Reinforcement Learning Homework

1. Suppose you are training an AI agent to drive a car down a perfectly straight road. The road has three lanes, and you want the AI to always stay in the middle lane.

   Suppose we set up our Markov Decision Process with three states, corresponding to the three lanes of the road, called Left, Center, and Right. So the agent will always be in the state corresponding to its lane.

   The agent may take one of three actions: Go-Straight, Shift-Left, or Shift-Right. These actions are designed to (in a perfect world), keep the car in the same lane, shift one lane to the left, or shift one lane to the right. Shift-Left is not possible from the left lane, and Shift-Right is not possible from the right lane.

   However, this agent is driving in Memphis, which has streets full of potholes, so each action only ever has a 0.75 probability of succeeding. The other 25% of the time, the agent will end up in a different lane than the one intended, according to this chart:

Agent in Lane	Action	Result and probability	
Left	Go-Straight	Left	0.75
""	""	Center	0.25
""	Shift-Right	Center	0.75
""	""	Left	0.25
Right	Go-Straight	Right	0.75
""	""	Center	0.25
""	Shift-Left	Center	0.75
""	""	Right	0.25
Center	Go-Straight	Center	0.75
""	""	Left	0.125
""	""	Right	0.125
""	Shift-Right	Right	0.75
""	""	Center	0.25
""	Shift-Left	Left	0.75
""	""	Center	0.25

   The agent receives a reward depending on the state it enters (after taking an action). The reward is +2 for being in the center lane, and +1 for being in either the left or right lane.

   a. Draw the MDP describing this situation. To make the diagram less cluttered, do not draw the rewards in the diagram, just the states (Left, Center, Right), and show arrows with the actions and probabilities.

   b. Suppose after running the value iteration algorithm on this MDP, with a gamma value of 0.9, the values that we converge to are all identical (this is actually true). V[Left] = V[Center] = V[Right] = 17.5.

      Use these three V-values and the recursive relationship between the V and Q equations (see

the RL notes handout) to fill in the following table of Q-values: (one below is already done for you)

Q[Center, Go-Straight] =
Q[Center, Shift-Left] = **17**
Q[Center, Shift-Right] =
Q[Left, Go-Straight] =
Q[Left, Shift-Right] =
Q[Right, Go-Straight] =
Q[Right, Shift-Left] =

Hint: This is the equation you want to use:

$$Q^*(s,a) = \sum_{s'} P(s' \mid s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

Walkthrough for computing Q[Center, Shift-Left]:

If the agent is in the Center state (s=Center), and chooses the action Shift-Left (a=Shift-Left), then there are two possible next states, either s' = Left (with prob=0.75) or s' = Center (with prob=0.25). R(Center, Shift-Left, Left) = +1, and R(Center, Shift-Left, Center) = +2, and the V-value of each state is 17.5, so the math becomes:

Q[Center, Shift-Left] = 0.75*(1 + (0.9)(17.5)) + 0.25*(2 + (0.9)(17.5)) = 17

c. Use your table of Q-values to derive the optimal policy for the agent (tell me the optimal action for each state). To be fair, this should be pretty obvious, but show your work anyway.

d. (1 bonus point extra credit): **Why** are all three V-values the same (17.5), given that we get 2 points for being in the center lane but only 1 point for being in the left or right lanes?

2. In the game of Nim, two players alternate taking sticks away from piles. On their turn, each player may remove any number of sticks, but only from **one** pile at a time. The player who is forced to take the last stick *loses*. *(Note: A player loses when they take the last stick of the entire game, not the last stick from a pile.)*

We will call the two players "A" and "B." We will represent states by the letter of the player who is about to move, followed by the number of sticks left in each pile. So if we start with three piles of 3, 4, and 5 sticks, then the first state of the game would be "A345." We will represent actions by two-digit numbers, the first digit being the number of pile the player chooses to remove sticks from (0-based, so

0=the first pile, 1=the second pile, etc), and the second digit being the number of sticks the player chooses to remove from that pile.

Let's imagine a game that is close to being over: say we are in state A012. At this point, pile 0 is empty, pile has 1 stick left, and pile 2 has 2 sticks left. Player A has three choices for actions: 11 (choose pile 1, take 1 stick), 21 (choose pile 2, take 1 stick), and 22 (choose pile 2, take 2 sticks). Each of these three actions would result in the next state being B002, B011, and B010. Note that there are no probabilities in this game; it is deterministic: each action only has one possible resulting next-state. (Makes the MDP much simpler).

The whole game only has two rewards. If "A" wins, they get +1000, and if "B" wins, they get -1000. (So "A" is the maximizing player, and "B" is the minimizing player.)

We will use gamma = 0.9, and alpha = 1. The reason we use alpha=1 is because in a deterministic environment (one with no randomness), Q-learning learns quickest with alpha=1 and still guarantees convergence to the correct Q* values.

