ECE 1508: Reinforcement Learning

Chapter 1: Introduction

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

Fall 2025

Defining RL Problem

Let's make an agreement

$$\textit{Reinforcement Learning} \equiv \textit{RL}$$

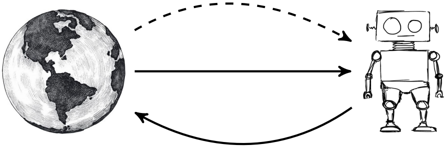
In each RL problem, we have three main components

1. *Agent who is the one who acts*
2. *Environment which responds to the agent's actoins*
3. *Interaction means which communicate actions and responds*

Let's mathematically model each component

Problem Formulation

The agent is interacting with the envronment

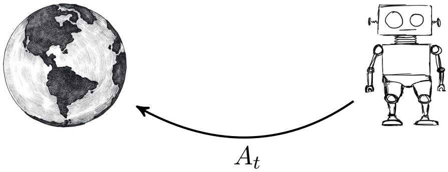


+ *How is exactly this interaction?*
– It happens by three means

action, *observation* and *reward*

Let's break it down

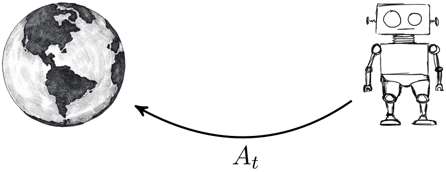
Problem Formulation: *Action*



At time t, agents selects an action from a set of possible actions

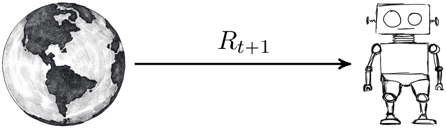
$$A_t \in \mathbb{A} = \left\{a^1, \ldots, a^M\right\}$$

Problem Formulation: *Action*



+ *Is this set always discrete?*
– *Not necessarily!* But, we may assume so for now ☺
+ *Is it always constant through time? Can't we get more or less options to act as time passes?*
– *Again: Not necessarily!* But, we may assume so for now ☺

Problem Formulation: *Reward*



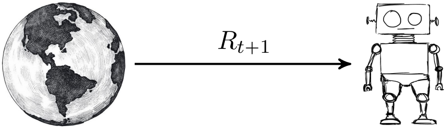
Once acted, the agent gets some reward from the environment

$$R_{t+1} \in \{r^1, \dots, r^L\}$$

We index by $t+1$, as agent gets the reward in the next time step

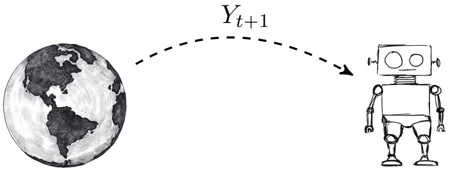
↳ *Again we assume a discrete set of possible rewards only for simplicity!*

Problem Formulation: *Reward*



+ *Does agent always get a reward?*
- No! It may have no reward in some time steps
  ↳ *We could think of zero (empty) reward in that case, e.g.,* $R_{t+1} = 0$

Problem Formulation: *Observation*

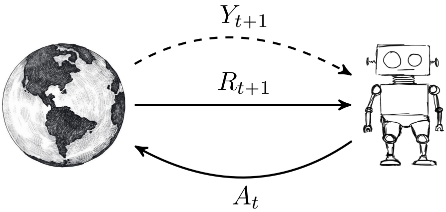


Before the next action, the agent also observes the environment

$$Y_{t+1} \in \mathbb{Y}$$

We index by $t+1$, as agent observes after each action

Problem Formulation: *Trajectory*



Let's make an agreement[1]

