Principles of Machine Learning

DSA 5105 • Lecture 10

Soufiane Hayou

Department of Mathematics

National University of Singapore

Mid-Term and Homework

- Homework 3 available on canvas (due 05/11/2022)

- Common mistakes in the Mid-Term

So far

We introduced two classes of machine learning problems

- Supervised Learning
- Unsupervised Learning

Today, we will look at another class of problems that lies somewhere in between, called **reinforcement learning**

Motivation

Some General Observations of Learning

- Interactions with environment
- Learning from Experience
- Reward vs Demonstrations
- Planning

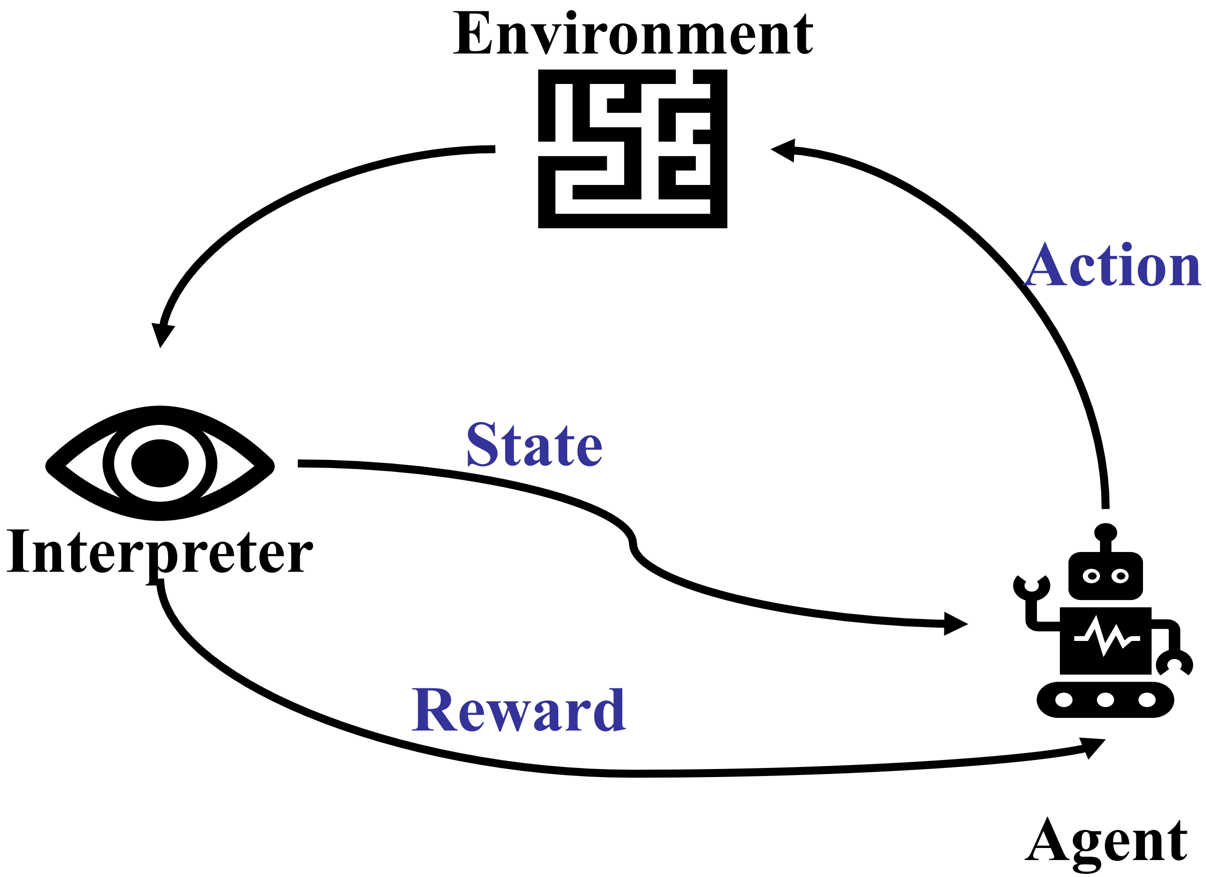
The Reward Hypothesis

All of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).

