Deep Reinforcement Learning for Blackjack

Marius Oechslein
Faculty of Computer Science and Business Information Systems
University of Applied Sciences Würzburg-Schweinfurt
Würzburg, Germany
marius.oechslein@study.thws.de

***Abstract*—**
***Index Terms*—Blackjack, Reinforcement Learning, Monte Carlo**

I. Introduction

A. Blackjack and Reinforcement Learning

Blackjack is a popular card game played in casinos worldwide. In 1962 the mathematician Edward Thorb published a Book on how to beat the house in Blackjack [1]. This made is possible for players to win money in the game of Blackjack in casinos. Although the casinos changed their rules to prevent the player from winning in the game of blackjack impossible now, the game is still an interesting subject to mathematical modeling due to its probabilistic nature.

Edward Thorb introduced the idea of card counting with the Complete Point Count System [1] which makes it possible for players to keep track of a score while playing that indicates their chances of winning. The Complete Point Count System keeps track of a counter that adds +1 or -1 based on each card that are seen by the player. This easy counting method makes it possible for players to count cards while playing the game.

B. Reinforcement Learning for Blackjack

The game can be modeled as a Markov Decision Process which makes it possible to apply Reinforcement Learning. With todays computation power it is now possible to let the computer play over 100.000.000 games, which wasn't possible in the 1960s. The baseline goal is therefore to at least achieve the basic strategy of Edward Thorb [1] with Reinforcement Learning methods.

All of the Reinforcement Learning methods play a large number of games while keeping track of a Q-Table containing the expected returns to each of the possible game states [**?**]. The different Reinforcement Learning methods like Sarsa, Q-Learning and Temporal Difference only take a different approach of updating this Q-Table [**?**]. Although the Reinforcement Learning methods are very powerful, they require an increasing amount of games to be able to model the Q-Table as the state-space of the game increases. In this paper we therefore investigate how well the Reinforcement Learning methods Sarsa, Q-Learning and Temporal Difference Learning perform as we increase the state-space by changing the card counting method. We change the card counting method by keeping track of the values of every seen card and not just keeping track of a counter.

C. Deep Reinforcement Learning

In 2013 DeepMind published the paper *Playing Atari with Deep Reinforcement Learning* [2], combining Deep Learning to Reinforcement Learning. The idea of Deep Reinforcement Learning is to replace the Q-Table with a deep neural network. This makes it possible to estimate the expected returns even for complex state-spaces.

In this paper we first investigate whether Deep RL is also able to achieve the baseline of the basic strategy of Edward Thorb [1]. Secondly we compare how the Deep RL performs compared to the more traditional RL-methods like Sarsa, Q-Learning and Temporal Difference Learning.

II. Methods

A. The Blackjack Environment

The game of Blackjack start by the dealer and the player recieving two cards where the player is only allowed to see one of the dealer's cards and his own cards The player only has the two actions of taking another card (Hitting) and not taking another card (Standing). Although the real rules of Blackjack extend this action-space by options like Doubling Down and Splitting, in this paper we only focus on Standing and Hitting. The Blackjack Environment used for the implementation of this paper only focuses on Hitting and Standing.

The player can then hit as many as he wants with the goal of coming as close to the cards value of 21 as possible. After the player finished his turn the dealer hits as long as his cards value are below or equal 16. At the end of one game the player's return is +1, -1 or 0 based on whether it was a win, loss or draw.

To model the basic strategy with Reinforcement learning, one game like that can be seen as one episode. This is possible since the basic strategy only focuses on whether the player should Hit or Stand based on the dealer's card value and the player's cards value.

When introducing card counting, one episode has to be extended by playing multiple games in one episode. This is necessary since card counting is dependent on seeing a high number played cards since the player's chances change depending on the ratio of low cards/ high cards played. In the case of one game the player on average see 3-5 cards which is not enough for an imbalanced ratio between high and low cards. Therefore one episode is extended to playing through one whole deck.

The Complete Point Count System [1] keeps a running score and updating it every time the player sees a card:

- +1 for: 2, 3, 4, 5, 6, 7
- -1 for: 10, A
- 0 for: 8, 9

It is favourable for the player when the running score is high [1]. And it is favourable for the dealer when the running score is low. This is because the dealer's chances of busting increases when there are only few low cards left in the deck.

B. State-action space

For learning the basic strategy the state consists of the **player's cards value**, the **dealer's card value** and if the player has an **usable ace**. This makes a total number of **250 states** possible states that need to be considered:

- 10 possible dealer states: 2,3,4,5,6,7,8,9,10,A
- 25 possible player states:
  No Ace: 4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
  With Ace: 13,14,15,16,17,18,19,20

For learning the Complete Point Count System the state has to be extended by **running score**. The running score in one game can at most be 24 (= 6 cards * 4 suits) and at least -20 (= 5 cards * 4 suits). This would increase the state-space to **11.000 states** (= 250 states * (24 - 20)) for the Complete Point Count System, although the running score around 0 are far more likely than 24 or -20.

And when making the card counting method more complex by counting every value of the seen cards, the state space increases to approximately **907.000.000 states** (= 250 states * ((52 cards / 4 suits) - 3 face cards)!) for playing through one card deck. Altough the actual state-space is a little less considering that the player can only see one card of the dealer. The most important thing to note here is that the state-space explodes when using the more complex card counting method.

With this huge state space it is interesting to observe how Deep Reinforcement Learning performs and how other Reinforcement Learning methods perform compared to that.

C. Reinforcement Learning methods

1) Sarsa:

2) Q-Learning:

3) Temporal Difference Learning:

D. Deep Reinforcement Learning

III. Results

A. Basic Strategy

B. Increased state space - counting all cards

IV. Discussion

V. Conclusion

References

[1] Thorp, E. O. (1966). Beat the dealer: A winning strategy for the game of twenty-one (Vol. 310). Vintage.

[2] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

[3] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.