Solutions Template

1 Written Questions

Write your solutions to the HW8 Written Questions in this template. You may either enter your solutions into the .tex file directly, or print out the pdf version and write your solutions by hand in the boxes provided. When you have completed all of the written questions, you should upload your solutions to Gradescope in pdf format. For the multiple choice question, shade in the box in the template document corresponding to the correct answer(s). For LaTeXusers, use ■ for shaded boxes, and don't change anything else.

1.1 Warmup Questions[14 points]

1.1.1 Value Iteration[6 points]

In this question you will carry out value iteration by hand to solve a maze, similar to what you will do in the programming part. A map of the maze is shown in the table below, where 'G' represents the goal of the agent (it's the terminal state); 'H' represents an obstacle; the zeros are the state values V(s) that are initialized to zero.

0	0	G
H	0	H
0	0	0

Table 1.1: Map of the maze

The agent can choose to move up, left, right, or down at each of the 6 states (the goal and obstacles are not valid initial states, "not moving" is not a valid action). The transitions are deterministic, so if the agent chooses to move left, the next state will be the grid to the left of the previous one. However, if it hits the wall (edge) or obstacle (H), it stays in the previous state. The agent receives a reward of -1 whenever it takes an action. The discount factor γ is 1.

1. (1 point) How many possible deterministic policies are there in this environment, including both optimal and non-optimal policies?

   number of deterministic policies = ☐

2. (3 points) Compute the state values after each round of synchronous value iteration updates on the map of the maze before convergence. For example, after the first round, the values should look like this:

-1	-1	G
H	-1	H
-1	-1	-1

Table 1.2: Value function after round 1

Note that you might not need to use all the tables provided below, if the algorithm converges within 6 rounds.

		G
H		H

Table 1.3: Round 2

		G
H		H

Table 1.4: Round 3

		G
H		H

Table 1.5: Round 4

		G
H		H

Table 1.6: Round 5

		G
H		H

Table 1.7: Round 6

		G
H		H

Table 1.8: Round 7

3. (2 points. **Select all that apply**) Which of the following changes will result in the same optimal policy as the settings above?

   ☐ The agent receives a reward of 10 when it takes an action that reaches G and receives a reward of -1 whenever it takes an action that doesn't reach G. Discount factor is 1.

   ☐ The agent receives a reward of 10 when it takes an action that reaches G and doesn't receive any reward whenever it takes an action that doesn't reach G. Discount factor is 1.

   ☐ The agent receives a reward of 10 when it takes an action that reaches G and doesn't receive any reward whenever it takes an action that doesn't reach G. Discount factor is 0.9.

   ☐ The agent receives a reward of -10 when it takes an action that reaches G and doesn't receive any reward whenever it takes an action that doesn't reach G. Discount factor is 0.9.

   ☐ None of the above

1.1.2 Q-learning [8 points]

In this question, we will practice using the Q-learning algorithm to play tic-tac-toe. Tic-tac-toe is a simple two-player game. Each player, either X (cross) or O (circle), takes turns marking a location in a 3x3 grid. The player who first succeeds in placing three of their marks in a column, a row, or a diagonal wins the game.

1	2	3
4	5	6
7	8	9

Table 1.9: tic-tac-toe board positions

We will model the game as follows: each board location corresponds to an integer between 1 and 9, illustrated in the graph above. Actions are also represented by an integer between 1 and 9. Playing action a results in marking the location a and an action a is only valid if the location a has not been marked by any of the players. We train the model by playing against an expert. The agent only receives a possibly nonzero reward when the game ends. Note a game ends when a player wins or when every location in the grid has been occupied. The reward is +1 if it wins, -1 if it loses and 0 if the game draws.

O	X	
O	O	X
		X

Table 1.10: State 1 (circle's turn)

To further simplify the question, let's say we are the circle player and it's our turn. Our goal is to try to learn the best end-game strategy given the current state of the game illustrated in table1.10. The possible actions we can take are the positions that are unmarked: $\{3, 7, 8\}$. If we select action 7, the game ends and we receive a reward of +1; if we select action 8, the expert will select action 3 to end the game and we'll receive a reward of -1; if we select action 3, the expert will respond by selecting action 7, which results in the state of the game in table1.11. In this scenario, our only possible action is 8, which ends the game and we receive a reward of 0.

O	X	O
O	O	X
X		X

Table 1.11: State 2 (circle's turn)

Suppose we apply a learning rate $\alpha = 0.01$ and discount factor $\gamma = 1$. The Q-values are initialized as:

$$Q(1, 3) = 0.6$$
$$Q(1, 7) = -0.3$$
$$Q(1, 8) = -0.5$$
$$Q(2, 8) = 0.8$$

Note: for the following questons, **Do not round you solutions**.

