CSE 4113 Assignment 3

Mahdi Mohd Hossain Noki
Roll 02

1 Introduction to OpenAI Gym

OpenAI Gymnasium is an API standard for reinforcement learning. It defines several environments as a generalized implementation of several popular Reinforcement Learning problems.

Following a simple code block to interact with an environment

```
import gymnasium as gym
env = gym.make("LunarLander-v2", render_mode="human")
observation, info = env.reset(seed=42)
for _ in range(1000):
   action = env.action_space.sample() # this is where you would insert your policy
   observation, reward, terminated, truncated, info = env.step(action)

   if terminated or truncated:
      observation, info = env.reset()

env.close()
```

env in the provided snippet define and create an environment.

Important methods in the given snippets are described below

1. **make()**: prepares an environment with its action and observation spaces. Difference environments have different action and observation spaces. Returns an environment
2. **reset()**: Resets the environment for the first observation of an environment. Environments with the same seed always resets to the same position. Returns an observation of the state and info about the environment
3. **sample()**: Returns a random action from the action space of the environment
4. **step(action)**: From the current state s, performs and action and moves to the next state s'. Returns a tuple containing the observation: the next state moved onto, reward: the reward for performing the action, terminated: a condition checking whether the current environment in terminated - either by success or fail, truncated: a value that indicated too many iterations have been performed and environment can be ended now and info: information about the environment

2 Exploring Markov Decision Processes (MDP)

The markov decision process involves an initial state s_i, a transition function $T(s, a, s')$ indicating the transition from state s to state s' with action a and a reward function $r : S \to \mathbb{R}$

we pick the **frozenlake-v1** environment for this assignment.

The environment plays out with a player starting at an initial position (0, 0) in an nxn map. The goal is to reach a reward at position (n-1, n-1). The map is a frozen lake with holes in it. If the player moves to a hole, the game ends without any reward.

1. **observation space**: The current state can be expressed with the player's current position. It is represented as an integer - if the player is at position (i, j) in an nxn map the state is $(n-1)*i+j$. So the observation space is $1, 2, \ldots, n*n-1$
2. **action space**: from any position the player can try to take the following actions
   (a) 0: Move left
   (b) 1: Move down
   (c) 2: Move right
   (d) 3: Move up
3. **reward**: The goal is to reach the end. Rewards are distributed as
   (a) Reach goal: +1

(b) Reach hole: 0

(c) Reach frozen: 0

4. **Transition**: Since the lake is frozen, when the player tries to move to a direction it moves to the direction with 1/3 probability. It moves to two perpendicular directions with a 1/3 probability as well

3 Implementing Value Iteration

For the frozenlake environment we implement the value iteration algorithm. The optimal policy derived from value iteration should theoretically perform well, but its effectiveness may depend on various factors.

1. **Efficiency of Value Iteration**: Value iteration typically converges to the optimal policy relatively quickly, especially in small grid worlds like FrozenLake. However, the time complexity of value iteration grows quadratically with the size of the grid, so larger grid sizes may require more computational resources and time to converge.

2. **Optimality of the Policy**: The policy derived from value iteration aims to maximize the expected cumulative reward over time. In the case of FrozenLake, this means that the policy will try to guide the player from the starting position to the goal while avoiding holes. Since value iteration accounts for the stochastic nature of the environment (i.e., the uncertain movements due to the probabilities associated with each action), the resulting policy should consider these probabilities when making decisions.

3. **Handling Uncertainty**: The stochastic nature of the movement probabilities in FrozenLake means that even with the optimal policy, there's still a chance of falling into a hole. However, the optimal policy should minimize this risk by favoring safer paths with higher expected rewards. The policy will likely direct the player towards paths with fewer holes and closer proximity to the goal.

4. **Performance Evaluation**: To evaluate the performance of the policy derived from value iteration, you can simulate the player's movements in the game environment using the policy and measure various metrics such as the average time to reach the goal, the percentage of successful runs, and the average cumulative reward obtained. By comparing these metrics with alternative policies or random strategies, you can assess the effectiveness of the derived policy.

