Week 1: Reinforcement Learning Preliminaries

BEH Chuen Yang

Abstract

This document provides an introduction into Reinforcement Learning by contrast with Supervised Learning. We discuss some formal underpinnings of RL, including the Markov Decision Process framework. We explore how RL appears in the real world, and how it can be used to solve problems. Lastly, we provide a mathematical formulation of the RL objective.

1 Introduction

Many real-world problems require an *agent* to make a sequence of *actions/decisions* over *time* in an *environment* in order to achieve a certain *goal*. For example, a robot may need to navigate a maze (Muller (2023)), or an air-conditioner must cool a room down to a certain temperature.

In these situations, the "most appropriate" action/decision at each time step depends on a multitude of factors, including the current state of the environment, the dynamics of the environment, past actions, and the agent's objective. We call problems like these *sequential decision making problems* (SDMPs).

2 Sequential Decision Making & Markov Decision Process (MDP)

The notation in this section is synthesized from Achiam (2018), Levine et al. (2023), and Sutton & Barto (2018).

2.1 Sequential Decision Making Problems (SDMPs)

Generally speaking, for a given *environment*, let S be the set of all possible *states* and A be the set of all possible *actions*. An agent *interacts* with the environment over a given timeframe $0 \leq t \leq n$ by observing the current state $s_t \in S$ and taking an action $a_t \in A$ according to a policy π.

We defer the discussion of policies to Section 4.2.

Since we may not fully understand the environment dynamics, let D be a placeholder set for all unobservable variables affecting the environment. Then, we have a *transition operator* $s_{t+1} \sim \tau(s_t, a_t, d_t)$ describing for current states $s_t \in S$, $a_t \in A$, and unknown $d_t \in D$, how the environment changes (possibly stochastically) over time.

We also have a *reward function* $r_t = r(s_t, a_t, s_{t+1}, d_t)$, telling the agent "how good" a particular transition is.

In short, we can describe SDMPs using the tuple (S, A, τ, r) and set D.

A ~~set~~ sequence of interactions $T = ((s_0, a_0), (s_1, a_1), \ldots, (s_n, a_n))$, where $s_0 \sim \rho_0$ is stochastically sampled, is called a *trajectory* or *episode*. The agent receives a reward r_t at each time step t based on the current state and action taken.

2.2 Markov Decision Process (MDP)

A Markov Decision Process (MDP) is a special case of SDMPs, where the transition function satisfies a nice property known as the *Markov property* (Sutton & Barto (2018)):

$$s_{t+1} \sim \tau(s_t, a_t, d_t) = \tau(s_t, a_t) \tag{1}$$

Informally, this means that the next state s_{t+1} depends only on the current state s_t and action a_t, and not on any previous states or actions.

This simplification allows us to reduce the solution space of the problem, and is a key assumption in many RL algorithms (Sutton & Barto (2018)). We can also rewrite the reward function as $r_t = r(s_t, a_t, s_{t+1})$, since we assume that d_t no longer exists. (we have fully captured the environment dynamics.)

3 Contrasting Supervised Learning (SL) and Reinforcement Learning (RL)

3.1 Supervised Learning (SL)

In SL, models are made to learn relationships between *features* and *labels/targets*, or *input* and *output pairs*. (Goodfellow et al. (2016)). Formally, let $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ be a dataset, and let $f_\theta : X \to Y$ be a function that maps inputs to outputs where $\mathbf{x}_i \in X$ is the input and $\mathbf{y}_i \in Y$ is the output. For a given loss function $\mathcal{L}$, SL seeks θ such that the **expected loss over** S is minimized:

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta) = \arg\min_\theta \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\in S}[\mathcal{L}(f_\theta(\mathbf{x}), \mathbf{y})] \tag{2}$$

where f_θ is the model parameterized by θ (Levine et al. (2023)).

In the next subsection, we discuss how SL is used to learn decision-making policies *by imitation*.

3.2 Imitation Learning (IL)

As an extension of SL, IL[1] learns a policy π_θ by trying to *imitate* expert demonstrations.

Writing the problem formally to contrast with SL, let $S = \{(\mathbf{s}_i, \mathbf{a}_i)\}$ be a dataset of expert demonstrations, where $\mathbf{s}_i$ is the state and $\mathbf{a}_i$ is the action taken by the expert. Let $\pi_\theta : S \to A$ be a policy that maps states to actions, where A is the action space. For a given loss function $\mathcal{L}$, θ is sought *analogously as in SL* (contrast with Equation 2):

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta) = \arg\min_\theta \mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a})\in S}[\mathcal{L}(\pi_\theta(\mathbf{x}), \mathbf{y})] \tag{3}$$

Note that the loss function $\mathcal{L}$ need not be directly computed from the environment dynamics and reward function. [2]

4 Reinforcement Learning (RL) vs SL

In contrast to IL, RL learns by directly interacting with the environment, in order to directly maximize the expected cumulative reward signal (Sutton & Barto (2018)).

