Homework 2 - Question 1: State-Action Value Function and Policy Iteration

Submission by: Apoorv Agnihotri (6604679), Gaurav Niranjan (6599177), Carla López Martínez (6637484)

Question 1

Part a) Gridworld Q-Values

To find the Q-values $q_\pi(s, a)$ for the given states and actions under the equiprobable random policy, we need to consider:

- Each action has probability 1/4 under the random policy
- Reward is -1 for all transitions
- The given value function can be used to determine future state values

```
| 0   | −14 | −20 | −22 |
| −14 | −18 | −20 | −20 |
| −20 | −20 | −18 | −14 |
| −22 | −20 | −14 | 0   |
```

Our griworld states are:

```
| 0   | 1   | 2   | 3   |
| 4   | 5   | 6   | 7   |
| 8   | 9   | 10  | 11  |
| 12  | 13  | 14  | 15  |
```

For each state-action pair:

$q_\pi(11, down)$:

- Moving down from state 11 leads to state 15 (terminal state)
- Terminal state has value 0
- Q-value = Immediate reward + Value of next state
- $q_\pi(11, down)$ = -1 + 0 = -1

$q_\pi(7, down)$:

- Moving down from state 7 leads to state 11

- State 11 has value -14 according to the given value function
- $q_\pi(7, down)$ = -1 + (-14) = -15

$q_\pi(9, left)$:

- Moving left from state 9 leads to state 8
- State 8 has value -20 according to the given value function
- $q_\pi(9, left)$ = -1 + (-20) = -21

Part b) Optimal Value Function

The optimal value function v*(s) in terms of q*(s,a) is:

$v*(s) = max_a q*(s, a)$

This equation states that the optimal value of a state is equal to the maximum Q-value over all possible actions in that state.

Part c) Optimal Q-Function

The optimal Q-function q*(s,a) in terms of v*, P, and R is:

$q*(s, a) = R(s, a) + \gamma \Sigma_{s'} P(s'|s, a) v*(s')$

Where:

- R(s,a) is the reward function
- γ is the discount factor
- P(s'|s,a) is the transition probability
- v*(s') is the optimal value function of the next state

Part d) Optimal Policy

The optimal policy π *in terms of q* is:

$\pi*(a|s) = 1$ if $a = argmax_a q*(s, a)$
$\pi*(a|s) = 0$ otherwise

This defines a deterministic policy that always selects the action with the highest Q-value in each state.

Part e) Bellman Expectation Equation for Q-values

The Bellman Expectation Equation for q^π(s,a) in terms of only q values is:

$q_\pi(s, a) = R(s, a) + \gamma \Sigma_{s'} P(s'|s, a) \Sigma_{a'} \pi(a'|s') q_\pi(s', a')$

This equation expresses the Q-value of a state-action pair in terms of:

- The immediate reward R(s,a)
- The discounted future Q-values of all possible next states s' and actions a'
- Weighted by their transition probabilities P(s'|s,a)
- And the policy probabilities π(a'|s')

Question 2.2

Part a)

The start state of MazeGrid has a non-zero value after 5 iterations of value iteration. This is because for the value iteration algorithm, the value of a state is updated based on the maximum value of the successor states. In this case, the start state has a path to the goal state with a reward of 1, and the value of the goal state is propagated back to the start state through the optimal path. As the algorithm progresses, the value of the start state increases with each iteration until it converges to the optimal value. Since the start state is at some distance from the goal state, it takes a few iterations for the value to propagate back to the start state.

Part b)

The reason that the agent doesn't dare to cross the bridge is because of the noisy actions in the environment. Even if the agent takes the optimal action to cross the bridge, there is a probability that it will fall into the water due to the noise in the action. This uncertainty in the environment makes the agent cautious and prevents it from taking the optimal action that leads to the goal state. The agent prefers to go back to the left, even if it is suboptimal (this terminal state has a reward of 1), to reduce the risk of falling into the water. We can make the agent value future rewards more by increasing the discount factor γ, which would encourage the agent to take the optimal action despite the noise in the environment, but we can't increase the discount factor to be more than 1. Another way to address this issue is to reduce the noise in the environment, which would make the agent more confident in taking the optimal action to reach the goal state. For our case, we could get the agent to cross the bridge by just reducing the noise in the environment to a value of 0.02 or less compared to the default value of 0.2.

Part c)

a) Prefer the close exit (+1), risking the cliff (-10).
Achievable with a noisy of 0.02 and discount factor of 0.1.

b) Prefer the close exit (+1), but avoiding the cliff (-10).
Achievable with a noisy of 0.4 and discount factor of 0.1.

c) Prefer the distant exit (+10), risking the cliff (-10).
Achievable with a noisy of 0.02 and discount factor of 0.9.

d) Prefer the distant exit (+10), avoiding the cliff (-10).
Achievable with a noisy of 0.02 and discount factor of 1.

Part d)

On Mazegrid with the default parameters and having 100 iterations of value iterations, we get the value of the start state as 0.28. Instead we got 0.001363 and 0.002232 as the value of the start state in the first exercie running 10 and 10000 episodes. We see there's a difference between the two values, because the values we were getting in the first exercise were for a random policy (with equally probable actions in all directions). In this exercise we first do value iteration to get the optimal policy and then calculate the value of the start state. The value of the start state is higher in the case of the optimal policy because the agent is more likely to reach (quickly) the goal state (with a high reward) following the optimal policy than a random policy. The optimal policy guides the agent to take actions that lead to the goal state with higher probability, resulting in a higher value for the start state.

Part e)

To be able to avoid the terminal states, we need need a discount factor of 0, but when we have a discount factor of 0, there's no way we can make the agent prefer going up (avoiding the cliff) because both the transitions have the same q value of 0. And in our implementation we choose the action corresponding to the index 0, when we have the same q values for all the actions, which corresponds to the right action.

Question 2.3

Yes, we created a new gridworld environment called "CircularGrid". Please check it out in the code below.

```
def getCircularGrid():
  grid = [['#','#','#','#',+10],
          [' ',' ',' ','#',' '],
          [' ','#',' ','#',' '],
          [' ','#','S','#',' '],
          [' ','#','#','#',' '],
          [' ',' ',' ',' ',' ']]
  return Gridworld(grid)
```