5. **Robustness and Generalization**: Value iteration provides a deterministic policy based on the specific rewards and transition probabilities in the environment. While this policy may perform well within the training environment (i.e., the given FrozenLake grid), its robustness and generalization to variations or unseen environments are not guaranteed. Changes in the environment's dynamics or reward structure may require retraining or fine-tuning the policy to adapt effectively.

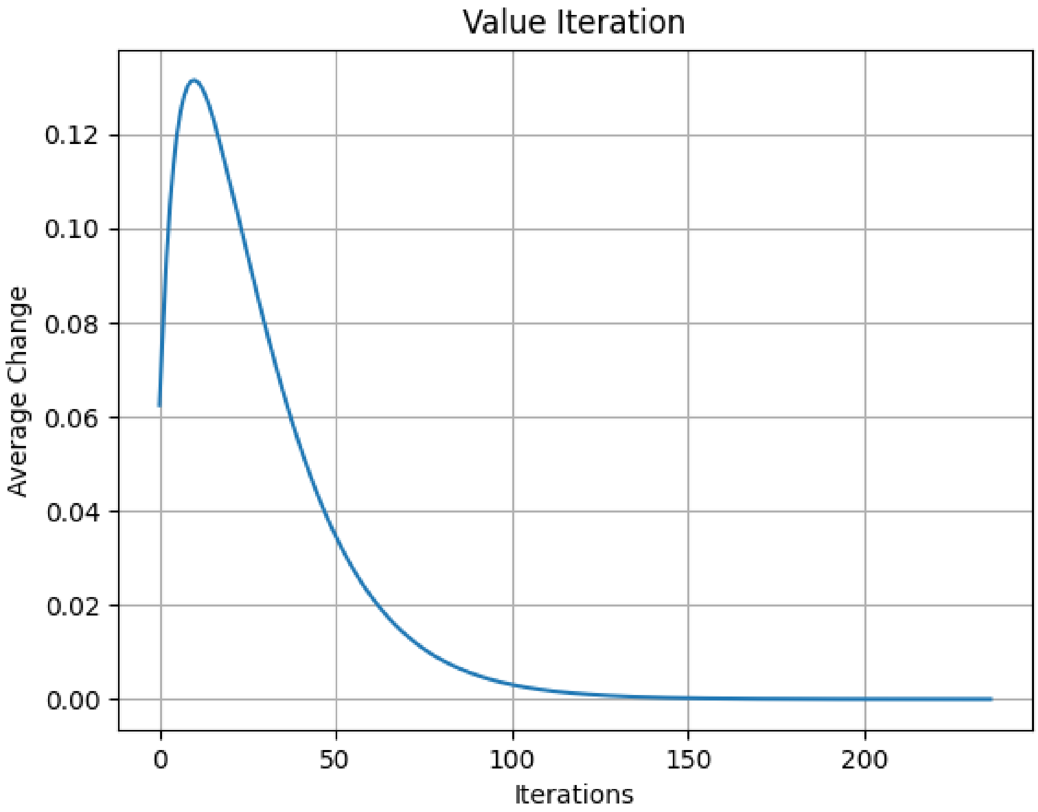

