Reinforcement Learning

Methods and Applications

Aman Arora

Supervised by: Prof. Vikas Vikram Singh
Department of Mathematics
Indian Institute of Technology, Delhi

March 26, 2021

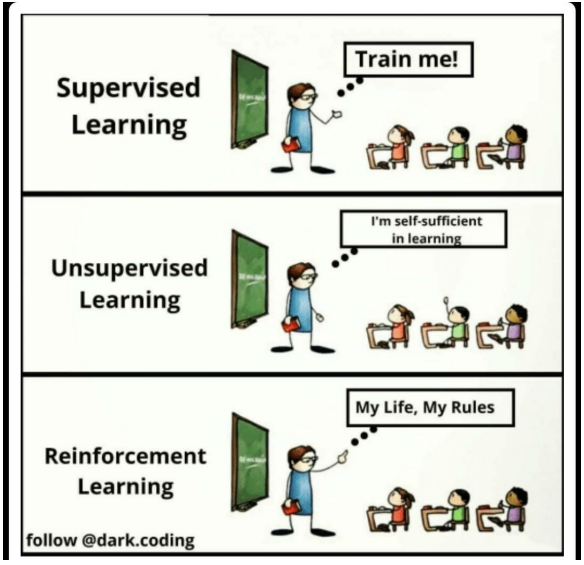
Overview

1. Introduction to RL
2. Elements of RL
3. Finite Markov Decision Processes
   - Markov Processes
   - Markov Reward Processes
   - Markov Decision Processes
   - Bellman expectation and optimality equations
4. Future Work

What is RL?

- Reinforcement learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward.
- Two crucial characteristics:
  - trial-and-error search
  - delayed reward

RL as a third paradigm of ML



Exploration vs Exploitation

- Challenge: Trade-off between *exploration* and *exploitation*
- Dilemma:
  - Prefer actions that are effective in producing reward.
  - Explore new paths to make better action selections.
- Reaching balance between exploration and exploitation is still and unresolved issue in RL.

Elements of RL

A *reward* R_t is a scalar feedback signal that indicates how well and agent is doing at step t. The agent's job is to maximise the cumulative reward.

Reward Hypothesis

All goals can be described by the maximisation of expected cumulative reward.

Elements of RL

- **Policy**: agent's behaviour function
  - Deterministic policy: $a = \pi(s)$
  - Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$
- **Value function**: how good is each state and/or action.
  - a *value function* specifies what is good in the long run
  - the *value* of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state
- **Model**: agent's representation of the environment
  - Given a state and action, the model might predict the resultant next state and next reward.
  - $\mathcal{P}$ predicts the next state
  - $\mathcal{R}$ predicts the next (immediate) reward, e.g.

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a]$$
$$\mathcal{R}^a_s = \mathbb{E}[R_{t+1}|S_t = s, A_t = a]$$

Learning vs Planning

Two fundamental problems in sequential decision making:

- Reinforcement Learning:
  - Environment initially unknown
  - Agent interacts with the environment and improves its policy.
  - Model-free methods: Trial-and-error learners
- Planning
  - Model of the environment is known.
  - Agent performs computations with its model, considering possible future situations.

Markov Processes

Markov Property

"The future is independent of the past given the present"

Definition

A state S_t is **Markov** if and only if:

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, ..., S_t]$$

Markov Processes

For a Markov state s and the successor state s', the *state transition probability* is defined by:

$$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$$

State transition matrix $\mathcal{P}$ defines transition probabilities from all states s to all successor states s',

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & \ddots & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix}$$

where $\mathcal{P}_{ij}$ represents transition probability from state i to state j and each row of the matrix sums to 1.

Markov Processes

Definition (Markov Process)

A *Markov Process* (or *Markov Chain*) is a tuple $\langle \mathcal{S}, \mathcal{P} \rangle$

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix,

$$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$$

Markov Reward Process

Definition (Markov Reward Process)

A *Markov Reward Process* is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix,

$$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$$

- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}[R_{t+1} | S_t = s]$
- γ is a discount factor, $\gamma \in [0, 1]$

Markov Reward Process

Definition (Return)

The *return* G_t is the total discounted reward from time-step t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The *discount* $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward R after $k + 1$ time steps is $\gamma^k R$
- This values immediate reward above delayed reward.
  - γ close to 0 leads to "myopic" evaluation
  - γ close to 1 leads to "far-sighted" evaluation

Bellman equation for MRPs

Definition (Value Function)