Examples

- Studying and getting good grades
- Learning to play a new musical instrument
- Winning at chess
- Navigating a maze
- An infant learning to walk

The Basic Components



Examples

Task	Agent	Environment	Interpreter	Reward
Chess	Player	Board state	Vision	Win/loss at the end
Learning to Walk	Infant	The world	Senses	Not falling, getting to places
Navigating a maze	Player	The maze	Vision	Getting out of the maze

Key Differences in Reinforcement Learning

- **Vs unsupervised learning:** not completely unsupervised due to a reward signal

- **Vs supervised learning:** not completely supervised, since optimal actions to take are never given

Example: The Recycling Robot

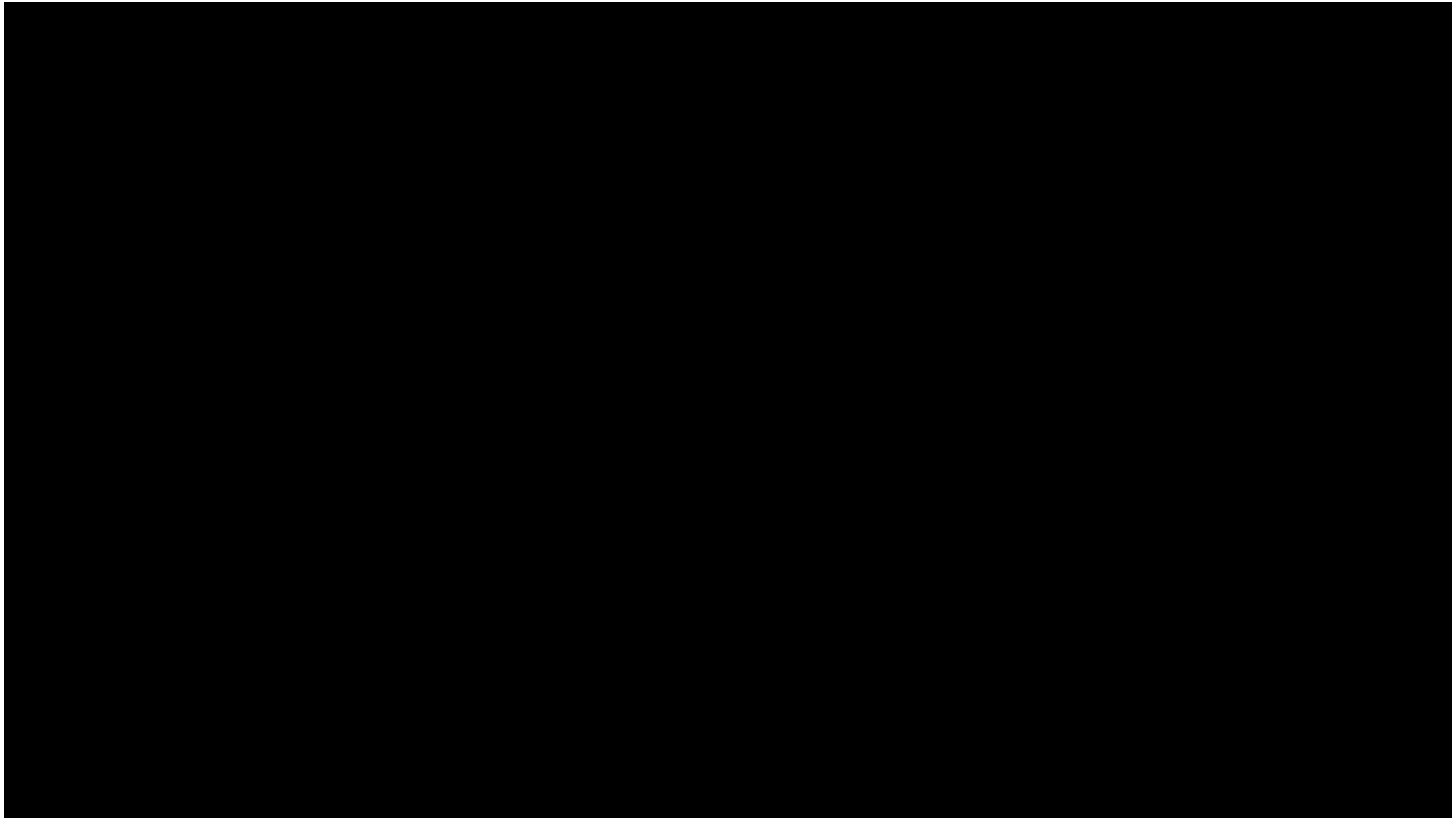
Actions

- Search for cans
- Pick up or drop cans
- Stop and wait
- Go back and charge

Rewards:

- +10 for each can picked up
- -1 for each meter moved
- -1000 for running out of battery

Another Example



The Reinforcement Learning Problem

The RL problem can be posed as follows:

An agent navigates an environment through the lens of an interpreter. It interacts with the environment through performing actions, and the environment in turn provides the agent with a reward signal. The agent's goal is to learn through experience how to maximize the ***long term*** *accumulated reward.*

Mathematical formulation of RL

Deterministic actions (state determines action) or Stochastic actions (each action has a probability)?

