player 1 to win close to 70% of the time, and I expected a high number of tie games. After looking at the previous set of results, I revised my expectations and thought the results of this testing would closely resemble the previous set, except they would take longer to reach the same point, or that maybe it would never even reach that point.

Instead, after 30 games, player 1 was only 2 games ahead, after 50 games, player 1 was actually behind player 2, and after 100 games, player 1 was again only 2 games ahead. There were also a higher number of tie games than in the first set of testing. After 1000 games, player 1 still remained barely ahead of player 2. There was not a significant gap until 2000 games when player 1 was 75 games ahead. After 5000 games, the two players remained fairly evenly matched, although there remained a higher number of tie games than expected. Figures 6, 7, and 8 show how the score stands at various points throughout the training.

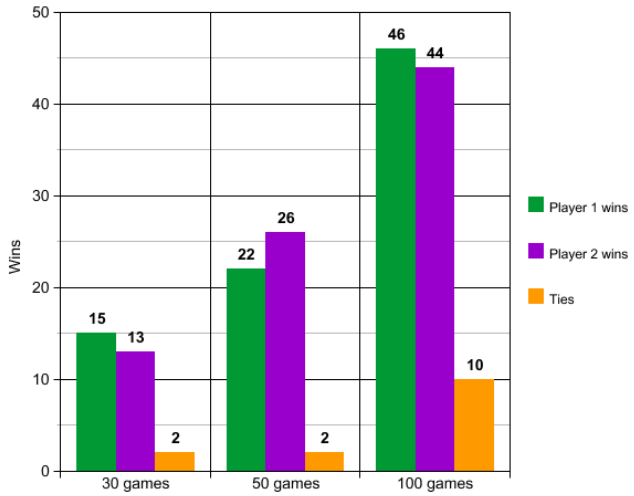

Figure 6: **Figure showing comparison of the win records for player 1 and player 2 and the tie games, for each of 30 games, 50 games, and 100 games.**

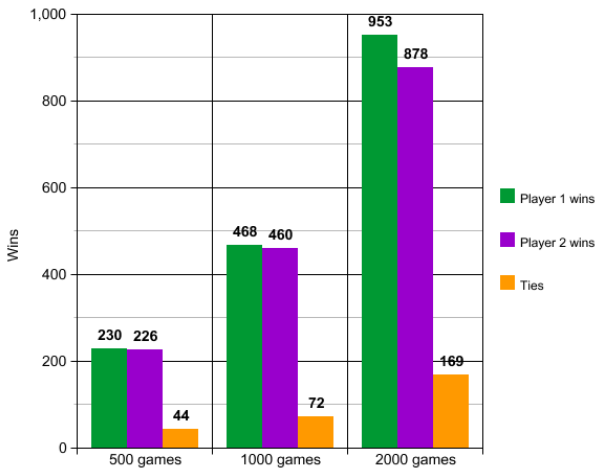

Figure 7: **Figure showing comparison of the win records for player 1 and player 2 and the tie games, for each of 500 games, 1000 games, and 2000 games.**

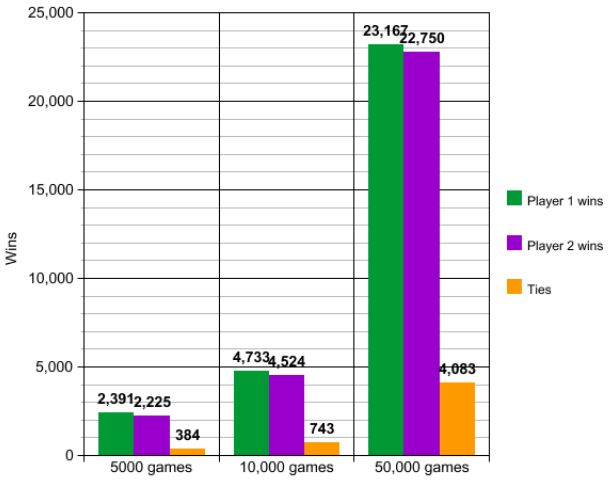

Figure 8: **Figure showing comparison of the win records for player 1 and player 2 and the tie games, for each of 5000 games, 10,000 games, and 50,000 games.**

3.3 Results of Adjusting by 90%-50%

Adjusting by 70% initially had the win ratio I was expecting, but player 2 closed the gap quickly, and adjusting by 50% actually led to player 2 playing better for a short time and being real close to player 1 the rest of the time. I was curious to see what effect adjusting by 90% for wins and 50% for losses had on the results.