↳ *At time* $t = 1$ *agent observes initial observation* Y_1 *and reward* R_1

  ↳ *It then decides for action* A_1

↳ *As a consequence it gets observation* Y_2 *and reward* R_2

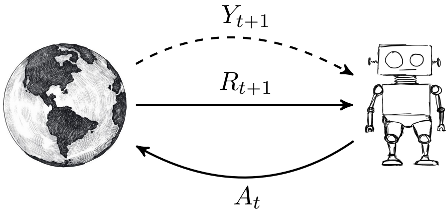
  ↳ *It then decides for action* A_2

$\dots$

This behavior is going to continue for ever, i.e., $t = 1, \dots, \infty$

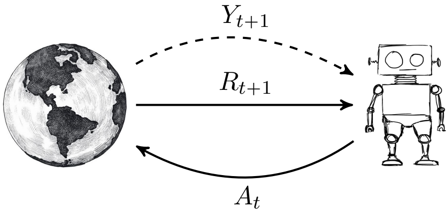
[1] We keep it as in *Sutton's book*, as it's a common notation

Problem Formulation: *Trajectory*



+ *But, shouldn't the agent stop at some point?!*
- Let's assume for now that *it never stops once started*
  - ↳ *We later define the concept of state*
  - ↳ *With this concept we can make agent stop without terminating* t
    - ↳ *The agent simply gets into a recurring state*

Problem Formulation: *History*

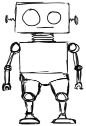


History contains all happened up to time t, before agent acts in time t

$$H_t = A_0, R_1, Y_1, A_1, R_2, Y_2, A_2, \dots, R_t, Y_t$$

This is the information that has been collected by the agent up to time t

Problem Formulation: *Final Goal of Agent*



The final goal of the agent is to

maximize what it is going to collect as reward in future

- + *Why does agent looks at cumulative future reward?*
- – Well it's a bit of philosophical argument: *we assume that*
  - ↳ *reward is all the agent would think about*
  - ↳ *the agent want to aggregate as much reward as possible*
  - ↳ *past has passed! Agents tries to maximize what it could get*

But, you have all the right *not to agree with this statement!*

↳ *In this case, you may look at it as a mathematical model* ☺

Problem Formulation: *Future Return*

We formulate *accumulated future reward* using the notion of return

Future Return

At time t, the future return of the agent is

$$G_t = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}$$

for some constant $0 \leqslant \gamma \leqslant 1$ referred to as discount factor

+ *What exactly is this return? Why do we have a discount factor?*
– Let's open it up!

Problem Formulation: *Future Return*

The future return at time t looks like

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

We can look at it this way: after we decide to act A_t at time t

- *We are going to get immediate reward R_{t+1} in next time step*
- *This will also impact our next action A_{t+1}*
  - ↳ *after this we will get R_{t+2}*
  - ↳ *after the next action we get R_{t+3}*

    $\dots$
- *We rather prefer to get the reward sooner than later!*
  - ↳ *large R_{t+1} is preferred to large R_{t+2}*
  - ↳ *large R_{t+2} is preferred to large R_{t+3}*

    $\dots$
- *We scale down gained reward with discount factor after each time step*

Problem Formulation: *Future Return*

> *The future return at time t looks like*
>
> $$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

Extreme choices of discount factor γ are zero and one:

- *Shortsighted agent sets $\gamma = 0$*
  - ↳ *It only cares about the immidiate reward*
- *Forethoughtful agent that sets $\gamma = 1$*
  - ↳ *It cares about the cumulative sum over time*
  - ↳ *It cares about future as much as it cares about now*
  - ↳ *It would accept less reward now, if it expects larger compensation in future*

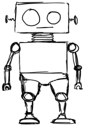
Problem Formulation: *Agent's Learning Task*

Given the concept of *future return*, we can formulate agent's goal as

> *At time t, agent looks at history H_t and decides for an action A_t that maximizes the future return G_t*

+ *But wait a moment! How can the agent optimize? How can it even know what the return value is?*
- We should model relations between the interaction means

A Stochastic Framework



Agent looks at the environment as a random object

- *Depending on agent's action, it returns a random reward and observation*
  - ↳ *Recall the multi-armed bandit example*
    - ↳ *Depending on choice of company, agent gets a random payment*

Agent may also act randomly

- *It is due to the randomness of environment*
  - ↳ *Recall the multi-armed bandit example*
    - ↳ *Agent tries each company for d days and decides after $2d$ days*
    - ↳ *It could go to Company A or B*
    - ↳ *The probability of going to Company A is higher though*

