Reinforcement Learning

Homework 2

Dynamic Programming - Value Iteration

Reykjavik University

Teacher: Stephan Schiffel

Fall 2023

Þórir Hrafn Harðarson – *thorirhh21@ru.is*

Modeling the environment

An environment was made to model the game of Blackjack. The model uses a deck consisting of infinitely many copies of each card, resulting in the probability of drawing a specific card is always 1/52. In this model of the environment a state is described by the phase of the game, the players total hand, if the player has a usable ace, the dealers total hand and if the dealer has a useable ace:

State: (phase, player hand, player ace, dealer hand, dealer ace)

The game will always start in the same initial state:

('start', 0, False,0, False)

And ends in the same terminal state:

('terminal', 0, False,0, False)

To simplify the problem and reduce the number of states the suit of the cards can be ignored. The jack, queen, and king all count as a value of 10 so they can be viewed as the same card as a 10. This will result in a simplified deck of:

[1 2 3 4 5 6 7 8 9 10 10 10 10]

where at each draw the cards all have a probability of 1/13, except for the 10 value card that has a probability of 4/13. From the initial state we can generate all possible starting hands for both the player and dealer, along with their probabilities, by using this simplified infinite deck.

Possible hand values for the player at the start are:

Ace = false : 4 - 20 (17 possibilities)

Ace = true 12 - 21 (10 possibilities)

Resulting in 27 possible hands.

The dealer can have 10 different values on their card, resulting in 270 possible states after the initial draw.

In total the player can have in their phase:

Ace = false : 4 - 21 (18 possibilities)

Ace = true 12 - 21 (10 possibilities)

With the dealer's card this results in 280 possible states in the player phase.

The dealer will always stand and end the game if they have reached a hand value of 17 or higher, the model will therefore never see a state where the dealer has a higher value than 16. Starting only with one card, the dealer therefore has possible hand values:

Ace = false : 2 - 16 (15 possibilities)

Ace = true 11 - 16 (6 possibilities)

This results in 21 * 28 = 588 possible states in the dealer phase.

With the start state and terminal state the state space therefore has in total 870 possible states.

During their turn the player and dealer have two possible actions: ['hit, 'stand']

Unless the player has become bust (hand value greater than 21) they will always have an option of performing either a ‘hit’ action or a ‘stand’ action.
The dealer will always perform a ‘hit’ action if their hand value is less than 17 and always perform a ‘stand’ action otherwise. During a ‘hit’ action the player's or dealer’s hand will get updated by drawing a single card. During a ‘stand’ action the phase will either shift from ‘player’ to ‘dealer’ or from ‘dealer’ to ‘terminal’ and the game ends. A ‘wait’ action is also included for phase transitions. If the player wins the game the agent will get a reward of 1 but if the dealer wins the agent gets a reward of -1. In case of a draw the reward becomes 0.

The model described above was implemented and tested by printing out possible states, summing up probabilities to see that they add up to 1, by printing out state transitions and by observing possible states from certain cases, i.e. observing if for a certain dealer state there exist 28 player states.

The resulting number of states from the model can be seen below:

```
The MDP has 870 reachable state(s).
player states: 280
dealer states: 588
starting states: 1
terminal states: 1
```

Figure 1 - Reachable states in the Blackjack model

Value Iteration

Having modeled the environment, value iteration was implemented for a MDP and then used to find the resulting value for each state by playing the optimal policy. Value iteration was performed with a discount factor of *gamma* = 0.99 and ran until the difference between value updates, *delta*, had reached a threshold less than *theta* = 0.001.
Having found the state values these could then be used to iterate through the (state, action) space and find the optimal policy by selecting the action at each state that gives the highest value. By looking at the value of the starting state we can find the resulting value of playing with the optimal policy, which for the model of this environment results in an outcome of -0.05.
Since the dealer does not have to act until after the player has taken the risk of going bust the dealer will eventually always win, even if the player uses an optimal policy to play the game.
The resulting values and actions for the starting hands can be seen in the figures below:

