Reinforcement Learning With Hado van Hasselt

Yaro Kazakov

September 10, 2024

Abstract

These are study notes to prepare myself for PhD applications.

1 Lecture 1 - Introduction to Reinforcement Learning

Prime Book for RL - Reinforcement Learning: An Introduction, Sutton & Barto 2018. The whole course in UCL was built based on this book.

Intelligence - being able to learn to make decisions to achieve goals.

Johns Hopkins Definition of Intelligence - Ability to solve complex problems or make decisions with outcomes.

1.1 What is RL?

RL is learning through interacting with our environment. It is goal-oriented, can learn without examples, and optimize for some reward signal. The interaction works as follows:

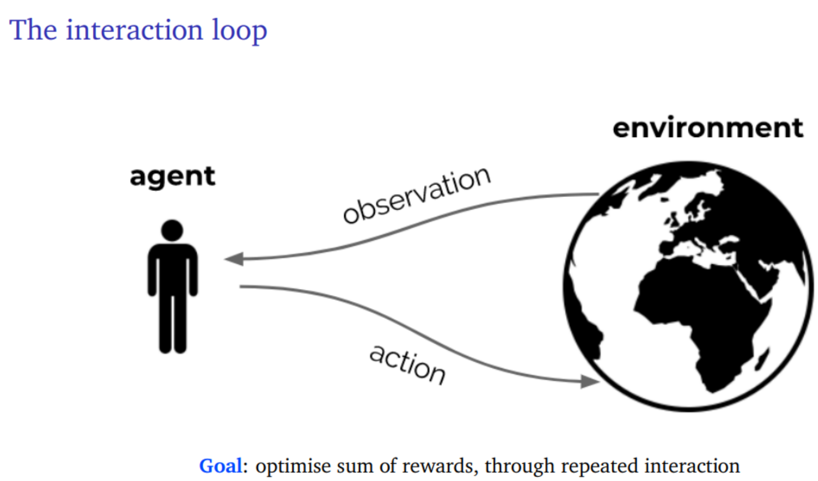

Figure 1: Interaction Loop

RL is based on reward hypothesis: Any goal can be formalized as the outcome of **maximizing a cumulative reward**.

Two main reasons to learn:
- Find Solutions
- Adapt Online

RL works for both cases. In short, RL is a science and framework of learning to make decisions from interaction. RL can sometimes be viewed as a cherry on top based on LeCake:

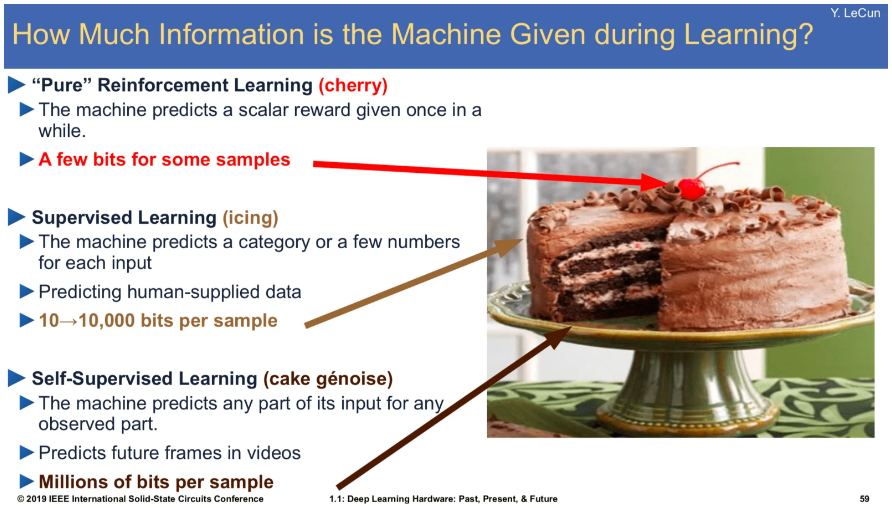

Figure 2: Mocking by LeCake

With Agent and Environment the interaction looks as follows . . .

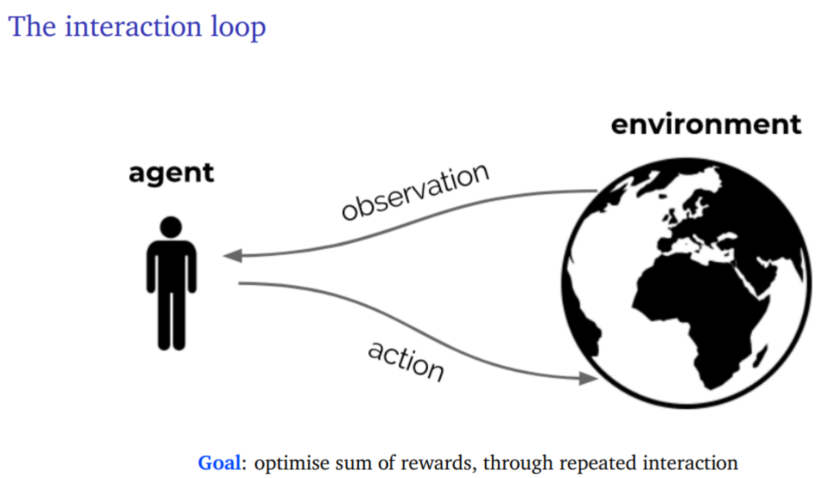

Figure 3: Interaction Loop

1. At each step t the agent:
   (a) Receives observation O_t and reward R_t
   (b) Executes action A_t
2. The environment:
   (a) Receives action A_t
   (b) Emits observation O_{t+1} (and reward R_{t+1})

