CIS 667 – Project Report

AI-Blackjack

https://github.com/DolorHunter/AI-Blackjack

Kevin Lin, Zixiang Wang

NetID: ltsungha, zwang161

Contents

[CIS 667 – Project Report 1](#_Toc122109572)

[AI-Blackjack 1](#_Toc122109573)

[Introduction 3](#_Toc122109574)

[Domain 4](#_Toc122109575)

[Tree AI description 5](#_Toc122109576)

[Neural Network 6](#_Toc122109577)

[Tree-based AI experiment 7](#_Toc122109578)

[Neural Network experimental 10](#_Toc122109579)

[Conclusion 11](#_Toc122109580)

[Bibliography 12](#_Toc122109581)

# Introduction

In this project, we set out to develop an artificial intelligence (AI) player for the popular card game Blackjack. Blackjack is a strategic game that requires players to make complex decisions based on the cards they hold and the actions of their opponents. To tackle this challenge, we employed a combination of Monte Carlo Tree Search (MCTS) and Convolutional Neural Network (CNN) techniques.

MCTS is a well-known tree search algorithm that uses random sampling to evaluate and select the best move at each decision point in the game. It has been successfully applied to a wide range of strategic games, including chess, Go, and poker. In our project, we used MCTS to guide the decision-making process of our AI player and help it choose the most advantageous moves.

In addition to MCTS, we also employed a CNN, which is a type of machine learning model that is particularly effective at processing and analyzing images. CNNs have been widely used in tasks such as image classification, object detection, and face recognition, and we leveraged their ability to learn and recognize patterns to improve the performance of our AI player.

Through a series of experiments, we were able to collect a range of data and draw several important figures that illustrate the performance of our AI player. These included win rate, node count, and final score figures, as well as train error/test error figures if applicable. Our results showed that the combination of MCTS and CNN was able to effectively navigate the complex decision-making process involved in Blackjack and achieve a high level of performance. This demonstrates the potential of these techniques for developing intelligent agents that can successfully play and even outperform humans in strategic games.

# Domain

Blackjack is a card game played with a standard 52-card deck. The goal of the game is to have a hand value that is greater than the dealer's hand value without going over 21, which is known as "busting." In this version of the game, there are two players: the dealer and the player. The blasting point is set at 21 by default, but it can be modified as we want.

The state of each turn is a tuple of poker, dealer and player. Poker is class instantiation of Poker class, dealer and player are instantiation of Player class. In Poker class, there are function and attributes like cards, shuffle, turn and next card. And in Player class, there are function and attributes like points, cards on hand, stop sign, alive sign and action.

At each step, there are two action for dealer and player, which is “hit” and “stop”. Hit means player want to take another card in hand, stop means player decided not take any cards.

There is a blasting point, if any player get more points than the blasting point, he lost the game. If no one exceed blasting point, the game continue, until both players choose stop action, and the player with more points in their hand win the game. If the dealer win the game, the final score is 1, and if the player win the game, score is -1, if tie score is 0.

Text

Description automatically generatedText

Description automatically generatedHere is an example about how game is played:

As the game start, it will ask about the Blast Point, default by 21. And we choose 23. Then it will ask for the strategies for Dealer and Player. We choose manual strategy and baseline strategy.

After initialized the game, Dealer get 8 and 2 in hand, and player get 3 and 8. And Dealer choose hit at first turn, getting 6 in hand and make total points to 16. Player chooses to stop. And next turn Dealer chooses to stop. Both players not exceed blasting point, so we need to comparing points, since Dealer get 16 points and Player only get 11 points, therefore Dealer win and final score is 1.

# Tree AI description

We choose MCTS as our Tree AI algorithm. MCTS is a well-known tree search algorithm that uses random sampling to evaluate and select the best move at each decision point in the game. And it is cited from [MCTS.ipynb.](https://colab.research.google.com/drive/1JuNdI_zcT35MWSY4-h_2ZgH7IBe2TRYd)

Pseudo code:

1. Starting at the root node, select a child node to expand based on the upper confidence bound of each child.
2. Expand the selected child node by adding a set of children to it.
3. Simulate a game from the current node until it is completed and return the result.
4. Backpropagate the playout result through the tree, updating the statistics (such as the number of visits and average score) for each node visited.

We simulate the game from node followed the rules, if player does not blast or stop, player has two children, hit and stop. If player is stop, it has one child as same as current state.

And the method we’re using for choosing child is UCT, we override a rollout function called rollout2, and it can pass best children we get, and we’ll use the action attribute in best children to pass action to node and determine the action for node. Also, we’ve manually prune the unwanted leaf from children, make sure every children is no redundancy. The score estimate is also modified, now it is dependent on the role of player, since we decided Dealer win is 1 and Player win is -1, so we also make sure if it’s Dealer score convergence to 1, and Player score to -1.

