Lab: Sequential Learning and Decision Making

INF581 Advanced Topics in Artificial Intelligence

1 Viterbi Algorithm

Recall Lab 1. In that lab you were encouraged to use a recursive function call to expand all paths (this is a valid choice due to the nature of the scenario). In the file `HMM.py` a very small test environment is also given with a reduced number of state dimension ($y \in \{0, 1\}$) and input dimension ($x \in \{0, 1\}$).

Task Now implement the *Viterbi algorithm*, using it to calculate *the most likely path.*

2 MDPs: Value Iteration

In the file `MDP.py`, a random environment is generated. Note in particular the `P` and `R` matrices. Optionally, there is a function to output the model as a `.dot` file which can be compiled into a graph with Graphviz. An example environment is shown in Figure 1.

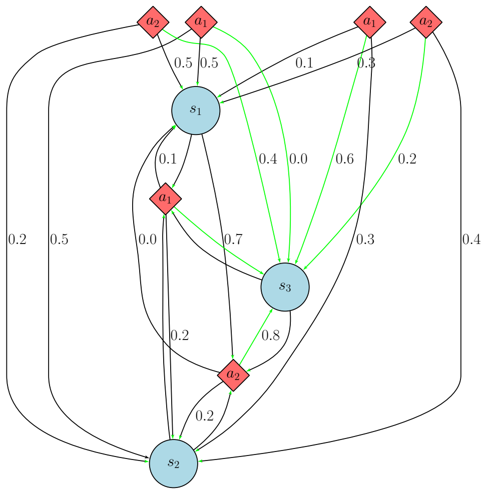


Figure 1: A random environment shown graphically. Transition functions can be inferred from the edges. Only edges shown in green give a non-zero reward (which is positive).

Task Implement Value Iteration on the same environment, as a function. This function should give you the value function and thereby also the policy.

Bonus You are given a shell of an OpenAI Gym environment (`myenv`). You can reformulate the MDP into the environment, and give your resulting policy to an agent; as defined in this library.

Bonus Again recall Lab 1. Rather than the ‘cat‘, consider the ‘rat’ as an agent. Suppose that it has full knowledge of all tiles (including their stochastic nature for producing sounds). Formulate a task by assigning suitable reward (from the point of view of the rat agent) and then map out the value of tiles.

3 MDPs: Policy Iteration

Following from Task 2: As discussed in the lecture, an alternative approach to value iteration is that of policy iteration.

Task Implement **Iterative Policy Iteration** (for the same environment). Note that as part of this task you should also implement **iterative policy evaluation**. Compare the policies obtained by both approaches (they should be the same).

4 Monte Carlo Control

1. Formulate Blackjack as MDP (episodic finite)
2. Monte Carlo algorithm (ϵ-greedy)

In this task we are going to build agents capable to deal with the famous Blackjack casino card game; specifically, based on the **Monte Carlo (MC) control** algorithm.

Blackjack is one of the most popular casino card games. The objective of player is to obtain cards the sum of whose numerical values is as great as possible without exceeding 21. In this lab, we consider the case in which each player competes independently against the dealer.

1. All face cards count as 10, and an ace can count either as 1 or as 11.
2. The game begins with two cards dealt to both dealer and player.
3. The player's cards is face-up.
4. One of the dealer's cards is face up and the other is face down.

After the initial cards are dealt, each player can choose to 'stick' or 'hit' (ask for another card). Actually, the player could request additional cards, one at each time ('hit'), until he either stops ('sticks') or the sum of his cards exceeds 21. If he goes bust, he loses. If he *sticks*, then it is the dealer's turn. The dealer *hits* or *sticks* according to a fixed policy (strategy). More specifically, the dealer *sticks* on any sum of 17 or greater, and *hits* otherwise. If the dealer goes bust, then the player wins; otherwise, the outcome — win, lose, or draw — is determined by whose final sum is closer to 21 (if the dealer is closer to 21, the dealer wins and the player loses, and vice versa).