Modeling Components: *State*

Let's start modeling this stochastic framework: *we start with history*

> *History at time t is collection of what agent has observed up to that time*
>
> $$H_t = A_0, R_1, Y_1, A_1, R_2, Y_2, A_2, \ldots, R_t, Y_t$$

But this is a long sequence as $t \to \infty$

- *It cannot be stored into a finite memory!*
  - ↳ *For any memory-size, there is a time after which the memory is full*
- *Not all of these entries in the history are useful*
  - ↳ *Information context of the history is always limited*

We need to replace history with a finite memory component

+ *This reminds me of some relevant discussion in deep learning!*
– Yes! *State-dependent systems in recurrent neural networks*

Modeling Components: *State*

We assume that environment is a state-dependent system

State-dependent Environment

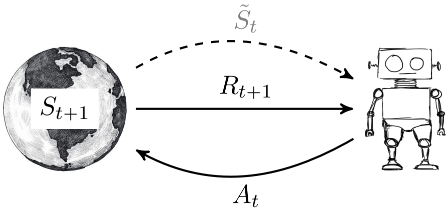
The environment at each time step has a particular state

$$S_t \in \mathbb{S} = \left\{ s^1, \ldots, s^N \right\} \quad \leftsquigarrow \text{set of possible states}$$

which along with the action A_t specifies its next state the reward it returns

+ *Is there necessary a discrete set of possible states?*
– Not necessarily! *But we keep assuming whenever needed*
+ *How does this assumption impact the RL setting?!*
– Let's see

RL Framework: *State-Representation*

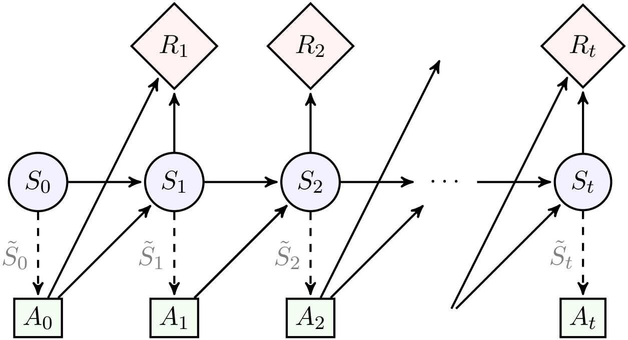


Let's say we are at time t and the environment is in state S_t

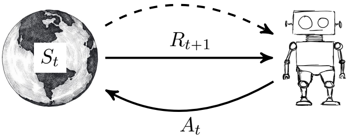
1. *Agent observes $\tilde{S}_t$*
   - ↳ *In ideal world, it can see the true state: $\tilde{S}_t = S_t$*
   - ↳ *In real world, it might see an incomplete version: $\tilde{S}_t \neq S_t$*
2. *Based in the observed state $\tilde{S}_t$, agents decides to act A_t*
3. *Environment receives action A_t*
   - ↳ *It rewards the agent as R_{t+1}*
   - ↳ *It changes its state to S_{t+1}*

RL Framework: *State-Representation*

So, we can plot the trajectory as a causal graph



RL Framework: *Markovity of State*



The *complete state* should *give all information about the environment at each time: this means that at time t*

the reward of next time R_{t+1} and the next state S_t

should be described completely by the action A_t and current state S_t

---

+ *Can we guarantee that we always find such state?*
– Well, we can assume that it exists, but maybe *we do not have access to it*

RL Framework: *Markovity of State*

We assume that the state S_t contains all information up to time t

↳ *This state describes a Markov process*

Markov Process

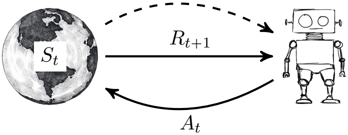
Sequence $S_1 \rightarrow S_2 \rightarrow \dots$ describe a Markov process if

$$\Pr\left\{S_{t+1} = s_{t+1} | S_t = s_t, \dots, S_1 = s_1\right\} = \Pr\left\{S_{t+1} = s_{t+1} | S_t = s_t\right\}$$

