Evaluating and Learning Blackjack Strategies with Monte Carlo Methods

1st Lennard Rose
Faculty of Computer Science and Business Information Systems
University of Applied Sciences Wuerzburg-Schweinfurt
Wuerzburg, Germany
5122737

***Abstract*—Blackjack is a card game played all around the world. With its simple rules that can be extended in many different ways, it has always attracted the attention of game theorists, mathematicians and computer scientist to explore its statistical properties. Based on these properties many came up with ideas how to break the advantage of the dealer. The approaches in this paper tried to achieve this using Reinforcementlearning methods. In the course of this paper it could be shown that Monte Carlo Methods are capable of approximating strategies that are on par with those of game theorists.**

I. Introduction

In the casinos of the world, Black Jack is considered the most played card gambling game [1]. The goal of the player is to draw cards until their sum is as close to 21 as possible. If he draws beyond that value, he goes bust and loses the game. If the player decided to "stand", that means stop drawing cards, his hands value is compared to that of the dealer who draws cards based on a fixed strategy. The one closer to 21 wins. For further explanations see [1].
The statistical nature of the game with the dealer acting according to a fixed and predictable policy suggests why it is particularly appealing to theorists. If the game was played with cards face-side up, every player could easily calculate his own chances to win. Without this, every bit of information has to be well considered to estimate the chances of winning. This led to vast mathematical systems suggesting how to use the little knowledge that can be gained during the game to beat the advantage of the dealer.
The purpose of this paper is to learn this mathematical systems by a computer using Monte Carlo methods from [2]. Monte Carlo Methods always use some kind of random-factor and are often found in "popular-science articles" to bring attention to the topic of statistical learning. A video[1] like this, seen at the time of the pandemic, was also the motivation for the author to choose these particular methods.

II. Theoretical Foundations

A. Blackjack

For this paper, a blackjack game was implemented to use as the environment. The environment was designed to be adaptable for different game rules and strategies to learned by the algorithm. For details see the provided Implementation. The following subsections contain information about different strategies and rule variations from [1] that have been used.

1) Basic Strategy: The basic strategy has the goal to minimize the advantage of the dealer by using the tables in figure 1 for the players decisions. All cells above the black ones suggests the player to stand. The player always stands on hand-values above 19.

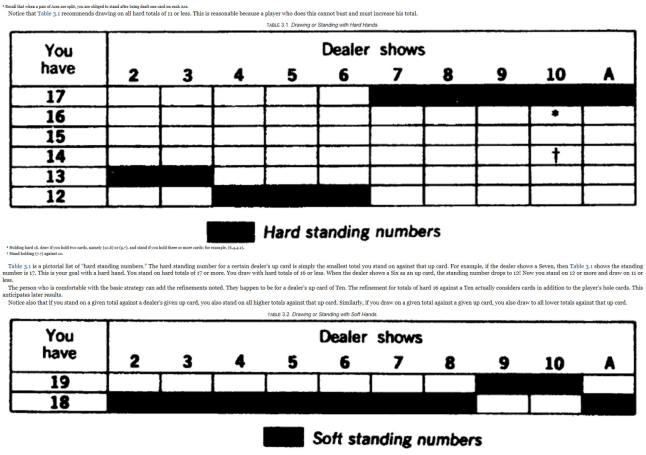

Fig. 1. Table for the basic strategy from [1]

A *hard hand* means the player holds no ace. Holding a ace is referenced as a *soft hand*.

2) Simple Point-Count System: The simple point-count system is easily applicable for players. The player only has to remeber one value he adds to or subtracts from, based on the cards he sees. For every card-value above 9 (including the ace), the player subtracts one of his count, for every value below 7, he adds one. The resulting value is used to adjust the placed bet. This is based on the assumption, that small values favor the dealer, which lead to a more cautious style of play being advised.

3) Complete Point-Count System: The complete point-count system needs an additional value to be remembered by the player compared to the simple point-count system. The

[1] https://www.youtube.com/watch?v=7ESK5SaP-bc

player now has to remember how many unseen cards still exist in the game. For this, he starts with 52 times the used amount of packs and subtracts one every time he sees a cards face-side, even if it was by accident. The point count that is also used in the simple version, is then divided by the unseen card count. Based on this ratio, also called the High-Low Index, the tables 2 and 3 are used for the decision-making.

