Agent Playing Blackjack with Advice

Final Artificial Intelligence Project Report

CS5100

Medhavi Mahansaria, Animesh Sinha, Dhara Bhavsar

December 14, 2015

College of Computer and Information Science, Northeastern University

Boston, MA 02115

Explanation of our Game of Blackjack

The objective of this game is to draw cards from a deck of **twenty-eight** playing cards to a total value of **fifteen**, begins with a deal of two cards to a player and a dealer.

Input: We are not using the standard deck; instead, we have used cards 2 through 7 and Ace only. Each of the cards dealt, except for the first card dealt to the dealer, lie face up in full view of the player and dealer.

Initially, a player may choose from four choices of play after dealing of the cards. A player may:

1. **Stand** – The player may choose to stand that is to stay with his current hand and take no more cards from the deck; or,

2. **Hit** – The player may choose to hit which is to take a card from the deck to add to his hand in an effort to more bring the total face value of all cards closer to the ideal twenty-one; or,

3. **Double Down** - The player may choose double down, only when holding two cards, to double his bet, hit his hand with only one more card, and then immediately stand; or,

4. **Split** – The player may split a hand into two hands if the hand is a pair of cards with matching values. A player may split up to three times in a game, resulting in four hands. Splitting a hand in this way results in each of the original cards becoming the first card in a new hand, which is then dealt an additional card. The player is required to back the new hand up with an ante (bet) equal to the original ante. On splitting the aces, player must agree to take only the additional card on each hand, unless that card is another ace in which case that hand may also be further split. A player may split into as many as four hands.

Card Value: The value of a hand is the sum of the values of each card in the hand. All cards 2 through 7 are valued according to their normal value. Aces are worth either 1 or 8 depending on which makes a better hand. A hand is known as soft if it contains an ace valued at 8 (e.g. Ace and a five is a soft 13).

Game Play: Here the player plays against the dealer, and attempts to obtain a hand with a greater value than that of the dealer but less than or equal to 15, a Blackjack.

1. A player may hit as many times as they wish before standing or busting, i.e. going over 15.

2. The dealer has to follow a strict set of rules and makes no decisions. In this game, the dealer must hit any hand with a value of less than 11 and stand on any hand with a value of 11 or greater.
3. If a player is dealt blackjack he automatically wins his bet unless, the dealer is also dealt blackjack (i.e. push).
4. If the dealer is showing a seven or an ace then any player holding a blackjack can take even money or wait to see what the dealer has, if the dealer has blackjack then the game is over and the dealer wins all bets and pushes, or ties, with any player with blackjack who did not settle for even money.

Description of the Project:

Our project is a blackjack simulator involving a dealer and a player, with the user of the simulator controlling the player. The simulator has two modes:

1. **Standard** Mode - Allows the user to play against the dealer on their own.
2. **Advice** Mode - Advices the user, which next step it is advisable for them to take.

The goal of the project is to create a feature that offers the player advice as he plays blackjack hands. The advice would include hand position decisions, when to stand, hit, double down, or split, and betting advice, when to increase or decrease bets.

In preparing advice for the player, the game attempts to make the best move based on the probability of future events and the value they will return. The ending value of any hand will be worth between +1.5 and –1.0. The former represents the 3 to 2 payout on a players blackjack and the latter represents a dealer victory, which will result in the loss of the bet.

One thing that makes Blackjack unique over most games of chance, especially those involving dice, is that it allows for some limited memory. Since most houses use a multi-deck shoe, four in the case of this software, each hand is dependent on the last. Although this makes the search space much larger for calculating the expected value of a move, it can also give a player who knows the count much greater advantages than most games afford.

Counting Cards Strategy:

The most common variations of card counting in blackjack are based on statistical evidence that high cards (especially aces and 7s) benefit the player more than the dealer, while the low cards, (especially 2s, 3s, and 4s) help the dealer while hurting the player. A high concentration of aces and 7s in the deck increases the player's chances of hitting a natural Blackjack, which pays out 3:2 (unless the dealer also

has blackjack). In addition, when the shoe has a high concentration of 7s, players have a better chance of winning when doubling. Low cards benefit the dealer, since according to blackjack rules the dealer must hit stiff hands (5-6 total) while the player has the option to hit or stand. Thus a dealer holding (5-6) will bust every time if the next card drawn is a 7, making this card essential to track when card counting.

In High-Low strategy, you start with zero and ace cards are revealed, count the cards in the following way:

- +1 for ranks 2, 3, 4
- -1 for ranks 7
- 0 for ranks 5, 6

In order to know the true count we divide the count by the number of decks in the shoe. The greater the true count, the more the player should bet and be inclined to stand, double, split, take insurance, and surrender.