We’ll do 500 rollouts to get the optimized children and using the action to play. Although we won’t get best results for just 500 rollouts, but it is rather efficient way to find the best children.

My baseline AI is built with a threshold with 15, also it can be called as dealer blast. It is the value of blasting point minus threshold. This idea supported by ["Winning at Blackjack - 21 Points Card Counting Method."](https://medium.com/@macaugamblinghk/%E6%B1%BA%E5%8B%9D21%E9%BB%9E-%E7%AE%97%E7%89%8C%E6%B3%95-3d8e3d3cff14), and it has a high winning rate, sometimes even beat MCTS.

# Neural Network

Explain the neural network you used in more detail. How did you integrate it into the tree search? Was it an evaluation function for minimax? A heuristic for A\*? A child selection decision for MCTS? Used in some other way? Include any mathematical formulas that define the model, and citations for any sources you used to implement/understand those formulas. What are the trainable parameters of your model? What optimization technique did you use to train them? How did you generate training data? How did you encode the data structure for a game state as an array of numbers that the neural network could use as input? What are the different model configurations you used in your experiments (e.g., number of neurons in a neural network hidden layer, or number of layers, or learning rate)?

# Tree-based AI experiment

The size of the problem is Blasting Point, and it can be modified at the start of game or test. In the experiment, I’ve tried different blasting point like 21, 27, 34, 42, 50. We set the number of players, actions to constant. And we use one poker in the game, but it is shuffled in every game.

I generate three figures for the results of experiments, to show the Winning Rate, Node Visited Counts and Final Score distributions.

Here are the results:

Winning Rate of Manual (random) vs MCTS:

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

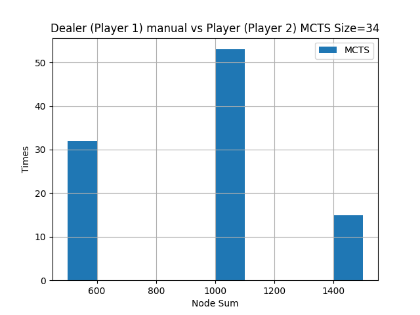
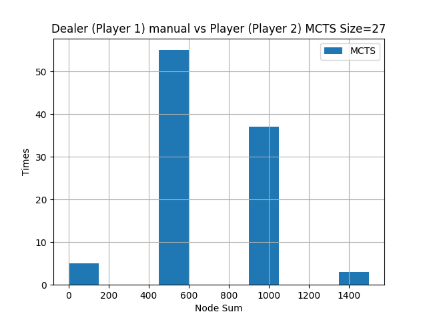
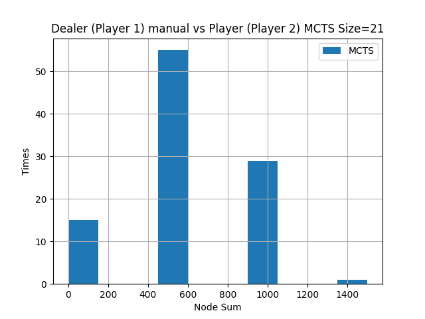
Description automatically generated

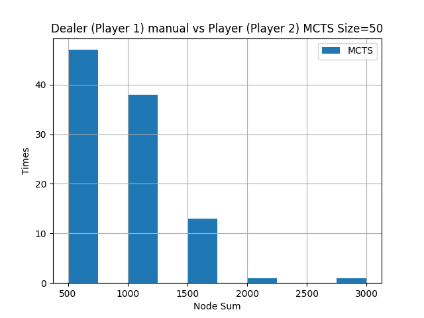
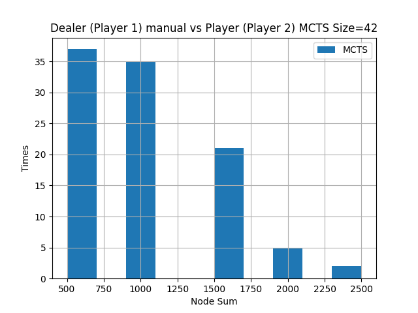
Chart, line chart