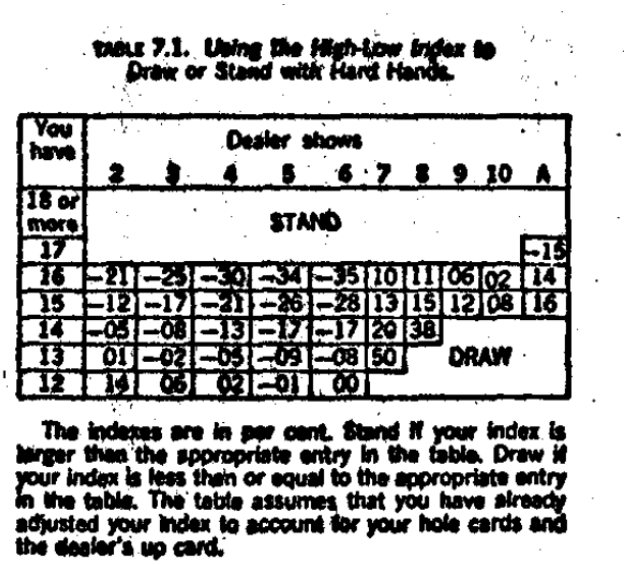

Fig. 2. Table for the Complete Count-Point System with a hard hand from [1]

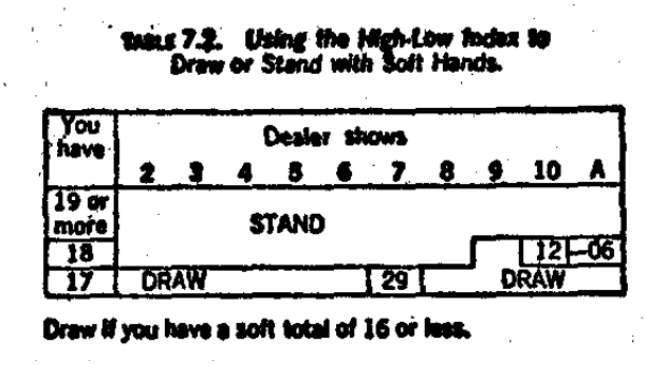

Fig. 3. Table for the Complete Count-Point System with a soft hand from [1]

B. Monte Carlo Methods

Monte Carlo methods are suitable learning algorithms for estimating value functions. Contrary to other methods, MC methods don´t need complete knowledge of the environment. They learn by sample sequences of state, actions and rewards also referred more general as ”experience”. This experience is learned in episodes to ensure its integrity and validity.

1) On-Policy Monte Carlo Prediction: Monte Carlo Prediction is a way to learn state-value functions of a policy [2]. For this, an episode of actions is performed, each corresponding to the policy with respect to the current states. The discounted returns for each state are then averaged to eventually converge to the policies state value.

There are two main ways of On-Policy Monte Carlo Prediction, First-visit MC Prediction and Every-visit MC Prediction. The former only takes account of the first visit of a state, if it appears again in the episode it is ignored and does not contribute to the state-value. The latter calculates the state-values with respect to all visits of a state. For the Blackjack task used in this paper, both methods have the same result. This is due to the fact that you cannot put back cards and every draw increases your hand value (leading to a new unseen state). Furthermore, only the last state has a reward. Figure 4 shows First-visit MC prediction, Every-visit MC prediction is accomplished by deleting the code-line that says ”Unless S_t appears in $S_0, S_1, ..., S_{t-1}$: and always append G to returns(S_t).

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated
Initialize:
  $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$
  $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):
  Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:
      Append G to $Returns(S_t)$
      $V(S_t) \leftarrow$ average($Returns(S_t)$)

Fig. 4. Pseudocode for the First-visit MC prediction from [2]

2) Monte Carlo Exploring Starts: Contrary to the first method in this section, Monte Carlo with Exploring Starts estimates state-action-values instead of just state-values. For these, the value is calculated separately for every action of the state. Exploring starts refers to the fact, that for the initial state, one of all possible state-action pairs is chosen randomly but with the same probability. This is done to ensure that every state-action pair is visited, for the limited state-action space of blackjack with its stochastic properties, this random starts are represented by the game logic. The pseudocode seen in figure 5 is analogous to the previous sections one for the first visit, meaning that for this method the first and every visit approaches are again the same.