Surprisingly, this set of results was the closest so far to what I had initially expected from testing. Not only did player 2 never win more games than player 1, but also it took player 2 longer to catch up to player 1 than in the previous testing results. There were also fewer tied games for the first 1000 games. Figures 9, 10, and 11 show how the score stands at various points during the training.

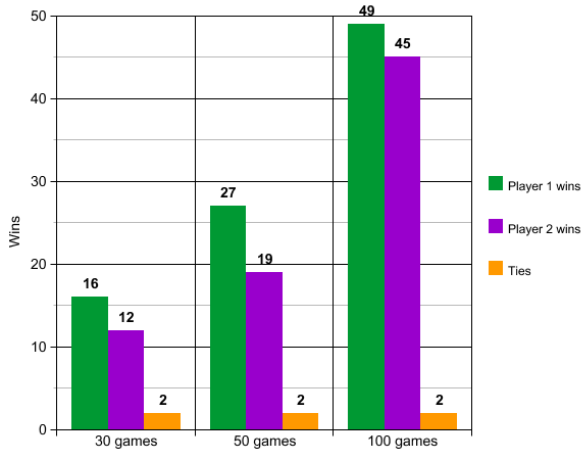

Figure 9: **Figure showing comparison of the win records for player 1 and player 2 and the tie games, for each of 30 games, 50 games, and 100 games.**

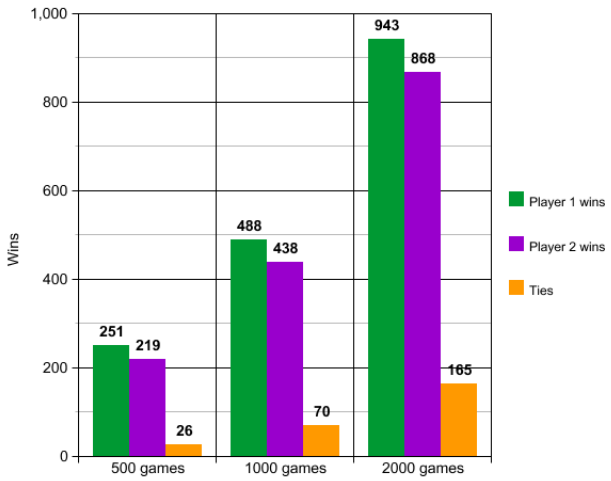

Figure 10: **Figure showing comparison of the win records for player 1 and player 2 and the tie games, for each of 500 games, 1000 games, and 2000 games.**

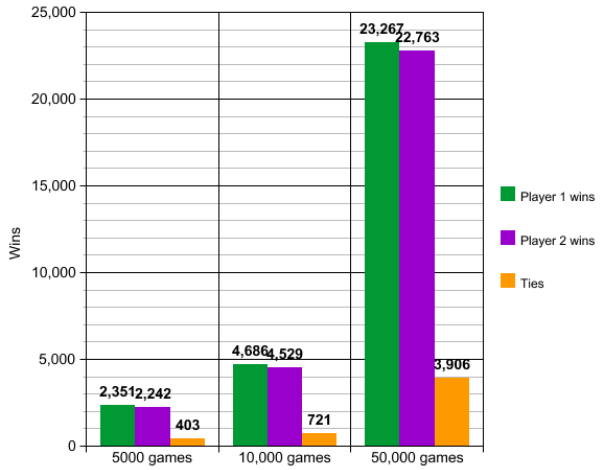

Figure 11: **Figure showing comparison of the win records for player 1 and player 2 and the tie games, for each of 5000 games, 10,000 games, and 50,000 games.**