A reward R_t is a scalar feedback signal. The agent's job is to maximize cumulative reward: $G_t = R_t + R_{t+1} + ...$ And as before we want to **maximize the cumulative reward** which is called **RETURN**

1.2 Value

The expected cumulative reward from a state s is called Value:
$v(s) = E[G_t|S_t = s] = E[R_{t+1} + R_{t+2} + R_{t+3} + ...|S_t = s]$

The value depends on the actions the agent take. Goal is to **maximize value**, by picking **suitable actions**. Return and values can be defined recursively:
$G_t = R_{t+1} + G_{t+1}$
$v(s) = E[R_{t+1} + v(S_{t+1})|S_t = s]$

A mapping from state to actions is called **POLICY**

It's possible to condition on states AND actions:
$q(s, a) = E[G_t|S_t = s, A_t = a] = E[R_{t+1} + R_{t+2} + R_{t+3} + ...|S_t = s, A_t = a]$

1.3 Core Concepts

The RL includes:

- Environment (dynamics of the problem)
- Reward Signal (specifies the goal)
- Agent, containing:
  1. Agent State
  2. Policy
  3. Value function estimate?
  4. Model

1.4 the Agent State

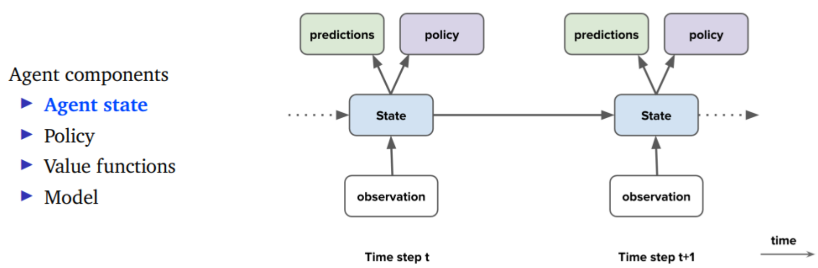

Figure 4: Agent's Pipeline

The history is the full sequence of observations, actions, rewards:
$H_t = O_0, A_0, R_1, O_1...$
The history can be used to construct agent state S_t

1.4.1 Fully Observable Environments

$S_t = O_t =$ environment state

1.4.2 Markov Decision Processes

$p(r, s|S_t, A_t) = p(r, s|H_t, A_t)$
- meaning the state contains all we need to know from the history. Doesn't mean it contains everything,

just that adding more history doesn't help. The agent's actions depend on its state. Generally:
$S_{t+1} = u(S_t, A_t, R_{t+1}, O_{t+1})$

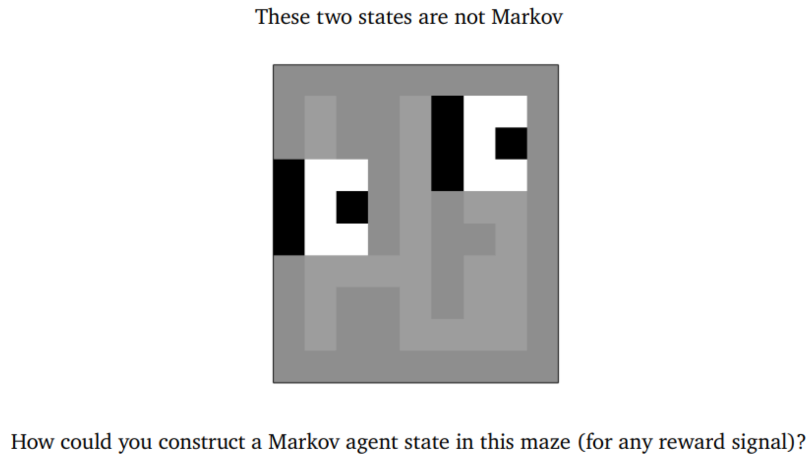

Figure 5: Not Markov States

To deal with partial observability, agent can construct suitable state representations

1.5 Policy

1. A policy defines the agent's behavior
2. It is a map from agent state to action
3. Deterministic policy: A = pi*(S)
4. Stochastic policy $pi(A|S) = p(A|S)$

1.6 Value Function

The actual value function is the expected return:
$v_{pi}(s) = E[R_{t+1} + \gamma * R_{t+2} + \gamma^2 R_{t+3} + ...|S_t = s, pi]$
Gamma is a discount factor that trades off immediate vs long-term rewards.
$v_{pi}(s) = E[R_{t+1} + \gamma * v_{pi}(S_{t+1})|S_t = s, A_t \; pi(s)]$

This is known as a **Bellman equation (Bellman 1957)**

A similar equation, that does not depend on a policy also holds:

$v_*(s) = max(E[R_{t+1} + \gamma * v_*(S_{t+1})|S_t = s, A_t = a])$

1.7 Model

A model predicts what the environment will do next. E.g. P predicts the next state:
$P(s, a, a') = p(S_{t+1} = s'|S_t = s, A_t = a)$

R predicts the next immediate reward:

$R(s, a) = E[R_{t+1}|S_t = s, A_t = a]$

1.8 Agent Categories

1. Model Free - Policy and/or Value Functions
2. Model Based - Optionally Policy and/or Value Function — Model

1.9 Subproblems of RL Problem

1. Prediction: evaluate the future (for a given policy)
2. Control: optimize the future (find the best policy)
3. These two are very strongly related

1. Learning: the environment is intially unknown. The agent interacts with the environment
2. Planning: A model of the environment is given (or learned). The agent plans in the model (without external interaction). a.k.a reasoning, pondering, thought, search, planning

IMPORTANT: ALL COMPONENTS ARE FUNCTIONS!!!

1. Policies
2. Value functions
3. Models
4. State update

We can use **Deep Learning** to learn those functions!

References