Figure 1: Value Iteration Convergence

4 Implementing Q learning

4.1 Learning Curve

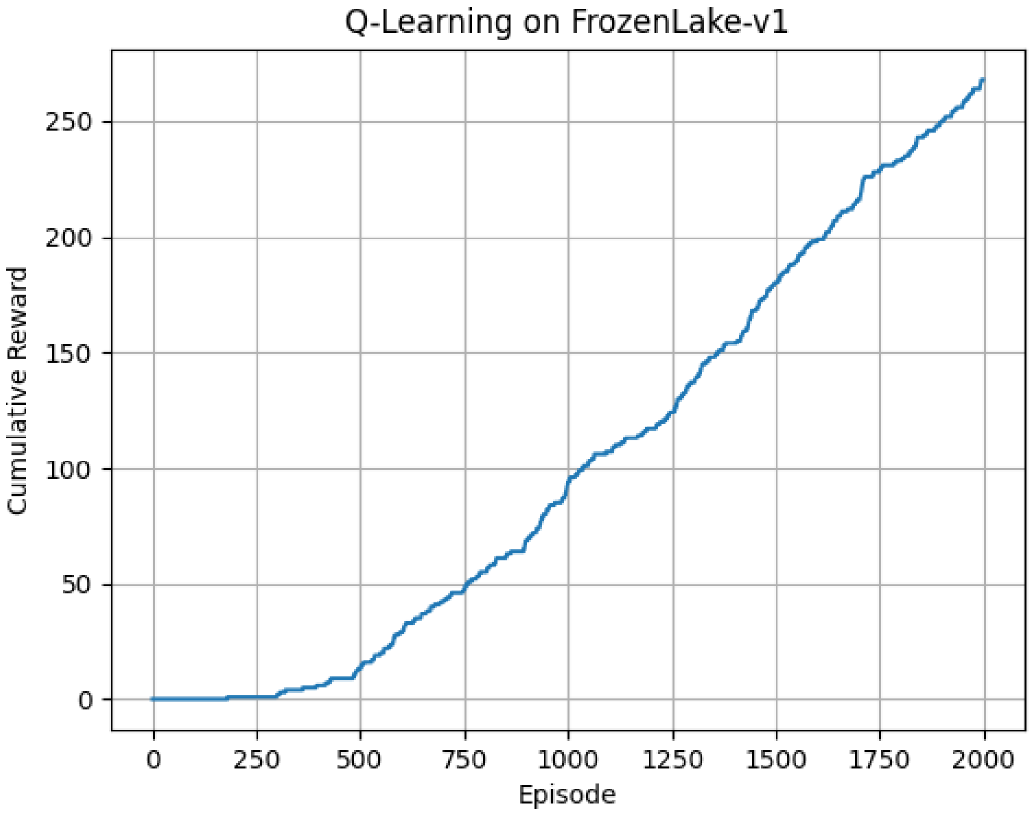

Figure 2: Q-learning reward over time (cumulative)

4.2 Performance Comparison

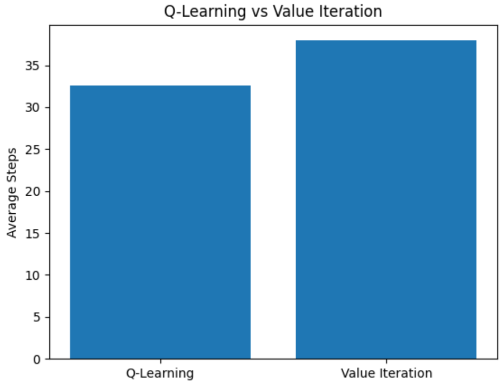

Figure 3: Average Number of steps taken to reach Goal

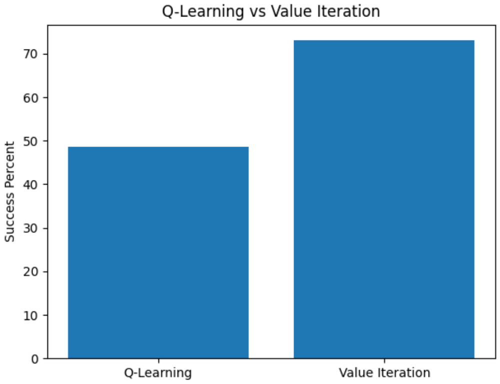

Figure 4: Average Number of Successful runs

As can be seen in the comparisons, Q-learning policy fails the game more times than value iteration algorithm, but takes less steps to reach goal when it succeeds