How uncertain are we about the environment?

(Similar to the fundamental question: is the world fundamentally random (Team Niels Bohr) or deterministic (Team Einstein)?)

Generally, deterministic rules do not perform well in highly complicated environments → probabilistic model

Finite Markov Decision Processes

Finite State, Discrete Time Markov Chains

- Sequence of time steps: $t = 0,1,2,\ldots$
- State space: $\mathcal{S}$ such that $|\mathcal{S}| < \infty$
- States: $S_t \in \mathcal{S}$

The states $\{S_t : t \geq 0\}$ forms a stochastic process, and evolves according to a **transition probability**

$$\mathbb{P}(S_{t+1} = s' | S_t = s, S_{t-1} = s_{t-1}, \ldots, S_0 = s_0)$$

Markov Property and Time Homogeneity

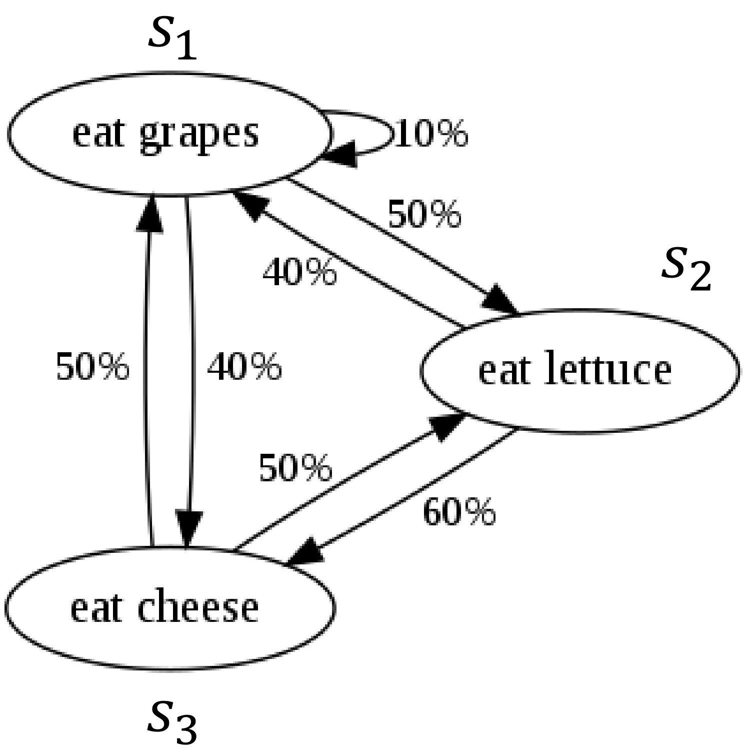
Markov Property

$$\mathbb{P}(S_{t+1} = s' | S_t = s, S_{t-1} = s_{t-1}, \dots, S_0 = s_0) \\ = \mathbb{P}(S_{t+1} = s' | S_t = s\ )$$

Time Homogeneous Markov Chain

- The transition probability is independent of time, i.e.
  $$\mathbb{P}(S_{t+1} = s' | S_t = s\ ) = P_{ss'} = p(s'|s)$$
- The matrix $P = \{P_{ss'}\}$ is called the **transition (probability) matrix**

Example



State space: $\{s_1, s_2, s_3\}$
Transition probability:

$$p(s_2|s_1) = 0.5$$
$$p(s_1|s_1) = 0.1$$
$$p(s_3|s_2) = 0.6$$

Transition probability matrix

$$P = \begin{pmatrix} 0.1 & 0.5 & 0.4 \\ 0.4 & 0.0 & 0.6 \\ 0.5 & 0.5 & 0.0 \end{pmatrix}$$

https://en.wikipedia.org/wiki/Markov_chain

Non-Markovian or Non-time-homogeneous Stochastic Processes

Example of non-Markovian process

- Drawing without replacement coins out of a bag of coins consisting of 10 of each \$1, 50c and 10c coins. Let S_t be the total value of coins drawn up to time t.