Description automatically generatedChart, line chart

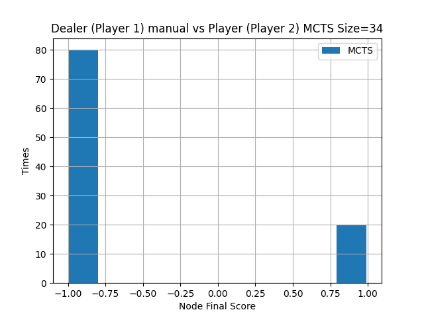
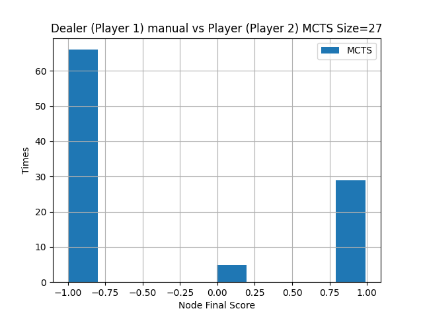
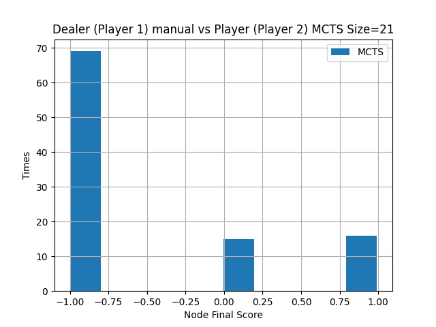
Description automatically generated

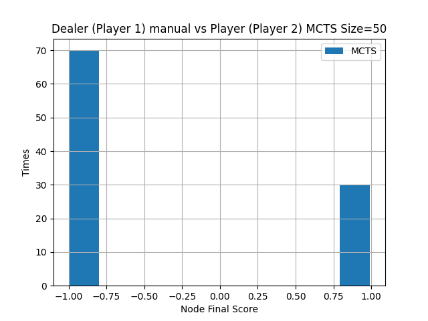
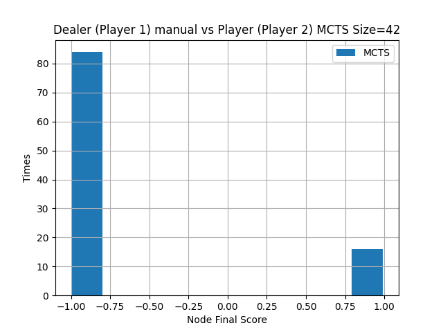
Node Counts of Manual (random) vs MCTS:





Final Score of Manual (random) vs MCTS:





Winning Rate of Baseline (threshold) vs MCTS:

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

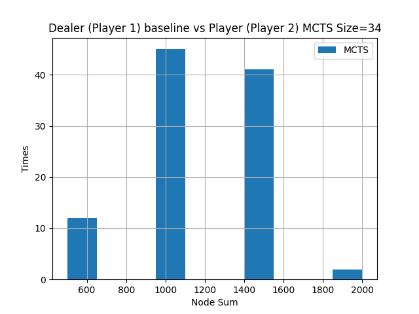
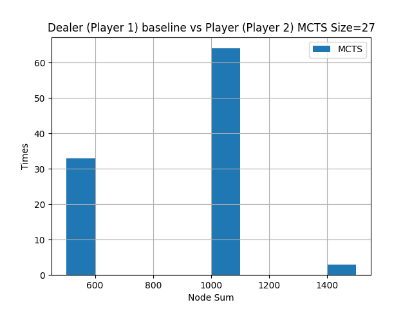
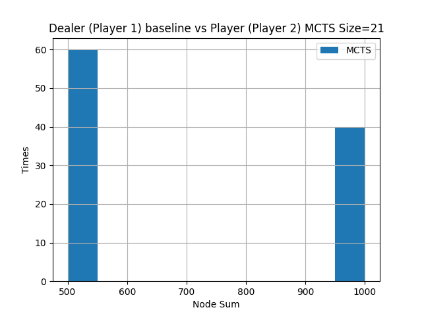
Description automatically generated

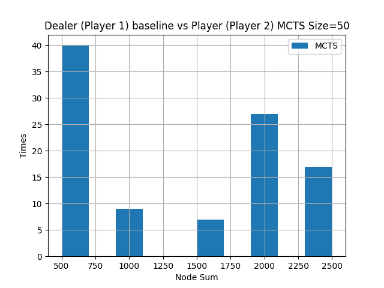
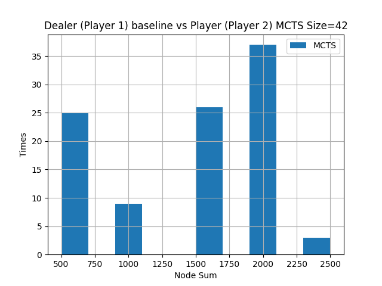
Chart, line chart, scatter chart

