Assignment: Reinforcement Learning

In this assignment you will apply reinforcement learning to the 4x3 grid shown below. You may use any programming language to implement the required algorithms (although Python is preferred and recommended).

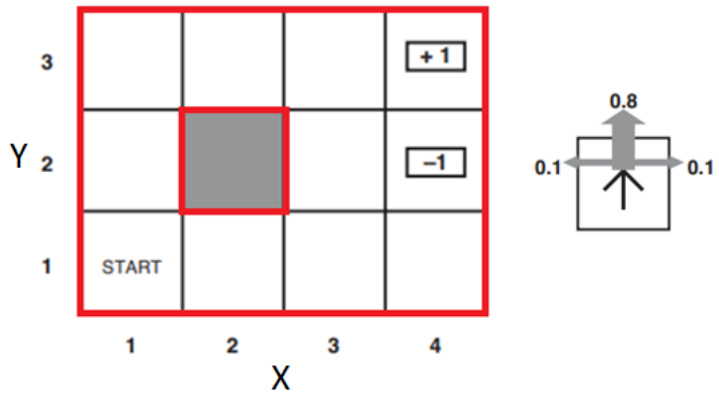

Figure 1: 4x3 world. Red lines show walls.

The rules of the environment are as follows.

1. An agent is supposed to start in cell (1,1).
2. Available actions, $A(s) = \{up,\ down,\ left,\ right\}$, where state s is some cell (x, y) with x being the column number and y being the row number.
    a. An action results in the intended movement with probability 0.8 but can cause movement in one of the perpendicular directions with probability 0.1 each.
    b. Bumping into a wall results in staying in the same cell.
3. $R(s)$ denotes the reward received in state s. $R\big((4,3)\big) = +1, R((4,2)) = -1, and\ for\ any\ other\ (x, y),\ R(x, y) = -0.04.$
4. The agent will exit the environment if any of the two terminal states, i.e., $(4,3)\ or\ (4,2)$, is reached.
5. The objective of the agent is to maximize the total sum of rewards received. Assume that rewards are additive. Use a discount factor $\gamma = 0.9$, wherever applicable.
6. The environment is fully observable – the agent knows exactly which sell it is in

Deliverables:

1. Answers to the 5 questions below in the form of a report in PDF format. Please copy and past the tables from this document as needed.
2. Source code in Zipped format (this should be separate from the PDF mentioned above).

Note: You can use a suitable value between 0 and 1 for the learning rate parameter α, wherever applicable.

Q1

Find the true utilities $U(s)$ and the optimal policy π^* by implementing either policy iteration or value iteration algorithm. Use discount factor $\gamma = 0.9$. Fill Tables 1 and 2 below with the values you obtain.

Hint: Use the rules of the environment given above (especially Rule 2) to compute the transitions probabilities $P(s' \mid s, a)$ that specify the probability of reaching state s' if action a was executed in state s.

Table 1: Utility values. In each cell enter the value of $U(s = (x, y))$

Y					
	1				
	2				
	3				
		1	2	3	4
			X		

Table 2: Optimal policy. In each cell, enter the $action \in \{up, down, left, right\}$ as specified by the optimal policy π^*.

Y					
	1				
	2				
	3				
		1	2	3	4
			X		

Q2

Implement a function $NextState(s, a)$ that returns the next state s' if action a is executed in state s according to Rule 2 above.

In order to test whether your function works as intended, for each $(state, action)$ pair shown in Table 3, input it to the function you implemented and record the next state returned by it. Repeat this process 100 times for each $(state, action)$ pair and fill the remaining columns of Table 3 indicating the number of times each adjoining cell or itself was reached as the next state s'.

Table 3: Input $(state, action)$ pairs and the frequency of output states.

$(state, action)$	No. of times each neighbor reached as next state				
	Itself	**Upper neighbor (if any)**	**Lower neighbor (if any)**	**Right neighbor (if any)**	**Left neighbor (if any)**
(1,1), Move Right					
(1,1), Move Up					
(3,2), Move Down					
(3,2), Move Left					
(3,3), Move Left					

Do your counts agree with what you would expect given the behavior of actions mentioned in Rule 2?

Q3

In this question we will assume that we do ***not*** know the state transition probability model $P(s' \mid s, a)$ and try to learn it using the $NextState(s, a)$ function you implemented in Q2. Note that the function $NextState(s, a)$ allows you to simulate how the agent will move in the given environment. In a real, physical environment, the environment itself will provide the function.

Now use the adaptive dynamic programming (ADP) approach to learn the transition probabilities $P(s' \mid s, a)$. For this, execute many trials such that in each trial, the agent starts at cell $(1,1)$ and follows policy π shown in Figure 2 using the function $NextState(s, \pi(s))$ to move around. Run as many trials as needed to achieve convergence or stop at a maximum number of trials that you can reasonably afford, whichever comes first. Provide your estimates for the following transition probabilities.

i. $P\big((2,1) \mid (1,1), \text{Move Up}\big) =$ ……
ii. $P\big((1,3) \mid (1,2), \text{Move Up}\big) =$ ……
iii. $P\big((2,3) \mid (1,3), \text{Move Right}\big) =$ ……
iv. $P\big((3,3) \mid (2,3), \text{Move Up}\big) =$ ……
v. $P\big((2,1) \mid (3,2), \text{Move Right}\big) =$ ……

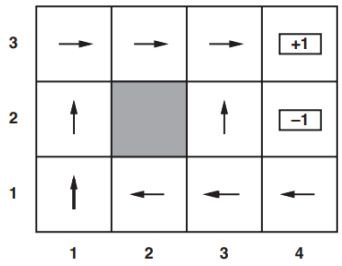

Figure 2: Policy π to be used in Q2.

Q4

Using the transition probabilities $P(s' \mid s, a)$ ***learned in Q3*** and implementing the GLIE (Greedy in the Limit of Infinite Exploration) scheme given below, estimate the true utilities $U(s)$. Note that the objective here is to learn the utilities from a learned model of the environment when the true model is *not* known (of course for this environment, the true model is given in Rule 2 but for this question we are pretending it's not available).

GLIE scheme

$U^+(s) = R(s) + \gamma f\left(\max_{a \epsilon A(s)} \sum_{s'} P(s' \mid a, s)\, U^+(s'), N(s, a)\right)$, where

- $U^+(s)$ - Optimistic estimate of the utility of state s.
- $N(s, a)$ – the number of times action a has been executed in state s.
- f is a function that should increase with expected utility and decrease with $N(s, a)$.

  Use $f(u, n) = \begin{cases} 2, & if\ n \leq 5 \\ u, & otherwise \end{cases}$

Fill Table 4 with the utility values $U(s)$ you obtained in this question.

Table 4: Utility values. In each cell enter the value of $U(s = (x, y))$

Y \ X	1	2	3	4
1				
2				
3				

Furthermore, obtain the optimal policy based on the utility values $U(s)$ you estimated in this question and fill Table 5.

Table 5: Optimal policy. In each cell, enter the $action \in \{up, down, left, right\}$ as specified by the optimal policy π^*.

Y \ X	1	2	3	4
1				
2				
3				

Q5

Compare the utility values and the policy you obtained in Q1 (where the true environmental model was known) with those obtained in Q4 (where the model was learned from trials) and comment on the accuracy of the reinforcement learning approach you used for this toy example.