Reinforcement Learning

Table of Contents

Section 1: Markov Decision Process (MDPs)

- Intro to MDPs
- Policies and Value Function
- Learning Optimal Policies and Value Function

Section 2: Q-Learning

- Intro to Q-Learning
- Exploration v/s Exploitation
- Intro to Deep Q-Learning

RL and RRC

Discussion and Questions

Markov Decision Process

MDP Notations

In an MDP, we have a set of states $\boldsymbol{S}$, a set of actions $\boldsymbol{A}$, and a set of rewards $\boldsymbol{R}$. We'll assume that each of these sets has a finite number of elements.

At each time step $t = 0, 1, 2, \cdots$, the agent receives some representation of the environment's state $S_t \in \boldsymbol{S}$. Based on this state, the agent selects an action $A_t \in \boldsymbol{A}$. This gives us the state-action pair (S_t, A_t).

Time is then incremented to the next time step $t + 1$, and the environment is transitioned to a new state $S_{t+1} \in \boldsymbol{S}$. At this time, the agent receives a numerical reward $R_{t+1} \in \boldsymbol{R}$ for the action A_t taken from state S_t.

We can think of the process of receiving a reward as an arbitrary function f that maps state-action pairs to rewards. At each time t, we have

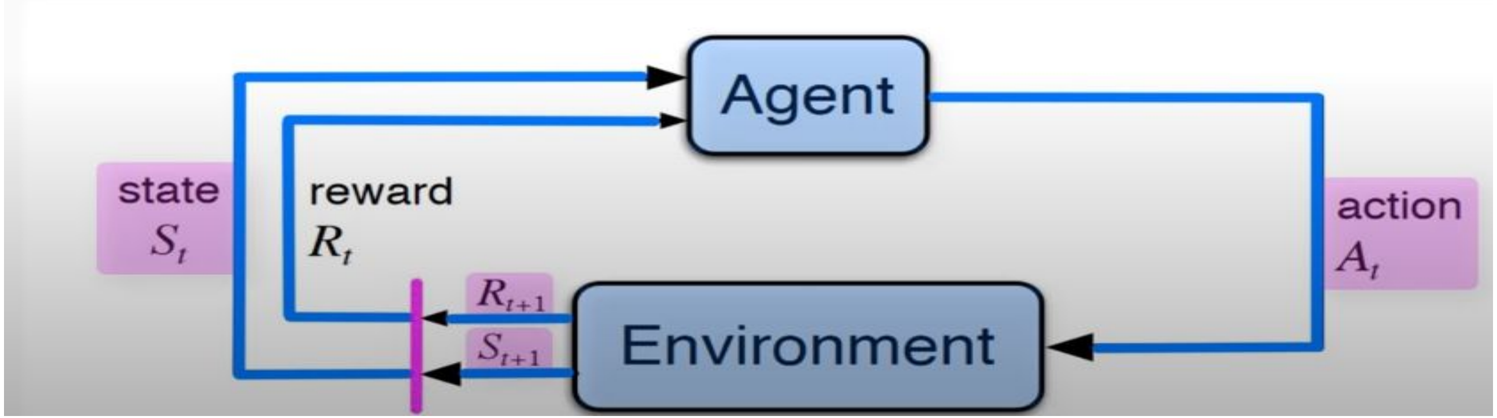
$$f(S_t, A_t) = R_{t+1}.$$

MDP Notations

The trajectory representing the sequential process of selecting an action from a state, transitioning to a new state, and receiving a reward can be represented as

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \cdots$$

This diagram nicely illustrates this entire idea.



