Blackjack Simulation

Anthony Rautmann

Mathematical Modeling and Analysis

12-16-2021

A picture containing text

Description automatically generated

**Introduction**

Last year at around this time, I traveled to Las Vegas for my twenty first birthday celebration. Previously, I had learned that Blackjack was a table card game that had the lowest “house odds” in Vegas, meaning that a given player had the best chance to win money playing Blackjack out of all the other games in the casino. I was pretty disinterested with all of the colorful and flashing slot machines because I felt that I understood the outcome of them less than the odds of drawing a given card out of a deck, so I invested a lot of my time sitting at a blackjack table. When I first started to play the game, I really didn’t have a strategy besides making sure that my hand added to 17 or higher so that I could beat the dealer. I knew that I could find out how to play a more intricate strategy than just asking for another card (called Hitting) if my hand was lower than 17. After some research, I learned about various card counting methods that some people were using to increase their knowledge of what was left in the deck as play went on. Essentially, players who use all of these different card counting methods are keeping track of what is being dealt. Each time a card is dealt, they update a count in their head so that they have a general idea of whether more high cards (ten through ace) or low cards (two through six) are left. The count starts from 0. Generally, high cards will subtract from the count and low cards will add to the count. They can then raise or lower their bet based off of how positive or how negative the count is. If the count is high and positive, that means there are more high cards left in the deck. The odds of winning a hand decrease if there are more high cards in the deck because the dealer will bust less often and make 21 more often. If the count is low and negative, that means there are more low cards left in the deck. The odds of winning a hand increase if there are more low cards left in the deck because the dealer will bust more often.

I will implement a simulation model that compares four card counting methods: Hi-Low, the Omega II System, Wong Halves, and Victor Advanced Point. The simulation could be a metaphor for a blackjack player that has perfect information and makes absolutely no mistakes in their card counting. The simulation will be using the same Blackjack ruleset as the casino I visited in Las Vegas. An eight-deck shoe will be used, meaning eight decks of cards are shuffled together and played from. Once 75 percent of the shoe is gone, a reshuffle will happen and another “round” will start with the reshuffled shoe. The minimum bet per hand is set to 20 dollars, so bets in the simulation won’t be lower than 20 dollars. The model will use the *same set of deals* to play hands of Blackjack using all four card counting methods so that results can be gathered and card counting methods can be compared usefully. After obtaining data on each card counting method and displaying some charts, I will answer the question: “*Which of the four card counting methods would I like to try and play in a casino*?”

**Model Overview**

In a typical game of Blackjack, one of the dealer’s cards are face up on the table while they deal each player two face up cards. A player can then decide whether they want to hit or not, weighing their choice partly off of what card the dealer has. I will implement a “policy” into my code, from which the computer player will make its decision to hit or stay (see *Figure 1*).

Text

Description automatically generated

*Figure 1.* “Policy” matrix, generated with Python

The numbered rows of the policy represent the total of a player’s hand, and the numbered columns represent the dealer’s face-up card. ‘H’ or ‘S’ correspond to ‘Hitting’ or ‘Staying.’ Each time the computer player must decide on whether to hit, it looks up an element in the policy matrix to decide based on the face up card totals.

Simulation starts by creating a randomized shoe of 52(8) = 416 cards. Then, hands are played until 75 percent of the shoe is gone. Before each hand is dealt, four bet values are generated based on what the given count of a card counting method is. Three cards are dealt. One to the dealer, and two to the player. Any time a card is dealt throughout the whole simulation, the card counting method counts are all updated based on the value of the dealt card. Then, the player chooses to hit or stay based off of the policy matrix in *Figure 1*. If the player hits and makes a score of 21 or busts, net winnings are updated, and a new hand starts. If no 21 was made or no bust occurred, the dealer will continuously hit until the value of their hand is 17 or higher. The dealer could make 21, bust, or stay at a value greater than 17. In any scenario, the net winnings will update, and a new hand will start. This process is repeated for 12 shoes. Data is stored for net winnings and bet values for each hand.

At the end of the simulation, we will have data for at least 58 hands per shoe (number of hands vary per shoe). Two separate data tables for each card counting method is created. One for the bet values, and one for net winnings (depicted in *Figure 2*).

Table

Description automatically generated with medium confidenceTable

Description automatically generated with medium confidence

*Figure 2*. Data frames for the High Low card counting method, generated with Python.