In this project, you will walk through a few rounds of (2-player) Q-learning with the game of Nim. Specifically, we will start with piles of 0, 1, and 2 sticks at the beginning of each game, and player A always moves first.

Here are the games I want you to use. (Remember, in the real world, these games would all be played according to some policy that balances exploration and exploitation, but I'm going to just tell you what moves to make in this case.)

Game 1: A plays action 22, then B plays action 11 (ends the game).
Game 2: A plays action 11, then B plays action 22 (ends the game).
Game 3: A plays action 11, then B plays action 21, then A plays action 21 (ends the game).
Game 4: A plays action 22, then B plays action 11 (ends the game).

At the end of this homework are two tables similar to those we used in class to practice Q-learning with blackjack. Fill them in for games 1-4 (you can ignore games 5, 6, and 7 unless you want more practice). The first game is already done for you (on the sheet in blue), and the walkthrough is below:

Game 1:

- The start state is A012 (so s=A012). A plays action 22, so a=22. (From pile 2, take 2 sticks.)
- The reward we get here is zero (the only non-zero rewards are when someone wins). [Here I wrote "0" in the "r" box.]
- Player B will go next, and they are left with piles of 0, 1, 0 sticks, so the next state (s') is B010. [I wrote B010 in the s' box.]
- The next step here is the max/min Q[s',a'] step. Player A is playing, but they use the "min" version of the Q-learning update equation because s' is a move for Player B. So we must calculate the best action for player B from s'=B010. We look in the table at the top of the page, and for B010, there's only one action (which should be obvious), which is Q[B010, 11], which is zero right now (since nothing's been updated). So the minimum over all possible actions a' of Q[s', a'] is 0. [I wrote 0 in the max/min box.]

- The next step is to run the update. To be clear, we are using the update equation $Q[s,a] \leftarrow Q[s,a] + \alpha \left[r + \gamma \min_{a'} Q[s',a'] - Q[s,a] \right]$, and we have all the pieces:

  s = A012 a = 22 r=0 s'=B010 max/min=0 alpha=1 gamma=0.9
  And at this moment, Q[A012, 22] = 0.

  So the update is: Q[A012, 22] = 0 + (1)[0 + (0.9)(0) – 0] = 0. So I put a 0 under Game 1 for Q[A012, 11].

- So now we are in state B010 (s=B010). B plays action 11, so a=11 (from pile 1, take 1 stick).
- The reward here we will get is 1000 (because B was forced to take the last stick), so they lose, and A wins, and thus we get +1000 points. [I put 1000 in the "r" box.]
- The next state here doesn't really matter, since the game is over, but you can put A000 in the box (since it represents that "A" **would** move next if they could).
- The next step is the max/min Q[s',a'] step. Because the game is over, this is zero, since there are no future rewards from the next state. [0 in the max/min box]
- The next step is to run the update. Now we are using Player B's update equation: $Q[s,a] \leftarrow Q[s,a] + \alpha \left[r + \gamma \max_{a'} Q[s',a'] - Q[s,a] \right]$, and our pieces are:

  s=B010 a=11 r=1000 s'=A000 max/min=0 (same alpha/gamma)

  And right now, Q[B010, 11] = 0.

  So the update is Q[B010, 11] = 0 + (1)[1000 + (0.9)(0) – 0] = 1000. So I put 1000 under Game 1 for Q[B010, 11].

Player A update: $$Q[s,a] \leftarrow Q[s,a] + \alpha \left[r + \gamma \min_{a'} Q[s',a'] - Q[s,a] \right]$$

Player B update: $$Q[s,a] \leftarrow Q[s,a] + \alpha \left[r + \gamma \max_{a'} Q[s',a'] - Q[s,a] \right]$$

Player A optimal policy: $$\pi(s) = \operatorname{argmax}_a Q[s,a]$$

Player B optimal policy: $$\pi(s) = \operatorname{argmin}_a Q[s,a]$$

Q[s, a]	Init	Game 1	Game 2	Game 3	Game 4	Game 5	Game 6	Game 7	Limit
Q[A012, 11]	0								
Q[A012, 21]	0								
Q[A012, 22]	0	0							
Q[B002, 21]	0								
Q[B002, 22]	0								
Q[B011, 11]	0								
Q[B011, 21]	0								
Q[B010, 11]	0	1000							
Q[A001, 21]	0								
Q[A010, 11]	0								

	s	a	r	s'	max/min Q[s', a']		s	a	r	s'	max/min Q[s', a']
Game 1	A012	22	0	B010	0	5	A012	21			
	B010	11	1000	A000	0			11			
Game 2	A012	11						21			
		22				6	A012	21			
Game 3	A012	11						21			
		21						11			
		21				7	A012	11			
Game 4	A012	22						21			
		11						21			