Defining & Solving RL Environments

Sivashanmugam Saravanan Omprakash

Department of Mechanical & Aerospace Engineering

University at Buffalo (SUNY)

Buffalo, NY 14260

saravan3@buffalo.edu

Abstract

The goal is to acquire experience in defining and solving RL environments, following OpenAI Gym standards. First, we focus on defining deterministic & stochastic environments that are based on Markov Decision Process (MDP).

1. Defining RL Environments

Theory

Everything the *agent* interacts with, comprising everything outside the agent, is called the *environment*. These interact continually, the agent selecting actions and the environment responding to these actions and presenting new situations to the agent [1].

Report

1. *Describe the deterministic & stochastic environments, which were defined (set of actions/states/rewards, main objective, etc).*

	For Deterministic & Stochastic Environment	
	Number	**Set**
Action	4	{Left, Up, Right, Down}
States	16	{(0, 0), (0, 1), (0, 2), (0, 3), (1, 0), (1, 1), (1, 2), (1, 3), (2, 0), (2, 1), (2, 2), (2, 3), (3, 0), (3, 1), (3, 2), (3, 3)}
Rewards	5	{-5, 1, 3, 5, 10}

2. *Provide visualizations of your environment.*

Figure 1 shows the layout of the environment. There are 9 states, 4 actions allowed, & 4 reward signals placed in the environment.

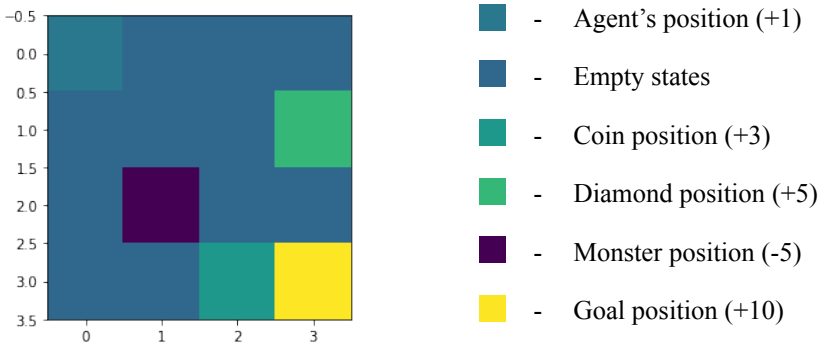

Figure 1: Default environment layout

3. How did you define the stochastic environment?

Stochasticity for the environment has been defined under the step function by choosing a sample from a uniform distribution between 0.0 & 1.0. If the random sample is greater than 0.75 the given action is executed, else an action from the action space is chosen at random.

4. What is the difference the deterministic & stochastic environment?

Deterministic Environment	Stochastic Environment
$P(s', r \mid s, a) = \{0,1\}$	$\sum_{s',r} P(s', r \mid s, a) = 1$
Outcome can be determined based on state	Outcome cannot be determined for certain based on state
Example: Making a move in Chess	**Example:** Rolling a die

*5. **Safety in AI:** Write a brief review explaining how you ensure the safety of your environments.*

In order to keep the agent within the deterministic/stochastic grid environment, the actions that makes the agent to exit the environment is clipped to the min & max limits of the grid. This means that, even if such an action is executed, the agent remains in its current state.

2. Applying tabular methods

SARSA method

SARSA, short for State-Action-Reward-State-Action, is an on-policy algorithm and this means it learns action values relative to the policy it follows. Agent that uses this algorithm interacts with the environment and updates the policy based on action taken.

The update function for the SARSA algorithm is as follows,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

The value for the state-action is updated by an error, adjusted by the learning rate (α). Q-value represent the possible reward received in the next step for taking action a in state s, plus the discounted (γ) future reward received from the next state-action ($Q(s_{t+1}, a_{t+1})$) observation.

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

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

Figure 2: SARSA Algorithm

Q-Learning method

Q-learning, Q meaning quality, is an off-policy algorithm and this means it learns action values relative to the greedy policy. This algorithm finds an optimal policy in the sense of maximizing the expected value of the total reward over any and all successive steps, starting from the current state.