We updated the count of the high-low player for each card that is dealt. In order to choose how much money to bet we use a formula that uses the true count.

Q Learning Strategy:

First we mapped the game to states – each states is the dealer up card, the score of the cards in the agent hand and whether the player has an ace or not. We tested our agent according to the number of wins and compared it to other agent.

In order to teach our agent we ran 100000 games and update the agent policy according to the following formula, the reward is the amount of money earned or lost.

We saved the path from the first state to the finite state, and updated it only at the end; because the state space is huge, there are many routes from the initial state to the finite state. The first action is the most important one (as there are more possible actions to it) and gave it the biggest reward, each state after until the finite state is 1/length (path) smaller.

Every state q-value is then calculated according to the Q-learning Formula:

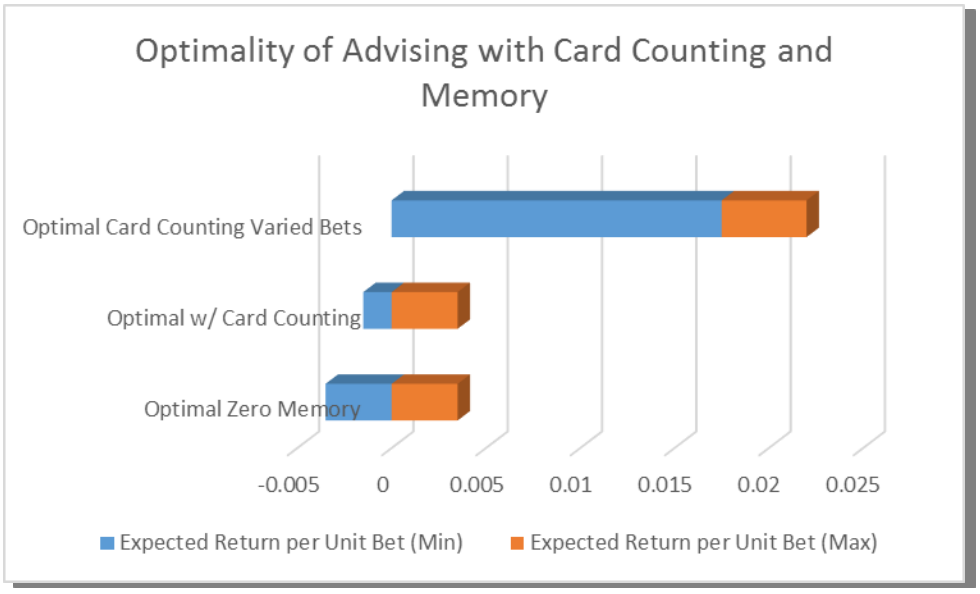
$$Q_{t+1}(s_t, a_t) = \underbrace{Q_t(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q_t(s_t, a_t)}_{\text{old value}} \right)$$

After some trial and error, we found out that the best values for Alpha is 0.9 and for gamma is 0 (no penalty).

Results:

Unlike in optimal zero-memory blackjack, decision tables that take into account the cards left in the deck are constantly changing. As a deck runs lower, knowledge of what it holds becomes very valuable when the variance is high. A deck of n cards should hold on average n / 7 of each card Ace through 7.

The expected gain using a 4-deck shoe and an optimum zero memory strategy is -.0036[1]. By using a card count to adjust decisions based on a given situation, we can improve this by about .0015. Finally, varying bets will allow an improvement to around +.020. Every deck has an expected value associated with it based on the anticipated gains from each hand that can be played from the deck. In theory, we would like to bet an infinite amount when we know the deck has a positive expected gain, and nothing when we know the deck has a negative expected gain. In reality, however, this is not possible given the social atmosphere of play and as a result, this simulation allows for a minimum bet of 10 and a maximum bet of 100.



[1] Manson, A. R., Barr, A. J., and Goodnight, J.H. *Optimum Zero-Memory Strategy and Exact Probabilities for 4-Deck Blackjack.* The American Statistician. Volume 29, Number 2, May 1975.