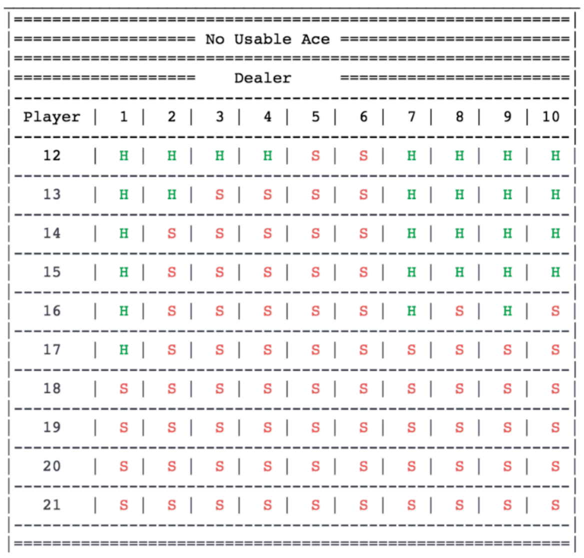
4.1 Formulate Blackjack as a MDP

We can formulate the blackjack game as an episodic finite *Markov Decision Process*

1. Each game of blackjack is an episode.
2. Rewards of $+1$, -1, and 0 are given for winning, losing, and drawing, respectively. All rewards within a game are zero, and we do not discount ($\gamma = 1$); therefore these terminal rewards are also the returns.
3. The player's actions are to hit or to stick.
4. The states depend on the player's cards and the dealer's showing card.
5. We assume that cards are dealt from an infinite deck (i.e., with replacement) so that there is no advantage to keeping track of the cards already dealt.
6. If the player holds an ace that he could count as 11 without going bust, then the ace is said to be usable. In this case it is always counted as 11 because counting it as 1 would make the sum 11 or less, in which case there is no decision to be made because, obviously, the player should always hit.
7. The player makes decisions on the basis of three variables: his current sum, the dealer's one showing card (ace-10), and whether or not he holds a usable ace.

Tasks

- Define a Q-table of the correct size to host the Q-function estimate.
- Define a structure to hold the number of visits of each state-action pair.
- Implement the Monte carlo agent with ϵ-greedy exploration.
- Visualize the optimal value function by using the `visualizeValue.py` function, which takes as input the Q-table.



```
 ___________________________________________________________
|===========================================================|
|=================== No Usable Ace ========================|
|===========================================================|
|===================     Dealer      ========================|
|-----------------------------------------------------------|
| Player |  1 |  2 |  3 |  4 |  5 |  6 |  7 |  8 |  9 | 10 |
|-----------------------------------------------------------|
|   12   |  H |  H |  H |  H |  S |  S |  H |  H |  H |  H |
|-----------------------------------------------------------|
|   13   |  H |  H |  S |  S |  S |  S |  H |  H |  H |  H |
|-----------------------------------------------------------|
|   14   |  H |  S |  S |  S |  S |  S |  H |  H |  H |  H |
|-----------------------------------------------------------|
|   15   |  H |  S |  S |  S |  S |  S |  H |  H |  H |  H |
|-----------------------------------------------------------|
|   16   |  H |  S |  S |  S |  S |  S |  H |  S |  H |  S |
|-----------------------------------------------------------|
|   17   |  H |  S |  S |  S |  S |  S |  S |  S |  S |  S |
|-----------------------------------------------------------|
|   18   |  S |  S |  S |  S |  S |  S |  S |  S |  S |  S |
|-----------------------------------------------------------|
|   19   |  S |  S |  S |  S |  S |  S |  S |  S |  S |  S |
|-----------------------------------------------------------|
|   20   |  S |  S |  S |  S |  S |  S |  S |  S |  S |  S |
|-----------------------------------------------------------|
|   21   |  S |  S |  S |  S |  S |  S |  S |  S |  S |  S |
|-----------------------------------------------------------|
|===========================================================|
```

Figure 2: MC Policy example on Blackjack game