Markovian Blackjack Simulator

Jonathan Yu
University of Massachusetts: Lowell
JonathanYu7@gmail.com
978-866-9456

ABSTRACT

In this paper we describe how to represent games of blackjack in discrete state spaces and use that representation to teach an AI how to effectively play blackjack. The AI uses Q-learning to develop a model of the expected value of available moves in each game state and then exploits that model to determine the best move for every game situation. We will outline the methods and challenges of developing such an AI, analyze the results of our AI to determine if it is learning successfully and discuss the broader educational value of our project.

Author Keywords
Blackjack, Q-Learning, Markov, MDP, AI, Game

INTRODUCTION

Blackjack is a game of chance. Even the so-called best blackjack players in the world can only expect to win slightly less than half the time. Yet its simplicity and quick pace have made it one of the most popular forms of gambling in casinos and card rooms around the world. Pure luck influences the outcome much more than a game such as Texas Hold'em. But like all games, there is a right and wrong way to play blackjack. Our goal is to have an AI learn the "right" way to play blackjack; to the point where it's performance can rival even the best human players.

PROJECT DESCRIPTION

Mathematically Modeling Blackjack

First we must make a finite, discrete state space representation of blackjack to give our Q-learning AI something it knows how to work with. A hand in blackjack has an approximately Markovian structure. The probability of what card is drawn next is almost independent of the cards that are already in play, given a large enough dealer's shoe (typically 6 decks.) For the state space of an entire hand, we don't have to consider every single possible combination of cards that the player and dealer can have. For our purposes, all that matters is the player's total and the dealer's total and to some extent the composition of the deck (hence why card draws are *almost* Markovian).

Dealer's Hand

When constructing the state space of Blackjack, we start with the composition of the dealer's hand. We only care about the value of the dealer's face up card because we don't know about the value of the hole card until after the player finishes playing his hand. We will treat all 10s and face cards as representing the same state because they have the same value and are interchangeable for all intents and purposes. Thus the dealer's hand has a state space of size 11, 10 possible values of the face up card and the state of the dealer getting a blackjack. For simplicity, we will assume there is no option for insurance bets and that the dealer reveals blackjacks immediately.

Player's Hand

The composition of the player's hand is a little more complex. In truth, we don't care about exactly what cards make up the total of the hand, only the total value and whether the hand is soft or hard. This is the Markovian aspect of the hand. An exception to this is when the initial 2 cards dealt are a matching pair because we must then offer the player the option of splitting. Each new hand is then dealt a second card and then played separately. The total payoff from both hands can then be attributed to the decision to split. We also make an exception if the player gets a blackjack because there are no further actions to be taken. In total, the player hand has the state space comprising of 29 total states:

- Soft hands with a value between 12 and 21 (10 states)
- Hard hands with a value between 4 and 21 (18 states)
- Blackjack (1 state)

Card Counting and Deck Feature Extraction

Finally, we should account for the state of the shoe. The probability distribution of the shoe is uniform for all cards (not all values) at the beginning of the game. However shoes, large ones especially, are not reshuffled after every hand. This means observant players can learn about the

distribution of the shoe by counting the cards that have already been played. While it would be trivial for a computer to track every card played, a human player can only realistically keep track of a running count using card counting methods such as Hi-Lo. Because the state space would explode in size if each state had a unique Hi-Lo score, we will break down the counts into arbitrary levels:

- Low: Count/D is greater than 3 (where D is the number of decks in the shoe). This means that there are relatively more low cards (2-6) in the shoe.
- Neutral: Count/D is between -3 and 3 inclusive. This means the ratio of low to high cards is relatively equal.
- High: Count/D is less than -3D. This means that there relatively more high cards (10-A).

Breaking down the count into levels like this can be considered a form of feature extraction. This method makes the learning more efficient by decreasing the total state space of the problem. Card counting is more effective in games with smaller shoes. The count resets to 0 whenever the shoe is shuffled. For our simulation, the shoe will be shuffled with 1-R/N probability after each game where R is the number of cards remaining in the shoe and N is the size of the shoe.

Total State Space

Finally, there is the special terminal state where the player busts. This is its own state because the Count and the dealer face up card don't matter when calculating payoff and there are no available actions. The hand ends when the player busts, only the count is carried over to the next game. The total state space is therefore 11 dealer hands x 29 player hands x 3 shoe states = 958 possible game states. The transition model gives the option to hit, stand, or double down at any game state. There are 3 actions for each possible state which means there exists a total of 958 x 3 = 2874 state-action pairs whose expected values the Q-learning agent must keep track of.