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated
Initialize:
  $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$
  $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):
  Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:
      Append G to $Returns(S_t)$
      $V(S_t) \leftarrow$ average($Returns(S_t)$)

Fig. 5. Pseudocode for the MC Exploring Starts from [2]

III. Experiment

In the experiment different Monte Carlo Methods are evaluated and compared with respect to different strategies.
For the Blackjack game, an environment was implemented to be able to simulate different strategies and special-rules. Especially the logic for the two counting systems mentioned in the previous section were implemented, to see the influence on the outcome of the game. In the experiments it turned out that the Complete Point-Count System only had an influence on the games if the corresponding strategy was used. Due to the many values the High-Low Index could become, constructing an special state-space with respect to the High-Low index wasn´t considered.
Every Algorithm was trained on 500000 episodes, representing the same number of Blackjack games.

A. On Policy Monte Carlo Prediction

The method was used with a gamma value of 0, meaning that previous states don´t get any shared reward from the last state. The Basic Strategy, the Simple Point-Count System as well as the Complete Point-Count System were evaluated using the Monte Carlo Prediction.
The Basic Strategy was used including special rules for the dealer showing a ten.
The Simple Point-Count System was considered using the Basic Strategy. The reward was adjusted to the given point count.
For The Complete Point-Count System, the corresponding strategy was used with respect to the calculated High-Low Index. The rewards were not further adjusted for this.

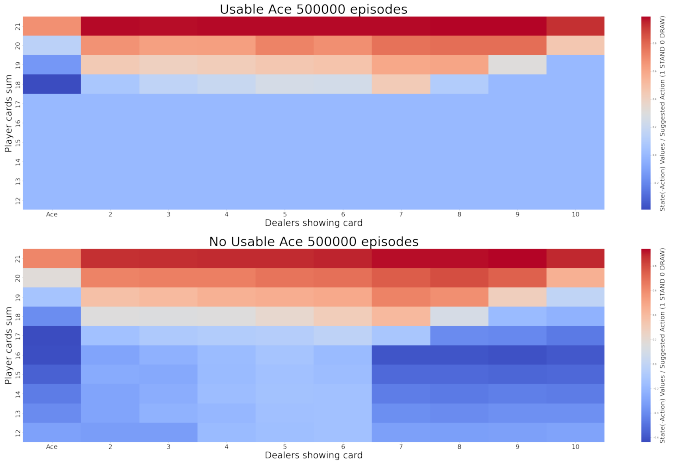

Fig. 6. State-value function of MC Prediction for the Basic strategy

All evaluated strategies eventually converged to a state-value function similar to the one in figure 6. Considering the red areas suggest to stand and the dark blue ones to draw, the state-value functions look very similar to the Basic Strategy table in figure 1. Contrary to expectations, the different point-counting systems had little influence on the results of the evaluation.

B. Monte Carlo Exploring Starts

For the MC Exploring Starts method, a random strategy was used that took the actions in every state by chance.

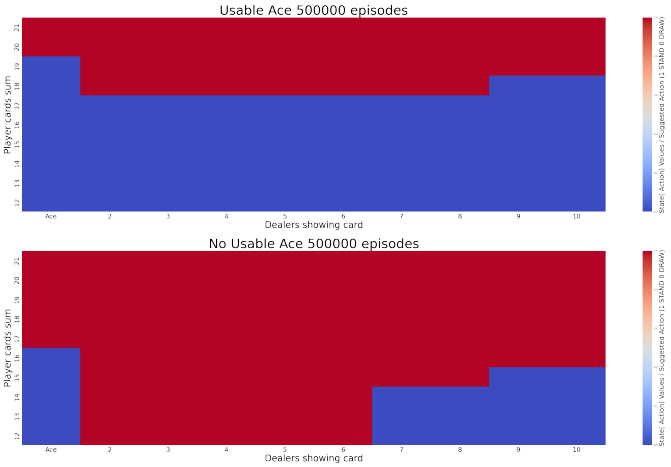

Fig. 7. Policy of MC Exploring Starts for the Basic strategy

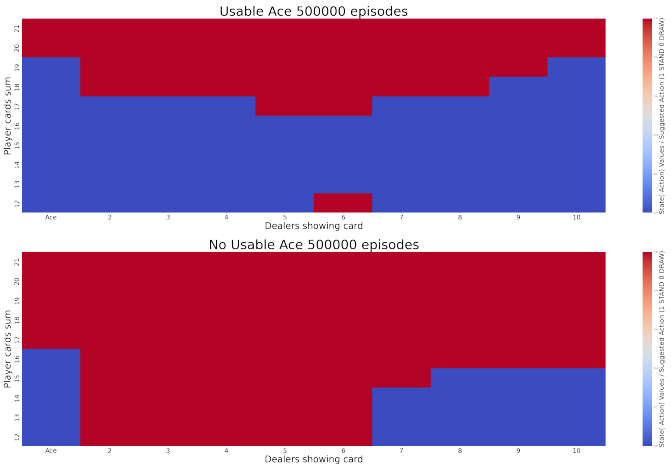

Fig. 8. Policy of MC Exploring Starts for the Basic strategy

The Policy represented in figure 7 is the result of the MC Exploring States algorithm with no special rules. The lower figure 8 was created using the Simple Point-Count System. Even though both are based on different reward calculations, the policies look almost the same. Both also are quite similar to the basic strategy.

