Reinforcement Learning Blackjack

November 25, 2025

1 Task

The aim of this project is simply to train an agent to play Blackjack, in an environment that can have a varying degree of complexity. For simplicity, we consider a stylised version of the game where there is no opponent.

2 Rules of Stylised Blackjack

- There is a single player. The dealer is passive (just deals the cards).
- The game is played with a number D of decks of Poker cards (52 cards per deck, 13 per suit ranging from 2-10 plus J/Q/K/A).
- Cards have a numerical value equal to their number for 2-10, figures J/Q/K are all valued 10, and Aces are valued 11 unless the sum in the hand goes over 21 and then they are valued 1. The suit is irrelevant.
- The D decks are shuffled randomly at the start of the game. The cards are dealt one by one from this random shuffle.
- The game lasts until all the cards are used. This defines an *episode* of the game (in RL terminology). NB the cards are *not* reshuffled between hands.
- The aim of the game is to obtain the maximum accumulated score over an episode, by adding up the score of each hand played, see Eq. (1) below.

- A hand represents one cycle of the game through the following sequence of steps:

  1. A card is dealt. We call its numerical value C_1.
  2. At iteration n, the player considers the total sum of the cards dealt in the hand, $C_1 + \cdots + C_n$, and decides whether to "stick" thus ending the hand and going to Step 4, or to "hit" to get another card from the deck and continue with the hand.
  3. A new card is dealt, we call its value C_{n+1}. The sum is updated to $C_1 + \cdots + C_{n+1}$. If the total is less than 21, the hand can continue and we go back to Step 2 (recall the rule about Aces).
  4. The hand is finished, either because the player stuck at Step 2 or because the total value of the cards is 21 or beyond. The resulting score for the hand is

     $$S = \begin{cases} (C_1 + \cdots + C_{n+1})^2 & \text{if} \quad C_1 + \cdots + C_{n+1} \leq 21 \\ 0 & \text{if} \quad C_1 + \cdots + C_{n+1} > 21 \end{cases} \tag{1}$$

     That is, the score is *quadratic* in the total if it does not exceed 21, and zero otherwise.
  5. NB there are no other moves such as splits or surrender as in real Blackjack.

3 Considerations

The goal is to maximize the value of each hand without going above 21, by making choices either to receive another card ("hit") or to end the hand ("stick"), see Fig. 1. In the setting of the rules above, the complexity of the game is determined by the number of decks shuffled together in the draw pile.

(i) Simpler setting: infinite number of decks. If the draw pile contains an infinite number of decks, $D \to \infty$, the card draw can be treated as completely random, i.e., each type of card is drawn independently and identically to all other draws. In this case, the information about past hands is irrelevant to the probabilities of outcomes in the current hand. In the $D \to \infty$ limit, it

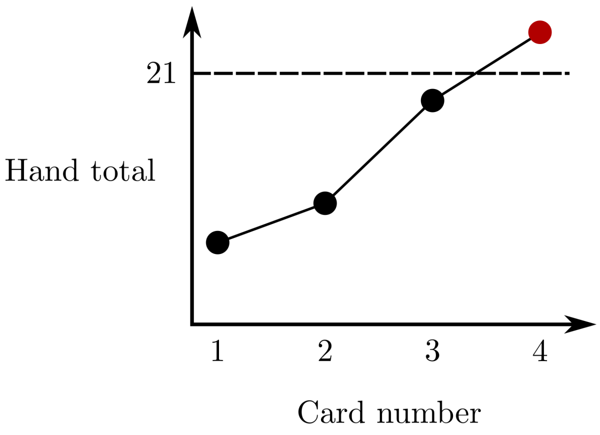

Figure 1: A turn by turn sketch of a hand in Blackjack. Sticking on card 3 would have resulted in a high score for the hand, while asking for another card leads to going over 21, and thus loosing the hand.

is not possible in finite time to play a whole game or episode. In practice one would play for a set finite number of hands. Under such conditions, given the infinitely large supply of cards each hand is independent. That is, each hand can be considered a separate episode of the game. This makes this setting the easier one to consider, and one would naturally try to solve this first.

(ii) More difficult setting: finite number of decks. If the draw pile contains a finite number of decks, $D < \infty$, then there is a fixed number of each type of card in the overall pile used in an episode, which changes over time as the game is played and the pile gets depleted. Thus, the probability of each draw is not independent of those which came before. In this case, while the game can still be played based purely on present information, better decisions are likely to be made by recalling some knowledge about past hands. As in many other computational tasks, there will be a need to strike a balance between potential advantage (for example in considering a large amount of past information) with computational cost/capability (as it will not be practical to recall everything).

In each of these cases, for RL you must first set up the problem by deciding on states and actions, defining the episodes, and choosing an appropriate reward function. This project will require both coding the Blackjack environment under the rules above and coding an agent with a reinforcement learning algorithm.

Note however that unlike the deterministic setting considered in the lectures, Blackjack is *probabilistic*, as there is no way to predict exactly which card will occur next. That is, the current state and the action taken does not fully determine the next state. Training an agent on this task will thus first require understanding how to extend reinforcement learning to a setting in which randomness is a key component (cf. Markov Decision Processes). See chapter 3 of the reference below for an introduction.

4 References

Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, second edition, MIT Press (2018), http://incompleteideas.net/book/the-book.html