The *state value function* $v(s)$ of an MRP is the expected return starting from state s

$$v(s) = \mathbb{E}[G_t | S_t = s]$$

The value function can be decomposed into two parts:

- immediate reward R_{t+1}
- discounted value of successor state $\gamma v(S_{t+1})$

$$\begin{aligned} v(s) &= \mathbb{E}[G_t | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \end{aligned}$$

$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

Bellman equation in Matrix form

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

where v is a column vector with one entry per state

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & \ddots & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

The Bellman equation is a linear equation and can be solved directly as:

$$\begin{aligned} v &= \mathcal{R} + \gamma \mathcal{P} v \\ (I - \gamma \mathcal{P}) v &= \mathcal{R} \\ v &= (I - \gamma \mathcal{P})^{-1} \mathcal{R} \end{aligned}$$

Markov Decision Processes

Definition (Markov Decision Process)

A Markov Decision Process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix,

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

- $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$
- γ is a discount factor, $\gamma \in [0, 1]$

Value functions

Definition (State value function)

The *value function* of a state s under a policy π, denoted by $v_\pi(s)$, is the expected return when starting in s and following π thereafter. For MDPs, we can define v_π formally by:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s\right], \text{for all } s \in S$$

where $\mathbb{E}_\pi[.]$ denotes the expected value of a random variable given that the agent follows policy π and t is any time step.

Note that the value of the terminal state, if any, is always zero.

Value functions

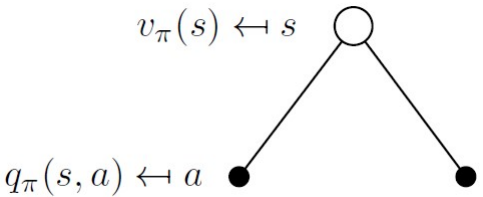
Definition (Action value function)

The value of taking action a in state s under a policy π, denoted by $q_\pi(s, a)$, is the expected return starting from s, taking the action a, and thereafter following policy π:

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

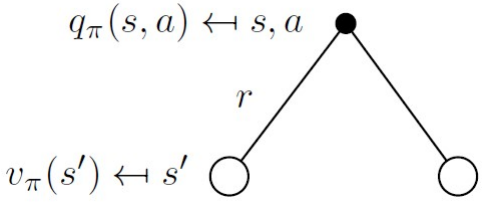
This is the key quantity that we use to help us optimize our MDP and pick the best actions.

Bellman expectation equation for $v_\pi(s)$ (1)



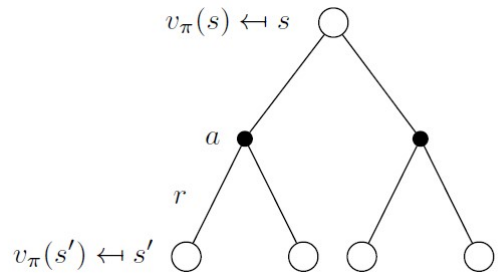
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

Bellman expectation equation for $q_\pi(s, a)$ (1)



$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

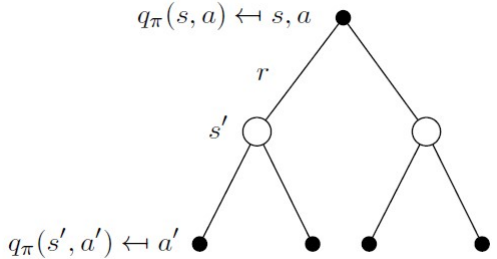
Bellman expectation equation for $v_\pi(s)$ (2)



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

The above equation is the **Bellman expectation equation** for $v_\pi(s)$

Bellman expectation equation for $q_\pi(s, a)$ (2)



$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s', a')$$

The above equation is the **Bellman expectation equation** for $q_\pi(s, a)$

Optimal value functions

Definition

The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

$$v_*(s) = \max_\pi v_\pi(s)$$

Definition

The *optimal action-value function* $q_*(s, a)$ is the maximum action-value function over all policies

$$q_*(s, a) = \max_\pi q_\pi(s, a)$$

Optimal policies

A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states, i.e.

