Machine Learning: Playing Blackjack

Richard Gaggioli
December 20, 2013
University of Wisconsin - Madison
ME 539
Instructor: Yu Hu

Introduction

There are two types of decisions a player makes while playing blackjack: how much to bet and whether to hit or stay. The purpose of this project was to use what I have learned in ME 539 to make decisions in the game of blackjack. In blackjack the dealer has an advantage over all of the players. Most of this advantage is a result of the dealer always playing last. Along with this, if a player busts he loses regardless of whether or not the dealer busts. For example if there are four players and two of the players and the dealer bust, the dealer pays the two who did not bust but the two players who did bust pay the dealer. Therefor on a hand that the dealer busted, he came out even. In order to gain advantage over the dealer I chose to use the k-nearest neighbor algorithm to make decisions for the players.

Works Performed

I started by making a simple blackjack implementation in Matlab where a player can bet 1, 2, or 3, and choose to hit or stay on their appropriate turn. The dealer stands on all 17's. I first attempted to gain advantage over the dealer where the players could only bet 1 each hand and the card count was not a part of the feature set. This proved to be largely unsuccessful so I tried using different features.

Table 1 shows the given features I ultimately chose. The first ten features show what percentage of the deck a given card occupies. Feature 1 is the number of aces left in the deck divided by the total number of cards left in the deck. Feature 2 shows the percentage of 2's, etc. Feature 10 shows the number of 10's, Jacks, Queens, and Kings divided by the total number of cards left in the deck. Feature 11 is the card count. For every card dealt if it is a 2-6 you add 1 to the card count. If the card is a 10, Jack, Queen, King, or Ace you subtract 1 from the card count. For the other cards the count remains the same.

The decision for how much to bet used features 1-11 and returned features 15-17. Where a 1 in feature 1 corresponds to a bet of 1, a 1 in feature 2 corresponds to a bet of 2 and a 1 in feature 17 corresponds to a bet of 3. For the hit or stay decision, features 1-14 were used. The classifier return features 18-19 where a 1 in feature 18 corresponds to the decision to hit and a 1 in the feature 19 means stay.

Table 1: Feature Set

Features 1-10	Feature 11	Feature 12	Feature 13	Feature 14	Feature 15	Feature 16	Feature 17	Feature 18	Feature 19
Percent of Card in Deck	Card Count	Unused Aces in hand	Showing Card	Players Score	Bet 1	Bet 2	Bet 3	Hit	Stay

In order to implement the kNN classifier I created a starting data set of 50 hands. Over these fifty hands there was one player who bet 1 if the card count was less than ten, bet 2 if the card count was greater or equal to ten but less than 24 and bet 3 when the card count was greater than 25. Also over these 50 hands the player stayed with a score 16 or greater.

After generating these fifty hands I simulated a blackjack table with two players. Each player placed there bet using the kNN classifier and then when their turn came they used the classifier to hit or stay. After every hand the feature vectors would be created based on the success of the decisions a given player made. These feature vectors were then added to the working data set. This created a reinforcement learning environment. If a player wins the hand all of the decisions are added to the data set. If the player loses, his bet amount is incremented down one, unless it is already the minimum. Along with this, his last decision in his hand is flipped. For example, if a player loses because he busted then the last option he had, he should have stayed. So this feature vector is relabeled to be a stay. If he stayed and lost, he should have hit. This is generally true.

Results

Several different k values were used to see which would have the best results. I found that for the betting decision it was best to use k = 5 nearest neighbors and for the hit or stay decision was best for k = 5 as well. For hit stay, k = 10 was very close however. Figure 1 shows how well the classifier worked when k = 5 for betting for both players and k =5 or 10 as labeled.

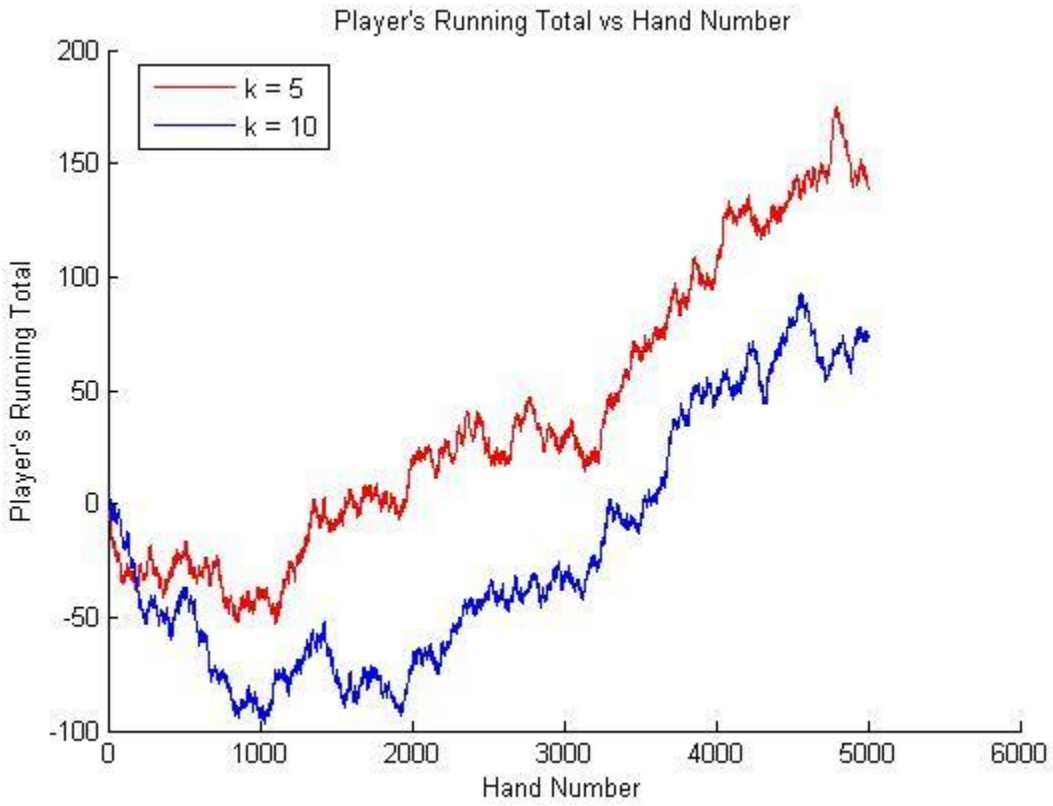

Figure 1: Player's Totals over 5000 hands. k = 5 for betting and k =5 or 10 for hit or stay as labeled.

Discussion

After my original attempt to gain advantage over the deal with a constant bet amount and without the card count as a feature, I made some vital observations. The main observation that I made is the best way to beat the dealer is by choosing the correct amount to bet. The winning or losing of any given hand is largely statistical. If the card count is high you have the best chance of getting a large score with the first two cards. Also if the dealer has to hit he is more likely to bust. By riding out the hands were the card count is low and cashing in on the hands when the count is high, a player can make the most profit.

By using a feature set generated from general strategy in blackjack as a starting point, I was able to make better and better decisions using kNN classifier in order to have consistently successful blackjack play. Using the success or failure of previous hands this program was able to make the best decision for a given state of a hand.