Situational Searches:

All searches in blackjack involve the same process of projecting the gain of hands based on the probability[2]. The differences lie in where the process of the shoe being played the search must begin. The simplest searches involve deciding whether the hit, stand, or double down after a non-matching pair of cards has been dealt to the player. The player must use what he knows is in the deck, what he currently holds in his hand and what the dealer is showing to calculate the best move. Since the player cannot know what the dealer is holding as the upside-down card, as far as he should be concerned the unknown card remains in the deck. The player may, however, know something about that card some of the time. Since a dealer, blackjack immediately ends the round it can be inferred that if the dealer is showing a seven card then the unknown card is not an ace, if he is showing an ace then the unknown card is not a seven. This will affect the probability of all other cards being the unknown card. The probability of a card being the unknown card should thus be assessed by the number of cards left in the deck divided by the total number of cards remaining less the number of aces or seven remaining instead of the number of cards left in the deck divided by the total number of cards. For example, a hand of 3 and 6 is dealt to the player and the dealer is showing a 5. The player would like to move to the position with the greatest gain possible. The value of standing is equal to +1 * the chance of the dealer beating 9 – 1* the chance of the dealer busting or standing on a hand short of 9. Each move of the dealer must be expanded out until a win, loss, or push can be evaluated for each hand. Once these have been found the evaluation function will travel back up the tree assigning value to each hand based on the probabilities of reaching each child hand multiplied by the probability of being dealt that child hand. The value of hitting the hand is equal to the value of each resulting hand multiplied by the probability of it being drawn from the deck. Again, we must expand each hand by looking at all the hitting and standing results and then choosing the more favorable gain.

Doubling down involves doubling the initial bet and drawing one more card to the hand. The evaluation of this must be handled a little differently since the bet is doubled. Doubling down will never produce a greater gain than hitting since drawing an ace or two on an eleven will most certainly cause the hitting evaluation function to draw additional cards. However, since the bet is doubled, a gain of .30 when doubling down is better than a gain of .40 by hitting. Therefore, to evaluate correctly a hand in such a circumstance, a weight equal to the factor of the bet increase should be multiplied to the gain. For all decisions among the options of standing, hitting, and doubling down, the simulation resolves these by the full evaluation of an exhaustive search.

The neural network accepts 9 inputs, one for each of the seven card counts, one for the card in the pair dealt, and one for the card the dealer is showing. The seven inputs for the cards represent (Actual Number Left / Expected Number Left) – 1 where:

---

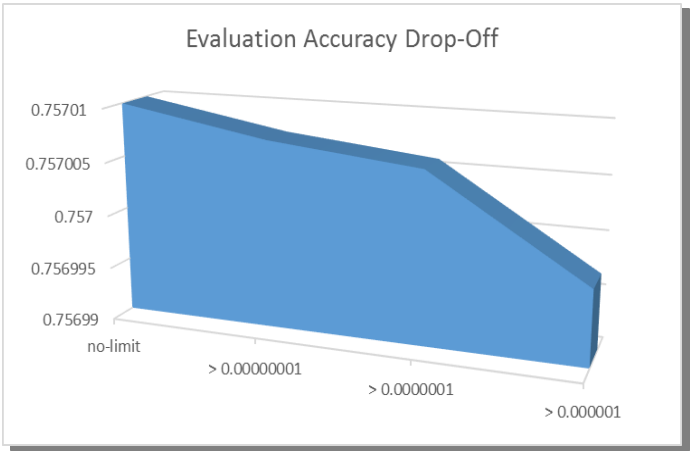
[2] Brett Hobbs, Near Optimal Card Counting Blackjack, 2002

Expected Number Left = Total Cards Left * Original Fraction of the Deck

The original fraction of the deck is 1/7 for all cards. The value of this number can be viewed as 0 means the exact number expected remain, > 0 means more than expected remain, and < 0 means fewer than expected remain.

The expected value is the (expected gain + 1) / 2, thereby reducing the range from [+1, -1] to [+1, 0]. From a base, we wrote portions of neural network code to allow for a varied number of inputs along with a varying number of hidden nodes.

Not much accuracy is lost when probabilities are constrained at early levels but the logarithmic scale eventually causes the expected larger drop-off in accuracy. The no limit evaluation took 8:04 seconds for a very complex hand; this cost is reduced to 0:54 by the greatest bounding shown in the graph.



The trade off here is a loss in accuracy. But, hopefully, this will be more than recovered by a great increase in the number of examples that can be generated for the neural network.

Overall, the predictions are 97.5 % accurate, which is a very good result.

We achieved most of my objectives in this project. We developed an artificially intelligent agent capable of earning money while playing blackjack, within the rules of the game. We were not able to account for the addition of other players to the game but this is more a function of programming than A.I. and does not disturb the fact the results in any case. The work with probability based expected outcomes and the use of neural networks in this project. Given the time, we would have liked to experiment with decision trees for deciding whether or not to split by using attributes such as is pair split mod 10 greater than dealer?, is pair together mod 10 greater than dealer?, and possibly is five?, and is ace?.