Q-Learning Process: Exploration and Learning Factors

The learning algorithm will prefer unexplored actions in each game state initially to ensure that as many states are explored as possible in the beginning. Once the AI is in a state, it will select the option with the highest expected utility 1-ε percent of the time and a random action ε percent of the time, where ε is our exploration factor. During the learning period, ε is 1 which means actions are chosen completely at random. Once an action is selected, that state action pair is given an expected utility based on the Bellman Equation and the reward function of that action. This value is updated every time that state-action pair is encountered. With enough encounters, the value should converge to the true expected value or at least a decent approximation.

Our learning factor α determines how much importance a new reward is given when updating the expected utility of an action. After a set number of simulations, exploration (ε) will be turned off and the algorithm will attempt to play a series of simulated blackjack hands using the strategies it has learned so far. Alternatively, the program can allow a human player to play against the dealer while giving move suggestions. During exploitation, learning (α) will be turned down but not completely eliminated. Even after large numbers of simulations, it's still possible that some state-action pairs are unexplored. If such a state is encountered during exploitation, the agent can still learn and improve its internal model.

ANALYSIS OF RESULTS

Our goal for this project was to build an AI that could eventually outperform the best human players. Our algorithm, after learning 20,000 simulated hands wins about 40% of the time on average. After 20,000 hands, we started seeing diminishing returns on our win percentage. Human players can expect to win about 48% of the time if they play according to sound strategy.

There are significant differences between our version of blackjack and true blackjack that makes direct comparison difficult. There are a number of advanced rules that were not implemented in our simulations. For example, in a real game, players have the option to split when they are dealt 2 cards of the same value initially. They are also given the option to surrender or place insurance bets against the possibility of the dealer's blackjack. Because none of these options are part of our model, it's hard to say whether our AI outperforms a human player because they each play slightly different games.

However, we can compare our AI to an agent that makes moves at random. Such an agent wins only 30% of the time, which our AI clearly outperforms. This demonstrates that our AI is definitely learning; a 10 percentage point increase in win percentage in a game so heavily dictated by chance is quite impressive. Even more significant are the metrics for monetary winning. In our model, bets for each hand are normalized to $2. Over 1000 hands, a random agent will lose about $1000 on average. Our AI cuts losses by about $800, an 80% reduction.

DISCUSSION

A major aspect of blackjack that we weren't able to model was the impact of betting and how that affects both the decision making process in a hand and the expected winnings of a player. In the real world, even though a player can only expect to win 48% of the time, their net winnings can be positive if they bet intelligently.

It is in betting where the benefits of card counting are the most felt. It is the impact of card counting on a player's bets rather than his in-hand decisions that cause casinos to

put so much effort into identifying and removing card counters from their floors. In simple terms, a deck is more favorable towards the player when there are relatively more low value cards than high value cards. This is because the players receive their cards before the dealer and so can bust first. The more low cards there are to draw, the greater chance that the player will not bust and win the hand, and the more the player should bet on that hand. In tables where there are multiple players, this is also an opportunity for other players to play "behind", placing bets on a hand that you control. Conversely, when the deck is loaded with high value cards, the dealer has the advantage and savvy players would elect to bet conservatively or simply move to a table with more favorable conditions.

There are many subtleties of blackjack that our project does not accurately model. The issue with modeling them all is our simple state space approach would no longer be adequate. With every additional option and freedom we give to the agent, our state space expands in size exponentially. Eventually, even running millions of simulations will not give us an adequately large sample size to ensure that the AI learns optimal moves for most possible game states. A potential solution for this would be abandon the pure state space approach and adopt feature extractors on each state. This way, an AI can process a much larger set of possible states without having to keep track of every single possible one. We leave this as a challenge to future researchers and we hope our

CONCLUSION

Ultimately, we feel we have met our goal of developing a blackjack playing algorithm. It's far from perfect and doesn't encompass even all the standard blackjack rules. However, it is a worthwhile demonstration of the effectiveness of Q-Learning and MDP algorithms on solving stochastic games such as this. Personally, I found the project entertaining, challenging, and extremely educational. Our hope is that someone will be able to use and build upon our work to design even more effective learning AIs that can solve problems far more difficult and practically useful than blackjack.

ACKNOWLEDGMENT

The work described in this paper was conducted as part of a Fall 2012 Artificial Intelligence course, taught in the Computer Science department of the University of Massachusetts Lowell by Prof. Fred Martin.

REFERENCES

1. Higher media. (2011). *Blackjack Rules*. Retrieved December 18, 2012, from Hit or Stand: http://www.hitorstand.net/strategy.php

2. Russell, S., & Norvig, P. (2010). *Artificial Intelligence: a Modern Approach Third Edition.* Upper Saddle River: Pearson Education, Inc.

3. Tamburin, h. (2003). *Numbers Don't Lie.* Retrieved December 18, 2012, from Golden Touch Craps: http://www.goldentouchcraps.com/tamburin_005.shtml

4. Wakin, M. B., & Rozell, C. J. (n.d.). *A Markov Chain Analysis of Blackjack Strategy.* Houston: Department of Electrical and Computer Engineering, Rice University.