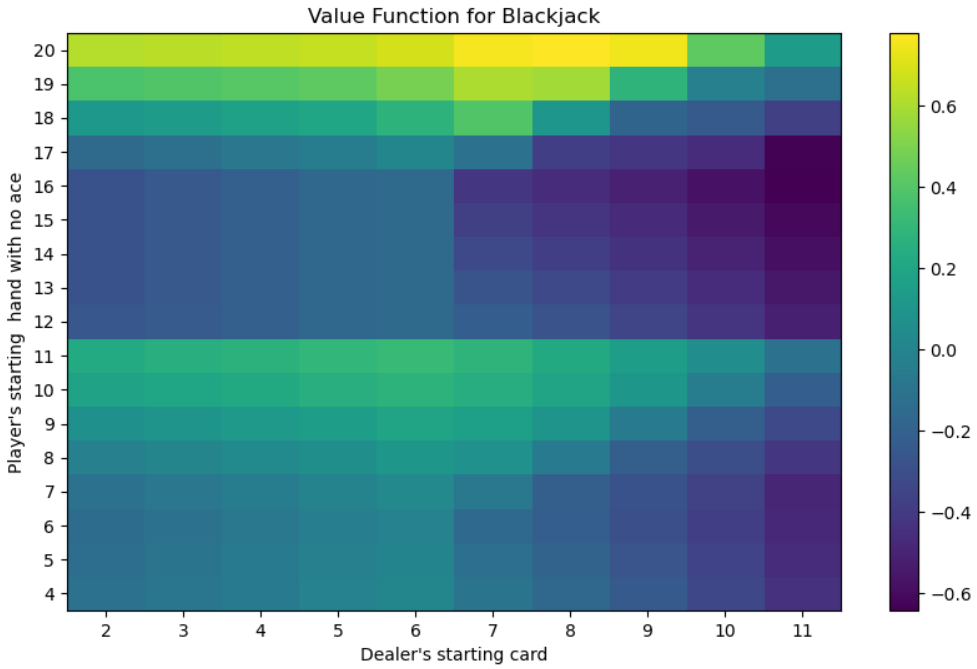

Figure 2 - Expected values for the initial hands when the player has no ace

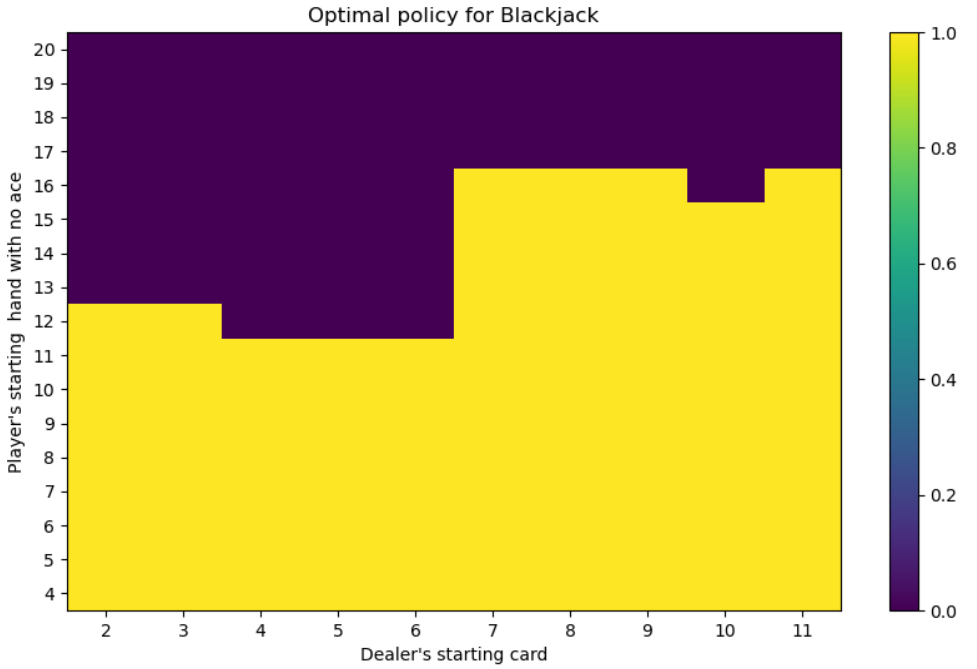

Figure 3 - Optimal actions for the initial hands when the player has no ace, purple = stand and yellow = hit