Description automatically generatedChart, line chart

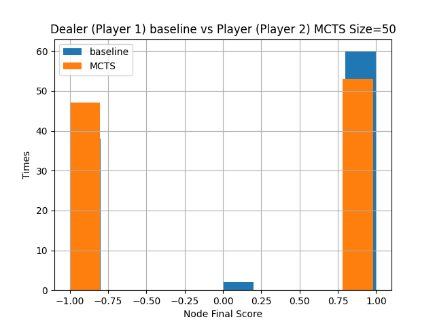
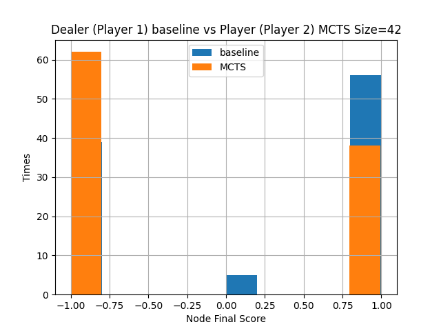
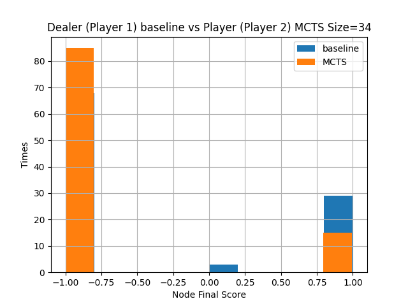
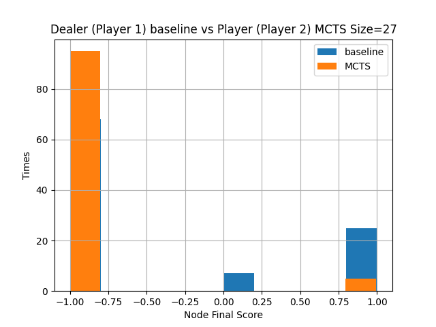
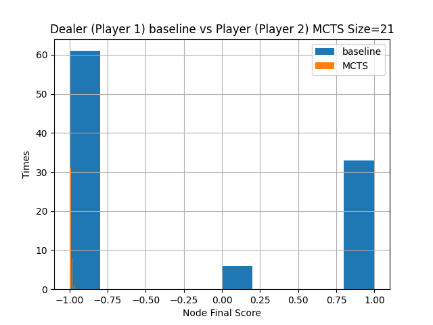
Description automatically generated

Node Counts of Baseline (threshold) vs MCTS:





Final Score of Baseline (threshold) vs MCTS:



Conclusion:

We can tell MCTS beats Manual (random) for sure, but Baseline (threshold) performance is better than expected. For Node Count, as the size increase, the upper bound of visited node increase, the distribution feature is no obvious, but it meets normal distribution in most of the cases. For Final Score, because we only have 3 cases, Dealer win as 1, Player win as -1, tie as 0, so the higher winning rate player has more win case, e.g., for Dealer it is 1.

# Neural Network experimental

Neural network experimental results: In this section you will evaluate the performance of your tree+NN AI. Each group member is responsible for training and evaluating a different configuration of the neural network. There should be one sub-section for each student in the group. For each of these sub-sections, the group member responsible should provide the following information:

▪ Which group member is responsible for this sub-section?

▪ What was the configuration of the neural network that this group member used?

▪ Plot the “learning curve” of training and testing error during gradient descent.

▪ Plot the histograms of node counts and final scores when using the trained NN in your tree search. If your domain is a two-player game, play the NN version against the baseline AI.

# Conclusion

***Conclusion: Summarize what you did. What were the most significant results? In which configuration and mode did your AI perform the best? What was the most challenging part of this project? If you were to continue working on it, what do you feel is the highest priority for future work, and why?***

# Bibliography

1. Macau Gambling. ["Winning at Blackjack - 21 Points Card Counting Method."](https://medium.com/@macaugamblinghk/%E6%B1%BA%E5%8B%9D21%E9%BB%9E-%E7%AE%97%E7%89%8C%E6%B3%95-3d8e3d3cff14) Medium.
2. Arnaud Buzzi, ["The statistics of Blackjack."](https://towardsdatascience.com/the-statistics-of-blackjack-e3b5fc29e67d) Towards Data Science.
3. Katz, Garrett. ["MCTS.ipynb."](https://colab.research.google.com/drive/1JuNdI_zcT35MWSY4-h_2ZgH7IBe2TRYd) Colab.
4. Yiu, Tony. ["Teaching A Neural Net To Play Blackjack."](https://towardsdatascience.com/teaching-a-neural-net-to-play-blackjack-8ec5f39809e2) Towards Data Science.