

# Dynamic Programming

Finding the Optimal Policy given a model of the environment

Nicole Orzan

November 28, 2021

# Recap

The mathematical framework of Reinforcement Learning:

- A set of possible **states**  $\mathcal{S}$  where  $s_t \in \mathcal{S}$  is the current state
- A set of possible **actions**  $\mathcal{A}$  where  $a_t \in \mathcal{A}$  is the current action
- A **transition function**  $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$
- A **reward function**  $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$  which returns  $r_t$

## Markov Decision Processes (MDPs)

These components allows us to define a **Markov Decision Process**

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r \rangle.$$

# Recap

## Key Concepts

- The agent aims to learn a **policy**  $\pi : \mathcal{S} \rightarrow \mathcal{A}$
- The goal of the agent is to maximize the (discounted) **cumulative reward**  $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ .
- We can find the optimal policy by making use of the **value functions**  $V^{\pi}(s)$  and  $Q^{\pi}(s, a)$
- We saw how to find an optimal policy in the specific case where **one single state** is available

Today we will consider the full Reinforcement Learning problem with its **sequential nature**.

# Today's Agenda

- ① Value Functions
- ② Dynamic Programming
- ③ Control
- ④ Policy Evaluation
- ⑤ Iterative Dynamic Programming Algorithms

## Change in Notation

### Attention!

So far we denoted the reward function as  $\mathcal{R}(s, a, s_{t+1})$ .  
For the sake of simplicity, hereafter we will denote it as  $r$ .

# Value Functions

The **state-value function**  $V^\pi(s)$ :

$$\begin{aligned} V^\pi(s) &= \mathbb{E} \left[ G_t \mid s_t = s, \pi \right] \\ &= \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, \pi \right] \end{aligned}$$

The **state-action value function**  $Q^\pi(s, a)$ :

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E} \left[ G_t \mid s_t = s, a_t = a, \pi \right] \\ &= \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a, \pi \right] \end{aligned}$$

# Bellman Equations

The **state-value function**  $V^\pi(s)$ :

$$\begin{aligned} V^\pi(s) &= \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, \pi \right] \\ &= \sum_a \pi(a|s) \sum_{s_{t+1}} p(s_{t+1}|s, a) [r + \gamma V^\pi(s_{t+1})] \end{aligned}$$

The **state-action value function**  $Q^\pi(s, a)$ :

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E} \left[ \sum_k \gamma^k r_{t+k+1} \mid s_t = s, a_t = a, \pi \right] \\ &= \sum_{s_{t+1}} p(s_{t+1}|s, a) [r + \gamma V^\pi(s_{t+1})] \\ &= \sum_{s_{t+1}} p(s_{t+1}|s, a) \left[ r + \gamma \sum_{a_{t+1}} \pi(a_{t+1}|s_{t+1}) Q^\pi(s_{t+1}, a_{t+1}) \right] \end{aligned}$$

# Optimal Policy

The value functions allow us to define a partial ordering over policies.

$$\pi \geq \pi' \text{ iff } V^\pi(s) \geq V^{\pi'}(s) \text{ for all } s \in \mathcal{S}$$

We can find the optimal policies by **maximizing** the state-value and state-action value functions results in the **optimal value functions**.

$$V^*(s) = \max_{\pi} V^{\pi}(s) \qquad Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$$

There is always at least one policy better than any other one:  $\pi^*$ .



# Bellmann Optimality Equations

$$\begin{aligned} V^*(s) &= \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, \pi \right] \\ &= \max_a \sum_{s_{t+1}} p(s_{t+1} | s, a) [r + \gamma V^*(s_{t+1})] \end{aligned}$$

$$\begin{aligned} Q^*(s, a) &= \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a, \pi \right] \\ &= \sum_{s_{t+1}} p(s_{t+1} | s, a) [r + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})] \end{aligned}$$

# Dynamic Programming

With the term Dynamic Programming we refer to a set of algorithms that, **given a model of the environment**  $p(s_{t+1}|s, a)$ , allow us to find the optimal policies.