Example of non-time-homogeneous process

- At time t, drawing a random number n $\in \{t, t+1\}$ of coins

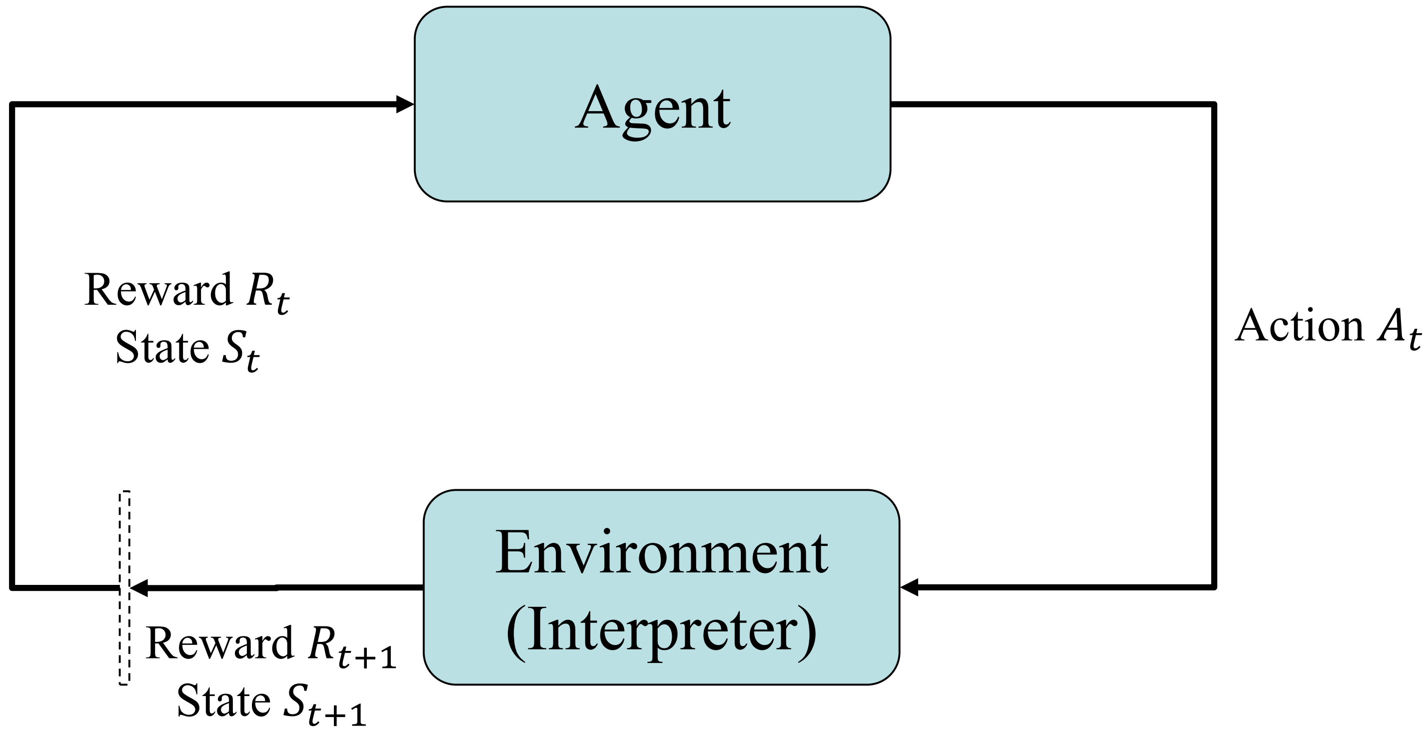
Essential Components are Markov Decision Processes

Markov decision processes (MDP) is a generalization of Markov processes, with **actions** and **rewards**

Essential elements

- Sequence of time steps: $t = 0,1,2,\ldots$
- States: $S_t \in \mathcal{S}$
- Actions: $A_t \in \mathcal{A}(S_t) \subset \mathcal{A}$ (union over all $s \in \mathcal{S}$)
- Rewards: $R_{t+1} \in \mathbb{R}$

State Evolution



Transition Probability

For Markov chains, we have the transition probability

$$\mathbb{P}(S_{t+1} = s' | S_t = s\ ) = p(s'|s)$$

For Markov decision processes, we need to account for additionally:

- The reward R_{t+1}
- The action A_t

Hence, we specific the **MDP transition probability**

$$\mathbb{P}(S_{t+1} = s', R_{t+1} = r | S_t = s\ , A_t = a) = p(s', r | s, a)$$

Markov Decision Processes

A **Markov decision process** (MDP) is the evolution of S_t, R_t according to

$$p(s', r | s, a) = \mathbb{P}[S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a]$$

A MDP is **finite** if $\mathcal{S}$ is finite and $\mathcal{A}(s)$ is finite for each $s \in \mathcal{S}$

Example: The Recycling Robot

State: $S_t = (x_t, c_t, w_t)$
(position, charge, weight)

