Homework 2: Dynamic Programming - Value Iteration

T-747: Reinforcement Learning

Fall, 2023

1 Introduction

The goal of this assignment is to get a better understanding of dynamic programming approaches for finding the optimal policy for a problem, by implementing value iteration for a specific problem.

2 Handing in

Hand in **a report (PDF document)** with the answers to all the questions below including the respective Python code that generated the plot or numbers for your answer, if any.

3 Environment

Consider the game of Blackjack played by a single player against the dealer. The game is played with a standard deck of 52 cards (2 to ace of the 4 suits). Each card has a value. The value of cards 2 to 10 is their pip value, face cards (Jack, Queen, and King) are worth 10 points and an Ace can be worth 1 or 11 points, as the player chooses. A hands' value is the sum of the card values. The goal of the game is to get a hand worth more than the opponent's hand, but not more than 21.

There are several variations on the rules of the game. For the purpose of this assignment, use the following rules:

To start with, the player gets two cards and the dealer gets one card from the deck (all face up). Now, as long as his hand value is below 21, the player can "hit", i.e., get one additional card in his hand. Alternatively, they can "stand" to end their turn. If the hand value of the player is over 21 the dealer wins. Otherwise, the dealer will now play according to this fixed strategy: A dealer will "hit" as long as their hands' value is less than 17 and "stand" otherwise. The dealer always counts aces as 11 points, unless that would bring their hands' value above 21. After the dealers turn, the outcome of the game is determined as follows (in this order):

- If the players' hand has a value above 21, the player looses.
- If the dealers' hand has a value above 21, the player wins.
- If player and dealer have hands with equal value the game ends in a draw.
- The player with the higher valued hand wins.

4 Tasks

1. Modeling the environment *(Total points: 40)*

   For this task, assume that the game is played with a deck consisting of infinitely many copies of each of the 52 cards. That is, even after having seen a card, the probability of drawing a specific card is still $\frac{1}{52}$.

   (a) (15 points) How would you describe a state of the problem? What are the possible actions, successor states, rewards and transition probabilities? Try to reduce the state space as much as possible (combine states whose values and best action do not differ), but make sure to keep the Markov property. To make it easier to implement the model later, try to keep the following in mind:

- Have only a single initial state (the game should always start in the same state). That means initially no cards should be dealt yet.
- To figure out the probabilities of state transitions, it is easier if a state transition does not involve too many changes to the environment at once. Rather have more and smaller state transitions than fewer and big ones.
- There may be states in the environment in which the player can not do anything but wait, but there is still something happening (e.g., dealing cards, the dealer playing, ...).

(b) (25 points) Implement your model of the problem by filling out the relevant functions in BlackjackMDP in `hw2.py`. Test your model! For example, testing can be done by

- generating some simulations of random game play, printing out the states and checking whether the states are what you would expect
- counting certain types of states in the set of all states and checking whether the number is what you would expect (e.g., how many states are there where it is the player's turn)
- computing the probability of certain state transitions or reaching certain states and comparing with the probability that the model computes
- checking whether for each state the sum of probabilities of all possible state transitions from that state is 1

How many reachable states does your model have?

2. Value Iteration *(Total points: 60)*

(a) (25 points) Implement Value Iteration for an arbitrary MDP (any object derived from the MDP class in the code). Decide on an appropriate stopping criterion and run your code on the Blackjack MDP.

Note, that for the code on the slides to work correctly, your model needs to fulfill certain criteria:

- There needs to be a single starting state.
- In every state (including the terminal state), the player needs to have at least one possible action (possibly just waiting for something to happen).
- From the terminal state, there needs to be a state transition to the terminal state with reward 0 and probability 1.

Hints:

- An easy way to store $V(s)$ is in a dictionary with s as a key. This works well if states are tuples, not lists.
- A list of all reachable states in the MDP is in `mdp.states`.

(b) (10 points) What is the expected outcome of playing Blackjack with the optimal policy? Who will win the the long run, the player or the dealer? Why?

(Hint: Think about which value you need to look at to answer the question. The algorithm should already have computed the value.)

(c) (15 points) Visualize the resulting value function and policy. For example, you could use a table similar to the one on Wikipedia, depending on your choice of state space.

(d) (10 points) Suppose the player had an additional action "Double down". Upon "Double down" the player will get exactly one more card and end his turn. He will also double his initial bet, that is, win or loose double the normal amount. Show how having this additional action changes the optimal policy and expected outcome of the game. (Essentially, you need to make a copy of your model, add the additional action and run the value iteration on the new model.)

(e) **(10 bonus points)** Playing with an infinite deck of cards is somewhat unrealistic. Assume to only play with one deck of cards (52 cards) and the player is able to count cards (remember the cards that have already been dealt). How many different states do you need to consider in this case? How does the value of the game change in this setting? Does card-counting give a player the edge it needs to win in Blackjack in the long run?

(f) **(10 bonus points)** Consider again, the game played with an infinite deck of cards and no "Double Down" action. Is the given fixed policy for the dealer optimal or could the dealer actually win more by using a different policy? Discuss how you could answer that question and which problems might arise.