## Key Idea

The key idea of Dynamic Programming is the **use of the Bellman equations** to organize the search for good policies.

# Dynamic Programming

The goal of Reinforcement Learning is to find the optimal policy  $\pi(a|s)$  for each  $s \in \mathcal{S}$  for a given problem.

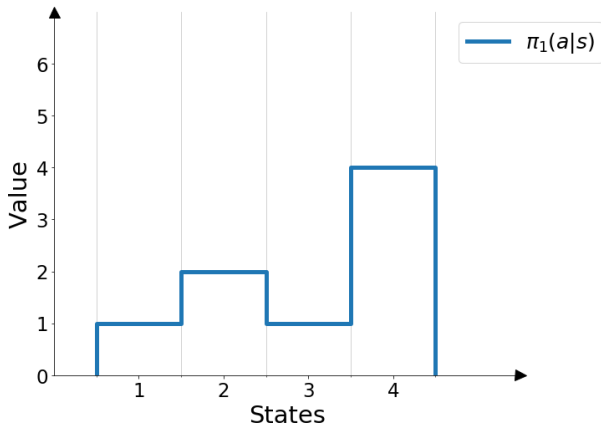
This task is called **Control**.

Often, to be able to solve the problem of Control, it is necessary to evaluate the goodness of a policy.

This task is called **Policy Evaluation**.

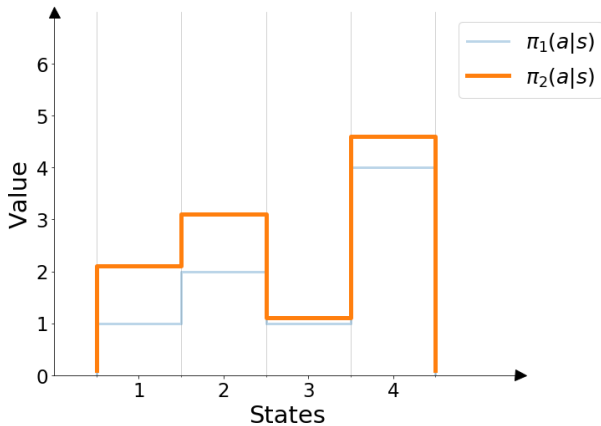
# Control

Improving a policy



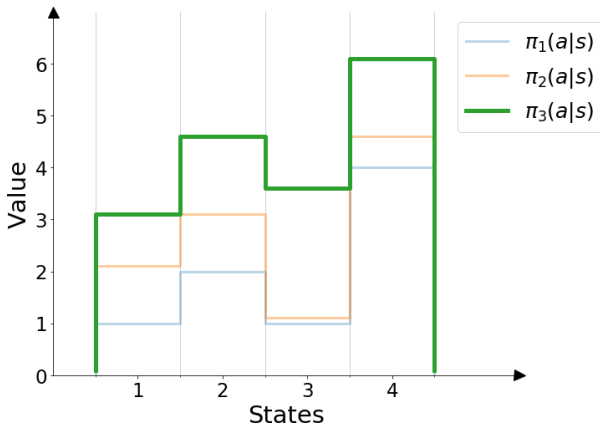
# Control

Improving a policy



# Control

Improving a policy



# Policy Evaluation

Evaluating a policy  $\pi$  means computing its state-value function (or state-action value function).