The update function for the Q-learning is as follows,

$$Q^{\text{new}}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

After some steps into the future, the agent will decide some next step for which the weight for that particular step is calculated with a discount factor (γ). The core fo the algorithm is a Bellamn equation as a simple value iteration update, using the weighted average of the old value and the new information.

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

Figure 3: Q-learning Algorithm

Report

Agents in Deterministic Environment

Note: Performance metrics on the left are of Q-learning agent and on the right is the SARSA agent.

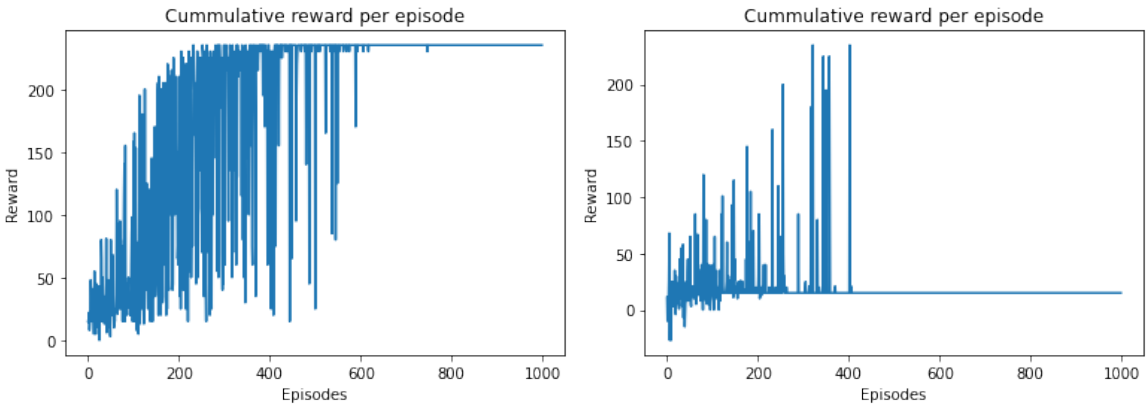

Figure 4: Q-agent vs SARSA agent - Cumulative reward per episode

Cumulative reward per episode - The cumulative reward per episode return by the Q-agent tends to linearly increase. Whereas in case of the SARSA agent, the agent starts to return linear increment in cumulative reward but as we go through the iterations the cumulative reward start to hover at around 15.

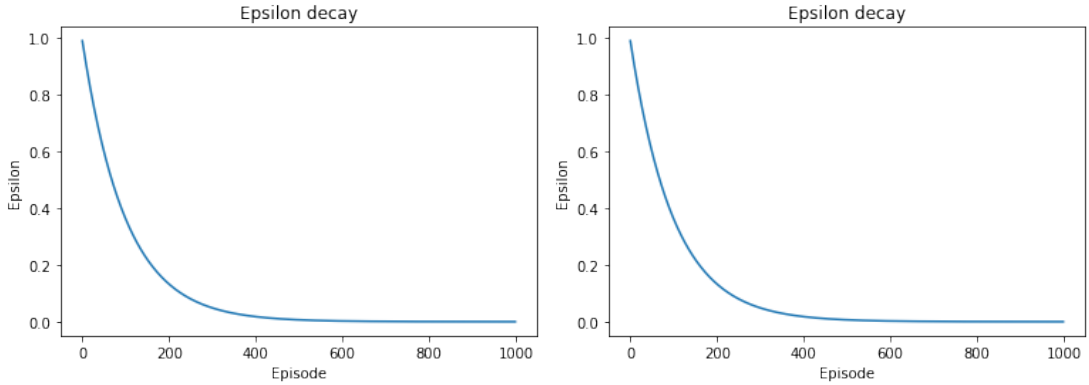

Figure 5: Q-agent vs SARSA agent - Epsilon decay

Epsilon decay - The epsilon (ϵ) value dictates the balance between the Exploration & Exploitation modes of the agent. The epsilon decay set to reduce by 1% inevitably returns identical decay for both the agents