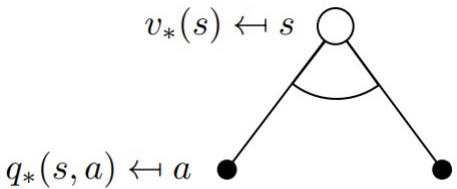
$$\pi \geq \pi' \quad \text{if } v_\pi(s) \geq v_{\pi'}(s), \forall s$$

Theorem

For any Markov Decision Process

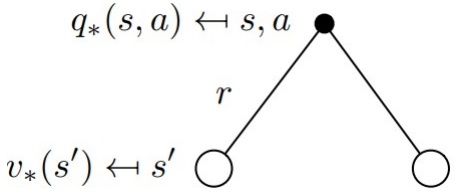
- *There exists an optimal policy π_* that is better than or equal to all other policies, $\pi_* \geq \pi$, $\forall \pi$*
- *All optimal policies achieve the optimal state-value function, $v_{\pi_*}(s) = v_*(s)$*
- *All optimal policies achieve the optimal action-value function, $q_{\pi_*}(s, a) = q_*(s, a)$*

Bellman optimality equation for $v_*(s)$ (1)



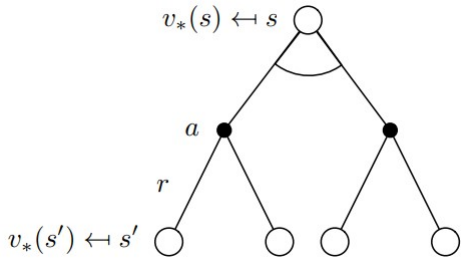
$$v_*(s) = \max_a q_*(s,a)$$

Bellman optimality equation for $q_*(s, a)$ (1)



$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

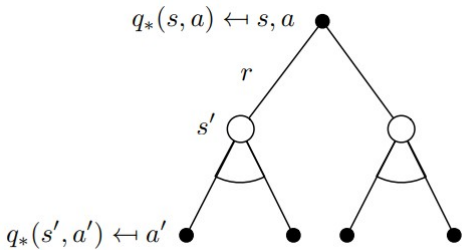
Bellman Optimality Equation for $v_*(s)$ (2)



$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

This $v_*(s)$ tells us how good it is to be in the state s and is the **Bellman optimality equation** for v_*

Bellman optimality equation for $q_*(s, a)$ (2)



$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s', a')$$

is the **Bellman optimality equation** for q_*

Future Work

Will work on the LunarLander-v2 problem on Open-AI Gym and try to maximize our average reward over multiple episodes using Deep Q-Learning and Deep Q-Network (DQN) that is the first deep reinforcement learning method proposed by DeepMind in 2015.

Problem Statement

In this task the agent has to land a lander on a landing pad.
Observation space. The state of the environment consists of 8 variables:

- Horizontal position: p_x
- Vertical position: p_y
- Horizontal velocity: v_x
- Vertical velocity: v_y
- Angle w.r.t the vertical axis: θ
- Angular velocity: ω
- Left leg contact: c_l
- Right leg contact: c_r

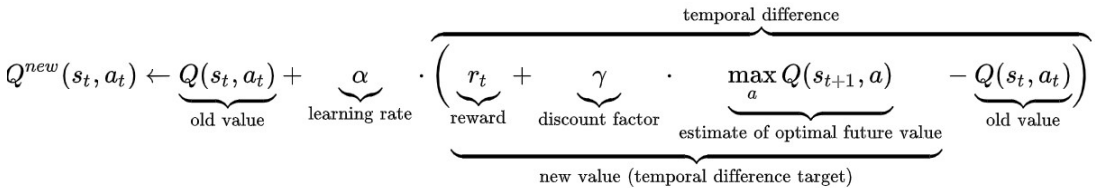
Problem Statement

Action space. The agent can perform 4 actions:

- All engines disabled: *nop*
- Enable left engine: *left*
- Enable maine engine: *main*
- Enable right engine: *right*

Problem Statement

- *Rewards*. The reward for moving from the initial point to the landing pad with final velocity of zero varies between 100 and 140 points. If the lander crashes it receives a reward of -100 points. If the lander lands correctly it receives a reward of $+100$ points. For each leg contact the agent receives a reward of $+10$ points. Firing the main engine in a timestep gives a reward of -0.3 points, while firing a side-engine gives a reward of -0.03.
- *Termination criterion*. The simulator ends if either 1000 timesteps are passed, the lander crashes or it passes the bounds of the environment.
- *Resolution criterion.* This task is considered as solved if the mean total reward $R \geq 200$ on 100 episodic runs.



"Life is like a Markov Decision Process. The decisions that shape our future depend solely on the choices we make given our current state-of-mind; all of our past experiences are implicitly incorporated in our current state, in that they led us to where we are today. So, let's try and let go of the past by making the best out of the state we are in at the moment"

Thank You