HW6 Intelligent Systems

Reinforcement Learning

Fardin Abbasi 810199456

School of Electrical & Computer Engineering

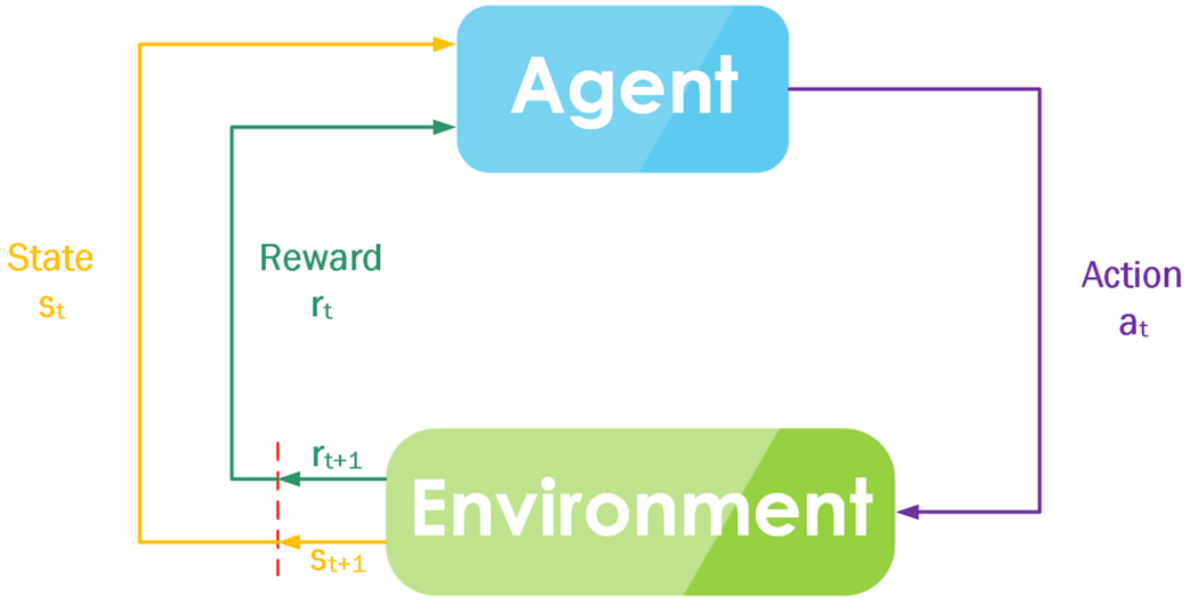
College of Engineering

University of Tehran

Questions

Implementation of Q-Learning

Part 1: Basic Concepts of RL



In reinforcement learning, there are a few key concepts to familiarize with:

- The **agent** is the ML algorithm (or the autonomous system)
- The **environment** is the adaptive problem space with attributes such as variables, boundary values, rules, and valid actions
- The **action** is a step that the RL agent takes to navigate the environment
- The **state** is the environment at a given point in time
- The **reward** is the positive, negative, or zero value—in other words, the reward or punishment—for taking an action
- A **policy** is agent's behavior from which take actions based on the observed state

Part 2: Implementation

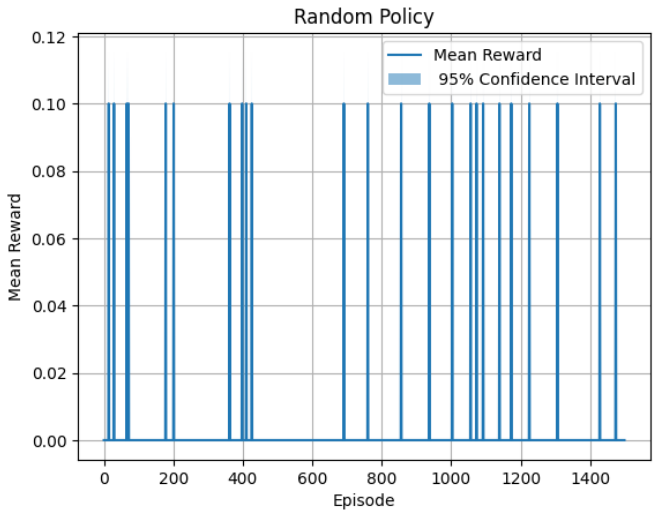
A: Environment

When *is_slippery=True* agent's chosen action is not necessarily the going direction, in other words environment's response to agent's action is stochastically.