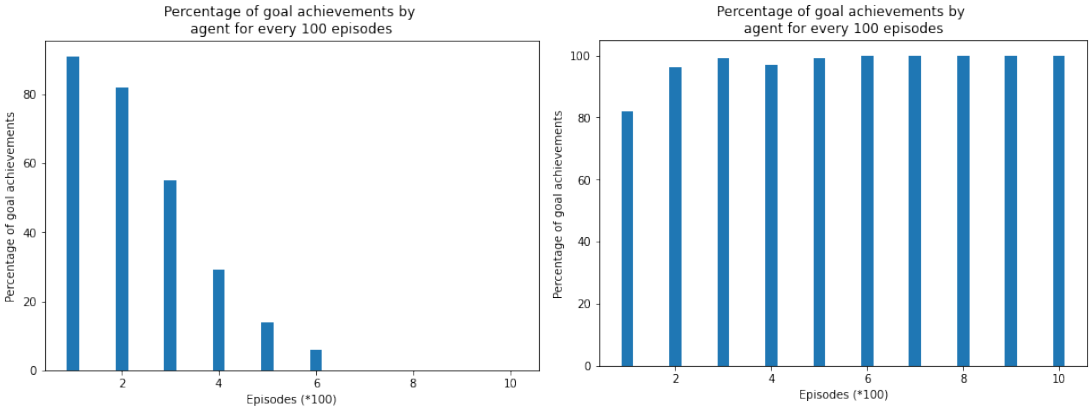

Figure 6: Q-agent vs SARSA agent - Percentage of goal achievement by agent for every 100 episodes

Percentage of goal achievement by agent per 100 episodes - In this metric, the SARSA agent repeatedly performs better that the Q-agent. The SARSA agent also reaches the goal all the time as we irate updating the model.

Table 1: Q-agent vs SARSA - Avg. time-step per episode and avg. penalty metrics

	Q-agent	**SARSA agent**
Average number of time steps per episode	41.745	11.3
Average number of times the agent receives penalty per episode	0.067	0.102

Average number of time steps per episode - The Q-agent tends to reach the goal after spending more time steps when compared to the SARSA agent.

Average number of times the agent receives penalty per episode - The SARSA agent seems to get penalized more often when compared to the Q-agent in the given deterministic environment.

Agents in Stochastic Environment

Note: Performance metrics on the left are of !-learning agent and on the right is the SARSA agent

Cumulative reward per episode - Again, in case of Q-agent, the cumulative reward linearly increases and in the case of SARSA agent the rewards start to stagnate as we iterate through the episodes. One thing to note is the

turbulence seen through the episodes. This is because of the uncertainty provided by the environment. Furthermore, given the fact that the epsilon value set to 1.0, the exploration is always preferred.

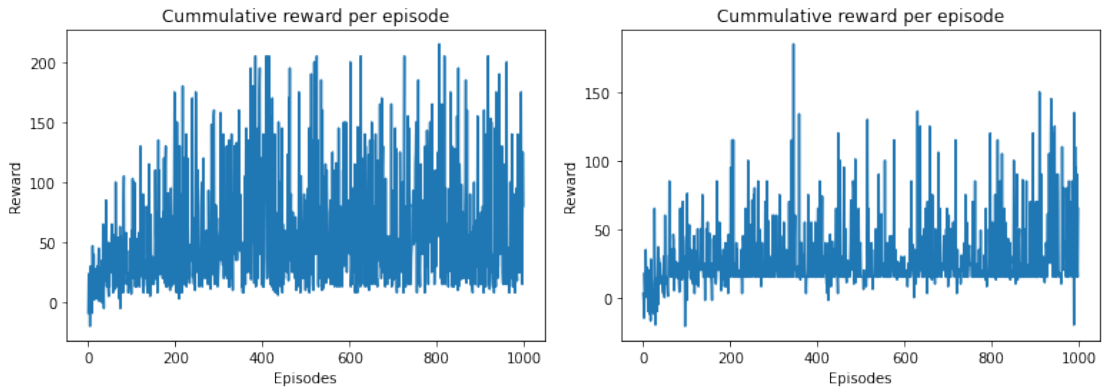

Figure 7: Q-agent vs SARSA agent - Cumulative reward per episode

Epsion decay - As seen in the deterministic environment the rate of epsilon decay remains the same in case of either of the agents.