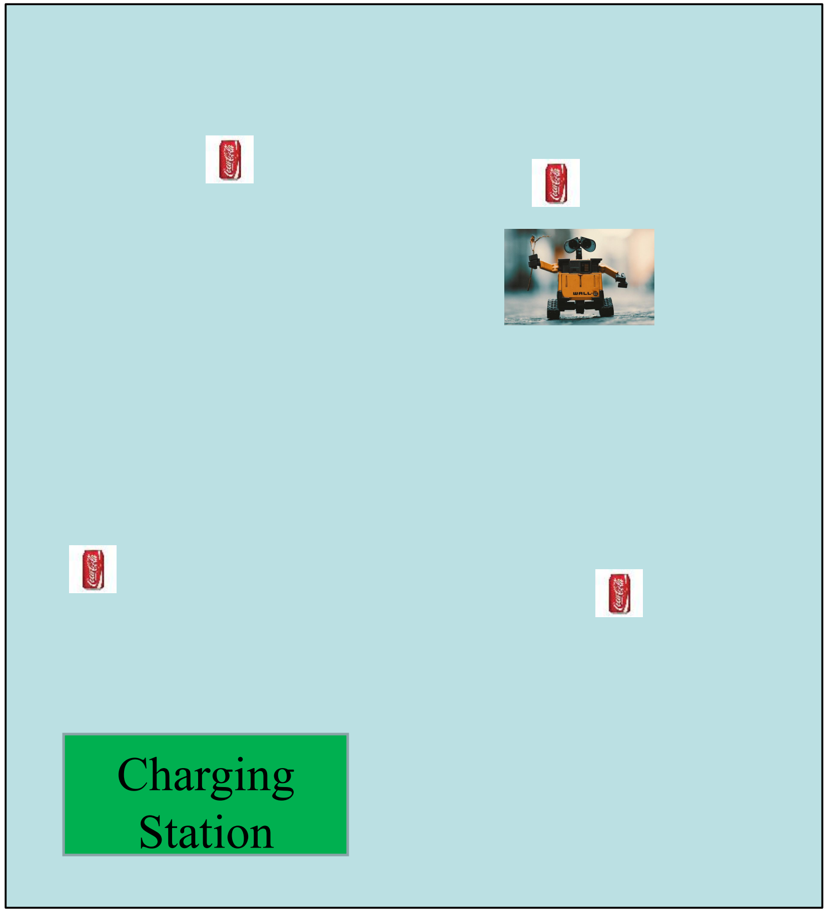
Actions:

$\mathcal{A}(S) \subset \{\text{move}, \text{pick}, \text{drop}, \text{charge}\}$

- If $S_t = (x_t, 0, w_t)$ then $\mathcal{A}(S_t)$ is empty
- If $S_t = (x_t, c_t\ w_t)$, such that $c_t > 0$, $w_t = 0$ and x_t has a can, then $\mathcal{A}(S_t) = \{\text{move}, \text{pick}, \text{charge}\}$
- …

Reward:

$$R_{t+1} = \begin{cases} -1, & A_t = \text{move and } c_t > 0 \\ -1000, & c_t = 0 \\ +10, & A_t = \text{pick} \\ & \ldots \end{cases}$$



The "Decision" Aspect: The Policy

The only way the agent has control over this system is through the choice of actions.

This is done by a specifying a **policy**

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$$

Deterministic policies: $\pi(a|s) = \mathbb{I}_{a=a_0(s)}$

Then we write $a_0 = \pi(s)$, i.e. deterministic policies are functions

$$\pi: \mathcal{S} \to \mathcal{A}$$

The Goal of Choosing a Policy: Returns

We want to maximize long-term rewards…

Define the **return**

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

Here, γ is the **discount rate**

This includes both finite and infinite time MDPs.

The Objective of RL

The goal of RL is the **maximize**, by choosing a good policy π, the expected return

$$\mathbb{E}_\pi[G_0 | S_0 = s] = \mathbb{E}_\pi[R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots | S_0 = s],$$