3.4 Results of Adjusting by 50%-90%

After seeing the results for adjusting wins by 90% and losses by 50%, I expected the results for adjusting wins by 50% and losses by 90% to be nearly identical. Instead, the results were contrary to anything I expected from any of my tests. Player 2 actually won the majority of games until 5000 training games, and was even then still closely matched to player 1 until 50,000 training games. At 30 games, 50 games, and 100 games, the ratio of player 2 wins to player 1 wins is close to the ratio I expected, except I was expecting player 1 to win a larger number of games than player 2. Figures 12, 13, and 14 show how the score stands at various points during the training.

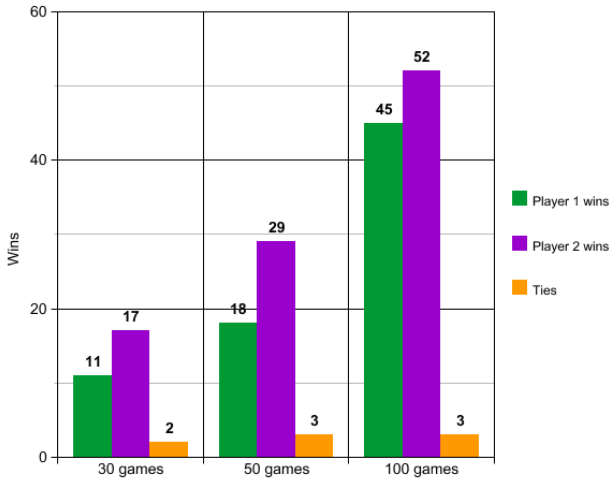

Figure 12: **Figure showing comparison of the win records for player 1 and player 2 and the tie games, for each of 30 games, 50 games, and 100 games.**

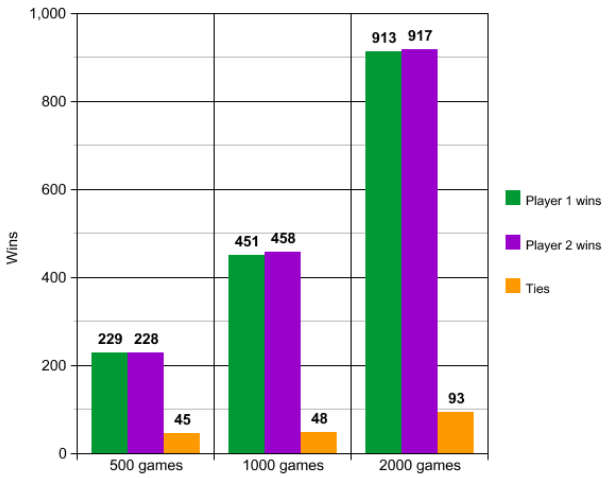

Figure 13: **Figure showing comparison of the win records for player 1 and player 2 and the tie games, for each of 500 games, 1000 games, and 2000 games.**

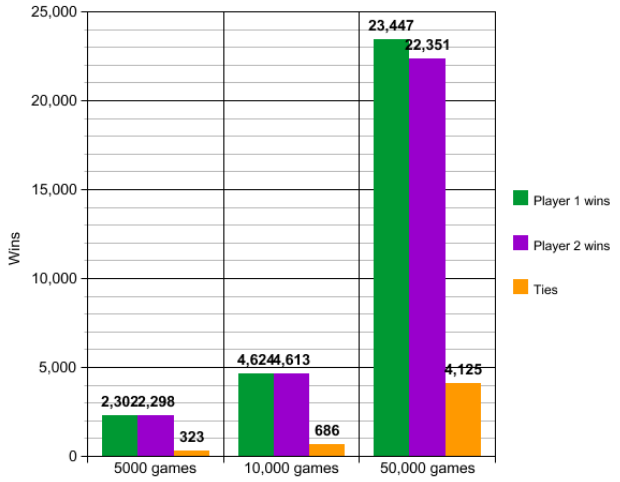

Figure 14: **Figure showing comparison of the win records for player 1 and player 2 and the tie games, for each of 5000 games, 10,000 games, and 50,000 games.**