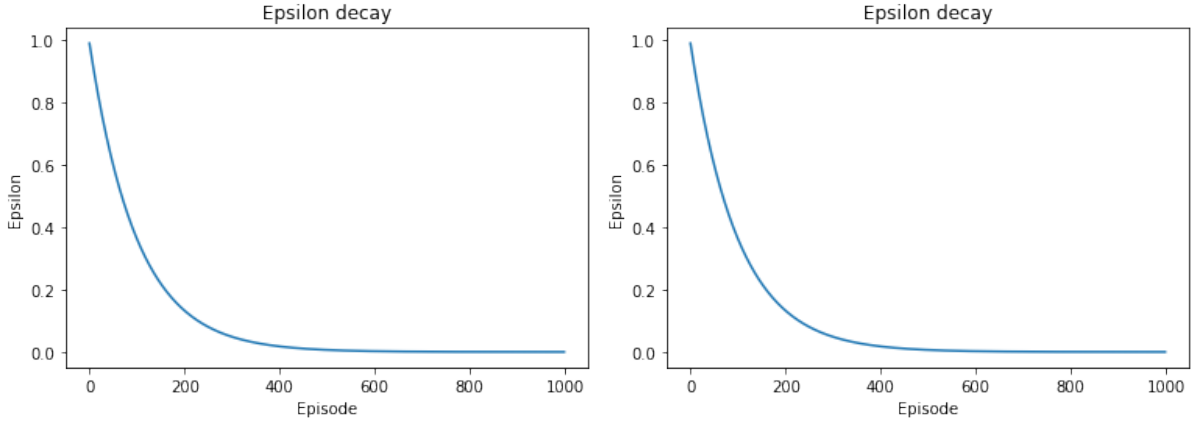

Figure 8: Q-agent vs SARSA agent - Epsilon Decay

Percentage of goal achievement by agent per 100 episodes - Owing to the inherent uncertainty of the environment it can be observed that either of the agents now have a higher chance of reaching the goal.

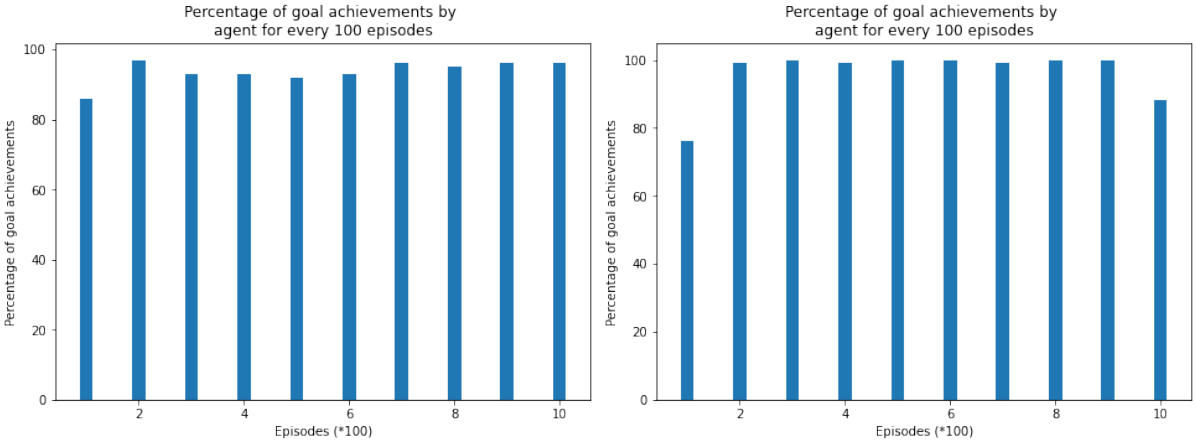

Figure 9: Q-agent vs SARSA agent - Percentage of goal achievement by the agent for every 100 episodes

Average number of time steps per episode - The Q-agent tends to reach the goal after spending more time steps when compared to the SARSA agent, but only marginally.

Average number of times the agent receives penalty per episode - The SARSA agent seems to get penalized more often when compared to the Q-agent in the given deterministic environment. But, the figures are considerably more than what we observed with in the deterministic environments. And this is because when the greedy action is chosen, the penalizing state is avoided, but when a random action is chosen the agent reaches the penalizing state inevitably.