Transition Probabilities:

For all $s' \in \boldsymbol{S}$, $s \in \boldsymbol{S}$, $r \in \boldsymbol{R}$, and $a \in \boldsymbol{A}(s)$, we define the probability of the transition to state s' with reward r from taking action a in state s as:

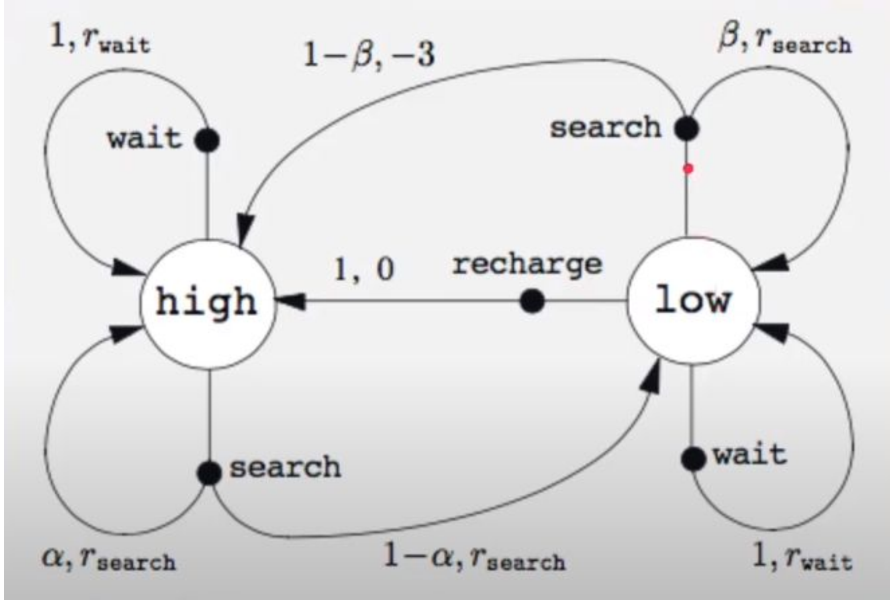
$$p(s', r \mid s, a) = \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}.$$

$$p(s' \mid s, a) \doteq \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r \mid s, a).$$

$$r(s, a) \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a),$$

$$r(s, a, s') \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}.$$

MDP Example : Recycling Robot



s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	r_{search}
high	search	low	$1 - \alpha$	r_{search}
low	search	high	$1 - \beta$	-3
low	search	low	β	r_{search}
high	wait	high	1	r_{wait}
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	r_{wait}
low	recharge	high	1	0
low	recharge	low	0	-

Expected Return : What drives an RL agent

For now, we can think of the return simply as the sum of future rewards. Mathematically, we define the return G at time t as

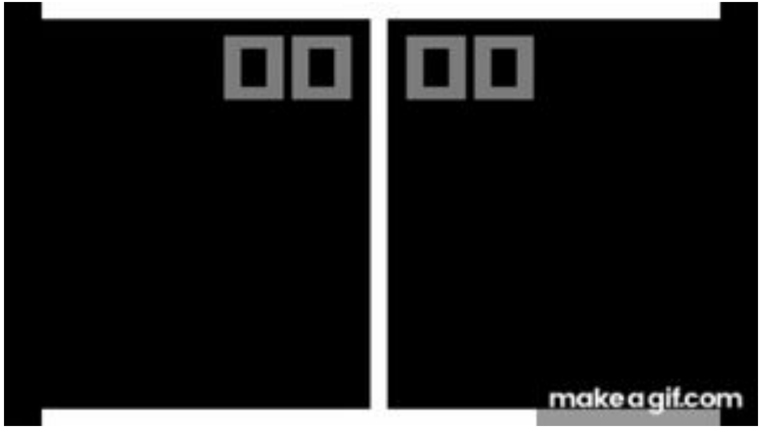
$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T,$$

where T is the final time step.

It is the agent's goal to maximize the expected return of rewards.

Types of Tasks

Episodic Task



Continuous Task

Discounted Return and Consistency Condition

To define the discounted return, we first define the discount rate, γ, to be a number between 0 and 1. The discount rate will be the rate for which we discount future rewards and will determine the present value of future rewards. With this, we define the *discounted return* as

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \\ &= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \end{aligned}$$

Now, check out this relationship below showing how returns at successive time steps are related to each other. We'll make use of this relationship later.

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+3} + \cdots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+3} + \cdots \right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Bored? Let's see AI learning to play Pong from Pixels!

Policies

When speaking about policies, formally we say that an agent "follows a policy." For example, if an agent follows policy π at time t, then $\pi(a|s)$ is the probability that $A_t = a$ if $S_t = s$. This means that, at time t, under policy π, the probability of taking action a in state s is $\pi(a|s)$.