If the environment dynamic is completely known, this problem is reduced to finding the solution to a set of  $|\mathcal{S}|$  linear equations:

$$\begin{aligned} V^\pi(s) &= \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, \pi \right] \\ &= \sum_a \pi(a|s) \sum_{s_{t+1}} p(s_{t+1}|s, a) [r + \gamma V^\pi(s_{t+1})] \end{aligned}$$

## Iterative Policy Evaluation

Since the above computation is complex, we can compute the value function for a given policy **iteratively**:

- We choose  $V^0$  arbitrarily
- We compute a sequence of improved approximations  $V^1, V^2, V^3 \dots$  by applying the Bellman equation to the previous estimate:

$$\begin{aligned} V^{k+1}(s) &= \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, \pi \right] \\ &= \sum_a \pi(a|s) \sum_{s_{t+1}} p(s_{t+1}|s, a) [r + \gamma V^k(s_{t+1})] \text{ for each } s \in S \end{aligned}$$

The sequence  $\{V_k\}$  is proven to converge to  $V^\pi$  as  $k \rightarrow \infty$ .



# Gridworld

- $\mathcal{S}$ : 16 states, with 2 terminal ones (0 and 15)
- $\mathcal{A}$ : Four possible actions:  $\uparrow$  (up),  $\downarrow$  (down),  $\leftarrow$  (left),  $\rightarrow$  (right)
- the function  $p(s_{t+1}|s, a)$  is deterministic
- $\mathcal{R}(s, a)$ :  $r_t = -1$  for every step

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

# Iterative Policy Evaluation

Let's evaluate the **uniform random policy**, which has probability 0.25 of taking each of the 4 possible actions.  $\gamma = 1$ .

$$V^{k+1}(s) := \sum_a \pi(a|s) \sum_{s_{t+1}, r} p(s_{t+1}|s, a) [r + \gamma V^k(s_{t+1})]$$

$$V^1(s=1) = \pi(\uparrow | s=1) [-1 + V^0(s=1)] + \pi(\downarrow | s=1) [-1 + V^0(s=5)] + \pi(\leftarrow | s=1) [-1 + V^0(s=0)] + \pi(\rightarrow | s=1) [-1 + V^0(s=2)]$$

$V^0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$V^1$

0	?	?	?
?	?	?	?
?	?	?	?
?	?	?	0

# Iterative Policy Evaluation

Let's evaluate the **uniform random policy**, which has probability 0.25 of taking each of the 4 possible actions.  $\gamma = 1$ .

$$V^{k+1}(s) := \sum_a \pi(a|s) \sum_{s_{t+1}, r} p(s_{t+1}|s, a) [r + \gamma V^k(s_{t+1})]$$

$$V^1(s=1) = 0.25 [-1 + 0] + 0.25 [-1 + 0] + 0.25 [-1 + 0] + 0.25 [-1 + 0] = -1$$

$V^0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$V^1$

0	-1	0	0
0	0	0	0
0	0	0	0
0	0	0	0

# Iterative Policy Evaluation

Let's evaluate the **uniform random policy**, which has probability 0.25 of taking each of the 4 possible actions.  $\gamma = 1$ .

$$V^{k+1}(s) := \sum_a \pi(a|s) \sum_{s_{t+1}} \sum_r p(s_{t+1}|s, a) [r + \gamma V^k(s_{t+1})]$$

$$V^1(s = 1) = 0.25 [-1 + 0] + 0.25 [-1 + 0] + 0.25 [-1 + 0] + 0.25 [-1 + 0] = -1$$

$V^0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$V^1$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

# Iterative Policy Evaluation

$$V^{k+1}(s) := \sum_a \pi(a|s) \sum_{s_{t+1}} \sum_r p(s_{t+1}|s, a) \left[ r + \gamma V^k(s_{t+1}) \right]$$

$V^0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$V^N$

0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	0

# Iterative Policy Evaluation

---

**Algorithm 1** Iterative Policy Evaluation

---

Input the policy to evaluate  $\pi$

Define threshold for accuracy  $\theta$

Initialize  $V(s)$ , for all  $s \in \mathcal{S}$  arbitrarily, except  $V(\text{terminal}) = 0$

$\Delta \leftarrow 0$

**while** True

**for** each  $s \in \mathcal{S}$

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s_{t+1}|s, a)[r + \gamma V(s_{t+1})]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

    if  $\Delta < \theta$  return  $V$

---

## Policy Improvement

What happens if, instead of following policy  $\pi$  in state  $s$ , we decide to take action  $a$ ?