In other words: we assume that

whatever we need to know about past is always in the last state

RL Framework: *State-Representation*

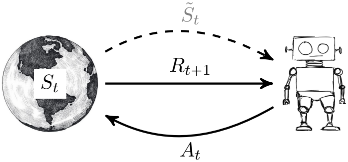


Moral of Story

Environment behaves based on a state which is a Markov process

- *At each time agent observes the state and acts based on that*
  - ↳ *It could be an incomplete observation*
- *Environment looks only on its current state and agent's action*
  - ↳ *it returns a reward*
  - ↳ *it changes its state*

RL Framework: *Agent's Observation*

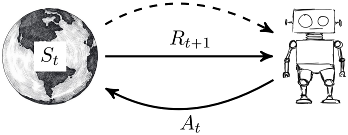


The agent could have different levels of access to the state

- *Full observation of the state:* $\tilde{S}_t = S_t$
  - ↳ *This is an optimal case, but does not happen always*
- *Partial observation of the state:* $\tilde{S}_t \neq S_t$
  - ↳ *This is more realistic, as some agents are restricted in their observation*
    - ↳ *A robot can only see a limited part of its surrounding area through its camera*

To keep things easy, we consider full observation for the moment

RL Framework: *Environment's Behavior*



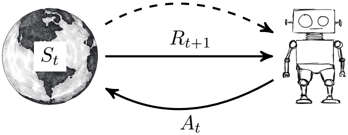
As we said: *environment decides for rewards and new state based on its current state and agent's action, we can describe this behavior mathematically by*

1. *Rewarding function that maps state S_t and action A_t to reward R_{t+1}*

$$\mathcal{R}(\cdot) : \mathbb{S} \times \mathbb{A} \mapsto \{r^1, \ldots, r^L\}$$

This mapping is in general random

RL Framework: *Environment's Behavior*



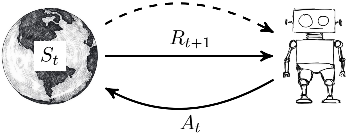
As we said: *environment decides for rewards and new state based on its current state and agent's action, we can describe this behavior mathematically by*

2. *Transition function that maps state S_t and action A_t to the next state S_{t+1}*

$$\mathcal{P}(\cdot) : \mathbb{S} \times \mathbb{A} \mapsto \mathbb{S}$$

This mapping is in general random

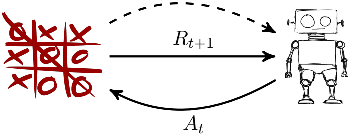
RL Framework: *Environment's Behavior*



- \+ *How do we get access to these mappings?*
- – It depends on the problem
  - ↳ *In some problems, we can find these mappings based on the rules of the problem, e.g., board games or physical dynamic systems*
  - ↳ *In some others, we simply don't know! We may assume some mappings or try to solve the problem without using these mappings*

OK! Let's take a break and see some examples

Example: *Tic-Tac-Toe*

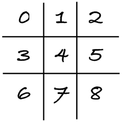


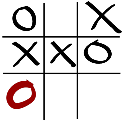
Let's consider the game of Tic-Tac-Toe: *in this game,*

- *Agent is one of the players*
  - ↳ *Action is the move done by the player*
- *Environment is the opponent and the game rules*
  - ↳ *State in each time is the status of the board*

Let's make it mathematically consistent!