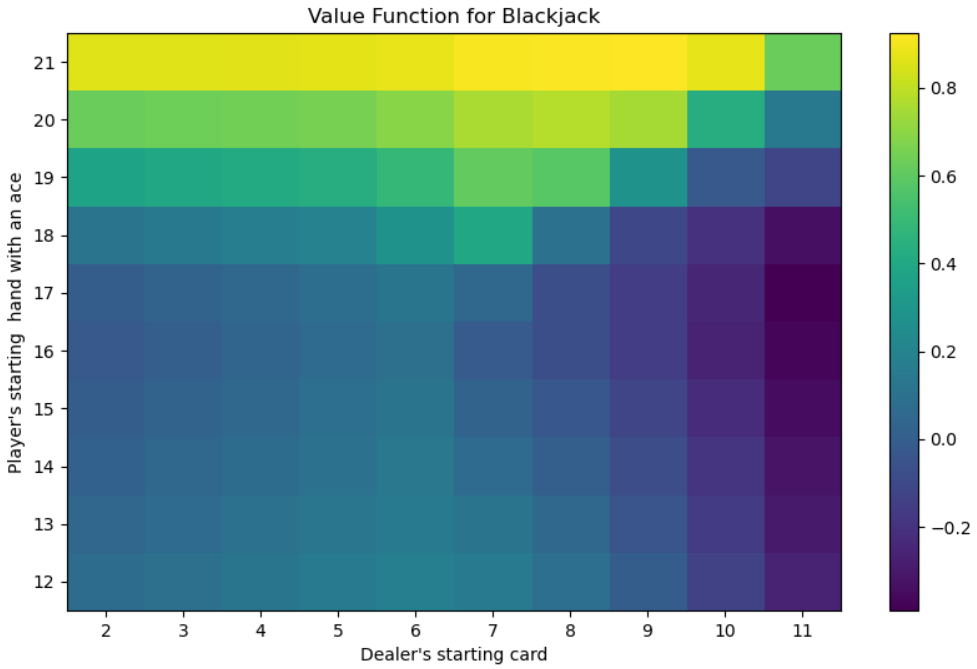

Figure 4 - Expected values for the initial hands when the player has an ace

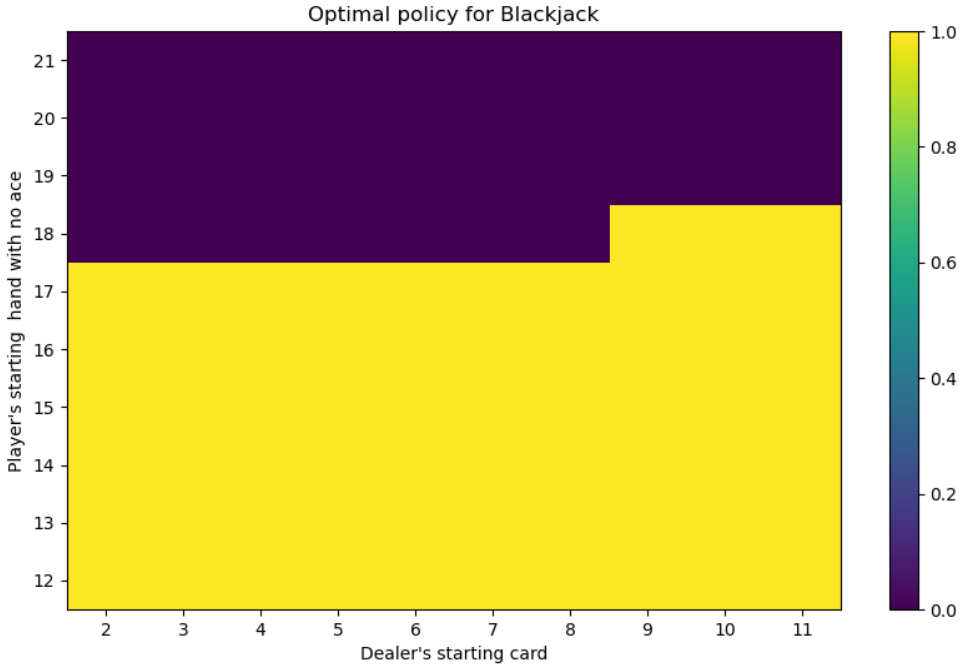

Figure 4 - Optimal actions for the initial hands when the player has an ace, purple = stand and yellow = hit

By adding in a double down action the player now has an option of getting a single card and ending his turn but getting a double reward. Adding this new action changes the value function slightly as can be seen in the figures below:

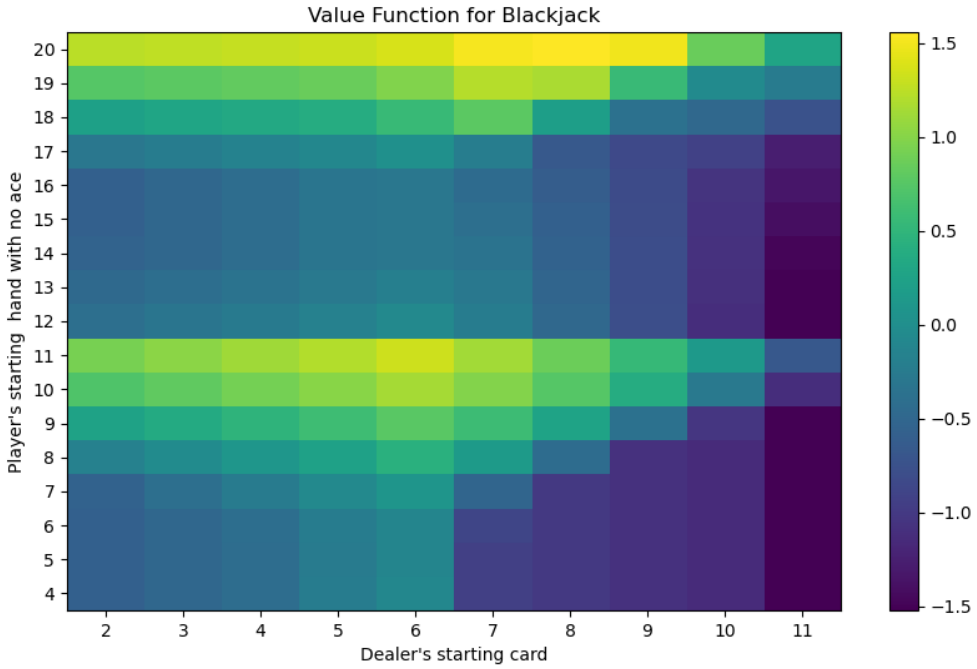

Figure 5 - Expected values for the initial hands when the player has no ace and can use a double down action

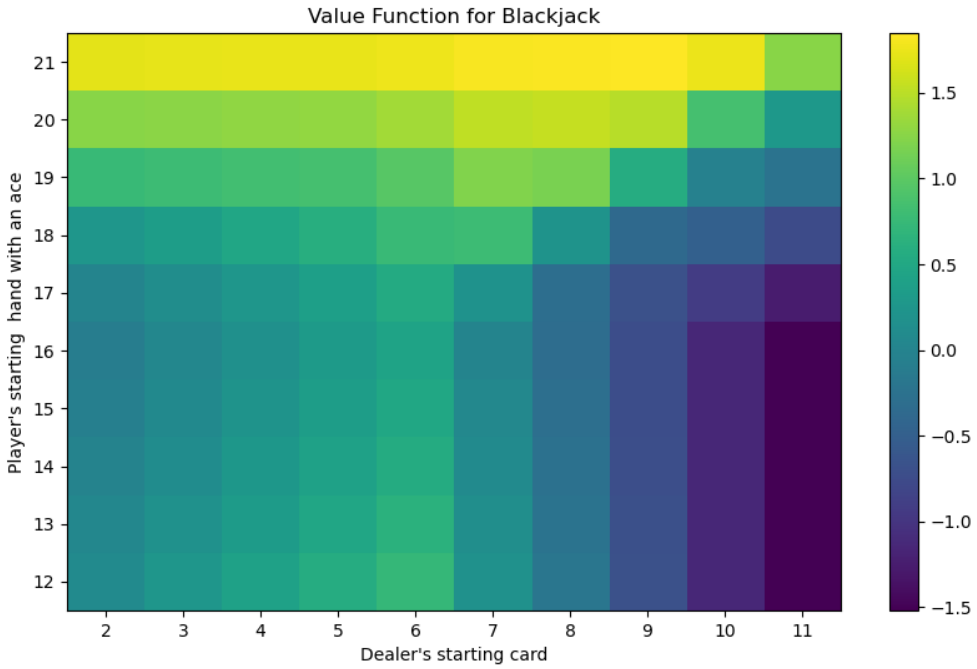

Figure 6 - Expected values for the initial hands when the player has an ace and can use a double down action

From the figures above it can be seen that an agent can now get an increased reward when it starts with a hand value of 9 - 11 which was not available to it before. An optimal policy would therefore include these actions in those cases. Sadly an error in the implementation caused the agent not behave correctly in those cases. This should bring the expected outcome from the optimal policy closer to a neutral value of zero but the dealer would still win in the long run.

In the case of having a finite deck of cards the modeled environment can longer rely on the simplification of looking only at the values of each card. This is still enough to accurately assess the state of play but to be able to correctly predict the probabilities of state changes the model will need to consider all cards and their suits. Some simplification could possibly be made since the amount of 10 value cards that can be in play is limited but the state space can be expected to increase by a magnitude of 4 to around 870^4.
The value function should now give increased value to good starting hands, especially those that include an ace, since good cards have now become limited in the deck. Even so the player is always forced to take a risk before it is the dealer's turn and will therefore still lose in the long run.

By using a policy of its own the dealer should in theory be able to increase its win rate by trying to maximize the probability of winning by doing a value iteration on the possible states that can arise after the dealer has seen the player's hand. It is hard though to say how much more the dealer could gain since it only reacts after the player has taken the risk of trying to maximize its hand without going bust. The dealer on the other hand doesn’t need to get as high a value as it can, only a higher value then the player. If the player is playing optimally, and has not gone bust, then the dealer would probably in most cases need to get to a value of 17 or higher to win. Only in rare cases would it be in the dealer's interest to get another card when it already has 17 or above and then that would be to try and beat a very good player hand, which would in most cases have a low probability.