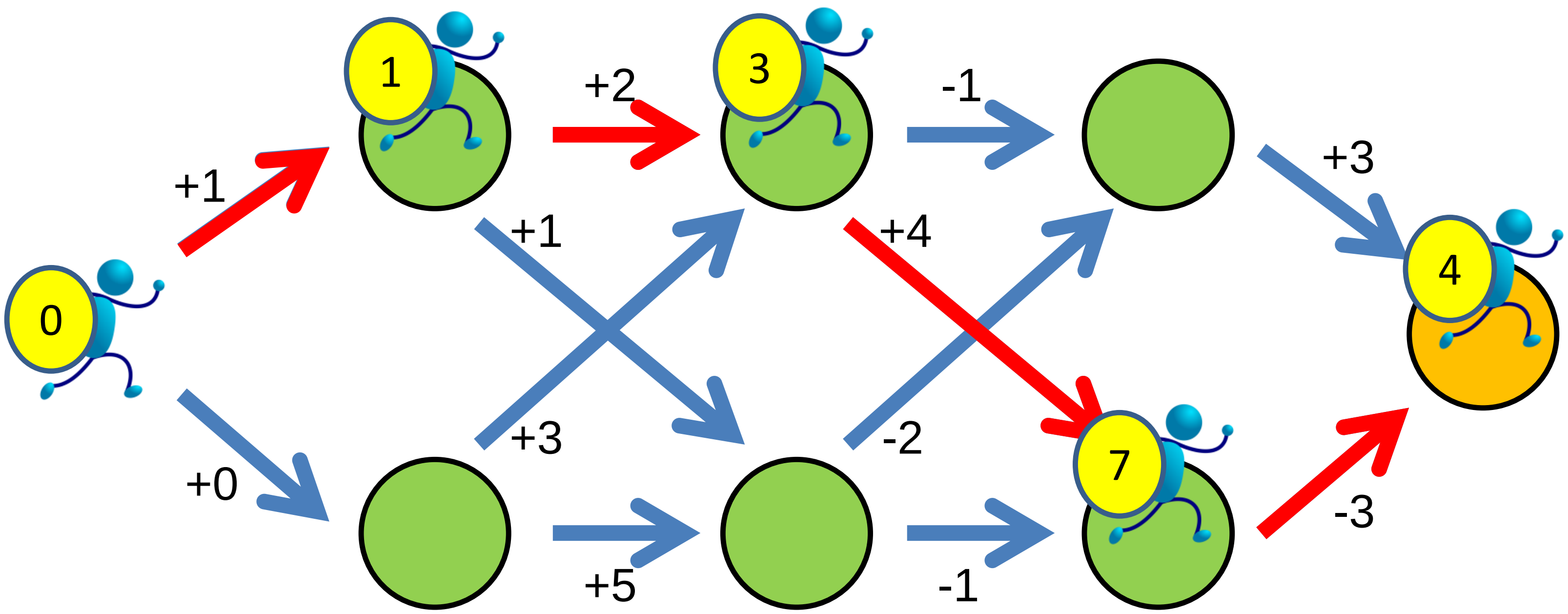
where we start from some state $s \in \mathcal{S}$.

We will consider time-homogeneous cases where this is the same as

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

Dynamic Programming

Example



How long does it take to check all possibilities?

The Curse of Dimensionality

A term coined by R. Bellman (1957)

The number of states grows exponentially when the dimensionality of the problem increases

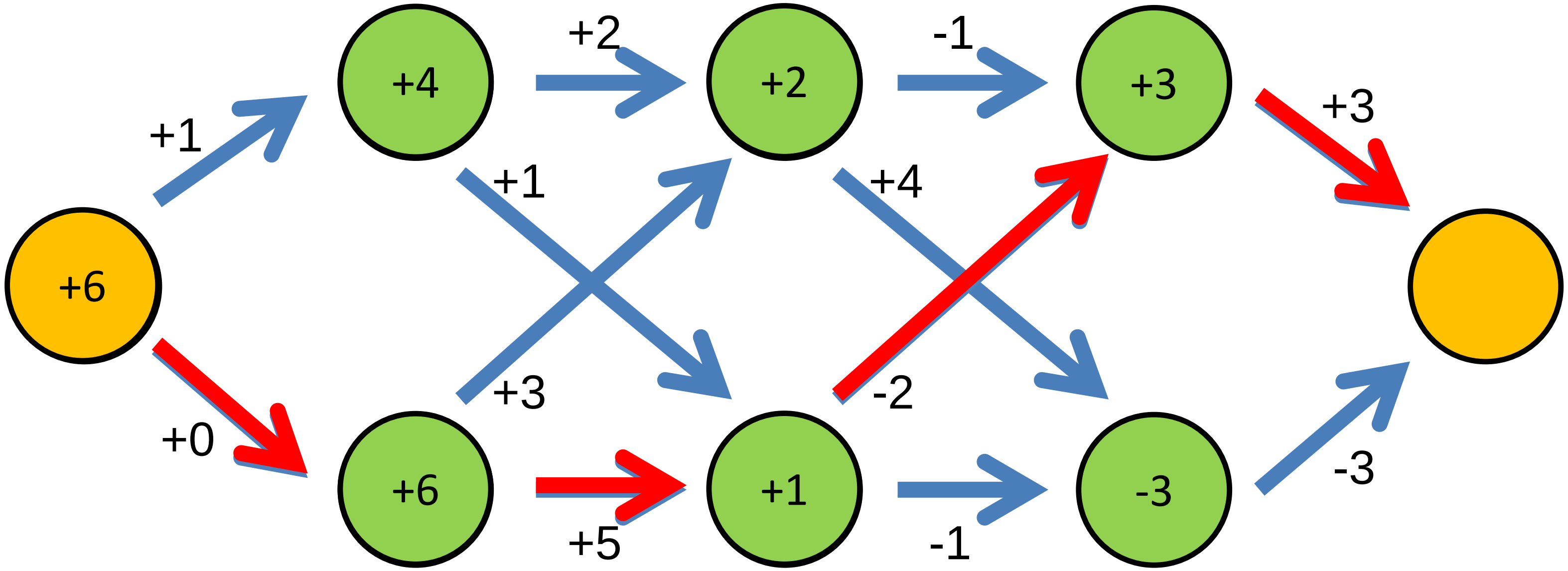
Can we have a non-brute-force algorithm?