C. Comparing win-rates and average returns

Every algorithms result was used as a strategy to play Blackjack. For every strategy 1000 epochs of 1000 episodes were played, resulting in 1000000 games of Blackjack. The state-value functions evaluated by the MC Prediction were used as a policy by counting state-values bigger than 0 as *stand*, otherwise *draw*. The policies of the Monte Carlo Exploring Starts method were used as they were. For further comparison all strategies were used without any learning

algorithm applied, for example mimicing the dealer (stand over 16) as seen in figure 9.

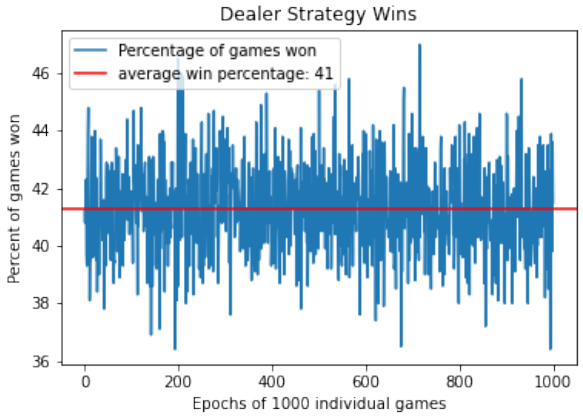

Fig. 9. Average win-rate of the "mimic the dealer" strategy

Comparing different strategies influence on the game		
Strategy	Win-rate	average result
Mimic Dealer	41%	-0.08
Basic Strategy	**44**%	**-0.04**
MC Pred. Basic Strategy	37%	-0.18
MC ES	43%	-0.05
Simple Point-Count	**44**%	**-0.04**
MC Pred. Simple Point-Count	37%	-0.18
MC ES Simple Point-Count	43%	-0.05
Complete Point-Count	43%	**-0.04**
MC Pred. Complete Point-Count	37%	-0.18

The results seen in table III-C provide a fairly clear picture. Non of the strategies provided by the Monte Carlo algorithms really worked better than the original strategies. Most of them scored worse than the "mimic dealer" strategy. As expected, all strategies derived from die Monte Carlo Prediction, resulted in similar values. The expandable results could have been caused by too few training episodes.

IV. Conclusion and open Questions

The advantage of the Monte Carlo Methods in this specific task is, that it is made for environment which episodes terminate eventually. Further, the algorithm was very easy to apply, because with the gamma value set to zero, the destabilizing influence on former states could be neglected. The feature that first and every visit have the same effect in blackjack was also useful.

To estimate the state-space of Blackjack was definitely a doable task. As already seen in the tables 2,3 and 1, the state-space consists of two matrices $\in \mathbf{R^{10x10}}$ with only two possible actions - stand and draw, easily representable by adding another dimension. It was convenient that values below 12 did not have to be taken into account, since a draw could not result in a bust. Separating The state space for the cases that the player has or has not an ace further simplified this.

However, it is also important to note that the strategies in the book are already optimized and tried-and-tested methods for changing the game as far as possible in one's own favor. Further, the Monte Carlo methods were never designed to further improve the already optimal methods, but to approximate them using the statistical properties of the game. For this, approximating the optimal strategies for Blackjack in virtually no time, instead of crazy-complicated calculations as the authors of [1] probably did, can be considered a success. It should also be mentioned positively that the optimal strategy was approached very precisely. Contrary to the expectation that changing conditions, such as card counting, would lead to a completely different strategy, Monte Carlo methods have consistently approximated the basic strategy. For future work, it is suggested to try a different state-space, considering the High-Low Index, to see how close the algorithm can approximate the suggested ones as on table 2 and 3.

References

[1] Edward O. Thorp, "Beat The Dealer" Vintage, New York, 1966
[2] Richard S. Sutton and Andrew G. Barto, "Reinforcement Learning: An Introduction" MIT Press, Cambridge, MA, second edition, 2018