Table 2: Q-agent vs SARSA - Avg. time-step per episode and avg. penalty metrics

	Q-agent	**SARSA agent**
Average number of time steps per episode	20.859	19.059
Average number of times the agent receives penalty per episode	0.117	0.272

Hyperparameter Tuning

Hyper-parameter Tuning (or) Hyper-parameter Optimization is the method of choosing a set of optimal hyper-parameters for a learning algorithm.

Table 3: Hyperparameter Tuning

Parameters	Discount rate (γ)	Number of episodes (k)
Values	{0.9, 0.66, 0.33}	{1000, 800, 600}

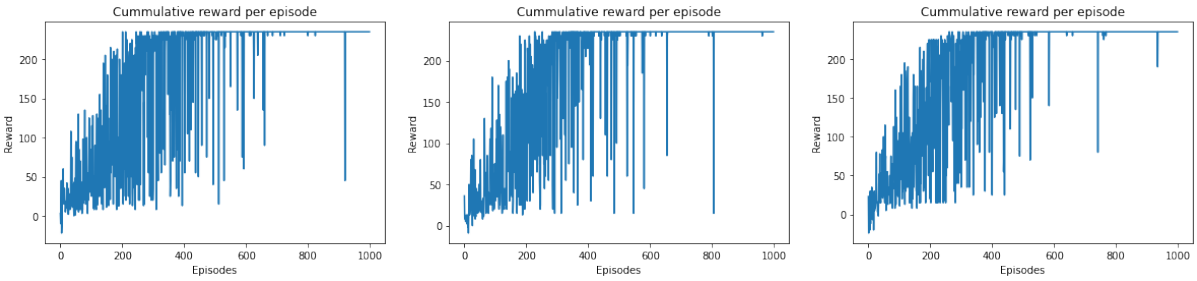

Figure 10: Q-agent in deterministic environment with 3 states of tune for the discount factor (γ)

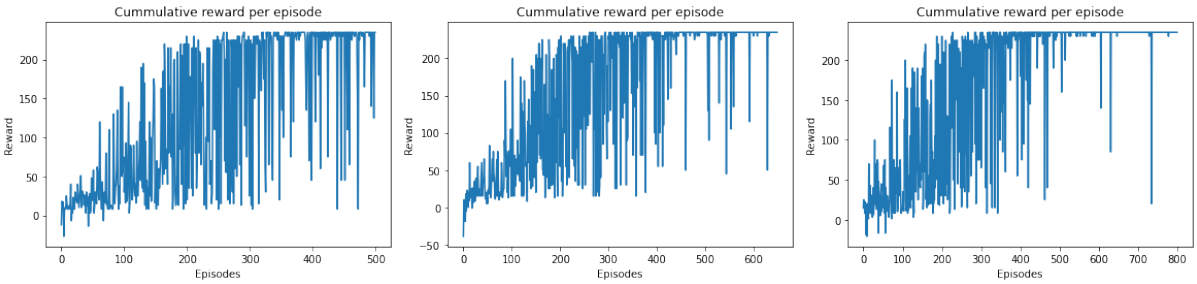

Figure 11: Q-agent in deterministic environment with 3 states of tuna for the number of episodes (k)

3. References

1. Reinforcement Learning: An Introduction by Richard S. Sutton & Andrew G. Barto - http://incompleteideas.net/book/RLbook2020.pdf
2. Q-learning [Wikipedia] - https://en.wikipedia.org/wiki/Q-learning
3. State-Action-Reward-State-Action (SARSA) [Wikipedia] - https://en.wikipedia.org/wiki/State–action–reward–state–action

4. On-Policy vs Off-Policy in Reinforcement Learning - https://leimao.github.io/blog/RL-On-Policy-VS-Off-Policy/
5. Introduction to Various Reinforcement Learning Algorithms. Part I (Q-Learning, SARSA, DQN, DDPG) - https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287
6. Summary of Tabular Methods in Reinforcement Learning - https://towardsdatascience.com/summary-of-tabular-methods-in-reinforcement-learning-39d653e904af