Dynamic Programming Principle

*On an optimal path (following the optimal policy), if we start at **any** state in that path, the rest of path must again be **optimal***

Dynamic Programming in Action

Define
$v_t(s)$ = best we can do starting from circle s at step t

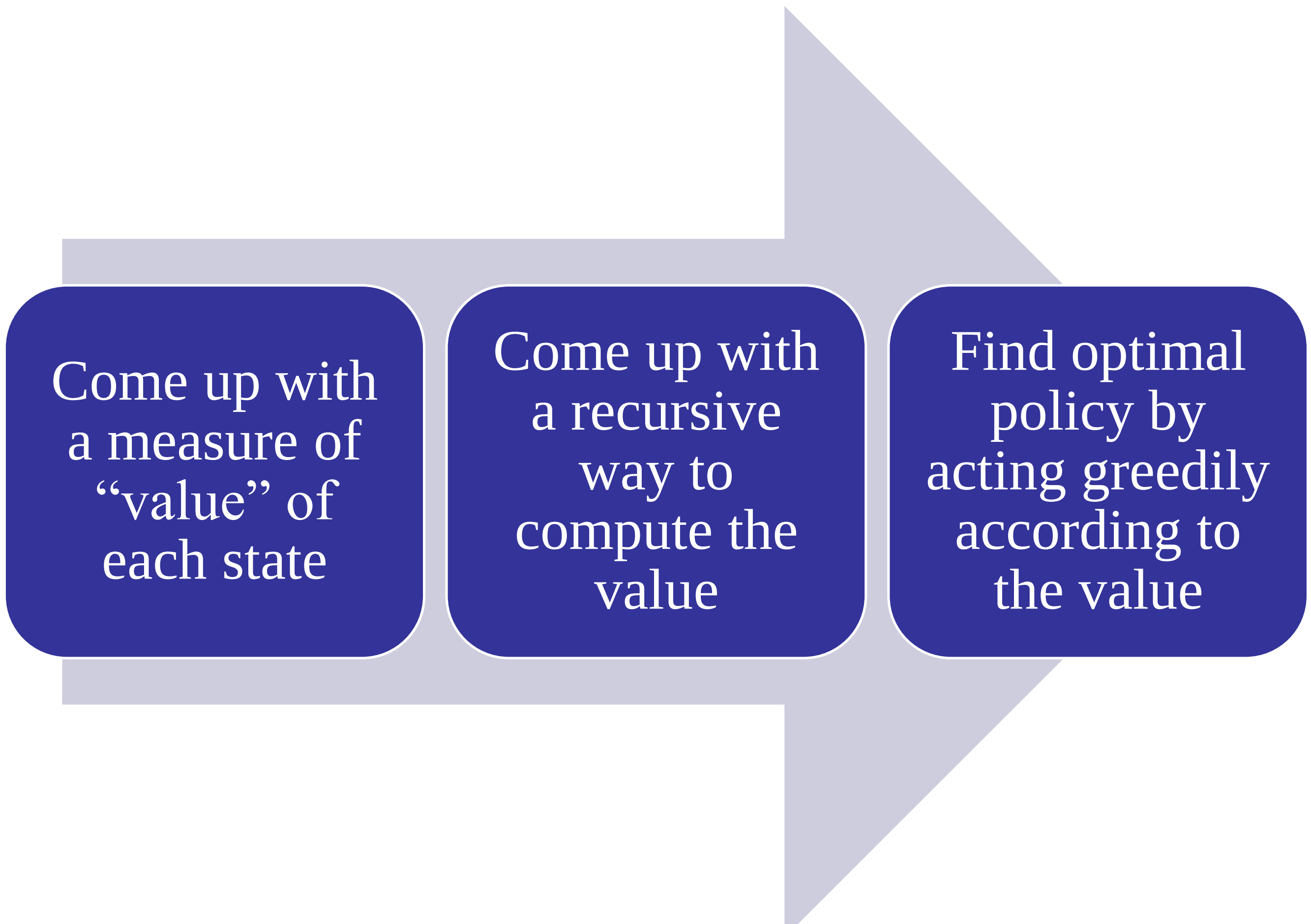


The Complexity of Dynamic Programming

We have shown that brute-force search takes at least N^T steps.