In the slippery variant, the action the agent takes only has **33% chance of succeeding.**

B: Random Agent Implementation

Mean Reward for a random agent after 10 runs of 1500 episodes is depicted below:



Behavior of the random agent during one episode is recorded in *random.gif*

C: Q-Learning Algorithm

The pseudocode of Q-Learning policy algorithm is written as follows:

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Exploration allows an agent to improve its current knowledge about each action, hopefully leading to long-term benefit. Improving the accuracy of the estimated action-values, enables an agent to make more informed decisions in the future.

Exploitation on the other hand, chooses the greedy action to get the most reward by exploiting the agent's current action-value estimates. But by being greedy with respect to action-value estimates, may not actually get the most reward and lead to sub-optimal behavior.

Epsilon-Greedy is a simple method to balance exploration and exploitation by choosing between exploration and exploitation randomly.
The epsilon-greedy, where epsilon refers to the probability of choosing to explore, exploits most of the time with a small chance of exploring.

$$a_t \begin{cases} = \max Q_t(a), & with\ prob\ 1 - \epsilon \\ \neq \max Q_t(a), & with\ prob\ \epsilon \end{cases}$$

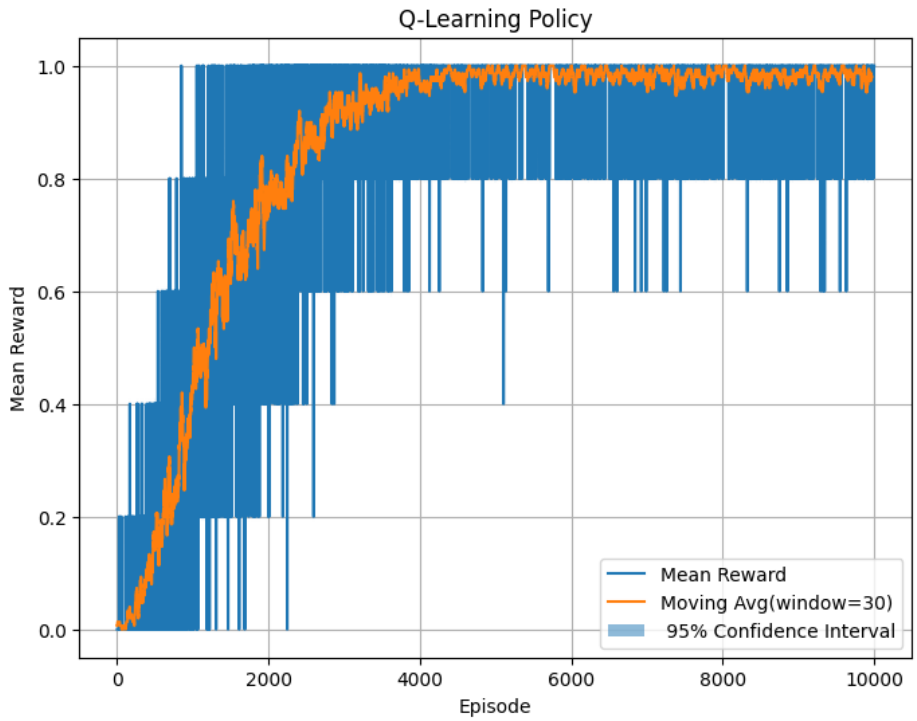
In reinforcement learning, our restaurant choosing dilemma is known as the **exploration-exploitation tradeoff**.

The Epsilon-Greedy Algorithm makes use of the exploration-exploitation tradeoff by

- instructing the agent to explore (i.e. choose a random option with probability epsilon)
- and exploit (i.e. choose the option which so far seems to be the best) the remainder of the time.

By choosing higher ϵ in the beginning of the learning process, we ensure agent has adequate exploration, then by decaying ϵ over time agent starts to exploit what it has learned.

Training results of a **Q-Learning policy** in the *frozen-lake* environment with *is_slippery=False* is depicted below:



As shown, the learned agent performs much better in the environment in compare with random agent.

Behavior of the agent with Q-Learning policy during one episode is recorded in *Q_learning.gif*