There are 65 columns and 12 rows stored in each table. The columns represent each hand, and the rows represent each shoe that was played. I chose to run the simulation with 12 shoes because after sitting at a Blackjack table for 12 straight shoe reshuffles (700 hands), it’s probably time for a break. We also have data for the number of cards left in the shoe in a data frame similar to the winning and betting data frames. Averages of each column were taken in all of the data frames. This leads to being able to graph average net winnings for an average number of cards that were dealt for a given hand. Also, it leads to a graph of the average bet values for an average number of cards that were dealt for a given hand.

**Results**

Chart, line chart

Description automatically generated

*Figure 3*. Line graph depicting winnings, generated with Python

In *Figure 3*, we can see the net winnings of the computer player for one shoe based on 12 averaged shoes. The methods start out pretty similar and produce relatively the same winnings. After more cards are dealt out of the shoe, that’s when the bet amounts for each method really start to vary. The 12 simulated shoes generated rather unlucky results (the dealer was winning more than the player), so the method that minimized losses was the Omega II System. If the player was winning more often, the graph would show that the method Victor Advanced Point maximized winnings. It is also interesting to note that all methods ended the averaged shoe with relatively the same amount of loss. The graph suggests that there are two methods that stand out: The Omega II System (minimized losses), and the Victor Advanced Point (maximized winnings). A graph of the bet values will help us further analyze the methods to determine which is better (see *Figure 4*).

Chart, line chart

Description automatically generated

*Figure 4*. Line graph showing bet values, generated with Python

In Figure 4, we can see the bet values of each card counting method for one shoe based on 12 average shoes. We can see that the Omega II System has the lowest bet values for hands, and the Victor Advanced Point method has the highest bet values for hands. This means that if a hand is won by the player playing with Victor Advanced Point method, the winnings will be significantly higher than the other methods. If a hand is lost, however, the losings will be significantly higher as well. This suggests that a Blackjack player that likes to take a lot of risk should play with the Victor Advanced Point method. A player that wants to minimize loss, and therefore minimize risk, should play with the Omega II System method.

To answer my initial question of “*Which of the four card counting methods would I like to try and play in a casino*,” I would say that each method has its own usage. I would choose the High-Low method if I was going to try out card counting. The High-Low method takes on some risk because it does not minimize losses, but also does not maximize winnings (and therefore maximize losings). As a side note, all of the card counting methods are extremely hard to play inside of a casino. There are so many decisions to make at a Blackjack table and adding more decisions to that with a card counting method makes the game more difficult. High Low is probably the counting method that is easiest for a person to implement inside of their own head, which is another reason why I would like to try it out.

**Discussion**

To conclude, I successfully produced and gathered data on a simulation of the game Blackjack. Paying attention to four different card counting methods, I discerned which method was most likely for me to actually try and play in a casino. In the future, some interesting aspects could be added into the simulation to make it even more realistic. Exploring the idea of splitting pairs in Blackjack and adding that aspect into the simulation would increase the winnings of the computer player. For example, if the computer is dealt two aces, they could split them into two separate hands and place another bet (which they would want to do because the odds of getting 21 when first being dealt an ace is high). Also, some model equation could be fit in *Figure 4*. That would allow me to be able to predict the bet value of each method based on what the number of cards dealt out of the shoe was. Over the course of this project, I learned that High Low is the best card counting method for me to try out in a casino.

**References**

Book – inspiration

Giordano, F. R., Fox, W. P., & Horton, S. B. (2014). *A first course in mathematical modeling*. Brooks/Cole, Cengage Learning.

python references

*What is the best way to create a deck of cards?* Stack Overflow. (1965, March 1). Retrieved December 17, 2021, from <https://stackoverflow.com/questions/41970795/what-is-the-best-way-to-create-a-deck-of-cards>

Pictures

*20,852 blackjack cards stock photos*. 20,852 Blackjack Cards Photos - Free & Royalty-Free Stock Photos from Dreamstime. (n.d.). Retrieved December 17, 2021, from <https://www.dreamstime.com/photos-images/blackjack-cards.html>

Policy

*Blackjack strategy charts - how to play Perfect blackjack*. Blackjack Apprenticeship. (2021, April 25). Retrieved December 17, 2021, from <https://www.blackjackapprenticeship.com/blackjack-strategy-charts/>

Card counting methods

wikiHow. (2021, October 21). *How to count cards in blackjack*. wikiHow. Retrieved December 17, 2021, from <https://www.wikihow.com/Count-Cards-in-Blackjack>

**Appendix**

My Python code script is too long to past here as text. I have uploaded my executable as text on GitHub so that you can see the text version here:

<https://github.com/18arautm/18arautm/blob/main/finalProjectCode.ipynb>