Note that, for each state $s \in \boldsymbol{S}$, π is a probability distribution over $a \in \boldsymbol{A}(s)$.

Value Function

- Functions of state / state-action pairs which estimates how good is to be for the agent be in a state or perform an action in a state.
- Value function provides the expected return. The expected return depends on the way agent acts which is influenced by the policy. Hence, there is a relation of policy and value function.
- There are two types of Value function:
    - State Value Function (Based on state)
    - Action Value Function (Based on state-action pair)

State Value Function

The *state-value function* for policy π, denoted as v_π, tells us how good any given state is for an agent following policy π. In other words, it gives us *the value of a state* under π.

Formally, the value of state s under policy π is the expected return from starting from state s at time t and following policy π thereafter. Mathematically we define $v_\pi(s)$ as

$$\begin{aligned} v_\pi(s) &= E\left[G_t \mid S_t = s\right] \\ &= E\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]. \end{aligned}$$

Action Value Function

Similarly, the *action-value function* for policy π, denoted as q_π, tells us how good it is for the agent to take any given action from a given state while following following policy π. In other words, it gives us *the value of an action* under π.

Formally, the value of action a in state s under policy π is the expected return from starting from state s at time t, taking action a, and following policy π thereafter. Mathematically, we define $q_\pi(s, a)$ as

Q-value

$$q_\pi(s,a) = E[G_t \mid S_t = s, A_t = a]$$
$$= E\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right].$$

Q-function

Too many equations? Another interesting video

Optimal Policy

In terms of return, a policy π is considered to be better than or the same as policy π' if the expected return of π is greater than or equal to the expected return of π' for all states. In other words,

$$\pi \geq \pi' \text{ if and only if } v_\pi(s) \geq v_{\pi'}(s) \text{ for all } s \in \boldsymbol{S}.$$

Remember, $v_\pi(s)$ gives the expected return for starting in state s and following π thereafter. A policy that is better than or at least the same as all other policies is called the *optimal policy*.

Optimal State Value Function

The optimal policy has an associated *optimal* state-value function. Recall, we covered state-value functions in detail last time. We denote the optimal state-value function as v_* and define as

$$\boxed{v_*(s)} = \max_{\pi} v_\pi(s)$$

for all $s \in \boldsymbol{S}$. In other words, v_* gives the largest expected return achievable by any policy π for each state.

Optimal Action Value Function

Similarly, the optimal policy has an *optimal* action-value function, or *optimal* Q-function, which we denote as q_* and define as

$$\boxed{q_*\left(s,a\right) = \max_{\pi} q_\pi\left(s,a\right)}$$

for all $s \in \boldsymbol{S}$ and $a \in \boldsymbol{A}\left(s\right)$. In other words, q_* gives the largest expected return achievable by any policy π for each possible state-action pair.

Bellman Optimality Equation

One fundamental property of q_* is that it must satisfy the following equation.