What about dynamic programming?

Summary of Key Ideas



Bellman's Equations and Optimal Policies

Value Function

As motivated earlier, we define the **value function**

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_k \gamma^k R_{t+k+1}\,|S_t = s\right]$$

and the **action value function**

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

Our goal: derive a recursion for $v_\pi(s)$ and $q_\pi(s,a)$

These are known as **Bellman's equations**

Relationship between v_π and q_π

Using the definitions, we can show the following relationships:

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$$

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

Combining, we get

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

This is known as the **Bellman's equation** for the value function

Bellman's Equation

For finite MDPs, the Bellman's equation can be written as

$$v_\pi = \gamma P(\pi) v_\pi + b(\pi)$$

This is a linear equation, and we can show that there exists a unique solution for v_π.

In fact, it is just

$$v_\pi = \left(I - \gamma P(\pi)\right)^{-1} b(\pi)$$

whose existence and uniqueness follow from the invertibility of $I - \gamma P(\pi)$, which in turn follows from $\|P\|_\infty = 1$.

Bellman's Equation for Action-Value Function

Using similar methods, one can show that the action value function satisfies a similar recursion

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a')\right]$$

Exercise: derive this equation and show that there exists a unique solution

Comparing Policies

We can compare policies via their values

- Given π, π', we say $\pi \geq \pi'$ if $v_\pi(s) \geq v_{\pi'}(s)$ for all s
- This is a **partial order**

Examples

- $\mathcal{S} = \{s_1, s_2, s_3\}$, $v_\pi = (2, 4, 6)$, $v_{\pi'} = (1, 3, 6)$. Then $v_\pi \geq v_{\pi'}$
- $\mathcal{S} = \{s_1, s_2, s_3\}$, $v_\pi = (2, 4, 6)$, $v_{\pi'} = (3, 3, 6)$. Then neither $v_\pi \geq v_{\pi'}$ nor $v_{\pi'} \geq v_\pi$ holds

Optimal Policy

We define an **optimal policy** π_* to be any policy satisfying

$$\pi_* \geq \pi \;\; \forall \text{ other policies } \pi$$

In other words, $v_{\pi_*}(s) \geq v_\pi(s)$ for all s and all π

- Does such a π_* exist?
- Is it unique?

Policy Improvement

We can derive the following result:

For any two policies π, π', if

$$\sum_a \pi'(a|s) q_\pi(s,a) \geq \sum_a \pi(a|s) q_\pi(s,a) \;\; \forall\, s$$

Then we must have

$$v_{\pi'}(s) \geq v_\pi(s) \;\; \forall\, s$$

In addition, if the first inequality is strict for some s, then the second equality is strict for at least one s.

Bellman's Optimality Condition

A policy is optimal if and only if for any state-action pair (s, a) such that $\pi(a|s) > 0$, we have

$$a \in \arg\max_{a'} q_\pi(s, a')$$

This means that an optimal policy must choose an action that maximize its associated action value function.

This then implies the existence of an optimal policy!

Bellman's Optimality Equation

Corresponding to an optimal policy

$$\pi_*(s) \in \arg\max_a q_*(s,a)$$

we obtain the following recursion

$$v_*(s) = \max_{a\in\mathcal{A}} \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')]$$

$$q_*(s,a) = \sum_{s',r} p\,(s',r|s,a)\left[r + \gamma \max_{a'\in\mathcal{A}} q_*(s',a')\right]$$

These are known as the **Bellman's optimality equations**

Some Remarks

The optimal value function v_* is unique. Is an optimal policy π_* unique?

Observe that the policy generated from v_* can be taken to be **deterministic**

$$\pi_*(s) \in \arg\max_a q_*(s, a)$$

In fact, for **every** policy π there exists a deterministic policy π' such that $\pi' \geq \pi$**!**

Summary

We introduced

- Basic formulation of reinforcing learning
- MDP as the mathematical framework
- Bellman's equations characterizing optimal policies

Next time: algorithms to solve RL problems