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E} \left[ \sum_k \gamma^k r_{t+k+1} \mid s_t = s, a_t = a, \pi \right] \\ &= \sum_{s_{t+1}} p(s_{t+1} | s, a) [r + \gamma V^\pi(s_{t+1})] \end{aligned}$$

If this value is greater than  $V^\pi(s)$ , we should definitely take action  $a$ .

# Policy Improvement

## Policy Improvement Theorem

Given any pair of deterministic policies  $\pi$  and  $\pi'$  such that, for all  $s \in \mathcal{S}$ :

$$Q^\pi(s, \pi'(s)) \geq V^\pi(s)$$

Then policy  $\pi'$  is as good as or better than  $\pi$ :

$$V^{\pi'}(s) \geq V^\pi(s)$$

An easy way to find a new policy  $\pi'$  which is always better than  $\pi$  is:

$$\pi' = \arg \max_{a \in A} Q^\pi(s, a)$$



## Policy Iteration

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi_* \xrightarrow{E} V^{\pi_*}$$

Since a finite MDP has only a finite number of policies, this process must converge to an **optimal policy** in a finite number of iterations.

# Policy Iteration

---

**Algorithm 2** Policy Iteration for estimating  $\pi \sim \pi^*$

---

**1) Initialization**

Initialize  $V(s)$  and  $\pi(s)$  arbitrarily for all  $s \in \mathcal{S}$

**2) Policy Evaluation Step** (Algorithm 1)

**3) Policy Improvement Step:**

policy-stable  $\leftarrow$  true

**for** each  $s \in \mathcal{S}$  **do**

old-action  $\leftarrow \pi(s)$

$\pi(s) \leftarrow \arg \max_a \sum_{s_{t+1}} p(s_{t+1}|s, a)[r + \gamma V(s_{t+1})]$

if old-action  $\neq \pi(s)$ , policy-stable  $\leftarrow$  false

if policy-stable = true then return  $\pi \sim \pi^*$  and  $V \sim V^*$

else go to 2

---

# Value Iteration

**Drawback** of Policy Iteration: in each iteration we need to perform policy evaluation.

The policy evaluation step can be **truncated** without losing convergence guarantees.

$$\begin{aligned} V^{k+1}(s) &= \max_a \mathbb{E} \left[ \sum_k^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right] \\ &= \max_a \sum_{s_{t+1}} p(s_{t+1}|s, a) [r + \gamma V^k(s_{t+1})] \text{ for each } s \in \mathcal{S} \end{aligned}$$

# Value Iteration

---

**Algorithm 3** Value Iteration Algorithm

---

Define threshold for accuracy  $\theta$

Initialize  $V(s)$  for all  $s \in \mathcal{S}$  arbitrarily, except  $V(\text{terminal}) = 0$

$\Delta \leftarrow 0$

**while** True **do**

**for** each  $s \in \mathcal{S}$  **do**

$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s_{t+1}} p(s_{t+1}|s, a) [r + \gamma V(s_{t+1})]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until  $\Delta < \theta$

Output a deterministic policy  $\pi \sim \pi^*$ , such that:

$\pi(s) \leftarrow \arg \max_a \sum_{s_{t+1}} p(s_{t+1}|s, a) [r + \gamma V(s_{t+1})]$

---

# Generalised Policy Iteration

We define generalized policy iteration as the **alternative execution of policy evaluation and policy improvement**, regardless of the granularity of the two processes.

This process stabilizes only when we found a policy that is greedy with respect to its own value function.

# Final Slide!

## Lecture Takeaway

1. Given  $p(s_{t+1}|s, a)$  we can find the optimal policy mathematically
2. But better use Dynamic Programming algorithms
3. It works by alternating the tasks of Policy Evaluation and Control