$$q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right]$$

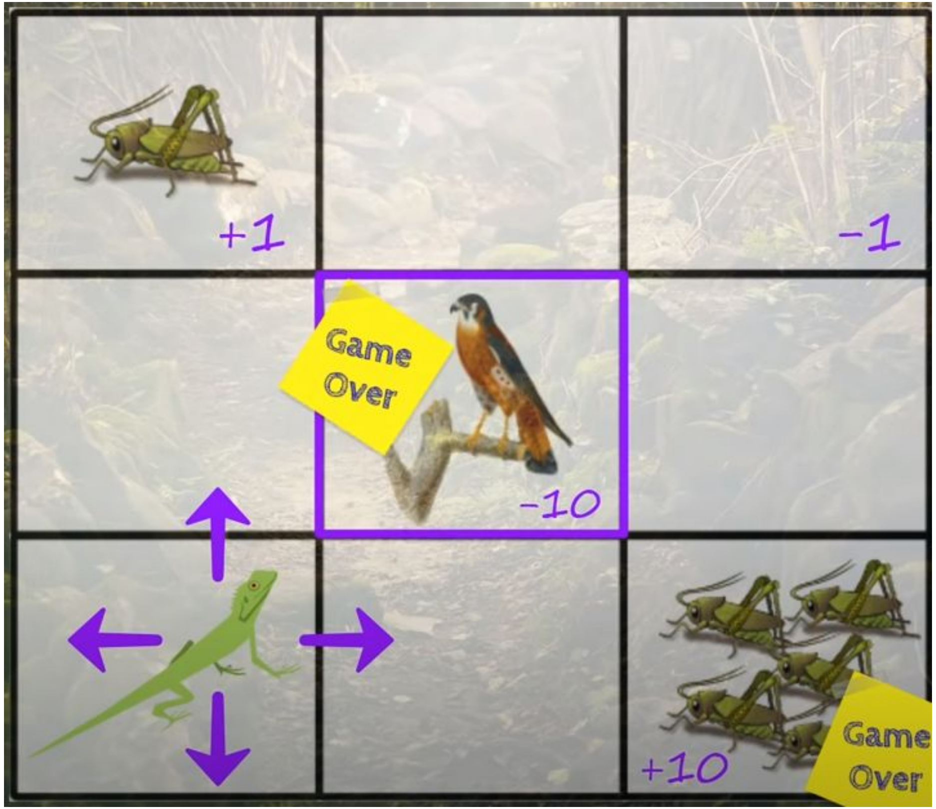
This is called the *Bellman optimality equation*. It states that, for any state-action pair (s,a) at time t, the expected return from starting in state s, selecting action a and following the optimal policy thereafter (AKA *the Q-value* of this pair) is going to be the expected reward we get from taking action a in state s, which is R_{t+1}, plus the *maximum* expected discounted return that can be achieved from any possible next state-action pair (s',a').

Q-Learning

- Iteratively updates the Q values using the Bellman Equation until the Q-function converges to the optimal Q-function.
- i.e. Bellman optimality equation:

$$\boxed{q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right]}$$

Q-Learning Example



1. At start of the game, the lizard has no idea of how good any given action from any given state. Therefore, Q-Values for each state-action pair would all be zero.

2. Throughout the game, Q-values would be iteratively updated using Value iteration.

Exploration v/s Exploitation : Epsilon Greedy Strategy

- At start we set, Exploration Rate = 1. At subsequent time-step we decay the Exploration Rate.
- To select to explore or exploit, we select a random number between 0 and 1, check if its > or < than exploration rate.
- If it's greater: explore, else exploit.

Updating Q-Value

To update the Q-value for the action of moving right taken from the previous state, we use the Bellman equation that we highlighted previously:

$$q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right]$$

We want to make the Q-value for the given state-action pair as close as we can to the right hand side of the Bellman equation so that the Q-value will eventually converge to the optimal Q-value q_*.

$$q_*(s,a) - q(s,a) = loss$$

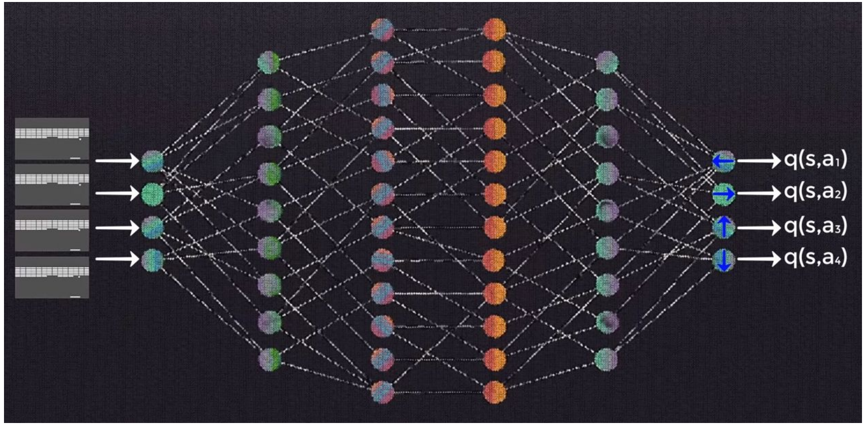
$$E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right] - E\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right] = loss$$