Example: *Tic-Tac-Toe*





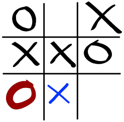
$S_t = \{0, 2, 3, 4, 5\}$
$A_t = 6$

We can number each cell on the board; in this case,

- *Played cells can be considered the sate*
  - ↳ *State in each time is the status of the board*
- *Next cell played is the action*

+ *Is it consistent with the Markovity assumption?*
– Sure! Let's see

Example: *Tic-Tac-Toe*



$S_t = \{0,2,3,4,5\}$
$A_t = 6$
$\rightsquigarrow S_{t+1} = \{0,2,3,4,5,6,7\}$

The player decides only based on the status of the board

↳ *Its action at time t relies on state S_t*

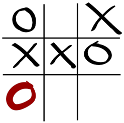
Next state also depends only on the current status and player's move

↳ S_{t+1} *is completely described by state S_t and action A_t*

$$S_{t+1} = S_t \cup \{A_t, O_{t+1}\}$$

with O_{t+1} being opponent's move

Example: *Tic-Tac-Toe*



$S_t = \{0, 2, 3, 4, 5\}$
$A_t = 6$

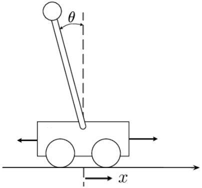
Transition function is probabilistic in this case in this case

$$\mathcal{P}\left(S_t, A_t\right) = \begin{cases} \{0,2,3,4,5,6,1\} & \textit{with probability } \Pr\{O_{t+1} = 1\} \\ \{0,2,3,4,5,6,7\} & \textit{with probability } \Pr\{O_{t+1} = 7\} \\ \{0,2,3,4,5,6,8\} & \textit{with probability } \Pr\{O_{t+1} = 8\} \end{cases}$$

Rewarding function is deterministic

$$\mathcal{R}\left(S_t, A_t\right) = \begin{cases} +1 & \textit{if player wins after playing } A_t \\ -1 & \textit{if player loses after playing } A_t \\ 0 & \textit{if players draw or the game is not over} \end{cases}$$

Example: *Cart-Pole Problem*



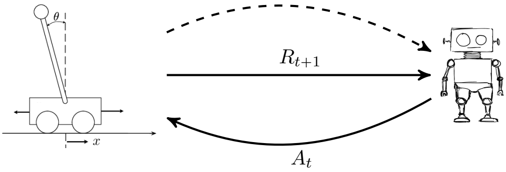
A pendulum is located vertically on a cart; at each time step, player can

- *either shift the cart to the right*
- *or shift it to the left*

The player gets \$1 for each time step that

- *the pendulum remains in an angle less than θ from the vertical line, and*
- *the cart is displaced with distance less than x*

Example: *Cart-Pole Problem*



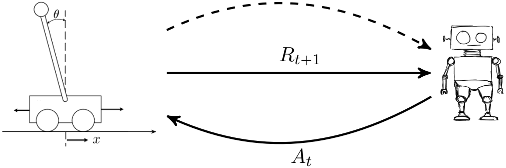
Agent is the player

- *At time t, it acts as $A_t = \pm 1$, i.e., right or left*

Environment is the cart-pole, game rule and the physical laws

- *State at time t is the collection of physical parameters*
  - ↳ *Distance of cart, its velocity, angle of pendulum, angular velocity, . . .*
  - ↳ *Current state and action completely describe next status of pendulum*
    - ↳ *This is a Markov state*
- *It returns reward $R_{t+1} = 1$, if game is not over after action A_t*

Example: *Cart-Pole Problem*



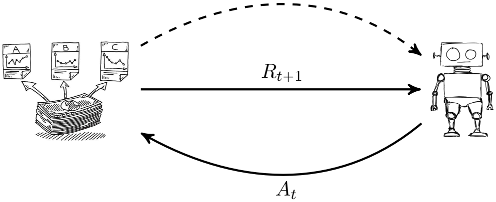
Transition function is completely described by dynamics of the system

$$\mathcal{P}\left(S_t, A_t\right) = \textit{Solution to dynamic equations}$$

Rewarding function is further specified by the game rules

$$\mathcal{R}\left(S_t, A_t\right) = \begin{cases} +1 & \textit{if game is not over after playing } A_t \\ 0 & \textit{otherwise} \end{cases}$$

Example: *Trading*



Agent is the one who invests

- *At time t, it acts by either buying or selling*

Environment is the market, investing portfolio, and transaction rules

- *State is a collection of market and portfolio features*
- *Reward R_{t+1} describes the profit made in each time frame*

Obviously in this case, transition function is not clear to us!