[1] There does not seem to be consensus on the definition of IL, despite that implied by Levine et al. (2023). For example, Tedrake (2023) seems to name this process *Behavioral Cloning*, a *type* of IL.

[2] Of course, the loss function could still ultimately *depend* on the dynamics and reward function. Consider the case where an expert is a pre-trained RL model. Clearly, if you try to track the expert actions, you are indirectly learning the environment dynamics. But what we mean here is that the loss function is computed as "closeness" to the expert's policy.

Symbol	Description
S	$96 \times 96 \times 3$ RGB Image depicting a God's eye view of the racetrack and car.
A	An integer $0 \leq a \leq 4$, respectively denoting: (1) Do Nothing, (2) Steer Left, (3) Steer Right, (4) Gas, (5) Break
$r(s_t, a_t)$	$\begin{cases} -0.1 + 1000/N & \text{if the car visits new tile, where N is total tiles.} \\ -100 & \text{if the car veers too far from track} \\ -0.1 & \text{otherwise} \end{cases}$
τ	Based on the actions described above, the car's position is updated. The car's position and velocity are updated by a fixed amount.

Table 1: Gymnasium's (Discrete) Car Racing Environment (Towers et al. (2024)).

Broadly speaking, the agent will use its policy to explore the environment (i.e. collect *rollouts*), and adjust the policy from the rewards it receives. It does *not* simply ape the expert demonstrations.

This process can appear quite contrived and abstract. We illustrate with the following example:

4.1 Reinforcement Learning Example

Consider this environment due to Towers et al. (2024):

> A car is driving on a race track and needs to get to the finish point as fast as possible. The car can accelerate, brake, or turn left/right.

Table 1 provides a description of the environment in more detail.

Owing to the absence of outside factors such as other cars, the weather, etc., the fact that any environment dynamics not observed by the agent are held constant, as well as the fact that the car's position and velocity are dependent on the car's previous position and velocity, such a model of the race track is *Markovian*, and this problem can be solved using RL.

4.2 The RL Objective

(Note that π can either be deterministic (hence $a_t = \pi(s_t)$), or stochastic (hence $a_t \sim \pi(\cdot|s_t)$). For convenience, we will assume π is stochastic in this section.)

Reusing notation developed in Section 2, the ultimate objective of an RL agent is to learn a policy π which maximizes the expected cumulative reward over time. Put formally,

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{T\sim\pi} \left[\sum_{t=0}^{n} r_t(s_t, a_t, s_{t+1}, d_t) \right] \tag{4}$$

where $\mathop{\mathbb{E}}_{T\sim\pi}$ is the expectation over all possible trajectories T sampled from the policy π.

Specifically in an MDP, we can ignore d_t, and so our agent's ultimate objective is instead:

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{T\sim\pi} \left[\sum_{t=0}^{n} r(s_t, a_t, s_{t+1}) \right] \tag{5}$$

If we further define $r(s_t, a_t) = \mathbb{E}_{s_{t+1}\sim\tau(s_t,a_t)}[r(s_t, a_t, s_{t+1})]$, we have the objective as:

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{T\sim\pi} \left[\sum_{t=0}^{n} r(s_t, a_t) \right] \tag{6}$$

(Contrast with Equation 3.)

Though this objective is arguably an over-simplification of the RL problem, it is a good algorithm-agnostic starting point for understanding RL.

5 Conclusion

In this report, we discussed the MDP framework as a special case of SDMPs which are friendlier to optimize for. By contrasting RL with SL, we highlighted what sets RL apart from typical Machine Learning problems.

It is hoped that this report serves as a good foundation for similar, future reports.

References

Joshua Achiam. Spinning Up in Deep Reinforcement Learning. 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Sergey Levine, Kyle Stachowicz, Vivek Myers, Joey Hong, and Kevin Black, 2023. URL https://rail.eecs.berkeley.edu/deeprlcourse/.

Derek Muller. The fastest maze-solving competition on earth, 2023. URL https://www.youtube.com/watch?v=ZMQbHMgK2rw.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Russ Tedrake. *Underactuated Robotics*. 2023. URL https://underactuated.csail.mit.edu.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments, 2024. URL https://arxiv.org/abs/2407.17032.