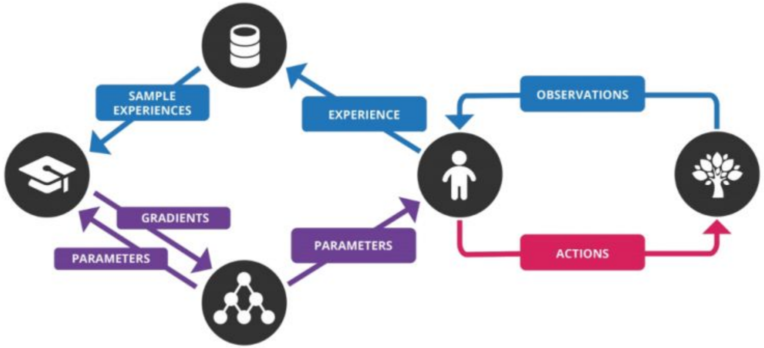
The formula for calculating the new Q-value for state-action pair (s,a) at time t is this:

$$q^{new}(s,a) = (1-\alpha)\underbrace{q(s,a)}_{\text{old value}} + \alpha\overbrace{\left(R_{t+1} + \gamma \max_{a'} q(s',a')\right)}^{\text{learned value}}$$

Intro to Deep Q-Learning

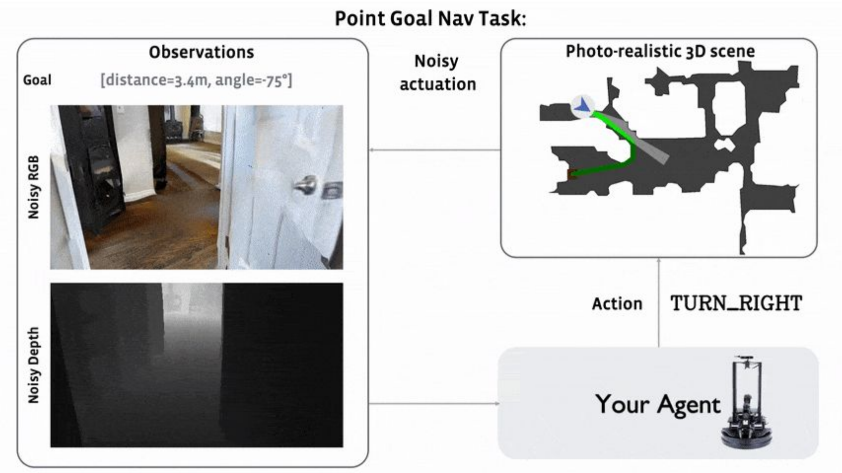
- Neural network learn the Q-function to estimate each Q-value state-action pair.



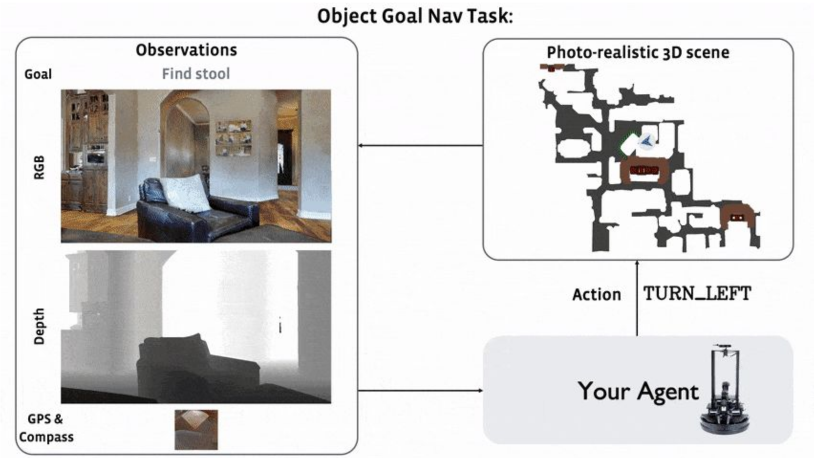


RL and RRC

Image Navigation / Visual Servoing



Object Navigation



Thank You. Questions?

Pushkal
katarapushkal@gmail.com