1. (1 point) In the first episode, the agent takes action 7, receives +1 reward, and the episode terminates. Derive the updated Q-value after this episode. Remember that given the sampled experience (s, a, r, s') of (state, action, reward, next state), the update of the Q value is:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma max_{a' \in A} Q(s', a') - Q(s,a))$$

Note if s' is the terminal state, $Q(s', a') = 0$ for all a'.

$Q(1,7) =$ ☐

2. (1 point) In the second episode, the agent takes action 8, receives a reward of -1, and the episode terminates. Derive the updated Q-value based on this episode.

$Q(1,8) =$ ☐

3. (2 points) In the third episode, the agent takes action 3, receives a reward of 0, and arrives at State 2 (1.11). It then takes action 8, receives a reward of 0, and the episode terminates. Derive the updated Q-values after each of the two experiences in this episode. Suppose we update the corresponding Q-value right after every single step.

$Q(1,3) =$ ☐ $Q(2,8) =$ ☐

4. (2 points) If we run the three episodes in cycle forever, what will be the final values of the four Q-values.

$Q(1,3) =$ ☐ $Q(1,7) =$ ☐ $Q(1,8) =$ ☐ $Q(2,8) =$ ☐

5. (2 points) What will happen if the agent adopts the greedy policy (always pick the action that has the highest current Q-value) during training? Calculate the final four Q-values in this case.

$Q(1,3) =$ ☐ $Q(1,7) =$ ☐ $Q(1,8) =$ ☐ $Q(2,8) =$ ☐

1.2 Empirical Questions[16 points]

The following questions should be completed after you work through the programming portion of this assignment.

We will ask you to solve the following two mazes using value iteration and q-learning.

Maze 1 (3x5):

```
**.*G
...*.
S*...
```

Maze 2 (8x8):

```
...*..*G
......*.
...*..*.
...*....
.*..*...
.....*.*
***...*.
S.......
```

1. (4 points) Solve the two mazes using value iteration. Report the time taken for execution, the number of iterations it took to converge, and the value function of the states marked in 1.12 and 1.13. **Please round to two decimal places**. Use discount factor $\gamma = 0.9$. We say that the algorithm converges when the change of $V(s)$ for all s is small (less than 0.001).

		(0, 2)		
(2, 0)				(2, 4)

Table 1.12: Maze 1 Map

(0, 0)							
							(6, 7)
(7, 0)							

Table 1.13: Maze 2 Map

Maze 1: time taken for execution: [ ] number of iterations: [ ]

value of (2, 0): value of (0, 2): value of (2, 4):

Maze 2: time taken for execution: number of iterations:

value of (7, 0): value of (0, 0): value of (6, 7):

2. (4 points) Solve the two mazes using Q-learning agent. Specifically, run the agent for 2000 episodes with epsilon-greedy policy. Report the time taken for execution, the average number of steps in each episode, and the value function of the states marked in 1.12 and 1.13. **Please round to two decimal places**. Use discount factor $\gamma = 0.9$, learning rate of 0.1 and epsilon $\epsilon = 0.2$.

Maze 1: time taken for execution: average steps:

value of (2, 0): value of (0, 2): value of (2, 4):

Maze 2: time taken for execution: average steps:

value of (7, 0): value of (0, 0): value of (6, 7):

3. (2 points) Examine and comment on the results for both methods. Which method runs faster? Do you see any difference between the value functions obtained? Are the optimal policies different? What do you think have caused the differences observed, if any?

4. (4 points) Now try using ϵ of 0.01 and 0.8 and compare the results with those you obtained using ϵ of 0.2. Examine and comment on the results **for Maze 2 only**: do you observe any difference? Especially examine the time taken for execution and the value functions obtained.

$\epsilon = 0.01$: time taken for execution: average steps:

value of (7, 0): value of (0, 0): value of (6, 7):

$\epsilon = 0.8$: time taken for execution: average steps:

value of (7, 0): value of (0, 0): value of (6, 7):

5. (2 points) Which method (value iteration or Q-learning) do you think is more suitable for this task of maze solving? Explain your reasoning. In what cases do you think the "worse" method here would become a better one?

6. **Collaboration Questions** After you have completed all other components of this assignment, report your answers to the collaboration policy questions detailed in the Academic Integrity Policies found here. (a) Did you receive any help whatsoever from anyone in solving this assignment? If so, include full details. (b) Did you give any help whatsoever to anyone in solving this assignment? If so, include full details? (c) Did you find or come across code that implements any part of this assignment ? If so, include full details.