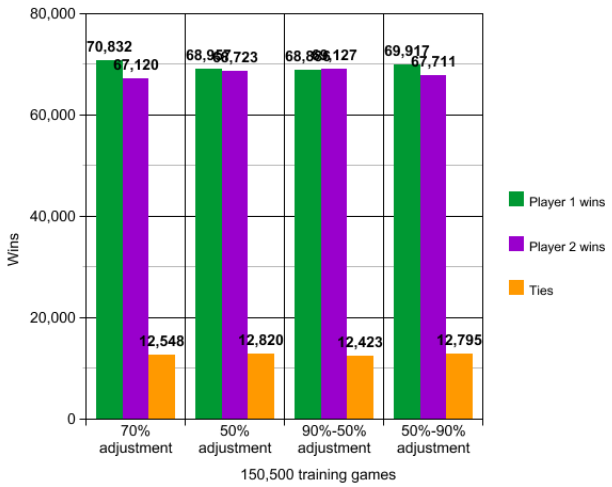

Figure 15: **Figure showing comparison of the win records for player 1 and player 2 and the tie games, for each of the 4 methods used to train an agent.**

3.5 Summary of Results

Although the results for each measured point in the training varied between training methods, I still expected the training to converge by the end. After running 150,500 training games, the win records had mostly converged, although not as much as I had expected. The 50% adjustment training and the 90%-50% adjustment training ended up with very close win records for player 1 and player 2, which is as I expected, although the player 2 wins for the latter was marginally higher than the player 1 wins. Neither the 70% adjustment training nor the 50%-90% adjustment training ended up with as close of win records as expected. Interestingly, the number of tie games was very similar across the board.

Before training the agent, I played a few games against it, both in the course of testing my code and in order to create a baseline. I was able to beat it easily. After training the agent, I played some more games, to compare against the previous games and see if its gameplay had improved at all as a result of the training. I was still able to beat it easily, but it scored much better than before training. Because the agent played noticeably better after 150,500 training games than with no training, I believe with more training, the agent could improve even more. Based on the level of improvement observed, I suspect it would take at least a couple million training games before the agent could pose a threat to a human player with any decent skill.

In an interesting comparison, I ran an untrained agent for 100,000 games against an agent that always chose randomly which hole to play. The agent that was learning as the games ran never showed significant improvement over the random agent, and the number of tie games is similar to the number of tie games with two learning agents playing each other. It would have been reasonable to expect the learning agent to win at least a majority of the games, but it only won 46,216 and there were 8,138 tie games. Perhaps the unexpected result is due to the fact that the learning

agent received one-third of the training it had received when playing against another learning agent. It would need to play approximately 301,000 games before it reached the same level of skill it had reached when playing against another learning agent.

The state-space complexity for Mancala is 1.31 X 10^{13} [Irving et al, 2000]. Therefore, the apparent lack of progress against a random agent makes sense. Because the random agent would be continually creating state-spaces the learning agent had not encountered yet, it would be unable to draw on its learning to respond. For each new state-space encountered, the learning agent begins by randomly choosing a move to make, so many of the games were likely similar to two random agents playing each other, which explains the closely matched results. When playing two learning agents against each other, since they are both using the same training to learn, the state-spaces explored are less random and fewer edge states are explored, so training progresses more quickly. Although it would take a massive number of games to train an agent fully, the agent was still able to learn and noticeably improve its gameplay after only 150,500 training games.

4. Conclusion

The conclusion I can draw from my research is that it is possible to train an agent to play Mancala, and it is possible to overcome player 1’s starting advantage. Although teaching it to play well enough to beat a human will take more training games than I was able to do, the results so far are promising.

5. References

[1] Irving, Geoffrey, et al. “Solving Kalah.” *ICGA Journal*, vol. 23, no. 3, 2000, pp. 139–147., doi:10.3233/icg-2000-23303.